Online Games as Social-Computational Systems for Solving NP-complete Problems

Charles A. Cusack, Jeff Largent, Ryan Alfuth, Kimberly Klask

Hope College

Abstract

This paper discusses the applicability of *human computing games* to solving instances of NP-complete problems, a collection of problems that cannot currently be efficiently solved with computers. The idea is to leverage the diversity offered by a large group of humans—that is, utilize the different skills humans have that computers don't as well as the different perspectives of the individuals who plays the games. To explore this possibility, we created *Pebble It*, a suite of games that can be used to solve instances of problems related to a mathematical concept called graph pebbling. Several examples are presented that demonstrate the benefits of this approach to problem solving. We conclude by discussing how these games can link players and researchers to form a social-computational system that can strengthen research—sometimes in unexpected ways.

Introduction

Ever since they were first created, computers have been a boon to mathematicians and scientists world-wide. There are many problems that computers can solve much faster than any human. However, there remain important problems that are beyond their abilities to solve but that most humans can solve easily, including recognizing written words, associating words with pictures, and determining if two audio clips are the same. There are many grey areas between and around these extremes. One of these grey areas is the collection of NP-complete problems. These are problems that computers are technically able to solve, but not in a reasonable amount of time and/or with a reasonable amount of resources. This is a grey area because it is unknown whether or not an algorithm will ever be developed to efficiently solve any of these problems. Whether or not humans can efficiently solve any of these problems is also unknown. Because many practical problems related to routing, resource distribution, scheduling, etc. are NP-complete, progress in this area is important for reasons beyond satisfying the intellectual curiosity of a handful of theoretical researchers.

Games that seek to utilize the ability of humans to solve real problems, called *Human Computing Games* or *Games With A Purpose*, have been around for several years. However, most of these games solve problems that are already known to be easy for most humans (e.g. image recognition). This paper presents a collection of games whose purpose is to determine if and to what extent untrained humans can assist in solving what are typically considered computational problems—specifically those that are NP-complete. Several examples are presented that demonstrate how the players, games, and researchers form a social-computational system, whereby the insights provided by untrained players assist the researchers.

Human Computing Games

Crowdsourcing is a method of using the collective wisdom a large and diverse group of people to solve a problem. *Human Computing Games* (*HCGs*) are a form of crowdsourcing that have proven to be an effective method of collecting large amounts of useful data. HCGs are an ideal vehicle for crowdsourcing, especially with abstract or theoretical problems, as they can present the problem in a non-threatening way and even transform it into a form of entertainment. HCGs cast a wide net, drawing participants with diverse backgrounds, education, talents, etc. This increases the chance that someone will see the problem in a different way and think of a solution that researchers might easily miss, since players have few preconceived notions of what may or may not work. HCGs may also lead to the discovery of unknown savants.

Related Work

In the past five years, over 40 HCGs have been presented at conferences and in journals by a growing number of researchers. These games collect common sense facts, annotate everything from music to web pages, train artificial intelligence algorithms, produce useful geospatial data, evaluate the sentiment of terms and sentences, help predict protein structures, and solve a variety of other problems.

The first and best-known examples of HCGs are the *Games with a Purpose (GWAP)* created by Luis von Ahn and his colleagues (von Ahn, 2006). The games include the *ESP Game* (von Ahn & Dabbish, 2004) and *Phetch* (von Ahn, S. Ginosar, M. Kedia, & Blum, 2007; von Ahn, Shiry Ginosar, Mihir Kedia, Liu, & Blum, 2006), which label/describe images on the Internet; *Peekaboom* (von Ahn, Liu, & Blum, 2006) and *Squigl*, which locate objects within images; *Verbosity*, which collects common sense facts; *TagATune* (Law & von Ahn, 2009; Law, von Ahn, Dannenberg, & Crawford, 2007), which annotates music and sound; *Matchin* (Hacker

& von Ahn, 2009), which determines images that users prefer; *FlipIt*, which determines whether or not two images are similar; and *PopVideo*, which annotates videos.

Many HCGs follow one of the three game-structure templates described by von Ahn and Dabbish: *output-agreement*, *inversion-problem*, and *input-agreement* (von Ahn & Dabbish, 2008). In all three of these templates, players are randomly paired and (presumably) cannot communicate with each other. The goal is for both players to "agree" on something, the idea being that if two humans can agree, then it is likely that what they agree about is correct. With output-agreement games, both players are presented with the same input (e.g. a picture) and produce outputs (e.g. words that describe the picture) until at least one of their outputs is the same (e.g. they both say "cheetah"). With inversion-problem games, one player is given an input (e.g. a word) and provides the other player with outputs related to the input (e.g. phrases that describe the word) until the second player can reproduce the input. Finally, input-agreement games provide each player with an input (e.g. a song), both players provide output to each other that is related to their given input (e.g. words describing the song—style of music, instruments used, etc.), and they need to determine whether or not they were given the same input.

The *Spectral Game* (Bradley, Lancashire, Lang, & William, 2009) is aimed at undergraduate students studying for organic chemistry. It presents the player with an actual spectral sample from the *ChemSpider* database and the player is to identify the structure of the molecule that the spectral sample represents. Not only does this help the chemistry student study spectroscopy, but because the user can leave comments on particular samples, it also serves as a tool for helping researchers determine if the samples are impure or incorrect. So far the game has resulted in three samples being removed from the *ChemSpider* database, and several impure peaks have been removed from the existing samples.

