Collaborative Problem Solving using a Cloud-based Infrastructure to Support High School STEM Education

Ms. Satabdi Basu, Vanderbilt University, Institute for Software Integrataed Systems

Satabdi Basu is a Ph.D. candidate in the Department of Electrical Engineering and Computer Science at Vanderbilt University. She has an undergraduate degree in Computer Science and Engineering from West Bengal University Of Technology, India, and a M.S. degree in Computer Science from Vanderbilt University. Her research interests include learning from multi-agent simulation models, computational thinking, scaffolding learning analytics and user modeling. She is currently a Research Assistant at the Institute for Software Integrated Systems and works on a NSF-funded project for teaching middle school students science and computational thinking simultaneously in curricular settings.

Dr. John S Kinnebrew, Vanderbilt University
Mr. Shashank Shekhar, Vanderbilt University
Mr. Faruk Caglar
Mr. Tazrian Haider Rafi, Vanderbilt University

Tazrian Haider Rafi is an undergraduate student studying computer science at Vanderbilt University.

Dr. Gautam Biswas, Vanderbilt University

Gautam Biswas is a Professor of Computer Science, Computer Engineering, and Engineering Management in the EECS Department and a Senior Research Scientist at the Institute for Software Integrated Systems (ISIS) at Vanderbilt University. He has an undergraduate degree in Electrical Engineering from the Indian Institute of Technology (IIT) in Mumbai, India, and M.S. and Ph.D. degrees in Computer Science from Michigan State University in E. Lansing, MI. Prof. Biswas conducts research in Intelligent Systems with primary interests in hybrid modeling, simulation, and analysis of complex embedded systems, and their applications to diagnosis, prognosis, and fault-adaptive control. He is also involved in developing simulation-based environments for learning and instruction. In his research, he has exploited the synergy between computational thinking ideas and STEM learning to develop systems that help students learn science and math concepts by building simulation models. He has also developed innovative educational data mining techniques for studying students' learning behaviors and linking them to metacognitive strategies. Prof. Biswas is a Fellow of the IEEE.

Dr. Aniruddha Gokhale, Vanderbilt University

Collaborative Problem Solving using a Cloud-based Infrastructure to Support High School STEM Education (RTP, Strand 2)

Abstract:

This paper discusses a challenge-based, collaborative, community-situated STEM learning environment - C^3STEM that is aligned with the next generation science standards. In C^3STEM, students synergistically learn STEM (Science, Technology, Engineering and Mathematics) and CS (Computer Science) concepts by solving realistic problems that provide a framework for applying scientific and engineering practices. A recent study conducted with 26 high school students in middle Tennessee showed that our approach resulted in the students making significant learning gains in both fundamental STEM and computational concepts. Furthermore, they were successful in working collaboratively in small groups to find good solutions to an overall challenge problem on optimizing traffic flow through adjacent city intersections. We discuss how students worked through the two components of the C^3STEM system: CTSiM (**C**omputational **T**hinking using **Si**mulation and **M**odeling) and C^2SuMo (**C**ollaborative **C**loud-based **S**caled **u**p **Mo**deling), present the experimental study we conducted, and discuss the results in detail. We end the paper with a summary of our accomplishments, and directions for future research.

1. Introduction

The Next Generation Science Standards place significant emphasis on combining the learning of fundamental concepts with scientific and engineering practices that help students develop useable knowledge that they can apply across multiple problems. In more detail, the standards emphasize that students develop the skills to build and use models, plan and conduct experiments, analyze and interpret data, and apply computational and mathematical thinking to explain phenomena and demonstrate understanding of core ideas. In addition, students are also expected to demonstrate understanding of several engineering practices, including design and evaluation[1]. Over the last three years, our research group has developed an innovative community-situated, challenge-based, collaborative learning environment (C^3STEM) that harnesses computational thinking, modeling, and simulation situated in the framework of real-world problem solving to support ubiquitous STEM learning in high school classrooms.

Real-world problems are typically quite different from the types of problems students solve in regular classroom assignments, primarily because they are ill-defined, messy, and knowledge intensive[2]. However, it is known that tackling real-world problems motivates students[3,4], promotes deep thinking, reasoning, and problem-solving skills[5,6], while providing opportunities for developing metacognitive and self-regulated strategies for becoming more independent and effective learners[7]. Unfortunately, even today, most high school students are rarely given opportunities to engage in real-world problem solving. By introducing modeling and simulation using computational thinking methods, along with more realistic problem solving in the high school curriculum, our goal is to make instruction more learner-centered, promote awareness and engagement by linking learning to real-world problems, and support deep learning of fundamental concepts through applying those concepts to solve complex problems.

C^3STEM incorporates two core learning environments based on modeling and simulation: (1) CTSiM (Computational Thinking using Simulation and Modeling) for fundamental physics- and calculus-based modeling of vehicle and traffic operation, while also incorporating parameters that imply driver behaviors[8, 9], and C^2SuMo (**C**loud-based, **C**ollaborative, **S**caled-**u**p **Mo**deling) for macro-level modeling that uses a Google Maps interface along with the high-fidelity **S**imulation of **U**rban **MO**bility (SUMO), a freely-available open-source visual traffic simulator for scaled-up, city-wide traffic simulations[10]. Students start with CTSiM, a visual modeling and simulation environment that emphasizes computational thinking[11, 12], to learn how vehicle dynamics and driver behavior models can be implemented using an agent-based paradigm. They combine their fundamental knowledge of Newton's laws of mechanics, i.e., relations between vehicle acceleration, speed, and distance to model vehicle movement, and then extend these models to include driver behaviors, such as look-ahead distance and gap acceptance to model traffic flow on city streets. CTSiM provides the basis for designing a progression of C^3STEM modeling and behavior analysis units that start from vehicles driving at constant speed to modeling traffic flow at intersections with traffic lights. After working individually on a set of CTSiM and C^2SuMo units, students work collaboratively in small groups to solve a challenge problem that requires them to find the right timing sequences for a set of adjacent traffic lights so as to maximize traffic flow (minimize waiting times) through the intersections.

This paper discusses the results of a recent study we conducted with 26 high school students in metro Nashville. The study demonstrated that our approach to teaching STEM learning and problem solving using real-world challenges kept students engaged and on task; they showed significant learning gains in both fundamental STEM and computational constructs, and all of the groups successfully completed the challenge problem. The rest of this paper is organized as follows. Section 2 provides a brief description of the two C^3STEM components CTSiM and C^2SuMo and the learning activity progression using these components. Section 3 discusses the study methods, and Section 4 discusses the results in terms of students' learning gains, their model-building activities, and their overall approach to solving the challenge problem. Section 5 presents the conclusions and directions for future work.

2. The C^3STEM learning environment and learning activities

We briefly discuss the two primary components of the C^3STEM environment: CTSiM and C^2SuMo in this section.

2.1 The CTSiM environment

CTSiM provides an agent-based, visual programming interface for constructing computational models and allows students to execute their models as simulations and compare their models' behaviors with that of an expert model[8,13,14]. In CTSiM, students design, build, simulate, and verify their computational models using three interrelated sets of activities:

1. **Construction**: Using the visual programming interface, students build a computational model in block-structured form using visual primitives that represent computational constructs that are parameterized using domain-specific variables, such as acceleration, speed, and distance for vehicle models and look ahead distance to capture driver behavior. Students select the visual primitives from a list and arrange them spatially, using a

drag-and-drop interface to build their computational models. Each behavior of the different agents in the simulation is modeled as a separate computational module.

2. **Enactment**: It involves a microworld that provides an interface for running multi-agent simulations of partial or complete domain models using NetLogo[15]. This environment allows students to set parameters for their simulations and employ a trace feature to link their computational model execution to animations of agent behaviors and visualizations of behavior variables as plots produced by Netlogo.
3. **Envisionment**: In this environment, students can design simulation experiments to analyze, refine, and validate the behavior of their models by comparing their model behaviors against the behaviors generated by an expert model that runs synchronously.

