Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing

On the Verification Complexity of Group Decision-Making Tasks

Ofra Amir
School of Engineering and Applied Sciences
Harvard University, USA
oamir@seas.harvard.edu

Yuval Shahar **Ya'akov Gal** **Litan Ilany**
Deptartment of Information Systems Engineering
Ben-Gurion University of the Negev, Israel
{yshahar,kobig,litanil}@bgu.ac.il

Abstract

A popular use of crowdsourcing is to collect and aggregate individual worker responses to problems to reach a correct answer. This paper studies the relationship between the computation complexity class of problems, and the ability of a group to agree on a correct solution. We hypothesized that for NP-Complete (NPC) problems, groups would be able to reach a majority-based correct solution once it was suggested by a group member and presented to the other members, due to the "easy to verify" (i.e., verification in polynomial time) characteristic of this complexity class. In contrast, when posed with PSPACE-Complete (PSC) "hard to verify" problems (i.e., verification in exponential time), groups will not necessarily be able to choose a correct solution even if such a solution has been presented. Consequently, increasing the size of the group is expected to facilitate the ability of the group to converge on a correct solution when solving NPC problems, but not when solving PSC problems. To test this hypothesis we conducted preliminary experiments in which we evaluated people's ability to solve an analytical problem and their ability to recognize a correct solution. In our experiments, participants were significantly more likely to recognize correct and incorrect solutions for NPC problems than for PSC problems, even for problems of similar difficulties (as measured by the percentage of participants who solved the problem). This is a first step towards formalizing a relationship between the computationally complexity of a problem and the crowd's ability to converge to a correct solution to the problem.

Introduction

One of the main applications of human computation is combining individual worker solutions to tasks such as classification, prediction and optimization problems (Lintott et al. 2008; Bernstein et al. 2010; Zhang et al. 2012). The focus of this paper is a class of problems for which there is a ground truth (which may or may not be known by the owner of the task), and workers' solutions to the problem can be immediately verified for correctness. Examples of such *analytical tasks* include protein folding, picture tagging, information extraction and translation.[1]

[1]We distinguish analytical tasks from estimation or prediction tasks (e.g., will a stock market crash occur in 2015) who have a ground truth which cannot be immediately verified.

On-line labor markets such as Amazon Mechanical Turk (MTurk) are a chief source of workers to many analytical tasks. However, the background and expertise of the participants of such markets vary widely, and their reports are inherently noisy. Thus a focus of research in the human computation literature is the aggregation of the solutions of individual participants into a collective choice (Little et al. 2009).

This paper provides preliminary support for the relationship between the computational complexity of analytical problems and the ability of the crowd to verify the correctness of a solution presented to them. We study two types of problems belonging to well known computational complexity classes. We consider Nondeterministic-Polynomial-Time-Complete (NPC) problems as "easy-to-verify", since they can be verified in linear or polynomial time. In contrast, Polynomial-Space-Complete (PSC) problems are "hard-to-verify", since they require an exponential number of steps and memory units to verify the correctness of a solution.

Our hypothesis was that for easy-to-verify problems, workers would be able to recognize a correct solution even if they were not able to solve the problem on their own. Thus, it would be sufficient for a single individual to solve the problem in order for the majority of the group to recognize and adopt the correct solution. As the likelihood of the existence of such a "solver" will increase, so should the ability of the group to converge to the correct solution, since group-members would be easily convinced of the correct solution. We contend such problems are suitable for crowdsourcing. In contrast, for hard-to-verify problems, those who were not able to solve the problem will not be easily convinced when presented with a correct solution. Thus, we hypothesize that the group would be far less likely to converge to the correct solution, and increasing the size of the group might actually be detrimental to the ability of the group to converge to the correct solution of hard-to-verify problems.

To test these hypotheses, we ran several experiments using the MTurk platform. We generated different instances of an NPC problem (Vertex Cover) and a PSC problem (Generalized Tic-Tac-Toe). Our empirical methodology simulated a (non-interacting) group problem solving scenario in which group members attempt to solve problems as individuals and subsequently vote on a collective choice from the solutions proposed by the group. Specifically, workers were assigned to one of two NPC or PSC problem instances, asked to solve the problem instance individually, and subsequently asked to recognize correct and incorrect solutions to the same problem that were provided for them from a predefined set of solutions.