In *Foldit* (Cooper, Treuille, et al., 2010; Cooper, Khatib, et al., 2010) players are presented with the task of folding three-dimensional proteins as close as possible to their "natural state." A better understanding of how proteins fold can help find cures for diseases. The game also allows players to help design proteins that could help with the prevention or treatment of diseases. Protein folding is a computationally difficult problem that may benefit not only from visualizations skills, but also from certain problem-solving skills possessed by humans (Z. Popović, personal communication, February 14, 2008) that cannot (at least currently) be programmed into a computer. In contrast to the *GWAP* games that use basic human recognition to attack problems, *Foldit* requires special insight from the player to solve the problem.

Cognition versus Recognition

Most HCGs follow a model in which two or more players are randomly grouped and cooperate and/or compete in some way, resulting in a piece of information that must be verified by other groups of players and/or researchers before it can be considered correct. In other words, they produce data that *might* be correct. Furthermore, these games are often solving problems that do not necessarily have a fixed answer (e.g. words that describe a picture).

Most of these games share another characteristic: they require no special insight from players. They solve what we might call *recognition problems*. In a sense, most of these games view the player like a battery that powers their algorithm, not like an independent computational unit that might have something unique to contribute. They afford players no opportunity to become better at the game in a meaningful way, and have no possibility of a savant coming up with an insight that the researcher missed. Only a few researchers are attempting to use games to solve *cognition problems*—those that require more insight and ability to solve. These include the previously mentioned *Foldit*; *FunSAT*, a game that has players solve instances of the NP-

complete problem *Satisfiability* (DeOrio & Bertacco, 2009); and a game to solve certain NP-complete scheduling and mapping problems (Lin & Dinda, 2009). The work of DeOrio and Bertacco seems very preliminary, in that although they refer to *FunSAT* as a 'massively multi-player puzzle game', the only version of the game that we found seemed to be single-player, and we found the gameplay to be non-intuitive. Similarly, the game created by Lin and Dinda does not seem to be available online, and the screenshot in their brief abstract does not look very game-like. On the other hand, *Foldit* has been around for over two years, has had over 57,000 players, with several hundred active players per week, which is impressive given the complexity of the game. The creators regard the contributions made by the players as so significant that they list "Foldit players" as co-authors on their recent articles (Cooper, Treuille, et al., 2010; Cooper, Khatib, et al., 2010). This bodes well for the prospect of using Human Computing Games to solve cognition problems.

Background

NP-complete Problems

Problems in the complexity class NP-complete share two important characteristics. First, no efficient algorithm (polynomial-time in the size of the input) is known to solve them. What this means is that only small instances of these problems can currently be solved, even with the use of very powerful supercomputers. Second, all of the problems are equivalent in the sense that an efficient algorithm to solve *any* of them would yield an efficient algorithm to solve *all* of them. One of the most important open problems in theoretical computer science is whether or not there is an efficient algorithm to solve any, and thus all, NP-complete problems (Cook, 1971). As many practical problems are NP-complete, there is a large amount of interest in

finding an answer to this question. The thousands of papers that have been published related to the so-called “P versus NP” question are a testament to this high level of interest.

Most scholars who research these types of problems are mathematicians and theoretical computer scientists—not the most diverse group of thinkers. In addition, some people think that globalization has led to more uniformity of thought (Gasarch, 2002). This is particularly problematic in this case: since most researchers believe that no efficient algorithm will be found for NP-complete problems, few researchers are willing or able to “go out on a limb” and spend too much time attempting to find such an algorithm. Finally, the tenure and promotion system makes it difficult for researchers to spend long periods of time trying to solve the problem with little to no evidence of progress.

The use of HCGs has the potential to contribute to this research area by bringing in the expertise of the more broadly educated masses. Since most people do not understand this problem, its difficulty, and its already assumed—but unproven—answer, they will be neither discouraged nor influenced by those “in the know.” They will be able to approach the problem from a perspective which is likely very different from that of a typical researcher, having no preconceived notions and typically not being subject to peer pressure and expectation. Finally, they will bring with them knowledge that perhaps no current researchers possess.

Graph Pebbling

Many important NP-complete problems are related to graphs or can easily be reduced to problems on graphs. Further, many graph problems can be visualized in a very game-like way. In light of this, games based on graph problems are good candidates to test the feasibility of applying the HCG model to computational problems. In fact, a collection of problems related to a mathematical construct called *graph pebbling* are ideal in that these problems are often

described using the language of games. These thoughts gave rise to *Pebble It*, a suite of games that solve various graph pebbling problems. Before we describe the games, a brief discussion of graph pebbling is necessary.

A *graph* **G** = (**V**, **E**) is composed of a set of *vertices*, **V**, and a set of pairs of vertices, **E**, called *edges*. Put simply, a graph is a set of vertices with edges connecting some of the vertices (see Figure 1a). Two vertices of a graph are *adjacent* to each other if they are connected by an edge. A *pebbling configuration* (or just configuration if the context is clear) is a function C from vertices to non-negative integers, where $C(v)$ is the number of pebbles on vertex $v \in V$. In other words, a pebbling configuration assigns zero or more pebbles to each vertex (see Figure 1b).

For any vertex v such that $C(v) \geq 2$ (that is, v contains at least two pebbles), a *pebbling step* consists of removing two pebbles from v and placing one pebble on a vertex adjacent to v. Thus, pebbles can be moved from one vertex to an adjacent vertex, but each time a pebble is moved, an additional pebble is lost (see Figure 1c). A configuration is called *r-solvable* if there is a sequence of pebbling steps that places at least one pebble on a specified vertex r. A configuration is called *solvable* if it is r-solvable for every $r \in V$—that is, if it is possible to place a pebble on any vertex of the graph using pebbling steps.

