Human Computation and Multiagent Systems: An Algorithmic Perspective

Andrew Mao
Harvard SEAS
mao@seas.harvard.edu

David C. Parkes
Harvard SEAS
parkes@eecs.harvard.edu

Ariel D. Procaccia
Harvard SEAS
arielpro@seas.harvard.edu

Haoqi Zhang
Harvard SEAS
hq@eecs.harvard.edu

"Best. HIT. Ever."

Anonymous Amazon Mechanical Turk worker reflecting on our human intelligence task

Abstract

Human computation systems such as online task markets provide platforms that allow us to harness the abilities of many human problem solvers to accomplish a variety of tasks. We argue that such platforms share some of the features of multiagent systems, and ask whether algorithms originally designed for multiagent systems can be leveraged for coordinating the problem solving process among human computers. We study this question in the context of the graph coloring problem where each human controls the color of one vertex, and compare the performance of humans using adaptations of two well-known distributed constraint satisfaction algorithms to the performance of humans that were asked to simply color the graph. We find that people provided with algorithmic instructions achieve significantly better performance than people who are left to their own devices.

1. Introduction

Human computation is a recently popular computational paradigm that outsources problem solving steps that are difficult for computers to many *human computers* connected by the Internet. Over the last few years, there have been two prominent streams to advance human computation. The first stream focuses on online applications that are often described as *games with a purpose* (von Ahn and Dabbish 2008), which are craftily designed so that people can contribute useful computations as a by-product of playing an enjoyable game. A well-known example is the *ESP game*, in which two players who cannot communicate see the same image and must agree on a suitable label, hence leading players to identify relevant labels. Another well-known example is *Foldit* (Cooper et al. 2010), where players collaborate in folding proteins into their stable shape.

The second stream is represented by *online labor markets*, where the paradigmatic example is *Amazon Mechanical Turk* (AMT).[1] AMT has two types of participants: requesters and workers. The requesters post *human intelligence tasks (HIT)*, that specify a task to be carried out, requirements for workers, and a payment for successful completion. The workers can view the available HITs and select the ones that best fit their objectives (usually making money, having fun, or both). Typical HITs are quite tedious, and often involve data entry or generating training data for various algorithms; common examples include labeling images, transcribing audio, and finding information on websites. Both the name and the amusing slogan of AMT, "artificial artificial intelligence," allude to the fact that humans are solving problems that still challenge the best AI systems today.

In general, online labor markets differ from games with a purpose in that they are significantly more generic and versatile. In fact, one can view platforms like AMT as a new type of multiagent system, where software agents are replaced by human computers.[2] Much like classic non-cooperative multiagent systems, these platforms are distributed and the agents (in an ironic reversal of roles) are autonomous, heterogeneous, and self-interested (at least to some extent). To date, most tasks on AMT are simple and request *independent* effort, and few tasks aim to coordinate problem solving among many actors within the system.

In this paper, we look at human computation through an algorithmic lens, and seek to leverage the connection between human computation and multiagent systems. Our main research question is:

> *Can successful algorithmic ideas from multiagent systems be adapted to effectively coordinate human computers to contribute jointly to a problem solving effort?*

There are obvious difficulties and challenges. First, a human agent may not understand a complicated algo-

[1] `http://www.mturk.com`

[2] We might also expect hybrid systems that aim to harness the respective abilities of both software agents and human computers.

rithm, and additionally needs sufficient context to effectively perform its role in the computations. In addition, information must be conveyed, through good user interface design, in an way that is intuitive for humans to understand and process. Finally, humans make errors, and unless a computer algorithm is somewhat robust to such errors, the performance of humans attempting to execute an algorithm can be poor. However, humans also have the ability to use instincts and make decisions in ways that a straightforward algorithm cannot, and a well-designed system should allow for this potential.

Overview of methodology and results. We focus on the distributed graph coloring problem, where agents must color the vertices of a graph such that no two adjacent vertices share the same color. Our focus on graph coloring is not because we expect human computation approaches to provide faster solutions than a computer. Rather, we view graph coloring as a paradigmatic computational problem, and one from which we hope insights will carry over to other problems. An advantage of the graph coloring problem is that non-algorithmic, human computation has been previously studied (Kearns, Suri, and Montfort 2006).