Figures 1 and 2 illustrate snapshots of the Construction and Envisionment interfaces for CTSiM depicting the code defining vehicle driving behavior, and an animation that depicts the flow of vehicles through an intersection with stop lights, respectively. As discussed, students use the drag and drop interface in the Construction world to drag parameterized computational blocks from the list on the left of the interface to build their computational model of vehicle behaviors. The organization of the computational primitives (e.g., conditionals, loops, mathematical expressions, and logical operators) regulate the flow of execution in the computational model, while the domain-specific primitives generally represent agent actions (e.g., change position, increase-velocity, execute-left-turn) and sensing conditions (e.g., approaching STOP sign, is left turn light green?). Students can study the behaviors generated by their models in the form of agent-based simulations in the Enactment interface, and compare their model behaviors against that of an "expert" model in the Envisionment interface. NetLogo visualizations and plotting functions provide students with a dynamic, real-time display of how their agents (in this case, vehicles and drivers) operate in the environment specified by the current problem. Students can observe vehicle behaviors in the animations, and study the emergence of aggregate system behaviors by studying the generated plots and the behaviors implied by the animations. As discussed, they can also turn on a debug feature, and trace the execution of their code block by block in the Enactment interface. Although the expert model is hidden from the students, they can observe its simulated behavior and compare it with their own models, through the synchronized side-by-side plots and microworld visualizations[8, 16].

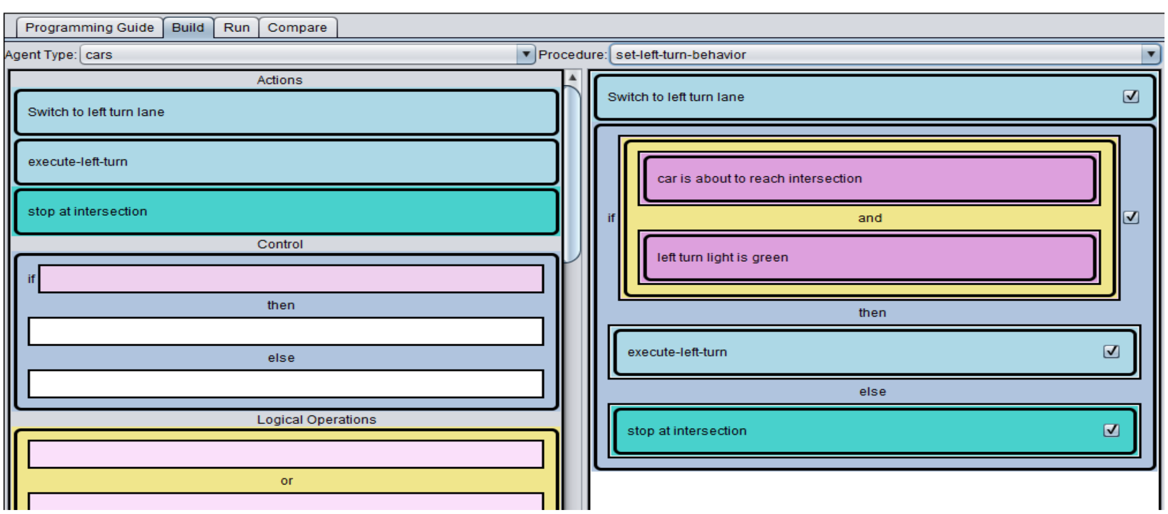

Figure 1: The Construction interface for modeling a left-turn-behavior at a traffic intersection

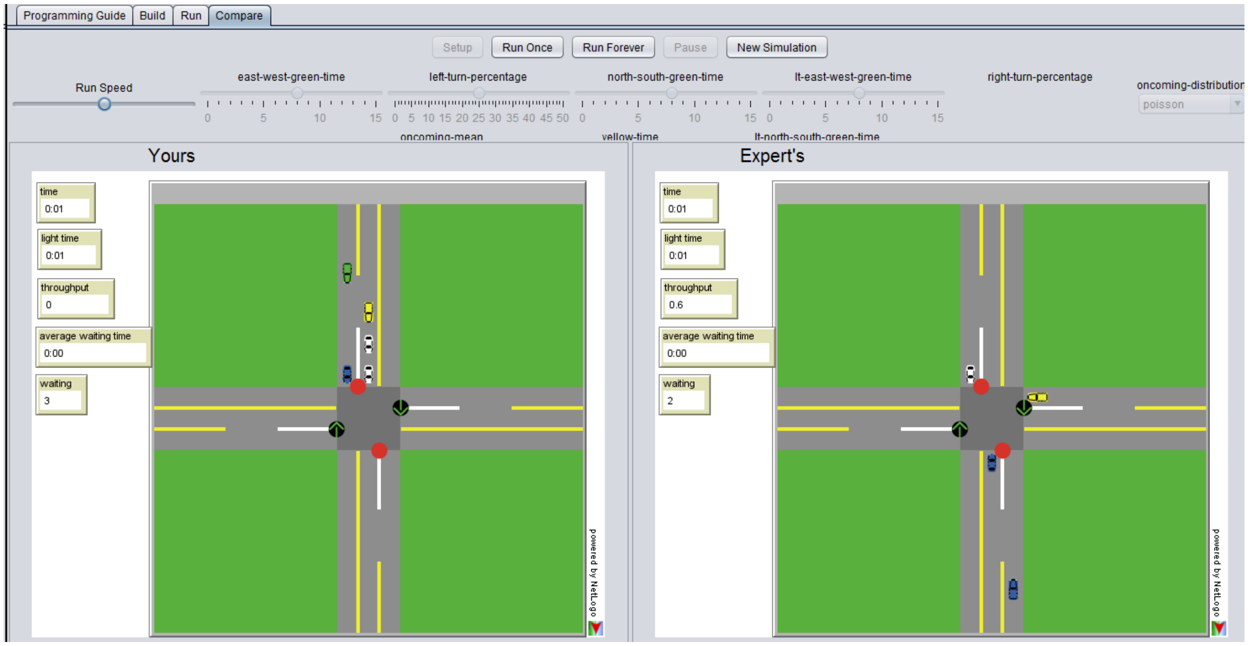

Figure 2: The Envisionment interface for comparing students' traffic intersection models against an expert model

Traffic modeling and simulation provides an excellent domain for learning a variety of STEM concepts in the CTSiM computational thinking framework. The modeling task can be performed at many different levels of abstraction and complexity. In the simplest model, students use the basic form of Newton's first and second laws to model vehicles traveling at constant speed, or accelerating and decelerating as they are driven. The level of complexity increases when students model vehicles making left and right turns. Students have to model more complex logic when considering driver behavior associated with following other vehicles, making unprotected left turns with oncoming traffic, and modeling stopping and starting behaviors at STOP signs and traffic lights at intersections. Additional logic has to be considered when changing lanes to make left and right turns at intersections. Note that all of the computational models of vehicle behaviors that the students construct are contextualized by their understanding of how actual vehicles are driven on city streets. This facilitates their monitoring and interpretation of the models they are constructing, thus providing them with opportunities for self-directed learning in contexts that they can relate to. We have designed a set of six CTSiM units that focus on fundamental STEM and computational concepts required to build simulation models. We briefly describe them as part of the learning progression in Section 2.3.

