Internet Mathematics Vol. 5, No. 4: 477–492

Atropos: A PSPACE-Complete Sperner Triangle Game

Kyle W. Burke and Shang-Hua Teng

Abstract. We create Atropos, a new two-player game based on Sperner's lemma. Our game has simple rules and several desirable properties. First, Atropos is always certain to have a winner. Second, the game is impartial, meaning that both players always can make the same moves. Third, like many other interesting games such as Hex and geography, we prove that deciding whether one can win a game of Atropos is a PSPACE-complete problem. We provide a web-based version of the game, playable at http://cs-people.bu.edu/paithan/atropos/. In addition, we propose other games, also based on fixed-point theorems.

1. Introduction

The relationship between computational complexity and game strategies has encouraged the development of both of these fields. Games, due to their enjoyable and competitive nature, create a breeding ground for analysis as strategies are discussed and revised. The ability to express strategies using computational complexity allows us to categorize them based on concrete classes of difficulty. Conversely, the ability to express complexity classes in terms of finding game strategies motivates the study of these classes.

Many two-player games can employ simple rules yet still resist simple methods to efficiently produce winning strategies. Games such as Geography, Hex, and Go all require PSPACE-complete capability to deduce these winning strategies in every case [Papadimitriou 94b, Reisch 81, Lichtenstein and Sipser 80]. For these games, the simplicity of the rules often masks the mathematical intricacies of the

1542-7951/08 $0.50 per page

underlying structure. In Hex, for instance, the existence of a winner has been shown to be equivalent [Gale 79] to Brouwer's fixed-point theorem [Brouwer 10].

Motivated by the equivalence of the fixed-point theorem to another result, namely Sperner's lemma [Sperner 28], we present a new two-player game: Atropos. As Eppstein argues, games with polynomial[1] strategies lose their ability to amuse once players learn the "trick" [Eppstein 06]. This concern might be amplified in our era, when we encourage the most talented human game players to compete against highly optimized machines. If polynomial strategies for a game exist, these machines can efficiently implement the strategies, and play becomes trivial. Thus, we continue by proving that Atropos is PSPACE-complete as our main result.

Furthermore, Atropos avoids one of the inelegant imbalances of games such as Hex in that it is not always to a player's advantage to make the next move. Indeed, in our game the final move results in a loss. Thus this game does not fall into the trap of clearly giving the first person to play any advantage. There is no need to incorporate an unnatural means to smooth this out (Hex uses the "pie rule" to negate this advantage).

Luckily, the rules for Atropos are very simple, and a player does not require any mathematical background to be a fierce contender.[2] Before we define the rules of the game, we discuss the Sperner triangle and the lemma that ensures that one of the players has won at the end of the game.

2. The Sperner Triangle

Sperner's triangle is simply a triangular array of nodes (see the left half of Figure 1), each colored in one of three colors with a simple boundary condition: each side of the outer triangle is assigned a different color, and nodes along that edge may not be given that color [Sperner 28]. Along each axis of the array, each node has two natural neighbors, aside from the boundary nodes, which may not have a second neighbor along some axes. Since the triangular array has three axes, each interior node of the triangle has exactly six neighbors.

Here, in lieu of colors, we use three different symbols: bars, shading, and filled, as demonstrated in Figure 2. We will often use the words *barring*, *shading*, and *filling* to describe the action of assigning the relative symbol to a node.

Using the triangle as our game board, the rest of our inspiration comes from the following brilliant result of Sperner [Sperner 28].

[1] That is, polynomial in the size of a description of the board.

[2] The first author often finds himself losing to friends and family members.

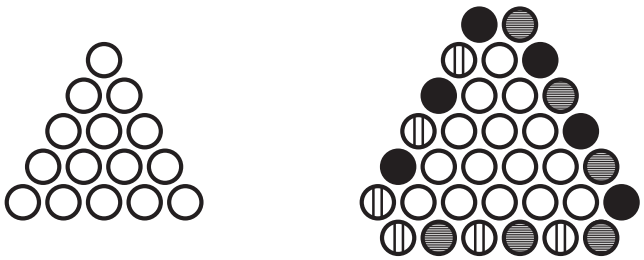

Figure 1. A triangular array of size 5 (left) next to our Sperner triangle game board, also of size 5, with added functional boundaries (right).