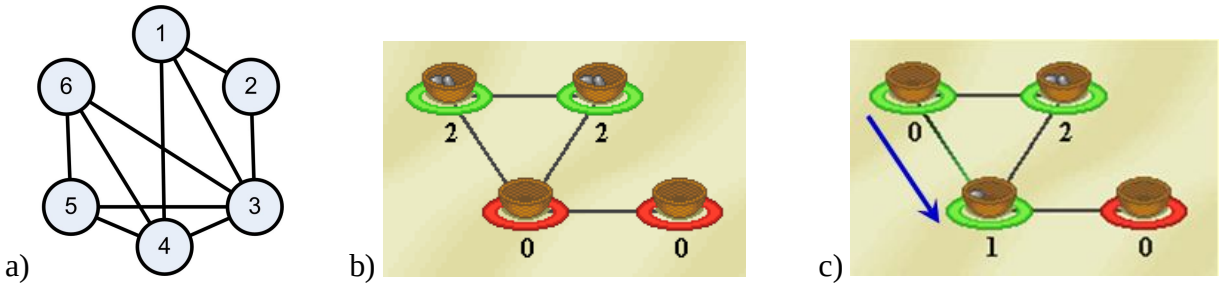


Figure 1 (a) A graph with vertex set {1, 2, 3, 4, 5, 6} and edge set {(1,2), (1,3), (1,4), (2,3), (3,4), (3,5), (3,6), (4,5), (4,6), (5,6)}. (b) A graph with pebbles on some of the vertices. (c) The graph from (b) after a pebble is moved as indicated by the arrow.

Determining whether or not a configuration is *r*-solvable for a given vertex *r* is NP-complete, which implies that determining solvability for the entire configuration is also NP-complete (Milans & Clark, 2006; Watson, 2005). These problems are referred to as *Reachable* and *Solvable*, respectively. An efficient algorithm to solve both of these problems is known for certain classes of graphs (Cusack & Bekmetjev, 2009). Other problems related to graph pebbling which are NP-complete include determining the minimum number of pebbles needed to ensure that any configuration is solvable (*Pebbling Number*), the minimum number of pebbles such that there exists at least one solvable configuration (*Optimal Pebbling*), and whether or not there is a sequence of pebbling steps that leads to a graph in which every vertex has at least one pebble (*Cover Pebbling*) (Milans & Clark, 2006).

The Games

As mentioned previously, *Pebble It* is a suite of games related to graph pebbling. They can currently be played at http://graph.computinggames.org. Each game is either tied directly to a particular problem or can be used to help answer questions about several problems. In most *Pebble It* games players either make moves corresponding to pebbling steps or place pebbles on a graph with a particular goal in mind. Every game consists of a collection of typically unrelated puzzles of that type. Every move made by the player is recorded and stored in a database so that the games can be replayed and studied by the player and/or members of the research team. Players are ranked on each puzzle and earn points based on how well they did relative to both their previous best score on the puzzle and the previous best score of all players, including the amount of time it took them to complete the puzzle. The points allow them to "level up" as well as be ranked according to their overall contribution to the project. Players can also earn achievements for things like attempting all puzzles in a particular game, completing a puzzle a

certain number or times, or getting first place on any puzzle. Puzzles are split into 6 skill levels, and players can only play puzzles in the next skill level when they have completed a certain percentage of puzzles in their current skill level for that game. These features are all designed to not only keep players coming back to play often, but to also encourage them to play better so that their contributions will be more meaningful.

Reach It

Reach It is the simplest and most straightforward game in the *Pebble It* framework. The puzzle consists of a graph and a pebbling configuration, and players are asked to make pebbling steps in order to move a pebble onto a target vertex. In other words, it helps solve the *Reachable* problem. Users have the option of restarting the puzzle at any time without penalty, as it is possible that they may run out of pebbles or wish to try a different strategy. Since we want to know the most efficient way to reach the goal vertex, scores are based on the number of pebbling steps, with fewer moves being better.

New *Pebble It* players are encouraged to play *Reach It* first, as it is the easiest to grasp conceptually and serves as a good introduction to the idea of graph pebbling. Another benefit of *Reach It* is that a configuration will often have many vertices that are easily reached, with only a small number of vertices that are difficult to reach. By marking one of these difficult vertices as the target, it is possible to narrow graph solvability problems down to just one goal for players to focus their attention on.

Rock It

Rock It extends the premise of *Reach It* from determining if a single vertex is reachable to checking if every vertex on a graph is reachable—solving the *Solvable* problem. Players are asked to show that each vertex on a graph is reachable by moving at least one pebble onto it.