For our study, we conducted a series of 90 experiments on AMT. In each experiment, fifteen people are instructed to color a fifteen node graph. Each person can view the entire graph but controls only one vertex, and can change its color at any time. To our knowledge, this is the first time experiments with *real-time* user interaction were carried out in an online labor market. Our methodology is detailed in Section 3, and may be of independent interest to the reader.

We test two adaptations of well-known *distributed constraint satisfaction* algorithms (Yokoo and Hirayama 2000): *distributed breakout* and *weak commitment search*. In each case, we provide normative but not fully prescriptive advice about how to participate in the task, in the spirit of the underlying algorithm. This approach centers on conveying algorithmic ideas in an intuitive way, while also allowing people room to differ from the precise mechanistic approach.

Our results show that instructions based on both algorithms yield a striking improvement over performance when no algorithm is provided, both in terms of completion time and number of moves required to color the graph. A statistical analysis with respect to completion time shows that the improvement yielded by the algorithms is highly statistically significant.

Related work. Kearns et al. (2006) study graph coloring with human subjects. They focus on the relation between graph topology and performance, and crucially do not provide algorithms. They find that graphs based on the preferential attachment model of social networks are harder to color than graphs with a cyclical structure, and "small worlds" graphs make the problem especially easy. Methodologically, Kearns et al. carried out their experiments in a laboratory rather than online.

There is a body of work in AI (see, e.g., (Anderson et al. 2000)) on *interactive optimization*, which uses humans to steer the search process in optimization problems. For example, humans can invoke, halt, and specify the scope of procedures, backtrack to previous solutions, and control the amount of effort spent on different subproblems. This is fundamentally different from our work in that humans do not play a role in making "base-level" decisions in solving the problem, and in that there is no need to coordinate between multiple humans.

Shahaf and Horvitz (2010) investigate *generalized task markets*, a generalization of online labor markets (which they refer to as "task markets") which admit both human and machine problem solvers. The challenge lies in optimizing the flow of tasks to humans and computers, based on their competency, availability, and price. Their algorithmic results focus on the problem of optimally assigning tasks to coalitions of agents in a way that respects a precedence order. In a similar vein, Dai et al. (2010) study the use of decision-theoretic techniques to control an *iterative* workflow that make decisions on requesting additional refinements based on costs and inferred work quality. In contrast with these works, our focus is on distributed algorithms for coordination between humans.

2. Distributed Constraint Satisfaction Algorithms

In a constraint satisfaction problem (CSP), the goal is to find an assignment of values to variables that is consistent with constraints over the variables. CSPs have a prominent place and rich history in AI research. Formally, a CSP has a set of variables $x_1, \ldots, x_n$. The values of the variables are drawn from discrete domains $D_1, \ldots, D_n$. To represent a constraint we use a prohibited combinations of variable values, called a *nogood*. The graph coloring problem is a CSP for which there is a variable for each vertex, the domains are the colors, and there is a nogood for each pair of adjacent vertices and identical colors.

In a distributed CSP (Yokoo and Hirayama 2000) the variables and constraints are distributed among agents. Agents communicate by sending messages, whose delivery can be delayed (but messages are received in the order in which they were transmitted). We assume below that each agent has exactly one variable and that all the constraints are binary, which is indeed the case in our experiments. For the purposes of this section we also assume that agents that participate in the same constraints, referred to as *neighboring agents*, can exchange messages.[3]

We focus on two algorithms for distributed CSPs. We give an informal description of both algorithms; the reader is referred to Yokoo and Hirayama (2000) for more details.

[3] Conveniently, in graph coloring agents that participate in the same constraint are actually neighbors in the graph.

Distributed breakout (DB). *Breakout* is a hill-climbing algorithm where the quality of the solution is iteratively improved by changing the value of a variable in a way that improves an evaluation function. A weight is defined for each violated constraint, and is initialized to one. The evaluation function is the total weight of violated constraints. When trapped in a local minimum, breakout increases the weight of all violated constraints by one. Eventually (possibly after several weight increments) this guarantees that the evaluation function of the current state becomes larger than that of states that are reachable by changing the value of one variable.

In the context of distributed CSPs it is not necessarily possible to change the value of the variable that most improves the evaluation function. Hence, under DB, neighboring agents exchange messages to determine who has the largest improvement, and only that agent changes its value. However, it may still be the case that two agents that are not neighbors change their values simultaneously.

