CoSolve: A System for Engaging Users in Computer-Supported Collaborative Problem Solving

Sandra B. Fan, Tyler Robison, Steven L. Tanimoto
Department of Computer Science & Engineering
University of Washington
Seattle, Washington, USA
{sbfan,trobison,tanimoto}@cs.uw.edu

***Abstract*—Recently there has been a trend toward online collaborative problem-solving. However, many systems either lack enough structure for participants to know where they can contribute or are too restrictive to allow collaborative solving. In this paper, we present our research prototype, CoSolve, a website that helps users cooperatively solve problems in a novel manner: solving sessions are represented visually as state-space search trees which solvers collaboratively generate, traverse and interact with online. We describe the problem-posing and problem-solving processes in our system, and present the affordances we designed for encouraging self-reflection and collaboration in the problem-solving process. Finally, we present observations from a user study conducted with teams of solvers who used CoSolve to solve a city-building problem. Users found CoSolve easy to use and helpful in problem solving. In addition, the study provides evidence that the state-space-search organization of problem-solving activity can serve effectively as the framework for human interaction in a computer-supported collaborative problem-solving system.**

Keywords-computer-supported collaborative work, problem-solving, collaboration

I. INTRODUCTION

The rise of social technologies for the web has led to the creation of new tools for collaboration. The idea of crowdsourcing problem-solving has gained momentum in the past decade, both commercially, e.g. Amazon's Mechanical Turk (MTurk) [1] and Innocentive [2], and in academia, e.g. Polymath Project [3]. MTurk allows users to hire remote workers on the web for small jobs. This structured approach has the advantage of providing huge numbers of workers with well-defined tasks; however more open-ended, creative, problem-solving tasks are less appropriate in this system. Additionally, many crowdsourcing systems are not directly collaborative; solvers either have no contact with one another, as in MTurk, or solvers compete with each other, as in Innocentive, rather than work together.

At the other extreme are loosely organized, very creative, problem-solving projects such as those found in Substepr [4], or in Tim Gowers' successful Polymath project [5] [6], which has gathered mathematicians together to cooperatively solve research problems in their field through blog entries, threaded online comments, and wiki pages. However, new participants have to wade through a lot of past discussion, and it may be difficult for them to understand what state the project is in, and where one can best contribute.

Is there a middle ground where collaborative problem-solving can occur, while allowing a structured process and manageable, defined tasks? To explore this idea, we developed CoSolve, a web-based collaborative problem-solving environment, which uses state-space search to provide structure while offering tools to enhance collaboration. CoSolve was briefly described in [7] and [8], but additional collaboration features have been added, and this paper covers the first formal user study of the system.

CoSolve draws on Herbert Simon's idea of design as state-space search [9] by visualizing problem-solving with state-space search trees that problem solvers collaboratively generate and explore, eventually finding possible solution states to their problem as they construct branches of the tree. While there has been much past work on the theory of problem-solving (for example, [10]) and on problem-solving environments, such as work by Vass [11], Berry et al. [12], Brodie [13], Jonassen [13], Dewan [14], CoSolve is novel in its use of the state-space-search paradigm to structure the collaborative problem solving process. CoSolve allows solvers to see an overview of the solutions others have tried, and to work towards solutions collaboratively. The tree-based interface provides an easy way to delve down and work on a specific solution in a manner that is easy for others to follow. Additionally, CoSolve's interface allows the possibility of humans and computer agents solving problems together, because the tree-based representation can be both visualized for human understanding and processed by a computer agent.

To test this notion of computationally visualizing state-space search, we created a prototype called TStar (described in [15], [16], [17]), a desktop program for visualizing a state-space search as a tree that users could generate. However, TStar was not easily usable for synchronous groupwork. As a result, CoSolve was then developed as a web application, and problems can be both posed and solved through its web interface.

Our system has been informed by much previous work in the field of user interfaces for collaboration. Dourish and

Support for this research by the National Science Foundation under grant 0613550, and by the Washington NASA Space Grant Consortium's Graduate Fellowship program, is gratefully acknowledged.

978-1-4673-0853-3/12/$31.00 ©2012 IEEE

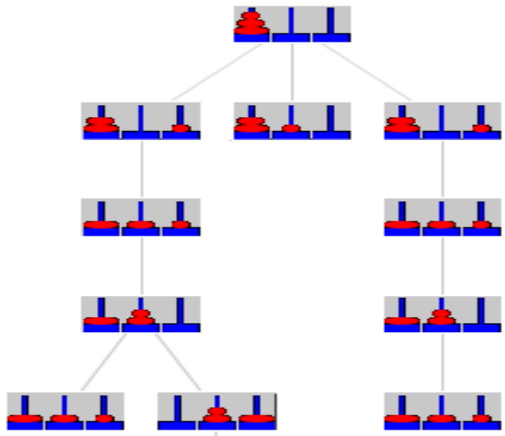

Figure 1. An example of a CoSolve tree, from a solving session for the Towers of Hanoi problem, with the problem's initial state shown at top.