2.2 The C²SuMo environment

Figure 3 illustrates the high-level architecture of the C²SuMo framework that is designed to help students to scale up their 'first principles' models in CTSiM to scenarios that represent flow of traffic flow through actual city streets. We tailored the C²SuMo environment to support both individual and small group work by students. First, students work individually to build traffic simulation models, and visualize and analyze the generated traffic flow using a Google Maps interface to solve a variety of traffic flow problems. Then they collaborate and work in small groups

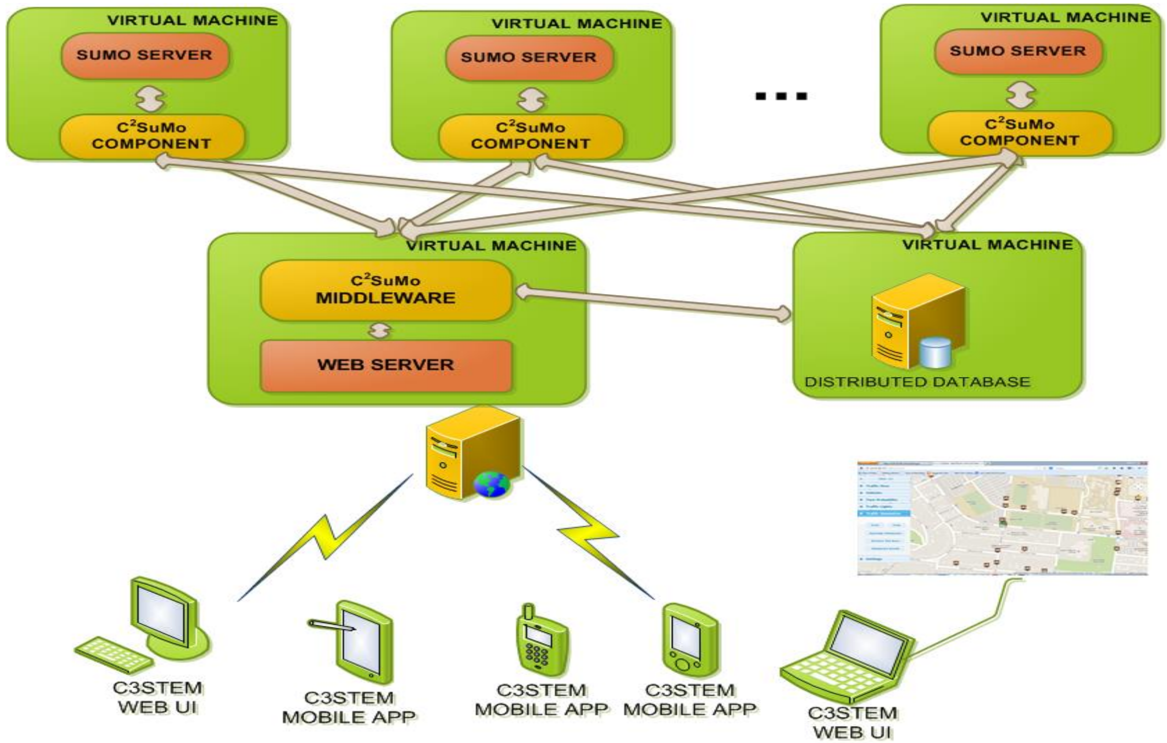

Figure 3: C^{2}SUMO Architecture and Components

of 2-4 students to build more complex traffic flow models that correspond to a complex challenge problem. In the study reported in this paper, students worked in groups of three to determine the timing of traffic lights that would optimize flow of traffic through two adjacent city intersections. They did this by running a set of experiments, using different timing sequences, and analyzing the results to empirically establish the timing sequence that maximized traffic flow through the two intersections.

Access to the C^{2}SuMo framework is ubiquitous: various client-side modalities such as laptops, Android tablets, and iPads are supported. To make the computational architecture flexible and applicable to solving a variety of problems, the C^{2}SuMo design is composed of pluggable components with loosely coupled user interface, middleware, and database components, all connected to the backend SUMO simulation engine. Scalability has been an important consideration while designing the system. This consideration has led to building a cloud-based elastic deployment model, which allows ubiquitous access and scalability based on the workload.

To enhance the realistic flavor of the SUMO simulations, the simulation results are projected onto a Google Maps interface, where the students can observe and study the results of their simulations on the map interface that represents real city streets. The web and mobile interfaces have been implemented using the jQuery-based feature-rich JavaScript library and the Google Maps API to disseminate the content-rich data. In addition to web browsers, the interface can cater to other mobile interfaces, such as Android and iOS-based tablets. The Google Maps API is a robust, powerful, and intuitive technology that can be embedded into webpages using JavaScript. It provides a customizable and extendable model to simulate the traffic and manipulate the traffic model. Some of its key functionalities include zoom in, zoom out, panning, and adding icons.

The C^2SuMo web user interface is depicted in Figure 4. This interface is accessed after successfully signing into the system. The left side panel comprises menu items, which allow students to control various traffic related parameters, such as vehicle acceleration, deceleration, length, and traffic light timing. For running experiments, students are provided with an *Experiment Mode* menu interface, which allows them to systematically change parameter values to produce results in the form of plots. As students switch modes, problem numbers listed in the *Problem Set* menu switch accordingly to the individual and collaborative (i.e., challenge) problems. The layout of the C^2SuMo web user interface is adjusted as the problem number being solved is changed. Additionally, the rest of the left side menu items are used to control simulation related tasks.

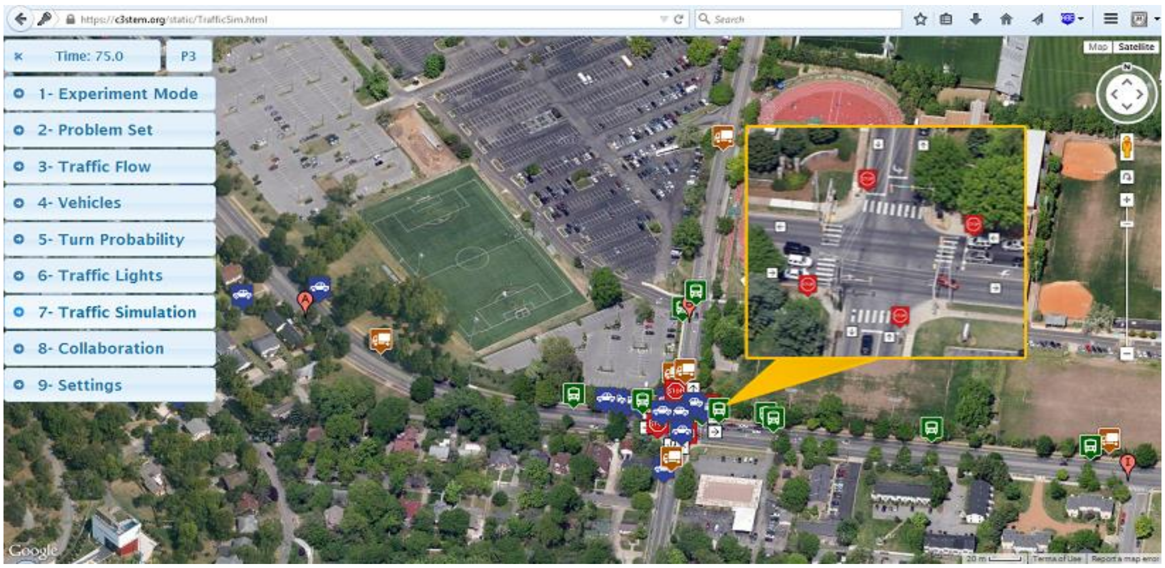

Figure 4: C^2SUMO Web User Interface

The application data including users, simulations, results, map region as well as images and videos are stored in a distributed, non-relational, and horizontally-scalable NoSQL database called MongoDB**Error! Reference source not found.**. It provides document-oriented storage, supports large amounts of persistent data storage, and also provides flexible data processing. This enables the students to work with large data sets, the need for which will increase when we have students working with real traffic streaming data.

The web server acts as a broker between the different C^2SuMo components. The mobile and web clients send requests to the web server, which delegates them to the C^2SuMo middleware and appropriately routes the replies with the results obtained from the middleware. The web framework is composed of the Apache web server, mod_wsgi Apache HTTP Server module and CherryPy object-oriented web framework. This setup supports high traffic demands that may arise when multiple students are running simulation experiments at the same time. This framework also supports static content and caching.

SUMO allows importing road networks using a Geographical Information System (GIS) map exported from OpenStreetMap[18] or any other GIS tool. These can be converted into a format compliant with the SUMO environment using the netconvert[19] application. The newly generated road network comprises components, such as lanes, junctions, traffic lights, points of interests, and edges, which are later utilized in generating the traffic simulation.