Figure 2. From left to right: A barred node, a shaded node, and a filled node.

Lemma 2.1. (Sperner's lemma.) *On any sized Sperner triangle, if all the nodes are colored, there will exist a triangle of three neighboring nodes, each of which has a different color. In fact, there will be an odd number of these triangles.*

3. Playing Games on the Sperner Triangle

Given the challenge of attempting to avoid tricolored triangles, natural games emerge in which players take turns coloring empty nodes on an initially uncolored Sperner triangle. In these games, a player loses when his or her play creates a tricolored triangle. Because of this, we refer to triangles with three colors as "bad" triangles. In addition, open circles at which any color played would result in a bad triangle are called "doomed" circles. Although playing at one of them is legal, it results immediately in a loss, and thus we often refer to them with the misnomer "unplayable." Also, we often speak of playable circles as not including those that are doomed. When we want to include doomed circles, we will use the term "open."

3.1. The Game Board

In order to elegantly enforce the border restrictions, we enhance the triangle by adding an alternating series of the other two colors to each side, as shown in the right-hand picture of Figure 1. Thus, a player playing the forbidden color along one of those sides immediately creates a bad triangle and loses the game.

Since it is often helpful to focus on one player in analyzing two-player games, we will often refer to one player as the hero and his opponent as the adversary.

3.2. Atropos Rules

The rules of Atropos are as follows:

1. On a turn, each player colors a circle on the triangle using one of the three colors.
2. The first player may choose any uncolored circle at which to play the first move.
3. If any circles adjacent to the last-colored circle are uncolored, the current player must choose one of them to color. Otherwise, the player may choose to play at any uncolored circle on the board.
4. The first player to color a circle that creates a tricolored triangle loses the game.

A playable version of Atropos is available at http://cs-people.bu.edu/paithan/atropos/.

3.3. An Alternative Game: Unrestricted Atropos

In this game, we follow the rules of Atropos except that players are never required to play adjacent to the last-colored circle. Instead, on every turn, a player may choose any uncolored circle on the board. We call this game *Unrestricted Atropos*. For more discussion concerning this game, see Section 6.

4. On The Complexity of Atropos

Our central complexity question concerns the following decision problem:

> ATROPOS: Given a legal Atropos state, determine whether the current player has a winning strategy.

Before analyzing the complexity of this problem, we must comment on the phrase "legal Atropos state." A legal state is one that can be realized from any legal sequence of plays from an initial game board. A game state then consists of either an initial Atropos board or a game board attainable from some sequence of moves on an initial Atropos board, with the last move identified.

As our main complexity result, we prove the following theorem.

Theorem 4.1. (Main theorem.) *ATROPOS is PSPACE-complete.*

Theorem 4.1 is true exactly when ATROPOS is in PSPACE and is also PSPACE-hard. We prove the former in Lemma 4.2, and use Section 4.2 and all of Section 5 to describe a reduction proving that ATROPOS is PSPACE-hard.

4.1. ATROPOS Is in PSPACE

Lemma 4.2. *ATROPOS is in PSPACE.*

Proof. Since the number of plays is at most the number of nodes in the game board, the depth of every branch of the game tree is linear in the size of the input. Thus, in polynomial space we can determine the result of following one path of the game tree. In order to search for a winning result, we can systematically try each possible game branch. Thus, we require only space enough to evaluate one branch at a time, as well as some bookkeeping to recall which branches we have already visited. This bookkeeping will require at most only $O(m^2)$ space, where m is the number of nodes on the board. Thus, in polynomial space, we can evaluate all the possible outcomes of the game tree until we either find a winning strategy or determine that none exists. □

4.2. Outline of the Reduction

It remains to be shown that strategies for Atropos are PSPACE-hard.

Classically, we show that problems are PSPACE-hard by reducing TQBF to them [Papadimitriou 94a]. In general, TQBF is the problem of determining whether a quantified Boolean formula, that is, a formula of the form $\exists x_1 : \forall x_2 : \exists x_3 : \forall x_4 : \cdots Q_n x_n : \phi(x_1, x_2, \ldots, x_n)$, is true. In our notation here, $\phi(x_1, \ldots, x_n)$ is a conjunctive normal form formula using the literals x_1 through x_n, while Q_n is a quantifier (either $\forall$ or $\exists$).