Bellotti's work on CSCW systems identified the need for group awareness [18] which informed our evolution from TStar to CoSolve. CoSolve users engage in *mixed-focus collaboration*, as discussed in [19] and [21], when they move between working on their own branches of the tree to evaluating their group's work on the entire tree; Section IV discusses their mixed-focus usage in more detail. Park et al. [23] and Heldal et al. [20] stress the importance of supporting an individual's ability to work independently when engaged in collaboration. To keep users' tree views in sync without disturbing each individual's work, we queued up their teammates' changes, similar to work done by Suthers et al. [22], but we empowered users by displaying a "Refresh" indicator (discussed in Section IV) that showed how many updates from teammates were in the queue, and allowing the user to click the indicator to update his or her view to include new changes when desired. There also exists previous work involving collaborative graph representations (e.g. [19], [22]). However, no system to date has explored using a state-space-search model for collaborative problem-solving visualization.

CoSolve's key premise is that the state-space-search problem-solving methodology can serve as the framework for human interaction within a collaborative problem-solving system. The main question then is whether human users are able to understand and successfully use state-space-search for problem solving or, on the contrary, they find such a representation too confusing or cumbersome to use. Secondly, how could such a framework aid solvers' understanding of their own collaborative problem-solving processes, and as a result, improve their collaborative efforts?

To answer these research questions, we conducted a user study with small teams of users collaborating to solve a city-simulation problem we formulated called CitySim. To address the issue of users' understanding of their collaborative process, we also tested the CoSolve Consultant, a specific set of tools we built within CoSolve for enhancing group awareness and metacognition during the collaborative problem-solving process. In the following sections, we describe CoSolve's problem-solving process and user interface. Then, we briefly explain CoSolve's problem-posing process and provide a short technical description of the CoSolve system. Finally, we discuss our user study and its results, and end with directions for future work in tree-based problem-solving interfaces.

II. CoSolve Description

In CoSolve, a user can perform the role of a problem *poser*, a problem *solver*, or both. A problem poser specifies problems by creating problem templates, and a problem solver initiates and participates in CoSolve solving sessions for a particular problem template. We first examine the problem-solving process. Then we describe the CoSolve Consultant, a tool which provides metacognitive prompts to the solvers in the form of additional views and the presentation of metrics that allow users to understand the dynamics of their problem-solving process. Finally we briefly discuss the problem-posing process.

A. Problem Solving in CoSolve

In a solving session, a team of solvers generates a state-space search tree that explores the solution space for a problem, typically with the goal of finding a state in the tree that represents an acceptable solution. Figure 1 shows an example of a solving session tree for the Towers of Hanoi problem. Towers of Hanoi is a simple puzzle with pegs and differently-sized disks. The goal is to move all the disks on the left peg onto the far right peg, moving only one disk onto a peg at a time, and never placing a larger disk on top of a smaller disk.

To solve this problem in CoSolve, a first member of a team of solvers uses the web interface to initialize a new solving session from the Towers of Hanoi problem template in CoSolve. The team is then presented with a single node, the root node, at the top of the tree. This node represents the initial state of the puzzle--all the disks on the first peg. Then any solver on the team clicks on the node to select and apply an operator to transform this state into another. For example, the solver might select the operator "Move smallest disk to far right peg." When the solver does so, CoSolve generates the next state, showing the disk moved to the right, and displays it to the user as a child of the root node, with a line connecting the two. From there, solvers can either continue to create children of the most recently created nodes, or they can backtrack to create new branches. Figure 1 shows several nodes that have been created by users, with multiple branches. Solvers can view all branches of nodes and can apply operators to nodes they created or nodes other solvers have created. The goal is to eventually find a state or path to a state that satisfies the criteria to solve the problem.

CoSolve can be used to solve a variety of different classes of problems. Obviously, it can be used for any well-defined problems with explicit, discrete moves that are applied in sequence, and that have clearly specified winning criteria, like chess or Towers of Hanoi. When we used CoSolve in class projects for undergraduate computer science courses, students have created problem templates for this class of problems, e.g., problem templates for sudoku, checkers, poker, stratego and other games, or puzzles like pentominoes or the 15-puzzle. Students have also created tools such as a class scheduler or transistor schematics diagramming tool. Other such templates created by undergraduate researchers in our lab have included