Once a pebble has been placed on a vertex, that vertex is designated as "reachable" (symbolized by the base of the vertex turning from red to green. See Figure 2), and remains reachable regardless of how many pebbles it has later. This means that users do not have to place a pebble on every vertex on a graph at once, and are free to "recycle" their pebbles at any time, restoring

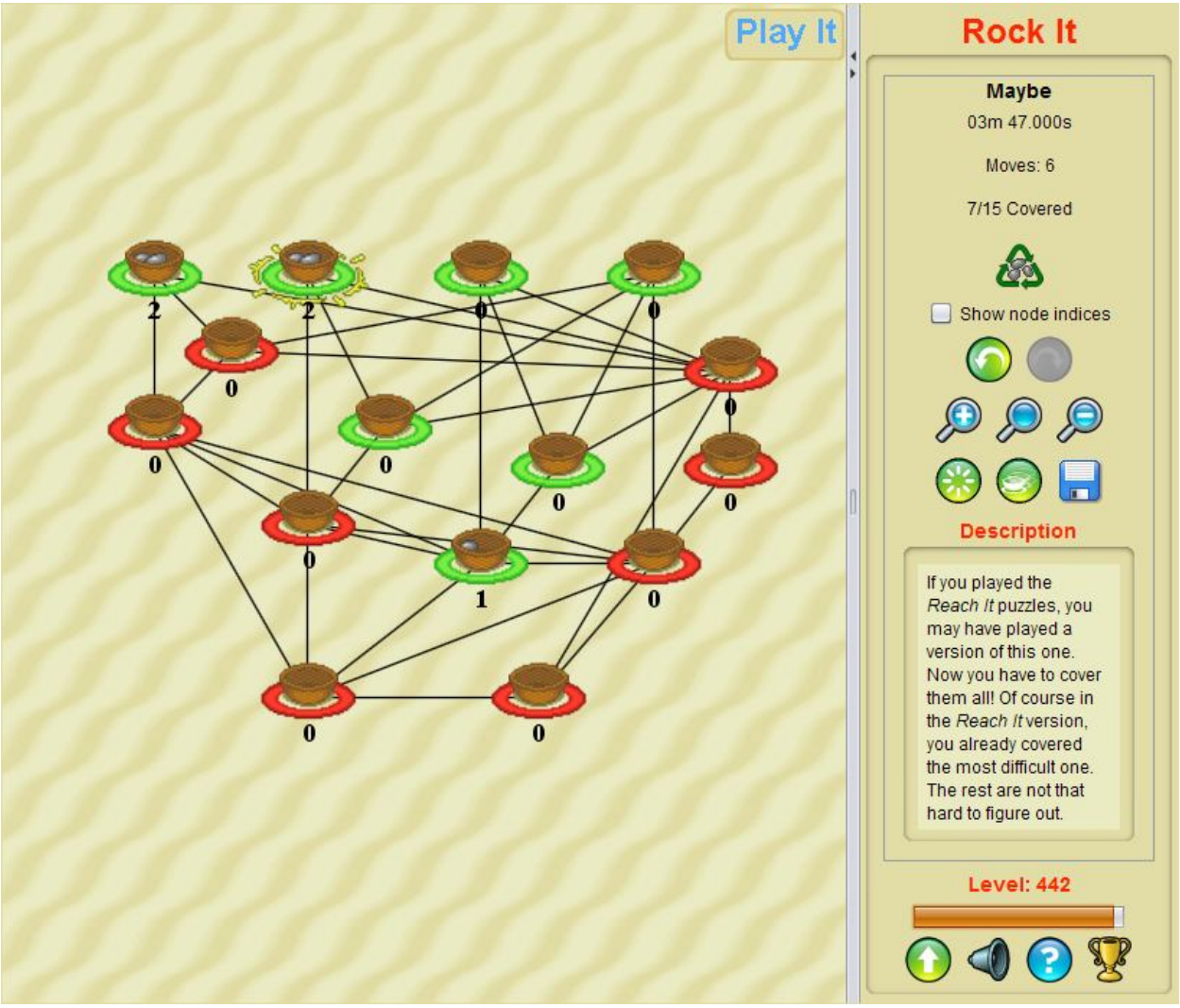


Figure 2. A screenshot of the play screen for *Rock It*. The player must make pebbling steps to place pebbles on each of the red vertices. Since in many cases it is impossible to reach them all simultaneously, they may use the "recycle" button to put the pebbles back to their initial configurations and continue making pebbling steps.

the initial pebbling configuration. They can also undo their moves, which reverses the effects of their last move but leaves reachable vertices colored green. As with *Reach It*, players are trying to finish the puzzle in the fewest number of pebbling moves.

Confound It

Rather than being concerned with determining the solvability of a pebbling configuration, *Confound It* asks players to place as many pebbles on a graph as possible in such a way that the configuration is unsolvable. In other words, they are trying to make sure at least one vertex cannot be reached by a pebble. Players can place as many pebbles on the graph as desired and, as long as they have at least one vertex with two or more pebbles on it, can click the "Check It" button to test their configuration. They have a choice of several algorithms to test the solvability of the puzzle, and if one of them determines that the configuration is unsolvable, they have completed the puzzle. Since it is more difficult to make a graph unsolvable with a large number of pebbles than it is with a small number, score is based on the number of pebbles they place on the graph, with higher scores being better.

Results from *Confound It* can help determine bounds on the pebbling number of a graph. Since the pebbling number is the minimum number of pebbles required to ensure that any arrangement of those pebbles on a graph will be solvable, the pebbling number must be higher than the number of pebbles used on any unsolvable configuration. In other words, if a pebbling configuration is found to be unsolvable, then the pebbling number has to be higher than the number of pebbles used in that configuration. Thus, by encouraging users to find unsolvable configurations with the greatest numbers of pebbles, we can lower bound the pebbling number for a graph based on the best solution to the puzzle.

Optimize It

Similar to *Confound It*, *Optimize It* presents players with a graph and asks them to place pebbles on vertices. However, the goal is just the opposite—players are told to come up with a solvable pebbling configuration using as few pebbles as possible. *Optimize It* helps determine bounds on the optimal pebbling number of a graph.