Our results revealed that workers varied widely in their ability to solve the different problem types as individuals. Furthermore, in our experiments workers were significantly more likely to recognize correct and incorrect solutions for NPC problems than for PSC problems, supporting our hypothesis. Specifically, workers that failed to solve problems as individuals were more likely to recognize correct solutions when they were presented to them in the NPC condition than in the PSC condition. The significance of this study is in empirically showing the correlation between the ability of workers to verify solutions to problems and their associated computational complexity class. The designers of analytical problems commonly outsource the tasks to the crowd and aggregate the results. Our findings suggest that majority voting provides a sound mechanism for reaching good solutions to easy-to-verify problems, but this mechanism may not suffice for hard-to-verify problems.

The remainder of this paper is organized as follows. First, we formally define our hypotheses. Second, we provide a more detailed explanation of the computational complexity classes and of the problems chosen for our experiments. We then describe the tasks that the subjects completed, the experimental procedures and the measures used to test our hypotheses. Next, we describe the results of our study. Finally, we discuss of the implications of our work to human computation and group decision-making research and directions for future work.

Related Work

Our study relates to prior work in the human computation and group decision-making literatures which we will review in turn. A common approach to solve problems using the crowd employs voting and other social choice mechanisms to aggregate solutions to a collective choice, relying on the assumption that this choice is better on average than the solutions submitted by the individual group members. Some of these approaches aggregate solutions automatically without actively involving the crowd in the aggregation process. For example, the Galaxy Zoo project chooses the most common response to classifying Galaxies (Lintott et al. 2008), while other approaches used agreement mechanisms in games (Law and Von Ahn 2009; Von Ahn and Dabbish 2008). Steyvers et al. (2012) proposed methods for aggregating individual solutions of PSC problems, the Minimal Spanning Tree and Traveling Salesman Problem by combining the local solutions of participants using the majority method. Recent studies have incorporated various AI approaches to increase the efficiency and quality of the aggregation process. For example, Kamar et al. (2012) have used an MDP formalism to decide whether it is beneficial to hire additional workers for a consensus task and predict group consensus using machine learning. Zho et al. (2012) use a minimal entropy approach to estimate the probability distribution over the ground truth in consensus tasks when each worker solve multiple problem instances. Lin et al. (2012) use a POMDP-based controller to decide what tasks to post in order to achieve efficient aggregation of crowd responses.

An alternative to the automatic aggregation approach is to harness the crowd itself to vote for the best solution. For example, workers vote for their favorite biological structures in EteRNA (Savage 2012), and on text edits suggested by other workers (Bernstein et al. 2010). The Turkit mechanism (Little et al. 2009) provides a framework for iterative construction of solutions and voting. Finally, Mao et al. (2013) combined crowd voting with social choice methods and have compared the performance of different voting mechanisms for aggregating people's ranking of solutions to optimization problems. Our study suggests that using the crowd to vote for a solution is a suitable approach for solving NPC problems, but not for solving PSC problems.

Our study also relates to the ongoing debate in the literature about the benefits and pitfalls of collective decision-making. On the one hand, groups have been shown to outperform individuals in the canonical study by Yetton and Bottger (1983) who found that interacting groups outperform nominal (non-interacting) groups when trying to solve analytical problems such as ranking items to take on a moon expedition task in order of importance. On the other hand, some group dynamics may seriously inhibit its overall performance, as shown in multiple studies documenting effects such as polarization, procrastination and groupthink. Most of these studies have investigated the effects of the communication and information exchange protocols on group behavior. A notable exception is the work by Laughlin et al. (1986) who have shown that for the type of analytical problems we use in this paper, the number of group members that is necessary and sufficient to solve a problem is inversely proportional to the extent to which solutions can be recognized by group members (deemed the "problem demonstrability", i.e. the degree to which it is easy to explain the solution to a problem). According to Laughlin et al. (1986), a necessary criterion for demonstrability is that group members must be able to recognize a correct solution when it is proposed by others. Our work formalizes the relationship between the computational complexity class of a problem, and this important condition of demonstrability.