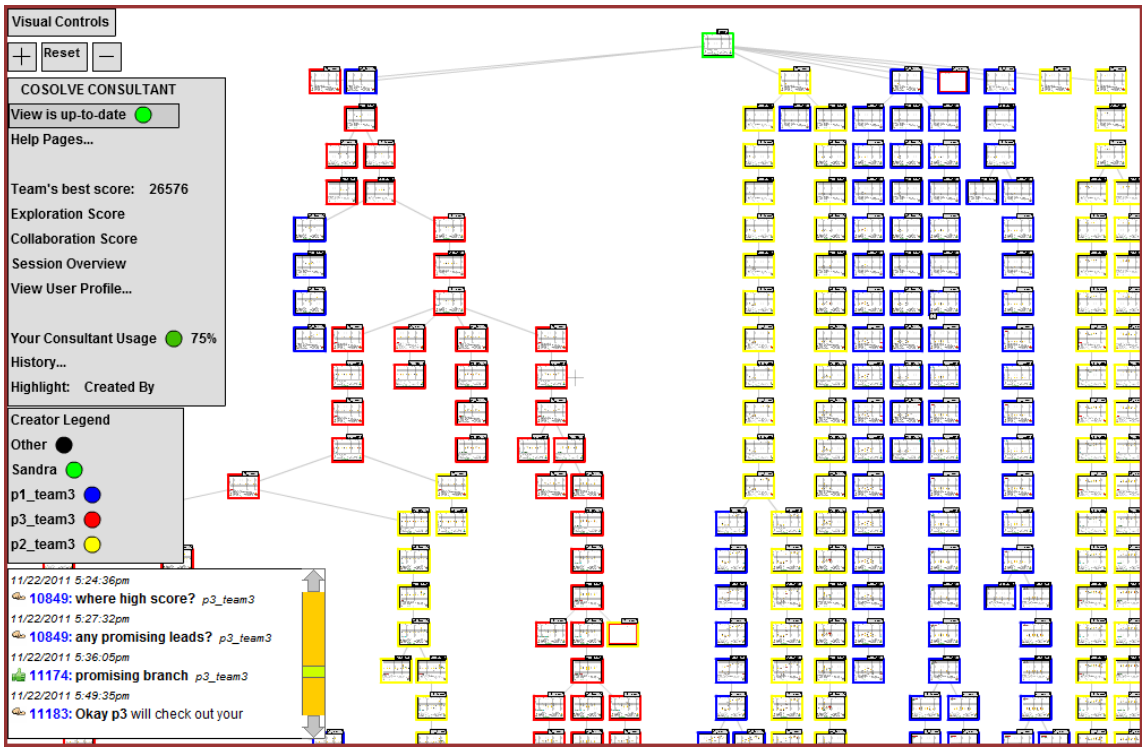

Figure 3. A CoSolve solving session, with the Full CoSolve Consultant. Pan and zoom controls are in the upper left, the Annotations List is in the bottom left.

minesweeper, a simple fantasy role-playing game, and two ecological/environmental simulation problems.

We have also explored CoSolve problem templates in a mathematics education context. A plane geometry construction template allows students to apply operators to draw a circle, draw a line, etc. to solve geometry problems. Another template allows students to visually explore functional composition. When used for education, CoSolve allows a teacher to see all the paths that students have explored and inspect their work at any step, allowing discovery of misconceptions and assessment of each student's contribution.

CoSolve can also be used in a very different manner: for creative problems that require subjective, human evaluation, such as design problems. For example, if a team is designing a playground, CoSolve states can be used to represent alternative layouts of the playground, and examples of problem operators might be "place a jungle gym in the northwest corner of the playground" and "use sand under the swing set." Another example is the Color Swatch problem template. A website designer and his client could use CoSolve to explore color palette possibilities. Each state is a palette of colors, and solvers can use operators to change RGB and HSV values of individual colors, or the whole set (e.g. "select complementary colors"). Users can explore, and the entire history of their exploration is saved. Here, there are no clearly specified goal criteria. Instead, CoSolve becomes a tool to aid discussion, brainstorming and consensus. We have also used CoSolve to aid in designing educational card games, as described in [7]. Each state is a game description, and designers apply operators to change game rules, for example, to change the number of cards allowed in a starting hand. Used in this manner, CoSolve becomes a way for human solvers to explore the space of possibilities in a visual way, and it provides a record that facilitates revisiting the many different ideas they may have developed.

Figure 2 depicts a solving session for the CitySim problem (described later in Section III.A). The tree is displayed in the main area of the viewport, and a variety of user controls are available on the left. In the upper left corner are controls for zooming, switching to different tree layouts, and switching

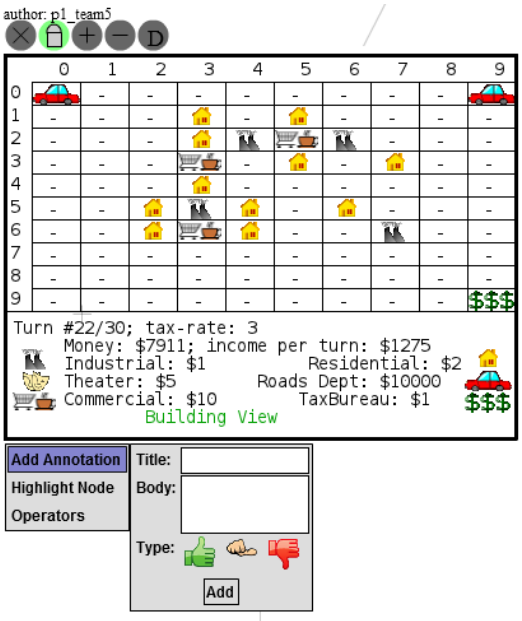

Figure 2. Close-up of a node in a solving-session tree for the CitySim problem. The node operations menu below the state image, and the buttons and information above, are hidden by default and displayed on mouse-over.