Asynchronous weak commitment (AWC). At every stage of the algorithm's execution, agents are assigned priorities. Each agent is only obligated to choose a value that is consistent with the values of its higher priority neighbors. If an agent has at least one such value, the agent chooses a value that is consistent with its higher priority neighbors and minimizes the number of conflicts with lower priority neighbors. Finally, if an agent cannot choose a value that is consistent with its higher priority neighbors, it increases its priority value to one that is higher than all its neighbors.

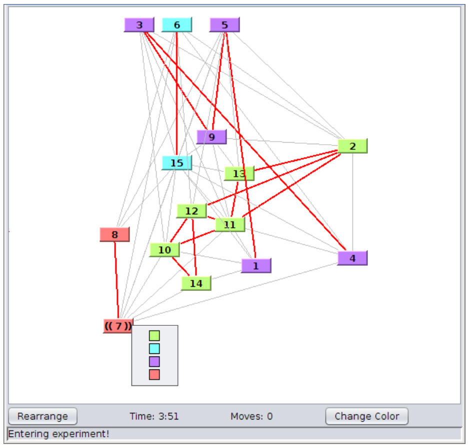
3. Methodology

We carried out experiments on AMT in three sessions. Each session focused on one of three graph coloring methods; each method was tested on the same five graphs with fifteen vertices, and for each graph we ran six experiments.

In each experiment, fifteen workers interacted in a real-time graph coloring game. Each worker could view the entire graph, and controlled the color of a single vertex; see Figure 1 for a screenshot. No communication between the workers was allowed. Workers were paid 20¢ for completing an experiment; an experiment is complete when the graph is colored or ten minutes have passed.

The three graph coloring methods we tested are *no algorithm* (NA), *distributed breakout* (DB), and *weak commitment search* (AWC). NA serves as a simple benchmark. When testing NA, the instructions workers were shown specified the goal of the game:

- The purpose of this task is to work with other Turkers to color blocks so that no two connected blocks are the same color.
- You control the color of one block in the graph, marked with (( )). The other blocks are controlled by other Turkers.



Entering experiment!

Figure 1: A screenshot of the graph coloring game. The worker controls the vertex labeled by 7.

- You can change the color of the block you control any time, as many times as you want, by clicking on the block or on the "change color" button.
- The game is won when no connected blocks have the same color. Connected blocks with the same color are joined by a thick red line.

When testing DB and AWC, the same general instructions also included the bullet "we suggest that you use the method described below when deciding when to change color and which color to change to." This was followed with a description of (the essence of) the algorithm. The language used was simple, informal, and designed to encourage workers to use their intuition.

The DB instructions are:

- When you click the "change color" button you will see a number next to each color.
- The larger the number, the closer to the goal choosing that color brings you. If the number is smaller than zero, choosing that color is bad.
- Even if you have a color with a positive number, maybe someone else has an even better move. You don't have to always pick the positive numbers. Try to think about the big picture and use your intuition.

The numbers mentioned in the instructions represent the evaluation function for the state obtained by choosing the corresponding color; these numbers are automatically computed.

The AWC instructions are:

- Each block is assigned a rank from 1 (highest) to 15 (lowest).
- If you have the same color as another block, lines to higher rank blocks are red, while lines to lower rank blocks are pink. Try to avoid having any red lines.
- If you have multiple colors that cause no red lines, try to reduce the number of pink lines.
- If you can't choose a color that doesn't cause any red lines, you may press the Rank Me Up button. This makes your rank higher than everyone you are connected to. Try to avoid pushing this button unless you have to.

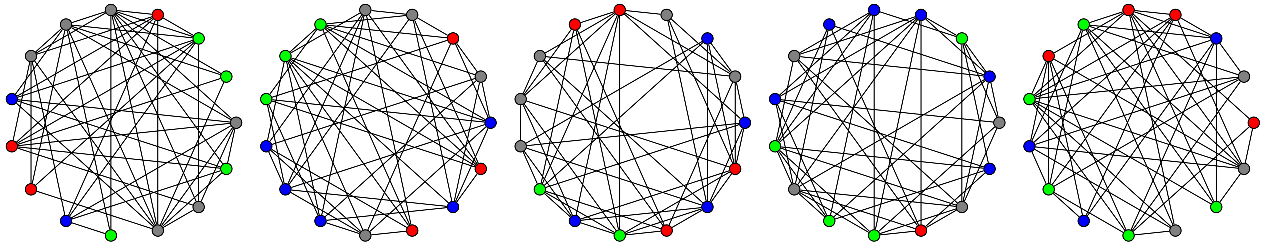

(a) Graph 1 (b) Graph 2 (c) Graph 3 (d) Graph 4 (e) Graph 5

Figure 2: The five graphs used in the experiments, along with their fixed assignment of colors to vertices.