Goals

There are several possible goals this sort of research could have. The most ambitious goal would be to use HCGs to help settle the P versus NP questions. For several reasons beyond the scope of this paper, this is an extremely unlikely outcome. A slightly less ambitious goal is to use player strategies to improve the currently best known algorithms for the problems that the games are based on. This is more difficult than it sounds, however. As an example, although we know that humans can generally differentiate between a cat and a dog in a picture, we really have no idea how they do it, so their "strategy" cannot be turned into an algorithm. Given that the focus of this project is on more computational problems, and that all of the moves the players make as they play the games are recorded, there may be a better chance of extracting strategies. The difficulty is that although their moves can be studied, doing so does not shed any light on *why* such moves are made—and users may not be able to articulate their reasoning. So in the end, it might be possible to learn from player strategies, but this is not necessarily an expectation.

There are two more reasonable goals, especially in the short-term. The first is that players will solve particular instances of problems that computers have failed to solve previously. Since humans solve problems very differently than computers, the hope is that players can use short-cuts that are difficult—if not impossible—to program algorithmically, allowing them to outperform computers on certain types of problem instances. In other words,

humans are sometimes able to "just look at" a problem and see the answer—or at least greatly narrow down the possibilities. In fact, there are *Reach It* and *Rock It* puzzles that humans can determine in just a few minutes whether or not they can be solved, but which our current algorithms cannot determine given hours or even days.

The second goal is that players will provide data that can be used to assist in establishing certain facts about the underlying problems. For instance, it is often the case that patterns emerge for things like the pebbling number or optimal pebbling number when considering a family of graphs (a collection of graphs with very similar structure but varying in size). However, these patterns may not be very obvious. A series of puzzles can be created so that players can determine such patterns—or at least provide rough estimates. Thus, a tedious task for the researcher can be turned into a game for others. Additionally, puzzles can be created to assist researchers in constructing examples and counter-examples. Sometimes researchers need to construct structures (in our case, graphs) with (or without) certain properties in order to support or refute a conjecture. This is a situation in which appealing to a larger group of people—in this case players—is certainly helpful since everyone has their own way of looking at things, and what might slip past some (e.g. the researchers) will hopefully not slip past everyone (e.g. the players).

Of course a fundamental question that must be addressed first is whether or not untrained humans can even solve instances of computational problems while playing a game. Can a grade-school kid, soccer-mom, construction worker, or retired insurance salesman be expected to actually solve puzzles based on mathematical problems without having to understand all of the complexities involved? Can the uninformed (or ill-informed) really contribute something

meaningful to the research of experts in a given field? In the next section, evidence is provided that suggests the answer to these questions is yes.

Results

We expect to fully deploy *Pebble It* to the public late in the summer of 2010. At this point *Pebble It* has been tested by over two hundred people. Most players have completed several *Rock It* puzzles (this was the first game to be developed), about 45 players have played *Reach It* puzzles, about 10 have played *Confound It*, and about 7 have played *Optimize It* (The latter two puzzles have only been available to the public for about a week.). Even with only this much data, the results so far have been both encouraging and surprising. Most of the players had no previous knowledge of graph pebbling yet they picked up the rules fairly quickly, especially after modifications were made based on the first round of player testing. Two particular examples are especially noteworthy.

The first example occurred when a player beat the high score on a relatively simple puzzle (See Figure 3). Since the puzzle was so easy, and since we clearly knew the best way to solve it, we assumed there must be an error in how the player's moves were recorded or in the formula used to compute his score. We replayed his solution and were stunned that we had totally overlooked a strategy. As seen in Figure 3b, it seems logical that in order to reach a vertex we should simply move all of the pebbles along the same path. However, the optimal strategy is to split them and then recombine them later, as shown in Figure 3c. With this technique, one additional vertex is covered with the same number of moves, saving a move later on. A non-expert in graph pebbling was able to make a simple observation that eluded a researcher who had focused on this problem for several years, along with three students involved in creating the game. This incident was the first evidence that the crowdsourcing approach has