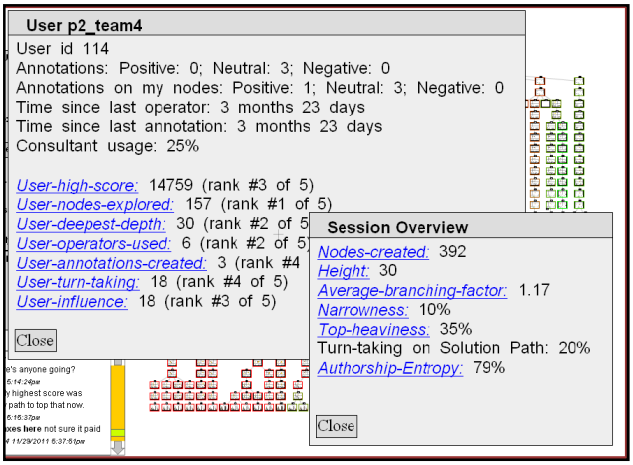

Figure 4. This figure shows some of the statistics and other information made available to solvers via the CoSolve Consultant.

between different *state views* (e.g. perhaps the solver wants to view a pie chart of the costs of playground equipment, or maybe a birds-eye view instead, etc.).

Figure 3 shows a close-up of a single node in a CitySim solving session tree. Underneath the node representation is a menu of node operations. Solvers select an operator to apply to the state, highlight the node for reference, or textually annotate the node as a form of communication with other solvers. There are three types of annotations: "positive" (thumbs-up), "neutral" (thumbs-sideways), and "negative" (thumbs-down). Solvers can use these to indicate whether they believe the node to be on the path to a good solution or not. All annotations created anywhere in the tree are immediately displayed in the Annotations List, a box in the lower-left corner of the user interface (Figure 2). Clicking on an annotation in the Annotations List will pan and zoom the user's tree view to display the associated node.

B. The CoSolve Consultant

To help users understand their own collaborative problem-solving process, and thereby possibly collaborate better, we developed the CoSolve Consultant for solvers to use within CoSolve while solving. We created this after the initial development of CoSolve. The Consultant interface can be seen on the left of the image in Figure 2, between the Visual Controls and the Annotations List. Some of the more important features include:

Highlighting. One of the most important features of CoSolve is being able to zoom out and view the entire solving session at once and thus reason about the session's history and what can be done next. In the early versions, it was difficult to show individual node details, such as node creator, and the presence of annotations, at a high-level view. We addressed this issue by allowing users to select automatic color highlighting of nodes. For example, highlighting by creator sets the highlight surrounding a node to be a different color based on who created that node. Solution-path highlighting applies one color to all the nodes in the tree that are along the path to the best solution found so far, and another color to the rest. Other types of highlighting include highlighting by number of annotations: the more annotations a node has, the darker the color. Other metrics that are highlighted in this manner are creation time (recency), node score, and annotation type balance (lighter color if node has more positive than negative annotations). These types of views encourage users to take a high-level view of their problem-solving process.

Session and solver statistics. In order to keep solvers informed about the progress of their team and contributions of their teammates, the Consultant offers a number of metrics that provide statistics about the work done so far. These include simple counts such as the total number of nodes and annotations, as well as richer measures that describe the shape of the tree, and the contributions of individual solvers.

High-level guidance. To most new solvers, CoSolve presents a new approach to problem-solving. As such, solvers are often unsure of how to proceed, and are unaware of high-level strategies that may make their task easier. These can, of course, be acquired through experience, but to speed up the process, the Consultant includes metrics intended to get solvers to think about the possibilities and implications of their actions, and it places judgments on some in order to hint at 'good' solving behavior. For instance, CitySim allows solvers to place six different types of buildings, and the profile computed for each solver displays the number that solver has tried, and is designed to encourage experimentation in this area.

Solvers access the last two features, statistics and high-level guidance, by clicking on the corresponding menu items in the Consultant interface. This brings up information windows as seen in Figure 4, and these information windows have links to additional information.

C. Problem Posing in CoSolve

To use CoSolve for problem solving, a description of the problem must first be programmatically represented within CoSolve as a *problem template*. This is done through a web form where problem posers enter scaffolded Python code fragments. CoSolve combines the fragments and applies them as needed during the subsequent solving activities.

A poser must specify two parts of a problem: (1) the *state* representation, or structure, of a problem, and (2) a set of *operators* for transforming each state into a resulting new child state. To specify the state structure to be used in a problem, posers enter key-value pairs into a special Python dictionary state variable that CoSolve provides. The keys represent a state's fields. The keys for Towers of Hanoi might be `peg1`, `peg2`, and `peg3`, and the values would be which disks are on that peg; if our disks are numbered 1, 2, and 3, then the initial state for the Towers of Hanoi problem might be: `{"peg1":[1,2,3], "peg2":[], " peg3":[]}`.

Optionally, when specifying the state variable, the poser may also provide a visualization method for the problem's states using scaffolded Python code; otherwise, a default representation outputting the key-value pairs of a state will be displayed to the user.