Methodology

NPC and PSC problems (Garey and Johnson 1979) are both computationally hard (all known algorithms for both computational classes have exponential time complexity), but they differ in the complexity of solution verification. NPC problems can be verified in polynomial time, while PSC problems, for practical purposes, can only be verified in exponential time. This serves as the basis for our hypotheses, which we state after defining the following: We refer to participants who did not solve correctly an analytical problem as *non-solvers*. We say a participant has *recognized a solution* if the participant was able to correctly classify the solution as correct or incorrect. We say a group has *converged to the correct solution* if the majority of participants in the group are able to recognize a solution when it is presented to them.

Our hypotheses can be stated as follows:

- Non-solvers would be more likely to recognize solutions for NPC problems than for PSC problems. Formally, we assert that for non-solvers (NS), the probability of recognizing the solution (VCS) in the NPC condition is higher than in the PSC condition:

$$P_{NPC}(VCS \mid NS) > P_{PSC}(VCS \mid NS)$$

- For NPC problems, convergence to the correct solution grows quickly with the size of the group because group members would be able to recognize a solution that is presented to them. Thus, the existence of a single solver is sufficient for the group to converge to the correct solution. In contrast, increasing group size is not expected to facilitate convergence in PSC problems, because the probability of recognizing a solution by non-solvers is small.

Problem Instances

We now turn to the question of which problem instances to generate from these two broad computational complexity classes. Examples of canonical NPC problem instances include the knapsack problem, the graph clique problem, graph coloring problems, the graph vertex-cover problem, and Boolean satisfiability problems. Examples of canonical PSC problems include playing generalized chess, verifying quantified Boolean formulas (QBF), and trying to form a row of k tokens in a two player, NxM board game [generalized Tic-Tac-Toe]. We chose problems for our study that meet the following conditions: (1) Do not require prior knowledge from subjects; (2) Can be explained reasonably within a few minutes; (3) Can be easily visualized.

For the NPC problem condition, we generated different instances of the Vertex Cover (VC) problem, which required participants to find a vertex cover of a given size for a given graph.[2] To examine the performance on PSC problems, we generated different instances of a Generalized Tic-Tac-Toe (GTTT) game, which required finding a move for an "O" player that will force a win within a given number of turns.[3]

We generated two instances of each problem type of varying difficulty levels, as determined by various parameters of the problem. For VC problems, we generated two 20-node graphs with a minimal vertex cover of size 12, as shown in Figure 1. We expected the first vertex cover problem instance to be easier for people to solve than the second instance, because its associated graph does not include crossing edges. For GTTT problems, we generated boards of varying sizes and configurations and required participants to choose the beginning move that would force a win for a given player. The two boards are shown in Figure 2. The first, harder, GTTT1 instance, which included a $10X10$ board, required the solver to force a win in 4 turns: a player wins by occupying 5 squares in a row (vertically, horizontally or diagonally). The second instance, which included a $6X6$ board, required the solver to force a win in 4 turns, where winning requires occupying 4 squares in a row.

[2]Note that finding a minimal vertex cover is not NPC, as verifying as solution requires checking that there does not exist a smaller vertex cover.

[3]Note that although GTTT is likely to be more familiar to participants than VC, this does not impede our study. Indeed, this fact may lead participants to solve GTTT problems more easily, making our hypothesis harder to support.