C^2SuMo middleware is integrated with the OpenStack-based cloud infrastructure[20]. The middleware is written in Python and uses OpenStack's Nova client API to launch virtual machines (VMs) that run SUMO servers to support multiple parallel simulations. The communication with SUMO occurs via the Traffic Control Interface (TraCI) API, which uses a client/server architecture and enables an application to communicate with the running traffic simulation via the TraCI protocol. TraCI allows a system to access the object information in the simulation environment and makes modifications on the objects' behavior at run-time. The protocol is modified to support multiple simultaneous server connections. SUMO is launched as a server application on the VMs and listens at a specified port number to establish a channel for incoming client connections. Vehicles, lanes, traffic lights, induction loops, routes, junctions, and edges can be retrieved and the state of a lane, traffic light, vehicle, route, edge can be manipulated through TraCI. Thus, the C^2SuMo communication middleware orchestrates the communication traffic between the back-end model execution modules and front-end user interfaces through the web server.

The C^2SuMo framework also incorporates a Google Hangouts[21]-based collaborative environment, where students communicate with each other remotely while collaborating to solve problems. Students can also use Hangouts to get guidance from instructors and domain experts. Our environment is currently set up to support 10 participants for video calls and up to 100 participants for sending and receiving messages. The Google Hangouts API provides rich, real-time collaborative functionalities, which allows users to share their C^2SuMo parameters to run experiments, compile results, and generate reports even when they are working remotely from different locations (e.g., their homes).

2.3 Learning activity progression

We developed a learning activity progression for high school students that includes a combination of 10 CTSiM and C^2SuMo traffic modeling and simulation units with accompanying problems for students to solve. For the research study described in Section 3.1, students worked on some of the units in class and the other units as homework. Table 1 describes each unit and its accompanying learning activities. Further, it indicates which of the units were assigned as homework (marked as HW[1]). During the periods when students met with the researchers, previous homework activities were discussed and students' difficulties and questions were answered. Students worked on units 19 individually, but they were allowed to discuss modeling concepts with each other or with their instructors. In other words, these units were guided learning activities supported by formative assessments that were stated in the form of problem assignments (see the Activities column of Table 1).

Unit 10 was the challenge problem that students worked on in groups of 3 both during and outside the study. For this C^2SuMo unit, the collaboration among students was structured so that one of three students first solved the traffic light problem for intersection 1, a second student

[1] The term homework is used rather loosely. It implies that the students worked on these modules outside of class on their own time

worked on the light sequence timings for intersection 2, and the third student was assigned the role of the administrator or coordinator. This student was given charge for running the coordinated simulations for the two intersections, and was responsible for documenting the results. Analysis of the results and deriving the optimal or best traffic light sequence was a collaborative venture. Since students worked on parts of the challenge problem as homework outside the study, we were not able to monitor all the ways in which students collaborated and make no claims of how they collaborated to solve the challenge problem. We will redesign the collaboration interfaces in the future so we can study the collaboration process in a more effective way.

Table 1: The C^3STEM learning activity progression

Unit name	System	Description	Activities
Unit 1: Constant velocity	CTSiM	Model vehicles traveling at constant velocity and calculate the distance traveled at each step	Run simulations to understand relations between velocity and distance
Unit 2: Stop sign	CTSiM	Model vehicles approaching, stopping, and speeding up at a stop sign	Run simulations to understand relations between look-ahead distance and max deceleration to ensure safe stopping
Unit 3: 4-way stop sign	C^2SuMo	Study how the flow of vehicles through a 4-way stop sign is affected by vehicle speed, acceleration, and length of the vehicle.	Document values of average waiting time, average vehicle speed and throughput, queue length and duration for different simulation parameters
Unit 4: Left right turn (**HW**)	CTSiM	Model vehicles making a smooth turn at an intersection with a stop sign	Observe the turning of the car for different number of discrete steps and discuss the effect on the smoothness of the turn as a function of the steps.
Unit 5: Safe left turn (**HW**)	CTSiM	Model a driver making an unprotected left turn at an intersection with oncoming traffic	Run simulations to study the effect of oncoming car velocity on 'safe' gap acceptance times for left turns
Unit 6: Left turn at an intersection (**HW**)	C^2SuMo	Compare traffic flow parameters when a vehicle makes a left turn at different intersections – unprotected turn, 4-way stop, and traffic lights	Document and compare values of average waiting time, vehicle speed and throughput, queue length and duration for the three types of intersections
Unit 7: Intersection with traffic lights	CTSiM	Model driver behaviors at intersections with traffic lights	Vary traffic light timing sequences and collect throughput and average wait time data with the goal of maximizing throughput and minimizing wait time at the intersection
Unit 8: A basic traffic light unit	C^2SuMo	Model the traffic light timing sequence for a single intersection	Experiment with the timing of traffic light cycles to produce the minimum average wait time at the intersection
Unit 9: Left-right turns at traffic light intersections (**HW**)	CTSiM	Model traffic flow through an intersection with lights that allows left and right turns. Left turns are protected.	Vary traffic light timings at the intersection and conduct a series of experiments to determine the timing sequence that achieves maximum throughput and minimum waiting time at the intersection
Unit 10: Challenge problem – Collaborating to solve flow through two intersections (**HW**)	C^2SuMo	Collaborate to study the timing sequence for adjacent traffic light intersections o to optimize traffic flow through the intersections.	Experiment with the timing of traffic light cycles to produce the minimum wait time and maximum throughput at both intersections; also study effects of driver behavior and lane closures on the average wait time

The progression of units listed in Table 1 was designed to capture the increase in complexity of the traffic scenarios and driver behaviors. Students started with modeling vehicles moving at a constant velocity and then proceeded to model how the velocity of vehicles changed as they approached and departed from stop signs. Next, they worked on modeling how vehicles make a smooth continuous turn at an intersection with a stop sign, followed by modeling unprotected left turns that required an estimation of oncoming traffic velocity. Finally, students modeled driver behavior at intersections with traffic lights that included making left/right turns at such intersections. This also involved more complex driver behaviors like making decisions on when to switch to the left-turn lane.

In previous studies, we sequenced the units such that students first worked on all of the CTSiM units before they switched to working on the C^2SuMo units. However, feedback from the students and their science teachers suggested that the progression would be more effective and lead to better learning and understanding if the related CTSiM and C^2SuMo units were grouped together, thus resulting in the interleaving of the units. For example, students worked on a Stop sign model in the Unit 2 CTSiM assignment, modeling how cars approaching an intersection slow down starting from a look-ahead distance that is a driver behavior parameter, come to a stop at the intersection, and then accelerate from rest till they achieve their initial speed, which corresponds to the speed limit specified for that road. The students were assigned the task of studying the relation between look-ahead time and the deceleration rate. As an extreme case, if the look-ahead distance was set to too small a value, the car would need to decelerate at a rate higher than the maximum possible for that car if it had to come to a complete stop at the intersection. In Unit 3, the corresponding C^2SuMo unit, the assumption was the students had figured out this relation from the CTSiM experiments, and the problem-solving focus was changed to studying the average wait times and throughput for cars at a 4-way Stop intersection. Similar relations exist between other CTSiM and C^2SuMo units (see Table 1). Our hypothesis was this would provide the students more opportunities for connecting fundamental concepts made transparent in the CTSiM models to behaviors observed in the scaled-up real-world but more black-box C^2SuMo simulations.

3.0 Methods

3.1 Participants and procedure

Twenty-six 10^{th}-grade honors students in an elective science course participated in the study. The honors program is implemented through a partnership between the local Tennessee public school district and the Vanderbilt School for Science and Math. Its curriculum connects sciences, technology, engineering, and mathematics through hands-on exploration with a progressive increase in depth and independence as students progress from the 9^{th}-grade through 12^{th}-grade. Students are selected through merit-based criteria. During the academic year, students attend class on the university campus one full day per week. Demographic data for students in the program was not released, but the large majority of students in the program are from local magnet high schools and the male to female student ratio for this group was 60% to 40%.