Next, to create operators, posers specify two things: the *precondition* code and the *state transformation* code. The state transformation code is Python code that alters the values in the current state. For instance, if the solver applies the "move

TABLE 1. TEAM RESULTS

	Minimal Consultant (MC)			Full Consultant (FC)		
	Team 1	***Team 3***	***Team 5***	***Team 2***	***Team 4***	***Team 6***
CitySim score	33087	26576	26865	31314	14848	20352
Number of Nodes in Tree	178	399	320	264	392	101
Total number of Annotations	38	25	5	21	22	55
- Positive Annotations	17	18	4	7	7	5
- Neutral Annotations	20	7	0	12	15	48
- Negative Annotations	1	0	1	2	0	2

smallest disk to far right peg" operator on the initial state, the Python code that the poser wrote for this operator would change the values of the first-peg and third-peg keys appropriately, and the resulting state would be represented as: `{" peg1":[2,3], "peg2":[], "peg3":[1]}`. The precondition code checks whether the operator can validly be applied to a state; for instance, if a solver tries to place a larger disk on top of a smaller disk, this would cause a precondition to fail and the solver will be unable to execute that operation.

In part because problem posing does involve some minimal coding, we decided to separate it from problem solving, so that it would be easier for non-programmers to solve problems with CoSolve. At the same time, posers with different levels of technical background have been able to create problem templates. For example, the CoSolve posing interface has been successfully used in undergraduate classes, both for computer science majors and non-majors, to create a range of problem template projects, as mentioned in Section II.A.

D. Implementation

CoSolve is a website built on top of Drupal (a PHP/SQL content management system) [24], and therefore uses many of its existing and add-on functionalities, such as user accounts and profiles, user groups, and so on. To implement CoSolve's backend, we wrote Drupal modules to create the problem-posing interface, and to generate and store solving-session trees and Consultant information. These are all independent of the solving session interface.

There are a number of alternative user interfaces for interacting with a solving-session tree. We have developed an HTML/Ajax interface and a Flash interface for the CoSolve website; there is also a separate Java interface. These all interact with CoSolve through a web service API. In this paper, we focus on the Flash interface, which is shown in all our screenshots.

III. USER STUDY

A. CitySim Problem Description

As mentioned in Section II.A, for the purposes of this user study, we created a problem template entitled *CitySim*, a turn-based city-simulation, for our subjects to solve. It was inspired by Maxis's SimCity game [25],[26]. A close-up of a CitySim state can be seen in Figure 3. A CitySim state consists of a 10x10 grid of a "city," upon which different types of buildings can be placed. Each building costs the city money to place, and each placed building has a different type of effect on its cell and the cells around it; some buildings provide employment, residential buildings provide population, other buildings provide or reduce happiness, which affects employment, some buildings increase taxes which generates income for the city. The goal of CitySim is to find a way to place 30 buildings in the grid such that the city earns the most money; the money value at a state is the "score" for that state, and the final, overall score for a solving session is the highest score achieved over the entire session.

B. Study Design

Eighteen subjects (11 male, 7 female, ranging from 18 to 31 years of age) were divided into six teams of three subjects each. They were recruited via the Internet and with flyers on a university campus, and by word of mouth, and were compensated with $20 gift cards. One team at a time was scheduled to come into the lab for the session together, with each subject seated at a separate computer.

After filling out consent forms, pretests on their knowledge of graphs and trees, and background questionnaires on their experience with collaboration tools, approximately 25 minutes were spent in a tutorial on how to use CoSolve, explaining CitySim rules, and on sample tasks in the problem. Then the participants began their task. They were told that they needed to achieve the highest overall score among the teams in the study, and that the winning team would receive an additional $20 gift card per person.

The task was divided into three 25-minute phases with a five minute break in between each, for a total of 75 minutes on the task. During all three phases, a team worked using the same solving session tree. Due to a scheduling issue, Team 1 only worked for two 25-minute phases, but we have included them in our data. While working on the task, subjects were asked to communicate with each other only through CoSolve, not verbally out loud, despite being co-located. To test whether the CoSolve Consultant was helpful, three teams used the Full Consultant (FC) version of the interface [Figure 2], whereas the other three teams saw only a stripped-down interface called the Minimal Consultant (MC), which did not have any of the exploration score, collaboration score, session overview, or user profile menu items, and only the default highlighting option (highlight tree by creator) was available to them.

After the activity, we conducted one-on-one post-activity interviews with each subject, looking over his or her solving session tree and asking them to describe their problem-solving processes. They also filled out a post-test and a wrap-up questionnaire on their impressions of CoSolve and their attitudes toward collaboration after having done their activity.

TABLE 3. SUBJECTS' POST-ACTIVITY QUESTIONNAIRE RESPONSES, AS PERCENTAGES OF TOTAL NUMBER OF RESPONSES (N=18)