The interface for testing AWC is identical to the interface for testing NA, except for the 'Rank Me Up' button and the differentiation between red and pink lines (in the NA interface all conflict edges are colored in red).

The five graphs were randomly generated as follows. To guarantee a chromatic number of at most four, i.e., that the graphs are colorable using at most four colors, the vertices are randomly partitioned into four subsets. Then, fifty edges are added between pairs of vertices that are chosen uniformly at random from different subsets. The connection between the number of edges and the difficulty of coloring a graph is well understood (see, e.g., (Cheeseman, Kanefsky, and Taylor 1991)), and the number of edges was chosen to make the task nontrivial.[4]

Each graph is associated with a fixed assignment of colors to vertices. In each experiment, the initial assignment of colors to vertices is a permutation of the fixed assignment associated with the graph. This allows for better control of the experiments, since the initial assignment of colors can have a great impact on the distance to a solution. The 15 workers for each experiment are randomly assigned to vertices. In addition, the spatial layout of the graph for each worker is generated randomly by an organic graph layout algorithm, with the option to choose a new layout at any time if overlapping edges are hard to see. We believe that together, these two randomizations make it essentially impossible for a worker to realize that he has already seen a given graph in a previous experiment. The five graphs and their fixed assignments of colors to vertices are illustrated in Figure 2.

Implementation details

The experiments reported in this paper are among the first to require multiple workers simultaneously working on the same task via AMT, and are (to the best of our knowledge) the first requiring real-time interaction. In order to simultaneously bring together multiple workers, we used methods that are inspired by the work of Mason and Suri (2010).

In the weeks leading up to the experiments we carried out numerous graph coloring games on AMT, involving graphs with either eight or fifteen vertices. The purpose of these games were twofold: first, to test the software, and second, to build a panel of experienced workers familiar with the game. When a team of workers completed a game (either successfully or unsuccessfully), the workers were offered the option of joining the panel. Ultimately, the panel included workers who opted in, as well as workers who played at least three games in the preliminary stage, for a total of 204 workers. Note that workers on the panel are not necessarily more successful than the average AMT worker; they have simply demonstrated a willingness to play a full graph coloring game and have some experience playing the game.

Before a scheduled experiment session, an email specifying the time of the experiment and providing a link to the appropriate HIT was sent to the workers on the panel. Workers who accept the HIT join a virtual lobby and wait for a game to begin. At any point, if the lobby contains at least fifteen workers, these workers are given the option to declare themselves ready for the experiment; the first fifteen workers to declare readiness then join a new experiment (and are removed from the lobby).

We monitored workers' activity during an experiment, to ensure that they were actually participating. When a worker does nothing (including mouse movements) for 45 seconds, an inactivity warning replaces the game on their screen. Workers were warned that being inactive for an unspecified amount of time would result in not being paid for the HIT. Fortunately, we did not find that worker inactivity was a significant issue. Only one NA experiment was clearly influenced by inactivity (in particular, one of the workers did not participate at all); this experiment did not complete in ten minutes. We consequently replaced this experiment with a different, valid one. Note that in our results, this replacement only decreased the difference between NA and each of DB and AWC.

4. Results

Table 1 shows the median completion time (in seconds) and number of moves for workers to color the graph using each of the three methods, grouped by graph.

[4] The average degree in our graphs is 6.66, whereas the most difficult average degree would be roughly 9.

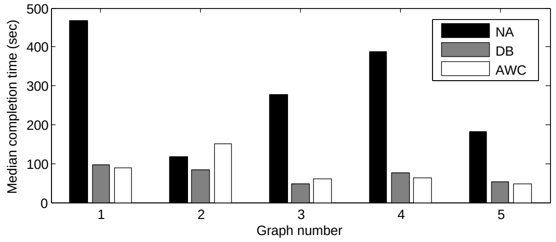


Figure 3: Median completion time.