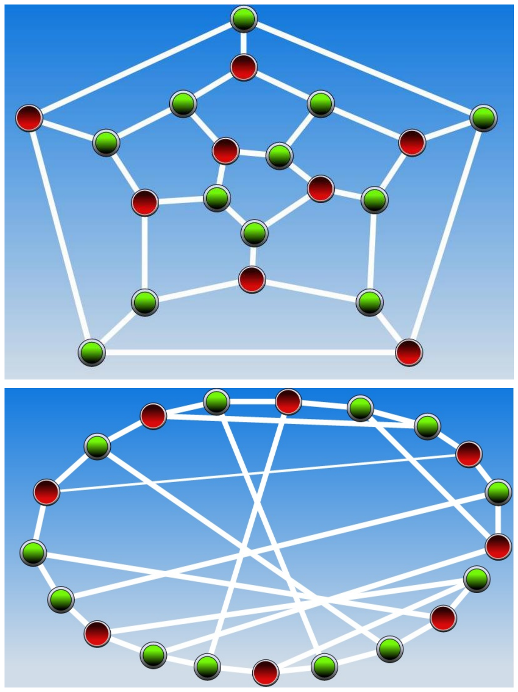

Figure 1: Two NPC vertex cover problems. Participants need to choose a set of 12 nodes, such that all edges are covered. The top graph (VC2) shows a correct solution which includes 12 nodes and covers all edges. The bottom graph (VC1) shows an incorrect solution, as one of the edges is not covered.

Experimental Design

The study followed a between-subject design, with the problem type (NPC or PSC) as the independent variable. We measured the proportion of participants who voted correctly for the proposed solution out of those participants who were not able to solve the problem on their own (i.e. non-solvers), in addition to the solution and verification time.

Tasks

Participants were presented with one of the problem instances in their respective conditions. For VC problem instances, the given instructions were: "Your task is to find a vertex cover of size 12 in the graph shown below. Your solution will be the set of vertices you chose (the vertices colored green when you click the continue button)." For GTTT problem instances, the given instructions were: "In the given configuration the O player can force a win in 4 turns even when the X player responds optimally at each turn. Find O's winning move." In all conditions, no time limit was imposed for solving the problem.

We developed GUIs for visualizing both problem types and for providing support for participants when reasoning about the problem instances. When solving VC problems,

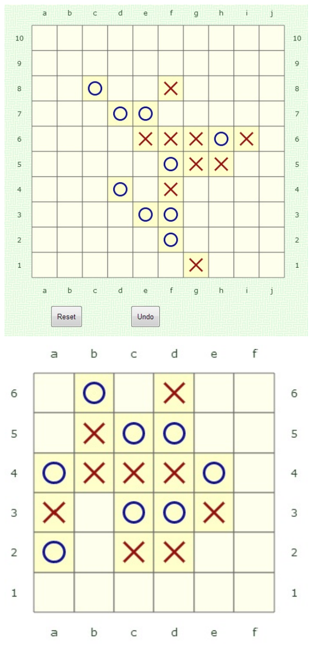

Figure 2: Two PSC General-Tic-Tac-Toe problems. In GTTT1 (top), participants were required to force a win for the O player in 4 turns, where winning requires occupying 5 squares. In GTTT2 (bottom), participants were required to force a win for the O player in 4 turns, where winning requires occupying 4 squares in a row.

participants could select vertices which would subsequently change their color and outline their associated edges with a thick outline. Selecting the vertex again would "reset" the color and the edge display back to normal. For example, the top (green) vertex in Figure 1 has been selected by the participant. When solving GTTT problems, participants could simulate moves by selecting a board position and adding an appropriate "X" or "O" icon, as well as "undoing" their moves to allow backtracking (using the "undo" and "reset" buttons shown for the top board in Figure 2).

Upon submitting their answers, participants were asked to explain why their solution was correct, using free language, and describe their strategy for solving the problem. Finally, participants were presented with three possible solutions to the problem they solved: the participant's own solution, and pre-generated correct solution and a pre-generated incorrect solution. These pre-generated solutions represented common wrong solutions that were submitted by participants (as determined in a pilot study).