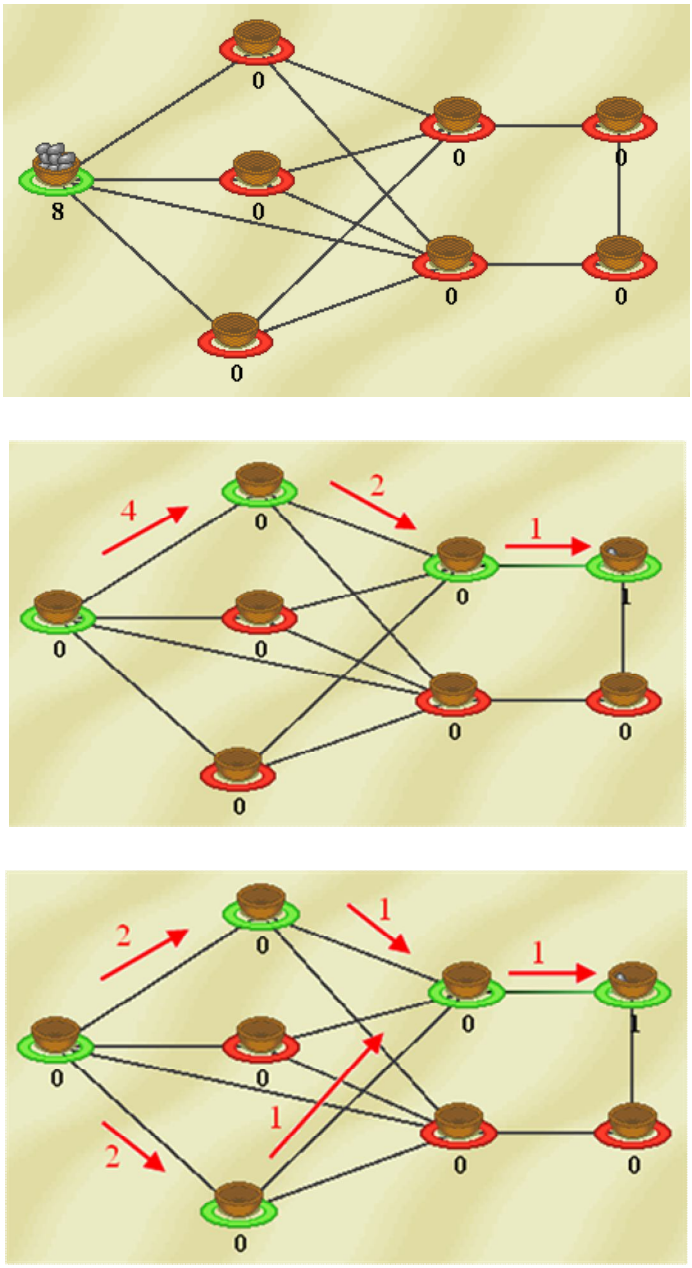


Figure 3. (a) *Rock It* puzzle *Melek* in its original configuration. (b) The path we believed was most efficient to move a pebble to the upper-right vertex. The number next to each arrow indicates the number of pebbles moved. After recycling, five more moves are required to reach the remainder of the vertices. (c) The path used by a player. Although it takes the same number of moves to reach the upper-right vertex, only four moves are needed to finish the puzzle after recycling—one less than required by our "optimal" solution.

potential for problems that might generally be thought of as too computational for the average player. Although a simple example, we believe that this is strong evidence that the average person may have much more to offer than some researchers might suspect. Moreover, the fact that this is such a simple example actually lends credence to the crowdsourcing approach since if such a simple optimization can be missed by several researchers, it is likely that more complicated optimizations might also be missed by researchers but discovered by players. We firmly believe that the "more the merrier" approach has a lot of potential to contribute to this sort of research.

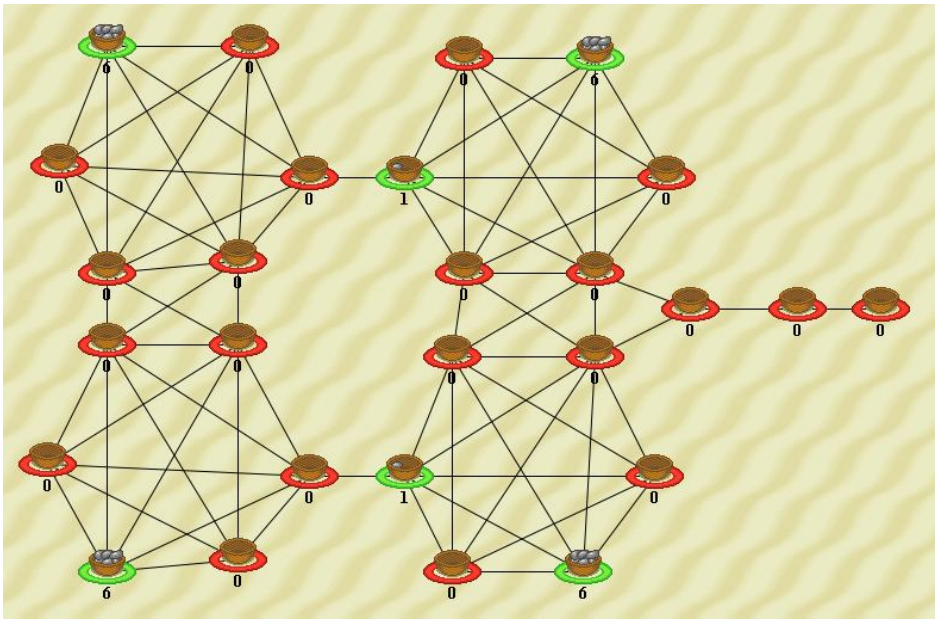
A second encouraging example is related to *Confound It*. Before the example is discussed, a little background is necessary. As mentioned previously, the goal of *Confound It* is to place as many pebbles on a graph as possible so that there is at least one vertex that cannot be reached with a pebble. Some method is needed to tell players whether or not their configuration is solvable. As there is no efficient algorithm to determine solvability of every graph, this type of puzzle presents a challenge. Several algorithms are provided for the players to test their solutions. If one is taking too long, they are free to stop it and try another one. The idea is that players are only presented with puzzles for which one of our algorithms can determine the solvability within a few minutes, if possible. In order to do this, we must be able to estimate how long the algorithms will take, which can be tricky. The approach we have taken thus far is to implement an algorithm that we think will perform reasonably on some set of graphs and configurations and to test its performance. When we find one or more graphs/configurations on which it does not perform well, we try to either improve the algorithm for those graphs, or create a new algorithm. When all of our sample puzzles are solvable with at least one of the current algorithms, we create new puzzles that we suspect none of the current algorithms will perform

well on. These puzzles help identify the weaknesses of the current algorithms and provide inspiration for the next algorithm. The following example will help to illustrate this process.

A puzzle called *FourSixesAndSome* (See Figure 4a) was created as an example of a puzzle that the *exhaustive search* algorithm would have difficulty completing. This algorithm failed to solve the puzzle and was aborted after 470 hours. An algorithm based on *shortest path* was developed to handle this and several other examples—it took only 125 milliseconds for this graph. Then we created *4ToTrickU* (See Figure 4b) by slightly modifying *FourSixesAndSome* so that the shortest path algorithm would no longer work.