In this study, students spent three of their weekly on-campus days, over the course of four weeks[2], working on the C^3STEM learning units. Each class day included 6.5 hours of class time and 30 minutes for lunch. In addition to this time in class, students were assigned some of the C^3STEM units and associated problems as homework. For both in-class and homework units, students built CTSiM or SUMO models, ran simulation experiments and used the results to answer questions associated with each unit. Their simulation experiments and answers were recorded in Google Docs, and the research team had access to these documents. For the final, challenge problem unit, students also worked collaboratively in groups to build the relevant SUMO models, run simulation experiments, and document their results. Students worked on the challenge problem both in and outside class. After they were done, they were asked to create a 10-minute presentation in Google Docs, which each group then presented to the class and the researchers on the final afternoon of the study.

Throughout their in-class work with the C^3STEM units (for all three class days of the study), researchers and program staff were available to answer questions and provide support to students. Two to four of the seven researchers were present at all times and one of the honors program staff was always present. For some of the time a second staff member also attended the classroom sessions. In addition to technical support in using CTSiM and C^2SuMo, researchers and program staff also held front-of-class discussions. In these discussions, which often involved back and forth conversations with the students, researchers presented material on agent-based modeling and simulation, modeling of dynamic systems using integral representations (calculus), the principles of physics (i.e., kinematics) that governed vehicle movement, and parameters that defined driver behavior. Some of the discussion was on fundamental knowledge, and some of it directly addressed issues that a significant number of the students were having problems with. In addition, there were a number of informal one-on-one discussions with individual students and the groups, where students described specific problems they were facing. The discussion with the researchers often involved a review of their current approach to modeling, checking the model, running experiment, and the expected results from these experiments. Sometimes, researchers suggested alternative approaches, but at no time were students given the code for building their models, or the answers to questions asked. In other words, students were given a lot of opportunities to explore and experiment, and guidance was minimal – it was provided to help students make progress on their own. This open-ended constructivist approach to learning has been shown to produce learning with understanding in science domains.[22,23] Students were required to work in assigned groups of three to solve the challenge problem (see Problem unit 10 in Table 1).

On the first study day, students were introduced to the C^3STEM project, followed by a brief introduction to the concepts of computational thinking, modeling and simulation. Then, the overall study goals were discussed by the researchers, which was followed by the pre-test that took most students about 60 minutes to complete. After the pre-test, the researchers provided two hours of instruction and hands-on exploration with the CTSiM computational modeling and simulation environment. Students then worked independently for approximately an hour on the first two C^3STEM units in CTSiM. Next, researchers provided an hour of instruction and hands-on training for the C^2SuMo environment and related traffic modeling and simulation concepts associated with the SUMO simulator. The usual logistic matters like creating accounts for document sharing and collaboration were also discussed. Following this instruction, students worked independently

[2] One of the weeks included a holiday during which there was no on-campus class day.

on unit 3 in C^2SuMo. At the end of the first day, students were introduced to units 4, 5, and 6 for homework. Given that students had other overlapping honors-course homework to finish for that week, completion of the C^3STEM homework was not made compulsory, and students were told they would have time at the beginning of the next class day to work on any units they had not completed as homework.

The second study day (the following week) began with a quick review of the in-class units from the previous week and then students worked to finish the homework units (4, 5, and 6) for about an hour. The majority of the day was spent working on units 7 and 8. The last hour and a half of the day was used to discuss the CTSiM and C^2SuMo homework units for the week (Units 9 and 10) with an emphasis on group collaboration for the C^2SuMo unit. During this period, students worked in their assigned groups for the first time. They were also introduced to Google Hangouts and told they could use Hangouts for subsequent collaboration outside of class. Students were randomly assigned to eight groups of three students each, and one group of two students.

The third and final day of the study (two weeks later due to Thanksgiving holidays) began with a review of the previous day's in-class units. Students then spent approximately three hours working in their assigned groups to complete their simulation experiments for unit 10 and prepare their final presentations. The nine groups required a total of approximately two hours for presenting their final presentations, and that was followed by an hour for the post-test.

3.2 Assessing students in the C^3STEM learning environment

Assessments included the traditional pre- and post-test tests for domain knowledge and computational constructs, as well as tracking students' model building efforts, and the reports the students created in the Google Docs repository. Exit interviews were also conducted with some of the student groups on day 3 of the study. We describe our assessments in more detail in the following sections.

3.2.1 Pre/post assessments for science and Computational Thinking

We designed separate paper-based domain and computational thinking (CT) pre/post tests to assess whether students understood the basic STEM concepts and CT skills emphasized in the modeling activities and were able to apply them to solve and reason about similar problems. The domain pre/post-test assessed whether students understood the kinematics concepts important to the traffic domain, such as the relations between speed, acceleration, time, and distance traveled, as well as the relations between traffic light cycles, vehicular speeds and traffic flow rates and wait times. The test comprised 12 questions and tested students' abilities to apply speed, time, distance, and acceleration formulae to solve for a given unknown variable. In addition, students were asked to interpret speed-time graphs, calculate gap-acceptance times for making safe unprotected left turns based on oncoming vehicle speeds, and calculate flow rates and wait times at intersections based on traffic light cycles. While designing the questions on the test, we tried to closely align them with the target concepts we hoped students would learn while modeling the different traffic scenarios and experimenting with the traffic models.

The CT test comprised 7 questions that required students to predict outputs of given code snippets and develop algorithms using pre-specified primitive constructs for scenarios described in text form. The simpler questions focused on use of single CT constructs, while the more complex questions required modeling of complex scenarios with multiple agents using a variety of CT constructs. The CT constructs tested included use of nested conditionals and loops, and other scenario-specific constructs. This tested students' abilities to apply abstractions to complex situations and then model agent behaviors using required algorithm structures. For example, one question required modeling the interactions between wolf and sheep populations in a given ecosystem. The behaviors of each population were discussed, and rules governing the energy flows between the two sets of agents had to be designed using given primitives.

3.2.2 Assessing students in the CTSiM environment

We assessed the computational models students built by calculating the alignment between the blocks in students' models and the pre-specified expert model. The calculation uses the edit distance between the abstract syntax tree (AST)[24,25] representations of the two models, i.e., the student's model and the corresponding expert model. This Tree Edit Distance (TED)[26] measure defines the distance between two models as the number of operations (node renaming, insertion, and deletion) that must be performed on the AST of one model in order to convert it to the AST of the other.

The TED measure places an emphasis on how blocks are combined to produce the model. In general, there are several ways to construct correct (and behaviorally equivalent) models. Therefore, additional transformations are applied to achieve a canonical form of the models. For example, a > b and b < a produce the same true-false result, but their AST representations are different. In the C^3STEM units, the modeling language for each is sufficiently targeted to the domain that we can easily construct a canonical form of the models in the AST representation. Hence, a TED of 0 means that the model is behaviorally similar to the expert model; a decrease in TED means that an incorrect block was removed or a correct block was added to its correct location in the model; while an increase in TED could point to an incorrect block being added, a correct block being removed, a block that belongs in the model being inserted in the wrong location, or an existing block being moved from a correct to an incorrect location in the model.

Using the TED metric, we characterized how students' models evolved as they worked on a unit and over the course of the intervention. For each change a student made to his/her model (adding or deleting primitives), we calculated the distance-to-expert of the resulting intermediate model. An edit was considered *effective* if it increased the accuracy of the student model and brought it closer to the expert model (i.e., reduced the TED metric). An aggregate edit '*effectiveness*' measure was calculated to be the proportion of model edits made that were effective. The effectiveness measure was computed for each student for each of the CTSiM units.

Beyond the overall effectiveness, we tracked students' effective and ineffective edits over time. For each student, we estimated the best-fit linear trend-line over their sequence of model distances for a unit to assess their performance over time. For each unit, all students' trend-lines had negative slopes, implying that their models got closer to the expert model as time progressed. However, the consistency of model changes over time (as measured by the coefficient of determination for these linear regressions) varied among the students. We use students' final-model-TED and the *effectiveness* and *consistency* measures to study students' modeling performance over the course of the intervention.