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Q1. The tree visualization of the problem-solving space was confusing	33.3%	55.6%	11.1%	0.0%	0.0%
Q2. The tree visualization of the problem-solving space was helpful	0.0%	11.1%	11.1%	50.0%	27.8%
Q3. The tree visualization of the problem-solving space was difficult to navigate	0.0%	11.1%	38.9%	44.4%	5.6%
Q4. The tree visualization of the problem-solving space helped encourage collaboration	0.0%	11.1%	44.4%	44.4%	0.0%
Q5. Regarding navigation of the tree, I was often lost or disoriented	5.6%	38.9%	27.8%	22.2%	5.6%
Q6. The CoSolve Interface was difficult to use	11.1%	44.4%	16.7%	22.2%	5.6%
Q7. The CoSolve Interface was helpful in playing the game	0.0%	5.6%	11.1%	77.8%	5.6%
Q8. The CoSolve Interface was easy to learn	0.0%	0.0%	16.7%	77.8%	5.6%
Q9. The CoSolve interface made collaboration more difficult	16.7%	44.4%	33.3%	5.6%	0.0%
Q10. The CoSolve interface was helpful in understanding my team's problem-solving process	0.0%	5.6%	38.9%	50.0%	5.6%
Q11. I was always well aware of what my teammates had done	5.6%	61.1%	22.2%	11.1%	0.0%
Q12. It was easy to collaborate with my teammates.	0.0%	50.0%	33.3%	16.7%	0.0%
Q13. I was aware of what my team-mates were doing most of the time.	5.6%	50.0%	22.2%	11.1%	11.1%

C. Study Results

shows each team's CitySim scores, number of nodes created, and number of annotations created, by type. The teams did well in terms of score; when we were developing the CitySim template, we did not achieve more than 30,000 points in a solving session, but two teams were able exceed that score. Also of note is that the teams varied greatly in terms of numbers of nodes created, with Teams 3 and 4 creating almost 400 nodes, while Team 6 only created 101 nodes, and Team 1, who had less time, created 178.

There was no significant difference in score and solution outcomes between the FC and MC groups, though there were differences in the collaborative process. FC users tended to create more neutral annotations than the MC users; on average, 76.5% of a FC user's annotations were neutral compared to 26.8% in the MC condition (two-tailed t-test, p=0.00041, significant). At the same time, the FC users had a lower percentage of positive annotations (19.7% versus 60.3%, p=0.00251, significant). There was no significant difference in negative annotations; both groups created very few of these.

We found that FC users did collaborate more actively with each other in creating a solution than MC users. To quantitatively capture the nature of each team's collaboration, we calculated metrics such as turn-taking and solution path inequality. Turn-taking is defined as the percentage of nodes a user created that were the children of nodes created by other teammates, rather than by the user himself. FC condition users had a higher rate of turn-taking than MC condition users (14.6% vs 6.0%, p=0.03, significant).

Solution path inequality (SPI) measures each team's users' contributions on the solution path from the root of the tree to the leaf node containing the final highest-scoring state in the tree. Since each solution path contains 30 nodes, one per CitySim building, a perfectly equal path would have ten nodes from each of the three teammates, hence SPI is calculated

TABLE 2. SUBJECTS' LIKERT-SCALE RESPONSES REGARDING THE COSOLVE CONSULTANT.

	Control (Minimal Consultant, n=9)					Experimental (Full Consultant, n=9)				
Question	***Strongly Disagree***	***Disagree***	***Neutral***	***Agree***	***Strongly Agree***	***Strongly Disagree***	***Disagree***	***Neutral***	***Agree***	***Strongly Agree***
Q1. The CoSolve Consultant was difficult to use	22.2%	44.4%	22.2%	11.1%	0.0%	22.2%	44.4%	33.3%	0.0%	0.0%
Q2. The CoSolve Consultant was helpful in playing the game	33.3%	11.1%	44.4%	11.1%	0.0%	0.0%	0.0%	22.2%	66.7%	11.1%
Q3. The CoSolve Consultant was difficult to learn	22.2%	44.4%	33.3%	0.0%	0.0%	22.2%	66.7%	11.1%	0.0%	0.0%
Q4. The CoSolve Consultant was helpful in understanding my team's problem-solving process	22.2%	33.3%	22.2%	22.2%	0.0%	11.1%	33.3%	22.2%	33.3%	0.0%
Q5. The CoSolve Consultant helped my team collaborate	11.1%	33.3%	44.4%	11.1%	0.0%	0.0%	33.3%	33.3%	33.3%	0.0%
Q6. The highlighting features of the CoSolve Consultant were not useful	44.4%	33.3%	22.2%	0.0%	0.0%	66.7%	11.1%	22.2%	0.0%	0.0%

Half of the teams were assigned to use the Full Consultant, and the other half to use the Minimal Consultant (as a control) but they all answered the same set of questions regarding the Consultant.

as: $\sum_{i=1}^{n} |10 - count_{user_i}|$, where n is the number of users on a team, and $count_{user_i}$ is the number of nodes $user_i$ created on the solution path. SPI=0 if all users contribute ten nodes. A completely unequal path would have all 30 nodes from one user, and 0 from each of the other two, and which gives the max SPI value of 40. We found that the FC group had less inequality than the MC group (SPI=21.4/40 versus SPI=34/40, p=0.04, significant)

Qualitatively, we found that the FC groups tended to work together more at the beginning than the MC group, though they quickly started splitting up. Figure 2 shows an example of this. On the left side of the tree, where the users began, we see more branching and turn-taking, but as time went on, on the right side we see long, straight paths consisting of mostly one user each.