In the case of VC problems, proposed solutions were presented by visualizing the solution on the graph. The correct solution showed a graph with exactly 12 green vertices that covered all edges. Incorrect solutions either included one more vertex than required, or a solution that had the correct number of vertices, but omitted an edge (the selection of wrong solution was randomized). In the case of GTTT problems, a solution was presented simply by showing the position chosen for the "O" player next move, for example "f3" for the correct solution in one of the conditions. Incorrect solutions either suggested a board location that is included in the sequence of winning moves, but is not the *first move*, or a move that was not on the winning path, but was close to the winning move (again, the selection of wrong solution was randomized). To assist with verification of the problem, the GTTT interface was presented with each solution, and participants could use it again (i.e. simulate plays on the board), similarly to its use in the individual problem solving stage.

Participants were asked to specify whether each solution was correct or incorrect. The ordering of the presented solution was counter-balanced in the experiments. The instructions were: "Next you will be shown three solutions - your own solution, and two solutions proposed by other people. Please examine all solutions (yours and the two proposed solutions); for each solution, vote whether you think it is correct or incorrect.", and then, for each presented solution, the following instructions were given: "Please examine the proposed solution below and vote whether you think it is correct."

We controlled for (the self-reported) gender, age and education; However, we did not find any significant effects for either of these, and therefore do not include them in our analysis. Participants recruited using MTurk were paid a base rate of 70 cents, and were further incentivized with bonuses for *solving* the problem correctly (by an additional 35 cents), and for *voting* correctly on the presented solutions (an additional 35 cents for those who voted correctly on all presented solutions).

Participants

Participants were recruited using MTurk and randomly assigned to one of NPC or PSC problem conditions. We restricted participation to participants from the US. In addition, for two of the instances (one VC and one GTTT), we recruited an additional group of undergraduate students. Since the performance of the student population was similar to that of participants recruited from MTurk, we aggregate their results and report them together. After answering a short demographic questionnaire, participants were presented with a tutorial detailing their respective problem and the appropriate use of the GUI. To verify that participants understood the problem and the use of the GUI, participation was contingent on successfully passing a quiz, which included questions regarding a simple instance of the problem.

A total of 200 participants completed the study. Gender distribution was close to uniform (99 female participants and 101 male participants). The mean age of participants was 30.4 (median 26). One participant completed less than high-school education, 67 participants completed high-school, 116 participants had college education and 16 had a graduate degree. We did not find significant effects for any of the different demographic variables.

Results

We define $P(Solve)$ as the probability that an individual participant solves the problem, computed as the proportion of participants who solved the problem out of the total

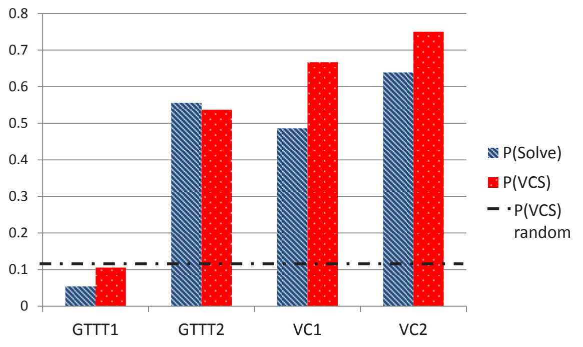

Figure 3: Probability of solving the problem correctly individually ($P(Solve)$), and the probability of recognizing solutions ($P(VCS)$), by problem instance. The dashed line shows the probability for a random verifier to correctly recognize all solutions (0.125).

number of participants. Figure 3 shows $P(Solve)$ (in blue stripped bars) as well as $P(VCS)$ (the probability of recognizing the solution, in red dotted bars) for each problem instance. We also include a baseline $P(VCS) = (\frac{1}{2})^3$ which is the expected probability for a random verifier to recognize all solutions.

As shown by the figure, GTTT1 was the hardest problem for participants to solve (it exhibited the lowest $P(Solve)$ measure), while GTTT2, VC1 and VC2 were more similar ($P(Solve)$ measured 0.46, 0.56 and 0.64, respectively). A positive difference between $P(Solve)$ and $P(VCS)$ indicates that some non-solvers were able to recognize solutions. We observed such positive difference for NPC problems. Interestingly, for the GTTT2 problem, this difference was actually negative, meaning that some solvers were not able to recognize solutions. For the GTTT1 problem, this difference was positive. However, $P(Solve)$ was extremely low for this problem, such that the improvement could also be achieved by random guessing.