Students also ran experiments with the models they built to answer specific questions that were posed to them. However, at this time the documents containing their solutions have not been graded. We will develop grading methods that give us a clear indication of how well students understood the concepts associated with the problem, and determine if the methods they used helped them generate the right answers. These results will be discussed in future publications.

3.2.3 Assessing students in the C^{2}SUMO environment

We also recorded students' simulation experiment runs (including their input and output parameter settings) in the C^{2}SUMO environment, along with their other activities in C^{2}SuMo. For the experiments, students varied input parameters, such as the number of vehicles, flow rates, number of lane, the type of intersections, and traffic signal logic. They captured the following output parameters from the simulation runs: vehicle trip duration, wait time at intersections, average speed, average queue length and average time spent at intersections, and overall throughput (i.e., flowrate),and used them to answer the C^{2}SUMO activity questions. We also calculated the time students spent collaborating using Google Hangouts, and analyzed the Google documents for inferences students made from their simulation experiments to answer the questions for the individual units.

Results reported in this paper correspond to solutions that were quantitative in nature. We are still developing the coding methods to analyze the answers for the other problems. For example, in Unit 6, students were asked the somewhat open-ended question "*Based on your experimental results, can you determine whether a particular traffic control mechanism performs better than others in a given scenario*? (the choices for the traffic control mechanism were: (1) unprotected left turns when there was no STOP sign for the oncoming traffic; (2) left turns at intersections with four-way stop signs, and (3) left turns at intersections with traffic lights)." The choice would depend on other environmental parameter settings that were chosen for the experiment. On the other hand, examples of problems where quantitative solutions could be derived were in Units 8 and 10, where students were asked to produce optimum traffic signal sequence timing that minimized the average waiting time of all vehicles. Students answered such questions by running a sequence of experiments, changing relevant simulation parameters in a systematic way, and tabulating the results. We evaluated students' responses by noting the smallest values of wait times observed.

The importance of collaborative learning and problem solving is understood better in the context of science and engineering practices, where complex problems are solved in teams. One of our main objectives for this study was to document the use of Google Hangouts for collaborative problem solving in the C^{2}SuMo environment. In this paper, we report a simple measure, i.e., the amount of time that students collaborated using Hangouts, and then determine if greater use of Hangouts, i.e., more collaboration resulted in better overall model building performance.

4. Results

4.1 Science and CT learning gains

Analysis of the pre-/post-tests showed that the intervention produced significant learning gains for both STEM and CT concepts (see Table 2). The corresponding effect sizes were large (> 0.8). We also analyzed students' learning gains for individual questions to study how effective the learning was for specific domain and CT concepts. On the STEM assessment questions, we found that students showed learning gains for 10 out of the 12 questions. Performing a multiple comparison test (Bonferroni's adjusted *p*-value = 0.004, *n* = 12) showed that the gains were significant ($p < 0.05$) on the two questions that required (1) calculating gap acceptance times for making safe left turns, and (2) deriving the acceleration and distance traveled from different segments of velocity-time graphs. Similarly, students also showed learning gains on most of the questions (6 out of 7) on the CT assessment, and the gains were significant (Bonferroni's adjusted *p*-value = 0.01, *n* = 7) for the transfer question, which required students to construct a multi-agent simulation model in a different domain (ecology) with a given a set of domain and CT constructs. Significant learning gains in the transfer task implies that students not only gained an understanding of the constructs, but were successful in applying them in a new problem domain.

Table 2: Pre/post learning gains for science and CT

	Pre-score mean (S.D.)	Post-score mean (S.D.)	2-tailed p-value	Effect size
Domain (*max score* =165)	41.56 (25.71)	64.10 (28.27)	< 0.01	0.83
CT (*max score* = 56)	40.39 (10.96)	45.39 (4.91)	< 0.05	0.91

4.2 Assessing students' computational models for different traffic scenarios

We calculated the tree-edit-distance (TED), effectiveness, slope, and consistency metrics described in Section 3.2.2 to study the accuracy of students' final models and how the accuracy of their models changed across each of the CTSiM units (see Table 3).

Table 3: Modeling performance across units

	Constant velocity (n=25)	Stop sign (n=25)	Left right turn (n=24)	Gap acceptance (n=24)	Intersection (n=25)	Intersection – left turn (n=21)
Mean final-model TED (S.D.)	0.26 (.44)	3.77 (2.86)	6.45 (3.18)	2.47 (2.14)	1.35 (1.08)	6.107 (2.35)
Average number of edits (S.D.)	9.6 (3.75)	128.72 (59.86)	49.52 (37.53)	44.04 (32.94)	18.28 (5.67)	75.86 (49.38)
Mean effectiveness (S.D.)	.69 (.19)	0.49 (0.10)	0.48 (0.26)	0.63 (0.36)	0.79 (0.13)	0.50 (0.08)
Mean slope (S.D.)	-0.18 (0.16)	-0.08 (0.05)	-0.18 (0.22)	-0.14 (0.17)	-0.35 (0.14)	-0.09 (0.05)
Mean consistency (S.D.)	0.53 (0.35)	0.70 (0.28)	0.56 (0.34)	0.45 (0.42)	0.85 (0.14)	0.72 (0.23)

Given the differences in the size of the expert models across units, we normalized the students' TED score at the end of each unit with respect to the size of the expert model AST for that unit. The plot of the normalized TED scores for each unit is shown in Figure 5. For example, the final TED score for the gap-acceptance unit is lower than that of the stop-sign unit (2.47 versus 3.77), but the gap acceptance expert model is smaller than that of the stop sign model (average of 8 blocks per procedure versus average of 14.75 blocks per procedure). This becomes evident in Figure 5, where the normalized TED score for the gap acceptance unit is greater than that of the stop sign unit (0.31 versus 0.26).