Subjects were given a 12-question pretest before the tutorial began to determine their familiarity with tree concepts, and then given a post-test after the activity. The tests had a multiple choice section testing graph-related vocabulary, and also had an abstract illustration of a tree and questions like "Find the root node of this tree" or "How many children does this node have?" etc. In the pre-test, subjects correctly answered a mean of 7.83 questions out of 12 total; the median score was 8. In the post-tests, we observed an increase in correct answers: 10.75 mean, 12 median. (Note: this data is for the last four teams, i.e. 12 subjects. Data from the first two teams, 6 subjects total, are not included due to changes in the tests). The FC subjects (n=6) showed a 3.67 increase in their mean scores compared to a 2.17 increase in mean scores for the MC (n=6), though the median score increase in both conditions is the same for both: a 3-question improvement.

In the post-activity Likert-scale questionnaire, subjects were asked to evaluate their usage of CoSolve's features. Table 3 shows some of these results (this data includes all 18 subjects). As shown in the table, none of the subjects felt the tree visualization interface was confusing (Q1), and 77.8% agreed or strongly agreed that the visualization was helpful (Q2). Our interviews with most of the subjects reflected this as well, e.g. one user said "Definitely the [tree] layout...that was great...if I had a couple good moves, and didn't know which one was good, I'd do all of them, and see the actual numbers instead of doing it in my head, and that was great, because otherwise I'd have to start over."

Regarding the usability of the entire CoSolve user interface itself, e.g. annotations, UI for applying operators, etc. (as opposed to just the tree visualization of a solving session), 83.4% of subjects agreed that the CoSolve interface was "easy to learn" (Q8), and the same percentage agreed that CoSolve was "helpful in playing the game" (Q7). Only 5.6% thought that CoSolve made collaboration more difficult (Q9).

However, subjects had trouble with awareness of their teammates' activity, with only 11.1% feeling they were "always well aware" (Q11) and 22.2% "aware...most of the time" (Q13) of what their teammates were doing, and although they did not think that CoSolve made collaboration more difficult (Q9), at the same time, 50% disagreed that it was "easy to collaborate with my teammates."

Table 2 shows subjects' responses to questions regarding the CoSolve Consultant specifically. The responses shown are divided between the subjects whose teams used the Full Consultant (FC), and those who saw the Minimal Consultant (MC). None of the FC subjects thought the Consultant was difficult to use, and only one MC subject thought so (Q1). 77.8% of the FC subjects agreed or strongly agreed that the Consultant was helpful, versus only 11.1% of the MC subjects (Q2). No one thought the Consultant was difficult to learn (Q3), and more FC subjects thought the Consultant was helpful for teamwork than the MC subjects (Q4, Q5).

IV. Discussion

Overall, our results show that subjects were able to effectively use CoSolve to solve the CitySim problem, although teams' performances varied greatly. Subjects found the tree-based solving interface helpful, and were able to learn about tree structures by using CoSolve. We can also see that the Full Consultant interface had a positive impact on users' attitudes, though not their team's actual solution scores. Users who had access to the Consultant were more effective collaborators: in those teams, each user contributed more equally to the solution than in teams where the users did not have access to it.

In our interviews, many subjects told us that the tree visualization was useful for collaborating because they could see the sequence of steps others took and learn from them. Very often, a subject would be working on his own branch, and notice that his nodes did not have the high score. So he would review a teammate's best branch to discern why she was able to get a higher score, and then apply the same technique to his own branches. CoSolve's ability to show everyone's work at once enabled solvers to learn, which is important in cooperative learning contexts. Another common occurrence we observed is when one solver would find a good technique or achieve a high score, and she would annotate that node with a thumbs-up "positive" annotation. Others would see that score, and then build branches from that node, perhaps applying their own techniques, enabling the team to get a higher score overall.

However, there were still obstacles to collaboration when using an interface like CoSolve's. For example, users had a difficult time maintaining awareness of their teammates' actions. One possible reason for this is that, as the tree grew larger and solvers started working on different parts of the tree, it became harder to navigate the tree. Several subjects said they were reluctant to pan a very large tree for fear of losing their current working location; as a result, they did not look at their teammates' work as often. Another reason may be that CoSolve's "Refresh" indicator, while designed to avoid disrupting an individual user's view, does so at the expense of some group awareness. Visually, a parent node's children are equally spaced beneath the parent; as each child appeared, existing children are animated to spatially move to make room for the new child. This can be jarring if nodes are moving as a user works on them, and so we implemented the "Refresh" indicator, located in the Consultant menu (Fig. 2) to allow users to update their views on demand, rather than automatically.

This prevents interruptions of users' views, but may also decrease awareness of others' actions.

One possible solution to the problem of awareness, which was also suggested to us by some of our subjects in the interviews, is to have a "mini-map" view of the tree, in one corner of the interface. That way, solvers could work on one part of the tree in the main view, but still have an idea of what the structure of the tree is like, and navigate, using the smaller, corner view of the entire tree. Another idea suggested by one subject was to allow users to split the screen to show different parts of the tree on different sides of the screen. This way, a solver could work on one part of the tree, while also following what a teammate is doing on a different part of the tree.

Finally, many subjects brought up the need for a conventional chat messaging feature. In CoSolve, node annotations are currently the main means of communication, and many subjects used the node annotations like chat—asking questions, talking back and forth—and they wanted to be able to communicate textually without having to associate each comment with a node.