(a) Median completion time (sec)

Method	Graph 1	Graph 2	Graph 3	Graph 4	Graph 5
NA	469	117.5	276	389	180.5
DB	97.5	83.5	47	75	52
AWC	88.5	152	61.5	63.5	47.5

(b) Median move count

Method	Graph 1	Graph 2	Graph 3	Graph 4	Graph 5
NA	220.5	62.5	115	233.5	73
DB	47.5	47.5	41	41.5	29.5
AWC	48.5	58.5	39	35	33.5

(c) Median adjusted move count

Method	Graph 1	Graph 2	Graph 3	Graph 4	Graph 5
NA	152	54	93.5	131	68
DB	45	45	38.5	38	28
AWC	42	52.5	34	32.5	32

Table 1: Median statistics for each graph.

With six experiments per method per graph, we give the average of the upper and lower medians. We consider medians rather than averages because some of the NA experiments did not complete after 10 minutes. The completion times are also visualized in Figure 3.

We see that uniformly across the five graphs, DB performs better than NA, and in most cases by a great margin. AWC is slightly worse than NA on Graph 2, but also clearly outperforms in the other four.

In addition to displaying the total number of moves, we display an adjusted move count computed by counting consecutive color changes by the same player as one move. During many NA experiments, we observed players signaling others with their color or randomly cycling through colors, or ‘thrashing’, when progress was not being made. The difference between the move count and the adjusted move count is much larger in the NA case, and makes a strong case that algorithmic guidance greatly reduced the necessity of thrashing behavior by players.

Statistical analysis

We analyze our data to test the statistical significance of the differences between completion time under NA versus each of DB and AWC. For this purpose we considered NA experiments that did not complete after ten minutes as completed after ten minutes, hence the differences are in fact even more significant.

We use Generalized Estimating Equations[5] to fit when the dependent variable is the completion time and the independent variable is the method. The algorithm estimates a linear model where the intercept as well as the coefficients (estimated with respect to using NA as the baseline method) are all highly statistically significant at $p < 0.001$. Using NA, the estimated completion time is 315.4 seconds (standard error 38.8). Using DB instead of NA, the estimated completion time is *reduced* by 234.5 seconds (standard error 36.1). Using AWC, the estimated *reduction* in running time compared to NA is 186.5 seconds (standard error 45.4).

To conclude, based on the data, we see a highly statistically significant effect that following DB or AWC instead of NA reduces the completion time.

Move analysis

We next analyze individual (adjusted) moves performed by workers, to see to what extent workers followed the algorithmic suggestions. We see that in 71.7% of all color changes in the 30 DB experiments the worker chose a color that maximizes the evaluation function. As a benchmark, we see that in the 30 NA experiments 56.6% of color changes maximize the DB evaluation function (computed after the fact using the logs). Similarly, 64.6% of all color changes in AWC experiments do not cause any conflicts with higher priority neighbors, whereas the value is 51.9% in NA experiments.[6] It is informative to focus on states where a worker did not have a color that eliminates all conflicts, but did in fact have a color that is consistent with higher priority neighbors; in these cases the difference is slightly larger, with 71.8% of AWC moves being consistent with higher priority neighbors versus 52.0% of NA moves. In the 30 AWC experiments workers used the ‘Rank me up’ button 75 times; 30 of these priority changes (40%) are in states where the worker indeed did not have a color that is consistent with higher priority neighbors.

We also ran simulations of DB and AWC on our five graphs, in order to compare the performance observed in experiments to perfect executions of these al-

[5] We have multiple experiments for each of five graphs, rather than generating a random graph for each experiment. While this enables the analysis presented above, it does violate the standard independence assumption of ordinary least squares (OLS) regression models. That is why for all the statistical analysis of the data we use *Generalized Estimating Equations*, an extension of generalized linear models (Nelder and Wedderburn 1972) that allows for the analysis of repeated measures or other correlated observations by adjusting the significance levels. We assume that measures from the same graph and method are independent, which gives the best fit, and use the identity link function and normal distribution.