Table 1 summarizes the results of the study for *non-solvers*. The results support the first hypothesis, in that the probability that non-solvers recognized solutions ($P(VCS \mid NS)$) was significantly higher for VC1 and VC2 than for GTTT1 and GTTT2. This difference was statistically significant for each problem pair of different complexity classes ($p < 0.05$ using χ^2 proportion test).

Contrary to our expectation, $P(VCS \mid NS)$ was relatively low for both NPC problems (0.46 and 0.54 for VC1 and VC2, respectively) [4]. That is, they were more likely to mis-classify their own incorrect solution as correct than to mis-classify other solutions. This can be seen in the table, in that $P(VCS \mid OWN, NS) < P(VCS \mid OTHRS, NS)$ for both VC1 and VC2 problems. For example, for VC1, $P(VCS \mid OWN, NS) = 0.57$ while $P(VCS \mid OTHRS, NS) = 0.76$. We did not observe this effect for PSC problem instances.

The difference in verification difficulty between NPC and PSC problems is also evident when comparing solution and verification times for GTTT and VC instances. Attempting to solve GTTT2 took less time on average (Mean = 498 seconds, STD = 403) compared to VC1 (Mean = 1205 seconds, STD = 4249) and VC2 (Mean = 585 seconds, STD = 426). Verification time, however, was higher for GTTT2 (Mean = 59.8 seconds, STD = 58.3) than for VC1 (Mean = 47.5 seconds, STD = 24.6) VC2 (Mean = 41.2, STD = 19.1). Note that the variance in verification time was significantly lower for the VC problem instances. Both individual solution times (Mean = 2365 seconds, STD = 11350) and verification times (Mean = 77.7 seconds, STD = 155.7) were longest for participants solving GTTT1.

We now turn to a qualitative analysis of the responses we collected. In addition to submitting their own solution and voting on the correctness of presented solutions, participants were asked to provide free text explanations justifying their solution and voting decisions. More specifically, after solving the problem, participants were instructed to provide an explanation as follows: “Please explain your solution, be as detailed as possible such that other people will be convinced that your solution is correct.” In accordance with our hypothesis, we expected explanations of solutions to PSC problems to be more complex than those provided for NPC problems.

As expected, explanations for the vertex cover problem tended to be short and provided a “visually justifiable” argument. A typical response explaining a solution of NPC problems (taken from VC2) was: *“The solution has exactly 12 vertices and all the edges are covered. ”*. Others explicitly noted the direct visual verifiability of the problem. For example, for the VC1 problem one participant explained: *“12 vertices are colored green and all of the lines have turned dark.”* Finally, some of the participants chose to include their solution strategy in their explanation: *“After several unsuccessful tries, I decided to alternate, working from outside to inside, choosing to highlight corners, then highlight non-corner nodes in the next layer toward the inside, etc. At this point, I got close. However, I ended up with 13 highlighted nodes. By trial and error, I simple selected/deselected interior nodes, until I found one that could be unhighlighted without deselecting a vertex.”*

For the GTTT problems, explanations varied in their level of detail. Some explanations were short and did not provide much evidence as to the correctness of the solution: for example, for the GTTT2 problem, one explanation was: *“Instead of being on the defense and first blocking player x’s chances player o traps player x into blocking their moves allowing them to secure places f3, f4 and f5”*, or *“Obtain 5 diagonally up to b6, create alternate pattern along column d for a win.”* However, others provided elaborate explanations which could help verify the correctness of their solution, such as *“ In order for O player to win in 4 moves, they must be willing to lose two options to X Player. First, they will place an O at f3. This will force X player to play c6. Then, O player HAS to play f5. This forces X player to once again concede a turn blocking a potential win at e5. This leaves O player with an open spot at f4, which is then played; leaving an open vertical line. Now; no matter what X player’s next move is, be it f2 or f6, O player just has to finish the line with the unfilled option and win the game.”*

[4] Yet it was still significantly higher than random guessing.