V. CONCLUSION

Online collaborative problem solving is a rich but difficult problem area for new models of computing and user interfaces. In this paper, we have described CoSolve, our novel, tree-based interface for problem solving structured around state-space search. We have discussed the problem-posing and problem-solving processes in our system, and then described the results of a user study which showed that subjects find this novel problem-solving interface helpful, easy to learn and easy to use. We found that the tree structure allowed solvers to explore and share solutions in ways they would not have been able to in a non-tree based interface. There are still improvements to be made, namely in improving tree navigation, awareness, and communication features in our system, but we believe CoSolve is a promising first step toward an effective framework for a computer-supported collaborative problem-solving system.

ACKNOWLEDGMENT

Thanks to Richard Rice, who created the Towers of Hanoi CoSolve problem template, and Chris Brenan and Robert Thompson, who both worked on the CoSolve Flash interface.

REFERENCES

[1] Amazon Mechanical Turk. URL: http://www.mturk.com/mturk

[2] Innocentive. URL: http://www.innocentive.com.

[3] The Polymath Blog. URL: http://polymathprojects.org.

[4] Substepr. URL: http://substepr.com.

[5] T. Gowers and M. Nielsen, "Massively collaborative mathematics," *Nature*, Vol. 461, 2009, pp. 879-881.

[6] J. Cranshaw and A. Kittur, "The Polymath Project: Lessons from a Successful Online Collaboration in Mathematics," In *Proc. CHI*, 2011, pp. 1865-1874.

[7] S. B. Fan, B. R. Johnson, Y.-E. Liu, T. S. Robison, R. R. Schmidt, S. L. Tanimoto, "Analyzing a process of collaborative game design involving online tools," In *Proc. VL/HCC,* 2010, pp. 75-58.

[8] S. B. Fan, "Roles in Online Collaborative Problem Solving," In *Proc. VL/HCC,* 2010, pp. 265-266.

[9] H. Simon, *The Sciences of the Artificial*, 3rd ed. Cambridge: MA: MIT Press, 1996.

[10] D. H. Jonassen, "Toward a design theory of problem solving," *Educational Tech. Res. and Devel*. 48(4), 63-85. 2000.

[11] M. Vass, J. M. Carroll, and C. A. Shaffer, "Supporting creativity in problem solving environments," In *Proc. Creativity & Cognition*, 2002, pp. 31-37

[12] L. Berry, L. Bartram, and K.S. Booth, "Role-based control of shared application views," In *Proc. UIST,* 2005, pp. 23-32.

[13] L. M. Brodie, "eProblem-based learning: problem-based learning using virtual teams," *European Journal of Engineering Education*, 34(6), 2009, pp. 497-509.

[14] P. Dewan, "Architectures for collaborative applications," In M. Beaudoin-Lafon (Ed.), In *Proc. CSCW*, 1999, pp. 497-509.

[15] S. L. Tanimoto, T. Robison, S. B. Fan, "A game-building environment for research in collaborative design," In *Proc. CIG*, 2009, pp. 96-103.

[16] S. Tanimoto, "Enhancing state-space tree diagrams for collaborative problem solving" In *Proc. DIAGRAMS* 2008, pp. 156-164.

[17] S. Tanimoto, and S. Levialdi, "A transparent interface to state-space search programs" In *Proc. ACM SoftVis* 2006, pp. 151-152.

[18] P. Dourish, and V. Bellotti, "Awareness and coordination in shared workspaces". In *Proc. CSCW* 1992, pp. 107-114.

[19] C. Gutwin and S. Greenberg, "Design for Individuals, Design for Groups: Tradeoffs Between Power and Workspace Awareness." In *Proc. CSCW* 1998, pp. 207-216.

[20] I. Heldal, D. Roberts, L. Bråthe, and Robin Wolff. "Presence, creativity and collaborative work in virtual environments," In *Proc. HCI*, 2007, pp.802-811.

[21] P. Dewan, P. Agarwal, G. Shroff, and R. Hegde, "Mixed-focus collaboration without compromising individual or group work." In *Proc. EICS,* 2010, pp. 225-234.

[22] D. Suthers, R. Vatrapu, R. Medina, S. Joseph, S., and N. Dwyer, "Beyond threaded discussion: Representational guidance in asynchronous collaborative learning environments," *Computers & Education*, Vol. 50, No. 4, 2008. pp. 1103-1127.

[23] K. S. Park, A. Kapoor, and J. Leigh, "Lessons learned from employing multiple perspectives in a collaborative virtual environment for visualizing scientific data," In *Proc. CVE* 2000, pp. 73-82.

[24] Drupal. URL: http://drupal.org

[25] SimCity. [Computer Game]. California, USA: Maxis, 1989.

[26] Lew, J. "Making City Planning a Game." New York Times, June 15, 1989.

[27] G. Fischer, and E. Giaccardi, "Meta-Design: A Framework for the Future of End-User Development." In Lieberman, H., Paternò, F., Wulf, V. (Eds): *End User Development--Empowering People to Flexibly Employ Advanced Information and Communication Technology*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.