Because of the inherent alternation in quantified Boolean formulas, many games with nonobvious strategies for two players have been shown to be PSPACE-hard [Papadimitriou 94a]. Indeed, we see that fulfilling a TQBF instance is much like playing a game. The hero will choose a variable assignment for x_1, and then the adversary will choose for x_2. The hero chooses x_3 in response, and so on.

Our reduction will model this behavior. We will create a legal Atropos state from any TQBF instance such that a winning strategy in the game exists if and only if the formula is true. Our reduction is inspired by the reduction of TQBF to GEOGRAPHY (see [Papadimitriou 94a, p. 460] for a clear description of this reduction). The game will proceed by letting the appropriate players make moves

corresponding to the assignment of values to the variables x_i. Each player will then make one further choice, and one of the literals in one of the clauses will be selected. We will "investigate" the literal through its interpretation in the game state and use it to force an end of the game.

In our resulting game boards, most of the moves players make will be very restrictive. In our construction, there is a resulting "prescribed flow of play" directing players to make choices in the order we described above. Our reduction provides punishment strategies such that any player violating the prescribed flow of play will lose in a constant number of turns. We describe these punishment strategies, ensuring that players must follow the prescribed flow in order to have a chance of winning the game.

Using our construction, each player's last choice is easily described: the adversary will choose a clause to investigate, and the hero will choose one of the literals in that clause. That literal will be evaluated, according to the assignment it received. If the literal is true, the hero should win; otherwise, the adversary should win.

We give the adversary the power to pick a clause because in the case in which the formula is true, all the clauses must be true; the adversary will have no power. However, if at least one of the clauses is false, the formula will be false, and the adversary should be able to select one of these false clauses in order to discredit the correctness. Conversely, inside each clause we will give the hero the ability to choose among literals. Thus, if at least one of those literals is true, the hero will be able to choose and identify it. Figure 3 illustrates the layout style we desire.

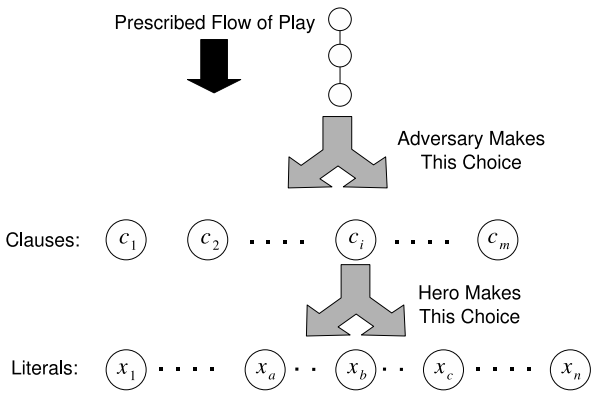

Figure 3. Construction sketch: end of the game. Clause c_i contains the literals x_a, x_b, x_c.

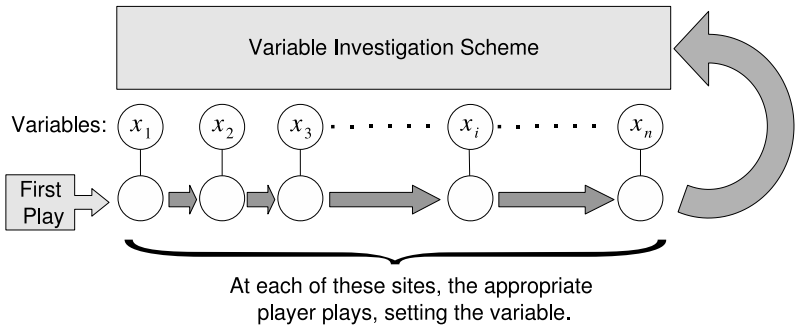

Figure 4. Construction sketch: setting the variables. The variable investigation scheme is laid out in Figure 3.

Before the flow of play reaches this point, we must have already set all of the variables. In order to accomplish this, the path of play must first pass by each of the variable settings, forcing the appropriate player to make a decision at each variable. Once the settings have been accomplished, we can move to the investigation procedure, as portrayed in Figure 4.