Another algorithm was developed that performed well on *4ToTrickU* and all of the other graphs, which led to the creation of a specialized graph that thwarted it. Even so, we still believed that this latest algorithm would perform efficiently on all diameter two graphs (graphs in which every pair of vertices is either directly connected or has a common neighbor) and were in the process of constructing a proof of this fact. We added two *Confound It* puzzles, one of them diameter two, that we were reasonably sure this algorithm could handle with ease. Then two players, a high school student and a college student, came up with very simple examples that stumped the algorithm. Upon seeing the examples, it was immediately clear why the algorithm performed badly. Although we would have eventually found this weakness, the timing of these examples saved us countless hours trying to prove something that now seems likely to be false. With our knowledge and effort we failed to come up with similar examples, yet two players, who had no knowledge of our algorithms and were not even trying to come up with hard examples, provided us with invaluable test cases. This experience has inspired a future *Pebble It* game called *Stump It* in which the goal will be to create a graph and pebbling configuration of reasonable size that none of our algorithms can solve in less than some predetermined time.

a)



b)

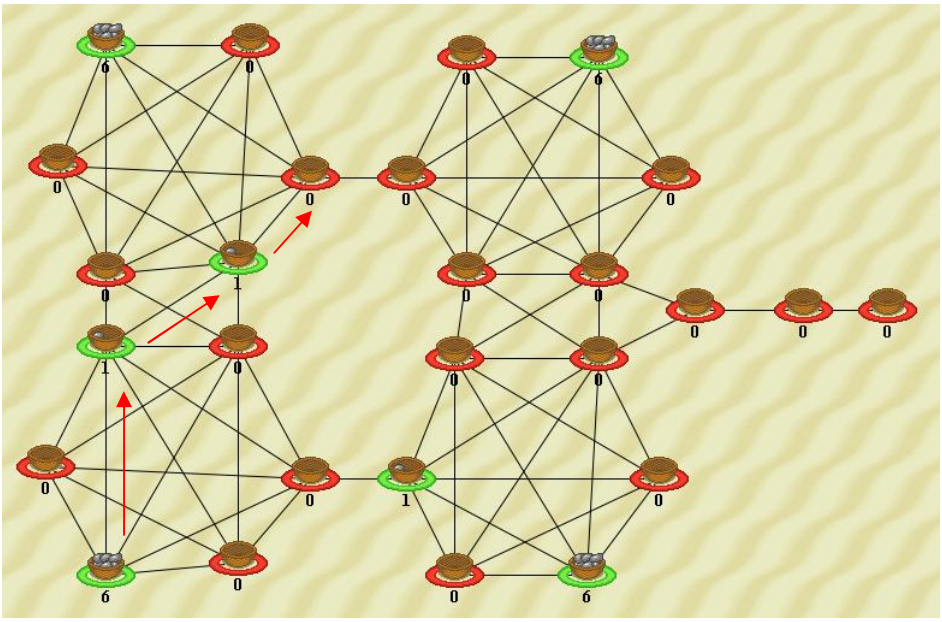


Figure 4. (a) *FourSixesAndSome.* Reaching the right-most vertex can be accomplished by always moving pebbles closer to it—the *shortest path* strategy. (b) *4ToTrickU.* Reaching the right-most vertex requires moving 1 pebble along the path indicated by the arrows, and then using the shortest path strategy. Using the shorted path strategy alone does not work on this puzzle.

These examples demonstrate that players can contribute to research in important and unintended ways, sometimes without even realizing it. They are concrete examples of how people with no specialized knowledge in a particular field can still contribute to research in that field, even if it involves algorithmic processes and complex mathematical concepts.

Challenges

Beyond the obvious challenges of creating games based on difficult problems that are fun to play and getting people to play them when there are thousands of other games to choose from, there are other challenges to address pertaining to using games to assist in solving NP-complete problems. Since graph pebbling problems map very easily to games, they are a good starting point to explore the feasibility of the techniques outlined here. However, beyond assisting a handful of mathematicians with theoretical results related to graph pebbling, there are currently no known practical applications of graph pebbling. Thus one challenge is to choose which problems to focus on. Four criteria come to mind: First, it must be possible to map the problem to an enjoyable game. Although some NP-complete problems, especially those related to graphs, may map nicely into games, this may not be the case with many (perhaps even most) NP-complete problems. Second, the games should allow for the collection of meaningful data. In other words, the players' actions should map to problem details in such a way that the players can make intelligent game choices that correspond to intelligent problem-solving choices. This is one thing that sets HCGs apart from *volunteer computing games* (Cusack, Martens, & Mutreja, 2006; Cusack, Peck, & Riolo, 2008), a paradigm in which the power of the player's computer is of more interest than the power of their mind. Third, the problems, along with the specific puzzles created for each, should be practical in the sense that they correspond to real-world problems and data. Since even progress on theoretical problems is of interest to some, this

criteria is not strictly necessary. However, there are benefits on all sides when real-world problems are involved. Finally, the problems should be those which do not already have reasonably efficient solutions. As with the previous criteria, this is not strictly necessary, but if a problem can already be solved with another technique, then it is not clear why an HCG would be needed.