[6] For the purpose of NA experiments, higher priority neighbors are considered to be vertices labeled by a smaller number.

gorithms.[7] In all cases the median of 10,000 simulations of the algorithm is smaller than the median adjusted move observed in our experiments, but in some cases the difference is small (e.g., for Graph 4 and AWC the difference is 4.5). Moreover, in several cases the minimum among six experiments is smaller than the median from simulation.

Robustness

One of the major shortcomings of AMT as an experimental platform, as opposed to laboratory experiments, is that it is difficult to control the population of workers participating in the experiments. This leads to a number of possible concerns that we discuss below.

A natural concern is that the same workers may participate in experiments involving different methods, and in particular use an algorithmic technique suggested for one method in an experiment focusing on a different method. However, the DB and AWC techniques require specific interface tweaks: scores for colors in DB, priorities and a "Rank me up" button in AWC. Therefore, we believe that we can safely dismiss this concern in our context.

A second concern is that there is a learning effect, in that workers improve their graph coloring skills over time, and therefore performance in earlier experiments would be inferior to later ones. However, we do not believe that this effect is significant, in particular because, as members of the panel, most workers who participated in the experiments took part in many graph coloring games beforehand (and indeed this was one of the insights behind the design of the panel).

Finally, the session of NA experiments was conducted in the morning (EST), whereas the DB and AWC sessions were conducted in the afternoon; the timing was arbitrary. It is known that most AMT workers hail from either the USA or India (Horton 2011). Because of the time difference, morning experiments attract a larger proportion of Indian workers than afternoon experiments, and this may bias the results. To verify that our results are robust to this effect we conducted a limited DB session in the morning, and were still able to observe performance that was significantly better than NA.

5. Discussion

The main message of this work is that people's performance greatly benefits from algorithmic instructions, and in particular that algorithms originally designed for multiagent systems can successfully coordinate between people. Although our experiments are specific to graph coloring, this problem is a paradigmatic CSP (which is NP-hard when the number of colors is at least three). One can easily think of problems that can be modeled as CSPs, where human computation is necessary, e.g., because some of the atomic operations can only be performed by humans. For example, think of constructing an itinerary for a trip; one would like to avoid scheduling two museums in succession. Such conflicts are difficult to specify in advance, but can be identified by a human looking at the itinerary.

Our results when using DB and AWC are comparable, with an advantage for DB. However, it is important to note that our DB implementation helps workers by automatically performing calculations when it updates the weights of edges. In contrast, AWC essentially does not perform any calculations and can be thought of as a pure coordination mechanism. In addition, AWC is a nontrivial algorithm and it is *a priori* unclear if people can benefit from it. For these reasons, we actually find our AWC results to be more striking than the DB results.

References

Anderson, D.; Anderson, E.; Lesh, N.; Marks, J.; Mirtich, B.; Ratajczak, D.; and Ryall, K. 2000. Human-guided simple search. In *Proceedings of the 17th AAAI Conference on Artificial Intelligence (AAAI)*, 209–216.

Cheeseman, P.; Kanefsky, B.; and Taylor, W. M. 1991. Where the really hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, 331–337.

Cooper, S.; Khatib, F.; Treuille, A.; Barbero, J.; Lee, J.; Beenen, M.; Leaver-Fay, A.; Baker, D.; and Popović, Z. 2010. Predicting protein structures with a multiplayer online game. *Nature* 466:756–760.

Dai, P.; Mausam; and Weld, D. S. 2010. Decision-theoretic control of crowd-sourced workflows. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 1168–1174.

Horton, J. J. 2011. The condition of the Turking class: Are online employers fair and honest? *Economic Letters*. Forthcoming.

Kearns, M.; Suri, S.; and Montfort, M. 2006. An experimental study of the coloring problem on human subject networks. *Science* 313:824–827.

Mason, W., and Suri, S. 2010. Conducting behavioral research on Amazon's Mechanical Turk. Manuscript.

Nelder, J., and Wedderburn, R. 1972. Generalized linear models. *Journal of the Royal Statistical Society Series A* 135(3):370–384.

Shahaf, D., and Horvitz, E. 2010. Generalized task markets for human and machine computation. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 986–993.

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

Yokoo, M., and Hirayama, K. 2000. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems* 3:185–207.

[7] At every step in an algorithm's simulation a random vertex is activated, and a move that is optimal according to the algorithm is performed, breaking ties uniformly at random.