Overall, this plan is very reminiscent of the reduction used for `GEOGRAPHY`. In addition, our topology meets some similar hurdles that must be overcome in that construction. For instance, our plan has a nonplanar design (paths must often intersect between the selection of a clause and the selection of a literal during investigation). Thus, we will need to produce widgets that allow for these logical crossovers to occur, as we do in the following section.

Our blueprints also seem to defy the rigid structure of the game board on which we are operating. We require widgets that provide pathways for our prescribed flow of play. Geography is played on a directed graph, so enforcing the flow of play is somewhat simpler. We will need to be very careful that players cannot defeat design plans by moving in unexpected directions. Also, we need widgets to handle variable assignment, path splitting, and other obstacles to realizing our layout on an Atropos board. We continue by exhaustively describing these widgets.

5. Reduction Widgets

Our reduction requires widgets to enforce various moves and allow for appropriate decisions to be made through others. In addition, the widgets must be able to connect, allowing us to build the overlying structure by fitting them together. In this section, we describe each of the widgets and specify how they are used.

Many of the widgets are simple and are only pathways to guide the flow of play. For more complex widgets, however, we need to be able to ensure that plays not following this flow correspond to trivially bad choices. This means that any player attempting to go against the prescribed pathway will be vulnerable to an optimal opposing winning strategy that is easily computable (here in constant time).

5.1. Paths

Paths are the simplest of the widgets in our construction, although we have two different versions for different circumstances. Players should not make nontrivial decisions along paths; thus we build them to strongly restrict playing options.

The first of our two versions is a path in which on any move a player has the option of playing exactly one symbol without immediately losing. In Figure 5, assuming that the flow of play comes in from the left, the leftmost circle can and must be filled. Then, the next player is forced to fill the circle to the right, and so on. This path pattern can be extended to any length. Turning widgets for these paths also exist, as we describe later.

The other type of path supports a chain of one of two different symbols. In this type of path, shown in Figure 6, whichever symbol is initially played between barred and filled forces the next plays to follow suit. If a space is barred, the next play must also be bars. The same is true for filled (we assume again that the flow of play is going from left to right).

This type of path is often the byproduct of play leaving other widgets. Since all our widgets use single-symbol paths leading in, it is vital to be able to force the path to switch from a two-symbol path to a single-symbol path.

Figure 7 shows a mechanism for this switch. The prescribed flow of play in this widget in the diagram is again from left to right, strictly horizontal, although there are free spaces both above and below. Indeed, if a player deviates by

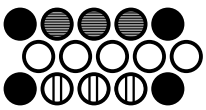

Figure 5. A single-symbol path.

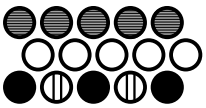

Figure 6. A two-symbol path.

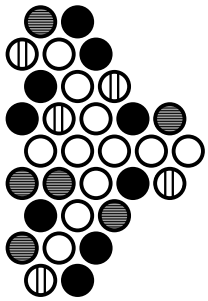

Figure 7. A switch from two symbols to a single symbol.

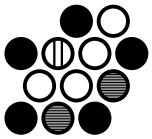

Figure 8. A 60-degree turn in a one-symbol path.

playing above the horizontal line, there is a winning response, simply by playing further above. The same is true for playing below.

The beginning pattern of the single-symbol path is clear on the right-hand side of the widget, and as before, no matter which play is made approaching that pattern, a sequence of fill plays can and must be made.

We must also be able to turn our paths in order to have them line up with other widgets. Figures 8 and 9 reveal 60-degree turning options. In order to turn more than 60 degrees, we can pair two or three of these together to attain 120- or 180-degree rotations. Note that in the second example (Figure 9) there are two possibilities for playing at the "elbow" of the turn. This does not affect the overall restriction; further plays after the elbow must return to the original symbol.

Unfortunately for fans of two-symbol paths, we do not bother to create turning widgets for them. Instead, whenever we are presented with a situation in which a two-symbol path occurs, we will immediately switch it to a one-symbol path.

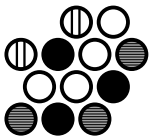

Figure 9. Another 60-degree turn in a one-symbol path.