Another challenge relates to the non-symmetric nature of NP-complete problems. As an example, to demonstrate that a pebbling configuration is solvable, a player only needs to make all of the legal moves required to place a pebble on each vertex. On the other hand, there is in general no easy way for the player to demonstrate that a configuration is *not* solvable. All NP-complete decision problems (that is, problems that ask a yes-no question) share this difficulty—essentially, when the answer is 'yes' there is generally an easy way to prove it, but when the answer is 'no' there is not.[1] In other words, if the answer to a particular problem is 'yes', then we might expect that eventually someone will be able to figure out the correct steps to demonstrate that this is the case. However, currently we have no way to allow the players to demonstrate that the answer is 'no', and it is not clear how, or even if, this can be done. If a puzzle goes unsolved for years, we may suspect that the answer is 'no', but cannot say this with any certainty.

Conclusion

Pebble It is a game that takes the human desire to be entertained and turns it into a useful tool for researchers. More than that, it forms the center of a social-computational system in which researchers create puzzles so that players can contribute using their unique perspectives.

[1] Technically speaking all NP-complete problems are decision problems, but in order to not bog down the discussion with details beyond the scope of this paper, the discussion of NP-complete problems has been simplified.

The players provide data that is larger in both quantity and diversity than a researcher could possibly create alone. Although the results are still preliminary, there is evidence that given the right games and the right puzzles, players will be able to provide useful data related to graph pebbling. This data will not only be in the form of solutions to particular puzzles, which is an expected outcome, but will also provide examples and counter examples that can be used to improve algorithms and future puzzles in unanticipated ways.

This concept is not limited to graph pebbling. Although it is not be possible to turn every problem into a game, there are many problems—especially graph-relates ones—that have potential. We believe that the wisdom of the crowds has something to offer researchers in fields like mathematics, theoretical computer science, and engineering, where conventional wisdom would suggest that only "experts" can make significant contributions, and that online games are a perfect venue in which to pursue this partnership. This approach to research certainly has its own set of challenges, but we believe that facing them is worth the risk in light of the potential benefits, both foreseen and unforeseen.

References

Bradley, J. C., Lancashire, R. J., Lang, A. S. I. D., & William, A. J. (2009). The SpectralGame: Leveraging Open Data for Education. *Journal of Cheminformatics*, *1*(9). doi:10.1186/1758-2946-1-9

Cook, S. (1971). The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (pp. 151-158).

Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., … Foldit players (2010). Predicting protein structures with a multiplayer online game. *Nature*, *466*, 756-760. doi:10.1038/nature09304

Cooper, S., Treuille, A., Barbero, J., Leaver-Fay, A., Tuite, K, Khatib, F., … >57,000 Foldit players (2010). The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games - FDG '10* (pp. 40-47). Presented at the the Fifth International Conference, Monterey, California. doi:10.1145/1822348.1822354

Cusack, C., & Bekmetjev, A. (2009). Pebbling Algorithms in Diameter Two Graphs. *SIAM J.Discret.Math.*, *23*(2), 634-646.

Cusack, C., Martens, C., & Mutreja, P. (2006). Volunteer Computing Using Casual Games. In *Proceedings of Future Play 2006 International Conference on the Future of Game Design and Technology*.

Cusack, C., Peck, E., & Riolo, M. (2008). Volunteer Computing Games: Merging Online Casual Gaming with Volunteer Computing. In *Proceedings of the International Academic Conference on Meaningful Play*.

DeOrio, A., & Bertacco, V. (2009). Human computing for EDA. In *Proceedings of the 46th*

Annual Design Automation Conference (pp. 621-622). Presented at the Annual ACM IEEE Design Automation Conference, San Francisco, California: ACM.

Gasarch, W. I. (2002). The P=?NP poll. *SIGACT News, 33*(2).

Hacker, S., & von Ahn, L. (2009). Matchin: eliciting user preferences with an online game. In *Proceedings of the 27th international conference on Human factors in computing systems*. Boston, MA, USA: ACM.

Law, E., & von Ahn, L. (2009). Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the 27th international conference on Human factors in computing systems*. Boston, MA, USA: ACM.

Law, E. L. M., von Ahn, L., Dannenberg, R. B., & Crawford, M. (2007). Tageatune: A Game for Music and Sound Annotation.

Lin, B., & Dinda, P. A. (2009). Experiences with scheduling and mapping games for adaptive distributed systems: summary. In *Proceedings of the 6th international conference on Autonomic computing* (pp. 73-74). Presented at the International Conference on Autonomic Computing, Barcelona, Spain: ACM.

Milans, K., & Clark, B. (2006). The Complexity of Graph Pebbling. *SIAM Jounral on Discrete Mathematics, 20*(3), 769-789.

von Ahn, L. (2006). Games with a Purpose. *Computer, 39*(6).

von Ahn, L., & Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. Vienna, Austria: ACM.

von Ahn, L., & Dabbish, L. (2008). Designing games with a purpose. *Commun.ACM, 51*(8).

von Ahn, L., Ginosar, S., Kedia, M., & Blum, M. (2007). Improving Image Search with PHETCH. In *IEEE International Conference on Acoustics, Speech and Signal Processing*

(pp. IV-1209-IV-1212).

von Ahn, L., Ginosar, S., Kedia, M., Liu, R., & Blum, M. (2006). Improving accessibility of the web with a computer game. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. Montreal, Quebec, Canada: ACM.

von Ahn, L., Liu, R., & Blum, M. (2006). Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. Montreal, Quebec, Canada: ACM.

Watson, N. G. (2005). The Complexity of Pebbling and Cover Pebbling. Retrieved from http://arxiv.org/abs/math.CO/0503511