	GTTT1	GTTT2	VC1	VC2
$P(VCS \mid NS)$	0.08	0.17	0.46	0.54
$P(VCS \mid OTHRS, NS)$	0.22	0.38	0.76	0.77
$P(VCS \mid OWN, NS)$	0.22	0.38	0.57	0.54
$\#(NS)$	36	24	37	13

Table 1: Performance results of "non-solvers", showing the probability of correctly verifying all solutions presented ($P(VCS \mid NS)$), the probability of correctly verifying other solutions (i.e. the two solutions not provided by the participant, $P(VCS \mid OTHRS, NS)$), and the probability of correctly verifying the participant's own solution ($P(VCS \mid OWN, NS)$).

Discussion

The results presented in the previous section support our hypothesis that non-solvers were significantly more likely to recognize solutions in the NPC condition than in the PSC condition. This result was consistent across problem instances of similar difficulties as measured by $P(solve)$. Specifically, the problems GTTT2, VC1 and VC2 had similar difficulty measures, but $P(VCS \mid NS)$ was higher for VC1 and VC2 than for GTTT2. We conjecture that it is the "verifiability" of a problem ($P(VCS)$), rather than the difficulty for individual solver ($P(Solve)$), that affects the ability of the group to converge to the correct solutions: When $P(VCS) >> 0.5$ (as was the case for the NPC problem instances), the likelihood that the group will converge (according to the majority vote) reduces to the probability that at least one of the participants solves the problem, given by $(1 - (1 - P(solve))^n)$, where n is the size of the group. This probability grows quickly with the group size, meaning that such problems are especially suitable for crowdsourcing. For the PSC problems we studied, group members had much more difficulty to recognize a correct solution, even when one was provided for them. We thus contend that for PSC problems, increasing the group size may not increase the likelihood of convergence to the correct solution.

We note that our study on the effect of verification complexity on group performance is preliminary, and further studies are required to validate our hypothesis and generalize our results. One limitation of the study is that it examined only one problem type of each computational complexity class, namely VC and GTTT, and only two instances of each. These problems differ not just in their computational complexity class, but also in other dimensions that may affect group performance and participants' ability to recognize correct solutions, such as the interface used and type of human reasoning required to solve and verify the problem. With regards to interface and visualization, in order to generalize our results and verify that the differences in performance were not due to the interface used, we are currently developing experiments with problems of different computational complexity classes that share the same graph visualization. With regards to the reasoning needed to solve and verify the problem, While we believe the type of reasoning is inherently related to the computational complexity classes – for example, to verify VC a person needs to look at all of the edges and nodes, which is linear, while to verify GTTT one needs to reason about the next possible moves of both players, which is exponential – further experiments with additional problem types are required to confirm that the difference in group performance is due to the computational complexity class, and in particular, due to verification complexity. Finally, more experiments with additional problem instances of each problems type (varying in their size and complexity) are needed in order to quantify the relationship between the computational complexity class and group performance.

We now turn to discuss another phenomenon revealed by our study. As described in the results section, non-solvers of NPC problems were biased to mis-classify their own solution as correct, when in fact it was wrong. A possible explanation for this finding is that participants did not exert sufficient cognitive effort when validating the solution they themselves submitted. To evaluate this possible explanation, we measured the average time for verifying participants own solutions and others' solutions in the NPC problem conditions. We found that the verification time for participants own solution was slightly shorter (between 5-10 seconds) than for others' solutions. However, this difference is reasonable given that they were already familiar with their own solution and does not seem to suggest that participants did not make the effort to validate their own solution, as they still spent significant time verifying it (over 35 seconds on average). Interestingly, this phenomenon did not occur for PSC problem instances. The apparent lack of this bias in the PSC problem may be attributed to the verification difficulty of these problems, making it much harder for participants to recognize solutions, regardless of whether the solution was their own.

We conclude with noting that the variance of participants verification time was significantly higher for PSC than for NPC problems. This suggests that people's verification behavior is more uniform for NPC problems than for PSC problems and shows another aspect of the easy-to-verify nature of NPC problems.