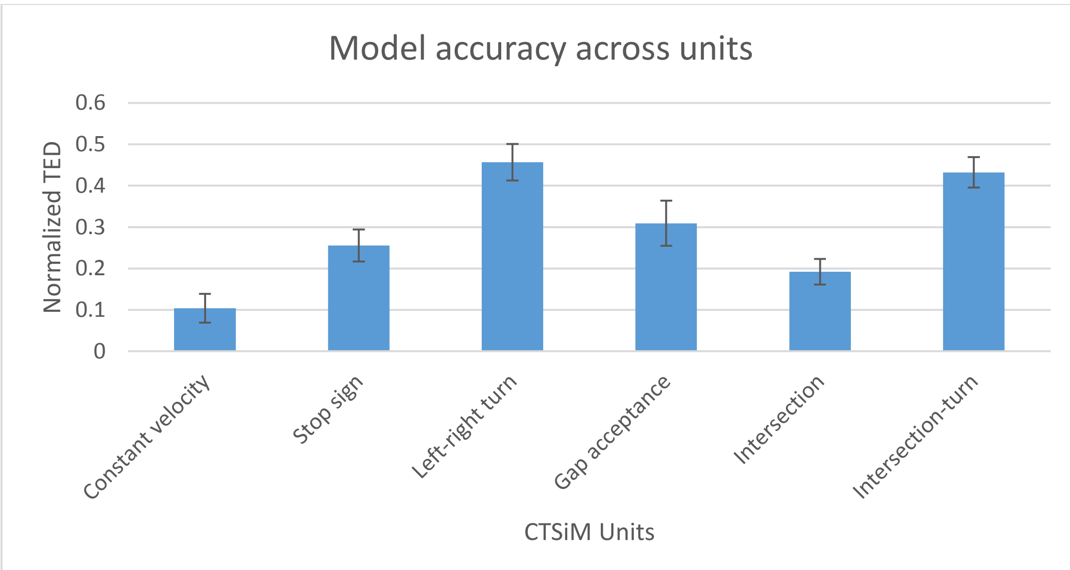

Figure 5: Normalized model distance progression over CTSiM units

Our results show that students had fairly accurate models (low normalized TED scores) in most of the units. Compared to the other units, students had more difficulty in deriving the correct model for (1) the left-right turn unit (Unit 4), and (2) the Intersection with-left-right turns unit (Unit 9). We further investigated which parts of the model had the most errors in Units 4 and 9. In Unit 4, we found a large number of students had problems correctly modeling the execute-turn procedure (18 of the 24 students who worked on this unit). To generate this behavior, students had to be cognizant of the fact that they were modeling a smooth turn with a number of discrete steps. Therefore, they had to take into account the tradeoff of using a smaller turn angle at each step to make the vehicle motion smooth and the fact that small turn angles implied a greater number of steps to complete the turn. Students' confusion on this issue led to an interesting front-of-class discussion led by one of the researchers about how simulation engines implemented on a computer were tuned to generate continuous behaviors. Unit 9 required students to model a complex set of *if-then-else* conditions to cover all possible vehicle behaviors at an intersection (start, stop, drive through, and turn left, right, or go straight). Students had to develop the logic so proper subsets of these conditions were combined correctly to capture the range of behaviors at a traffic light intersection. Some students (9 of 21 students who worked on this unit) had problems with modeling the more complex left turns where the car must first move to the left turn lane, proceed to the intersection, and then check the left turn light to determine whether to execute the left turn (or else stop and wait at the intersection). However, most students (18 of 21 students) were overwhelmed by the complex logic where multiple conditions had to be put together, and, as a result they had difficulties in constructing the correct model.

On the other hand, the Intersection unit (Unit 7) seemed to be the easiest with low final-model TED score (average normalized TED score of 0.19; 12 of the 25 students who worked on this unit had correctly or near-correct final models), and high values of effectiveness, slope, and consistency. In other words, most students did not find it difficult to replicate the expert model behavior that included how vehicles slowed down, sped up, or followed the car in front of it at a traffic intersection with traffic lights. Students had intermediate challenges in describing quantitative relations, as well as modeling complex driver behavior in Unit 9 where mechanisms of making left and right turns had to be defined explicitly. We also note that the slope of TED score over time was (on average) negative for all units, meaning students' models generally improved with time in each unit.

In previous studies using CTSiM kinematics and ecology units with middle school students who worked in a more formal classroom setting, we had noticed that students who make more consistent model edits generally show higher learning gains[27]. Hence, we wanted to check whether there were any relations between students' learning gains in this study and their final model TED scores, edit consistencies and slopes. Since there was more than one dependent variable, we ran a multi-dimensional regression analysis with students' final model TED scores, edit consistencies and slopes as regressors, and studied effects of changes in these regressor variables on students' domain and CT learning gains. However, none of the regressors predicted students' learning gains with significance for this study. It is more likely that a combination of factors - model building, the problem solving tasks associated with model building, and the discussions students had with each other and with the researchers as they worked on their individual units - may have together supported the learning of domain and CT concepts. In exit interviews, students did express that the modeling, simulation, and problem solving tasks linked to the real-world was a real eye-opener to them, and they would really like their high school teachers to discuss the relevance of the material being learned to real world problems and problem solving. Furthermore, students felt that the open-ended and exploratory nature of the tasks, and the opportunities they had to discuss concepts and how to develop the solutions to problems with each other and the researchers, played a very important role in their gaining a deep understanding of the relevant concepts. In future work, we will analyze students' approaches to developing hypotheses to solving open-ended problems, and the experimental procedures they conduct to study their hypotheses and develop solutions to the given problems so that we can further refine our units and the activities that we have developed to support student learning.

4.3 Analyzing students' collaborative problem solving skills

In this section, we present preliminary results of students' use of C^2SuMo with a focus on collaborative learning as derived from the simulation logs. When working individually (i.e., C^2SuMo units 3, 6, and 8), students ran a total of 1082 successful simulations as part of their experimental and problem solving tasks. The students spent the most time on experiments for the first C^2SuMo problem (i.e., unit 3), executing 321 simulations varying vehicle parameters and familiarizing themselves with the system. The second C^2SuMo problem (unit 6) had three parts for which 221, 136, and 124 simulations were run, respectively. For the last individual problem (unit 8), students ran a total of 280 simulations. Student generated documents of their solutions to these problems have not been analyzed, but we will do so in the near future to determine the impact of the experiments on learning.

For the challenge problem, where students worked in groups of three (one of the groups had only two members), students ran experiments by modifying the traffic light timing sequence at one intersection to determine how this affected traffic buildup at the adjacent intersection. The nine student groups executed a total of 240 simulations, consisting of 130, 51, and 59 simulations for problems 10a, 10b, and 10c, respectively. Half of the simulations (122) were performed in class during the on-campus days of the study, and the rest (118) were performed outside of class. This demonstrates that the students were able to collaborate successfully, whether they worked together in class or accessed the system remotely, while conversing with each other and sharing results using Google Hangouts and Google Docs.

Separately, we also logged the simulation results and the time students spent in collaborations using Google Hangouts. However, it is harder to get a complete picture of how students collaborated because students could have also used other modes of communication (such as cell phones, email, and texting) to share experimental results and discuss with each other. Just like in class, they may have worked together in close proximity of each other outside of class. . Also, due to a limitation in the current system, the collaboration time in Hangouts gets saved only if the user accesses it from the C^2SuMo environment. Out of the 118 simulations run outside of class, we have evidence that at least 53 were executed while the groups were collaborating using our Google Hangouts-based environment. Only two of the groups, group 2 and group 9 used the collaboration system extensively. They executed 24 of 25 and 29 of 30 total simulations, respectively, during remote collaboration sessions in the C^2SuMo environment. While other groups may have also collaborated, we do not have any direct evidence of their doing so using Hangouts.

In Problem 10a, students were solving the problem of minimizing the overall traffic waiting time by changing the traffic signal cycle times for the two adjacent intersections. Figure 6 shows the results of the experimental runs conducted by the different collaborative groups for this problem. The y-axis represents the average waiting time for all vehicles at the two intersections, and the x-axis is the experimental run number for a group. The figure shows that group 9 achieved the lowest waiting time, followed by group 5 and then group 2. Since groups 2 and 9 used the collaborative environment extensively when solving the challenge problem, this provides us with some initial evidence that the Google Hangouts based collaboration was successful in helping students work together to solve the complex optimization problem. In future work, we will study this feature in a more systematic way. Another key observation is that all groups systematically explored the space of parameters as they ran their experiments to find the best traffic light timings. Also, none of the groups stopped after they had found the first minimum, and they explored further to check if they could obtain better results. In other words, students were well aware that they were using an empirical solution method to solve a complex problem, and a lot of different experiments (i.e., search) would have to be performed before they could be convinced that they had a “good” (and possibly not the “optimal) solution.

In their final presentations, a number of the students stated that they learned two important facts from this study: (1) traveling at a steady speed was better than a lot of speeding up (and forced) slowing down; and (2) light sequences that changed color at small intervals but were proportional to the volume of traffic in either direction were better than longer sequences. The first conclusion made sense, but the second one may not be very accurate – through discussions the students realized that the simulation models they worked with were oversimplified, because reaction time and other driver behavior parameters were not taken into account. They realized if relevant delays were built into their models, they may have come up with more realistic results.