Conclusions and Future Work

This paper provides preliminary evidence that the computation complexity class of a problem, and in particular its verification complexity, affects the ability of a crowd to reach a correct solution to that problem.

We conducted a series of experiments testing the problem solving capabilities of individuals and their ability to recognize correct and incorrect solutions, when posed with problems belonging to NP-Complete and PSPACE-Complete problems. The results showed that participants that failed to solve problems as individuals were more likely to recognize correct solutions when they were presented to them in the NPC condition than in the PSC condition. Based on these results, we conjecture that the likelihood that the group converges to the correct solution grows quickly with the group size for NP-Complete problems but not for PS-Complete problem. We thus contend that NP-Complete problems are especially suitable for crowdsourcing.

This work is a first step towards exploring group problem solving in a formal way, relating group performance with computational complexity problem classes. Future research directions include extending these initial experiments to include more problem instances and additional NPC and PSC problems of different types, exploring the effect of adding explanations to proposed solutions, and evaluating performance in interacting group scenarios. Investigating these directions further could have implications for designing better mechanisms to support group problem solving, as well as for improving the design of crowdsourcing applications that require complex problem solving.

Acknowledgements

Yuval Shahar was partially supported by EU award FP7-ICT-2011-7, #287811 "MobiGuide"). Ya'akov Gal was supported in part by grants FP7-ICT-2011-9 #600854 and Marie Curie #268362. We thank Uri Gur-Arie and Lizet Pinya for developing the visualization tool for VC and Steven Komarov for developing the visualization tool for GTTT. We thank Krzysztof Gajos and Steven Komarov for helpful discussions.

References

Bernstein, M. S.; Little, G.; Miller, R. C.; Hartmann, B.; Ackerman, M. S.; Karger, D. R.; Crowell, D.; and Panovich, K. 2010. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, 313–322. ACM.

Garey, M. R., and Johnson, D. S. 1979. *Computers and intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Kamar, E.; Hacker, S.; and Horvitz, E. 2012. Combining human and machine intelligence in large-scale crowdsourcing. In *Proc. of International Conference on Autonomous Agents and Multiagent Systems*.

Laughlin, P., and Ellis, A. 1986. Demonstrability and social combination processes on mathematical intellective tasks. *Journal of Experimental Social Psychology* 22(3):177–189.

Law, E., and Von Ahn, L. 2009. Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1197–1206. ACM.

Lin, C. H.; Weld, D. S.; et al. 2012. Dynamically switching between synergistic workflows for crowdsourcing. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Lintott, C. J.; Schawinski, K.; Slosar, A.; Land, K.; Bamford, S.; Thomas, D.; Raddick, M. J.; Nichol, R. C.; Szalay, A.; Andreescu, D.; et al. 2008. Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society* 389(3):1179–1189.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2009. Turkit: tools for iterative tasks on mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, 29–30.

Mao, A.; Procaccia, A. D.; and Chen, Y. 2013. Better human computation through principled voting.

Savage, N. 2012. Gaining wisdom from crowds. *Communications of the ACM* 55(3):13–15.

Von Ahn, L., and Dabbish, L. 2008. Designing games with a purpose. *Communications of the ACM* 51(8):58–67.

Wasson, T. 2009. *Evaluation of the effects of solution demonstrability on group performance outcomes*. Hofstra University.

Yetton, P., and Bottger, P. 1983. The relationships among group size, member ability, social decision schemes, and performance. *Organizational Behavior and Human Performance* 32(2):145–159.

Yi, S.; Steyvers, M.; Lee, M.; and Dry, M. 2012. The wisdom of the crowd in combinatorial problems. *Cognitive science*.

Zhang, H.; Law, E.; Miller, R.; Gajos, K.; Parkes, D.; and Horvitz, E. 2012. Human computation tasks with global constraints. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, 217–226. ACM.

Zhou, D.; Platt, J.; Basu, S.; and Mao, Y. 2012. Learning from the wisdom of crowds by minimax entropy. In *Advances in Neural Information Processing Systems 25*, 2204–2212.