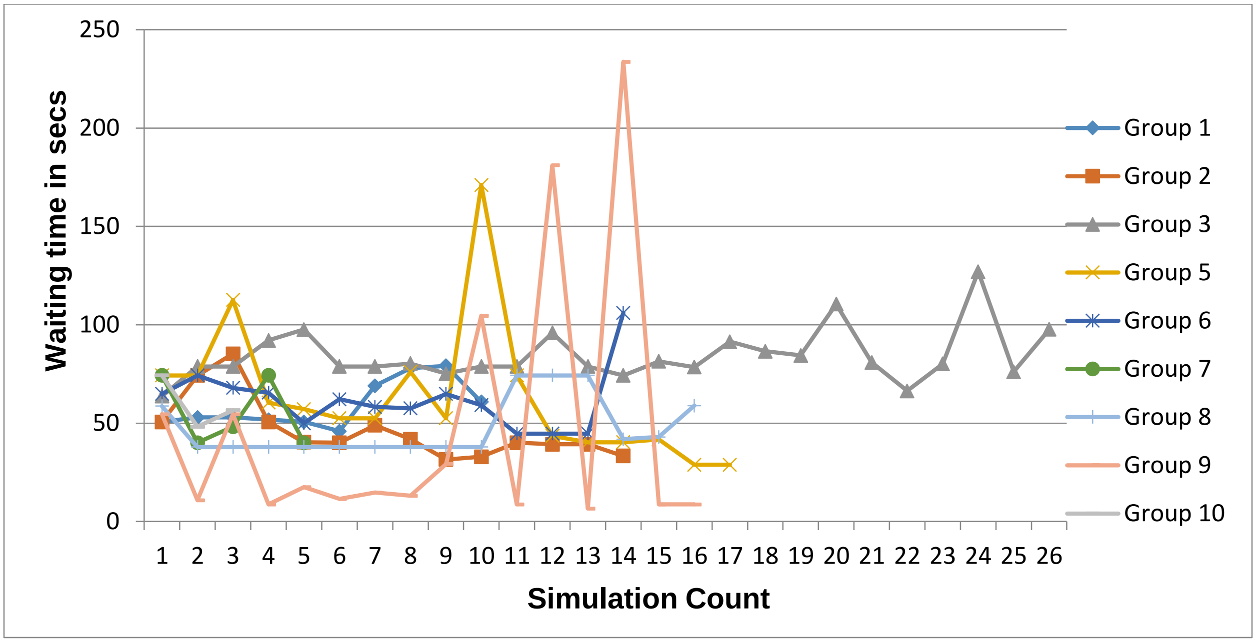

Figure 6: Waiting time by group for the collaborative optimization problem in Unit 10

5. Discussion and Conclusions

In this paper, we have presented the C^{3}STEM learning environment that harnesses the powers of computational thinking, modeling, and simulation situated in the framework of collaborative, real-world problem solving to support ubiquitous STEM learning in high school classrooms. A study conducted with 26 high school students showed that the environment helped them explore, study, work collaboratively, and learn the relations between individual vehicle properties like speed and acceleration and aggregate traffic properties like throughput, wait time, and queue length at traffic intersections. They also studied the effect of traffic light timing and coordination on the flow of traffic through multiple intersections. We have collected a lot of rich data of student work from this study. Some of it has been analyzed and presented in this study, but we are continuing to analyze the remaining data, and hope to gain a much better understanding of how to use real-world problem solving to motivate and promote deep learning of STEM and computational concepts.

Our overall conclusions from the study are that the students remained engaged and motivated throughout the intervention, they were enthusiastic, they sought new information and clarifications by asking lot of questions, and they were able to apply what they learned to solve problems. Furthermore, they collaborated extensively, spent considerable time outside class working on their modeling and problem solving assignments, and their group presentations were of very high quality. They were clearly able to define their approach to solving the challenge problem, the various observations they made while running experiments, and justify the solution they derived, and what the implications of their solution were. Students were also assessed using pre/post assessments and our results demonstrate significant learning gains in both STEM and CT as a result of the C^3STEM intervention. We also analyzed the computational models students built as part of the intervention and their collaborative problem solving activities. Students built relatively accurate models in most of the units, but their modeling performance was not a strong predictor of their learning. We identified other possible factors that may have also influenced learning. In order to improve our understanding of the mechanisms by which the C^3STEM intervention produces strong learning gains, we will continue to analyze students' efforts further by studying how they experiment with their models and their submitted 'google doc' documents.

Acknowledgements
This research was supported by NSF CNS EAGER grant #1257955

References

1. The Next Generation Science Standards http://www.nextgenscience.org/sites/ngss/files/NGSS%20Combined%20Topics%2011.8.13.pdf

2. Novick, L. R., & Bassok, M. (2005). The Cambridge handbook of thinking and reasoning. Problem Solving, 321-349.

3. Bransford, J. D., Sherwood, R. D., Hasselbring, T. S., Kinzer, C. K., & Williams, S. M. (1990). Anchored instruction: Why we need it and how technology can help. Cognition, education, and multimedia: Exploring ideas in high technology, 115-141.

4. Cognition and Technology Group at Vanderbilt. (1997). The Jasper project: Lessons in curriculum, instruction, assessment, and professional development. Mahwah, NJ: Erlbaum.

5. Barrows, H. S. (2000). Problem-based learning applied to medical education. Southern Illinois University School of Medicine.

6. Hmelo-Silver, C. E. (2004). Problem-based learning: What and how do students learn?. Educational psychology review, 16(3), 235-266.

7. Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. Educational researcher, 18(1), 32-42.

8. Basu, S., Dickes, A., Kinnebrew, J.S., Sengupta, P., & Biswas, G. (2013). CTSiM: A Computational Thinking Environment for Learning Science through Simulation and Modeling. In Proceedings of the 5th International Conference on Computer Supported Education (pp. 369-378). Aachen, Germany.

9. Dukeman, A., Shekhar, S., Caglar, F., Gokhale, A., Biswas, G., & Kinnebrew, J.S. (2014). Analyzing Students' Computational Models as they Learn in STEM Disciplines. In the 121st American Society for Engineering Education Annual Conference & Exposition. Indianapolis, IN.

10. Shekhar, S., Caglar, F., Dukeman, A., Hou, L., Gokhale, A., Kinnebrew, J.S., & Biswas, G. (2014). A Collaborative K-12 STEM Education Framework Using Traffic Flow as a Real-world Challenge Problem. In the 121st American Society for Engineering Education Annual Conference & Exposition. Indianapolis, IN.

11. Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

12. Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), 3717-3725.

13. Basu, S., Kinnebrew, J., Dickes, A., Farris, A.V., Sengupta, P., Winger, J., & Biswas, G. (2012). A Science Learning Environment using a Computational Thinking Approach. In Proceedings of the 20th International Conference on Computers in Education (pp. 722-729). Singapore.

14. Sengupta, P., Kinnebrew, J.S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating Computational Thinking with K-12 Science Education Using Agent-based Computation: A Theoretical Framework. Education and Information Technologies, 18(2), 351-380.

15. Wilensky, U. (1999). NetLogo. Center for Connected Learning and Computer-Based Modelling (http://ccl.northwestern.edu/netlogo). Northwestern University, Evanston, IL.

16. Clark, D. B., & Sengupta, P. (2013). Argumentation and Modeling: Integrating the Products and Practices of. Approaches and Strategies in Next Generation Science Learning, 85.

17. Chodorow, K. (2013). *MongoDB: the definitive guide*. " O'Reilly Media, Inc."

18. www.openstreetmap.org

19. http://sumo.dlr.de/wiki/NETCONVERT

20. https://www.openstack.org/

21. http://www.google.com/+/learnmore/hangouts/

22. Herrington, J., Reeves, T. C., & Oliver, R. (2014). Authentic learning environments (pp. 401-412). Springer New York.

23. Jonassen, D. H. (1999). Designing constructivist learning environments. Instructional design theories and models: A new paradigm of instructional theory, 2, 215-239.

24. Saikkonen, R., Malmi, L., & Korhonen, A. (2001, June). Fully automatic assessment of programming exercises. In ACM Sigcse Bulletin (Vol. 33, No. 3, pp. 133-136). ACM.

25. Yang, W. (1991). Identifying syntactic differences between two programs. Software: Practice and Experience, 21(7), 739-755.

26. Chris Piech, Mehran Sahami, Daphne Koller, Steve Cooper, and Paulo Blikstein. Modeling how students learn to program. In Proceedings of the 43rd ACM technical symposium on Computer Science Education, pages 153–160. ACM, 2012.

27. Basu, S., Kinnebrew, J., & Biswas, G. (2014). Assessing student performance in a computational-thinking based science learning environment. In *Proceedings of the 12th International Conference on Intelligent Tutoring Systems* (pp. 476-481). Honolulu, HI, USA: Springer International Publishing.