This does not present a problem, since only one of our widgets results in an outgoing two-symbol path: the variable widget. The outgoing paths for these widgets will be followed by the two-to-one symbol-switch widget.

5.2. Variables

Having described these devices, we are prepared to reveal the widget for modeling variables, presented in Figure 10.

Here the flow of play enters initially from the bottom left and exits through the lower right path. During this time, the "playability" of the circle corresponding to some Boolean variable x_i is determined. If that variable is investigated at the end of the game, then the flow of play will enter through the entrance in the upper right corner, and will terminate inside the widget.

The choice of symbol played in the space directly below x_i determines the playability of x_i (and corresponds to the assignment of true or false). Since the plays up to that point must all either be fills or—in the case of the last space—bars, the deciding play must also be either a fill or bars, and can be either, independent of what the previous play was. Notice now that if a fill is made, the location x_i will be playable later, whereas a play of bars means that x_i is unplayable. Thus, we associate a play of fill as setting x_i to true, while a play of bars sets x_i to false.

After this choice is made, the prescribed flow of play continues rightward, which is clearly forced in the case that x_i is made unplayable. We now must describe a winning response strategy to a player deviating from the play flow, which occurs when x_i is played prematurely.

Lemma 5.1. *Prematurely playing at x_i is suboptimal.*

Proof. In order to prove this lemma, we must show a winning counterstrategy to a premature play in x_i. We use the numbers from Figure 11 to refer to play locations in this strategy.

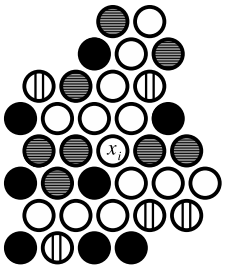

Figure 10. Variable widget.

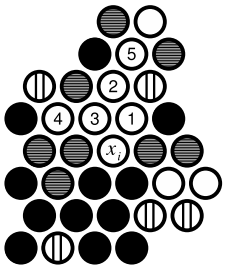

Figure 11. Responding to a premature play at x_i.

Assume that an offending player played prematurely at x_i. If that player has not already lost, then he or she must have either shaded or filled x_i. In either case, the winning punishment strategy begins by filling location 1. The offending player will have to respond in one of the following ways:

(a) Fill location 2. A winning response to this play consists in filling 3. Now the offending player must play at the unplayable location 4.

(b) Filling or shading location 3. The winning response to this is to fill 2. Now the offending player can play only at 5, which is unplayable.

Thus, playing at x_i prematurely is a losing strategy, and is suboptimal. □

Assuming that the players follow the prescribed flow, the parity of playable spaces in the upper portion of the widget is defined by the playability of x_i. If x_i is investigated, then the flow of play will enter at the upper right from a single-symbol shading path. Notice that these incoming plays cause location 1 in Figure 11 to be unplayable. Thus, the following sequence of nonsuicidal plays is at locations 2, 3, and then—if it is playable—x_i.

Thus, if x_i is playable, the play at 3 loses (a play at x_i forces the loss at 1). Otherwise, the play at 3 wins, because all three neighboring spaces are unplayable.

5.3. Splitting and Rejoining Paths

When determining which variable to investigate, players need to be able to make choices to follow different paths. In turn, multiple paths must converge to the same variable, since multiple clauses can contain the same literal from our instance of `TQBF`. Thus, we require widgets for splitting and rejoining paths.

In the course of any game, exactly one path through the variable investigation process will be used. We will enforce this through the design of these splitting and rejoining widgets.

The splitter (Figure 12), with play flowing from left to right, gives the second player the ability to choose between the two paths.

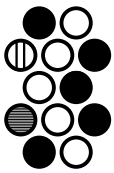

Figure 12. This widget splits a path.

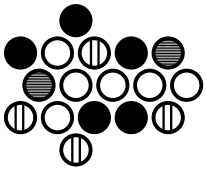

Figure 13. This widget converges two paths.

Joiner widgets (in Figure 13) have the additional responsibility of protecting the flow of play (again, left to right) to prevent a player from "going" backward. The widget performs this automatically. Independent of which path is taken from the left, the play at the intersection prevents plays through the other left path.

5.4. Path Crossing

The graph of paths in our model is not necessarily planar, meaning that we have to be able to handle paths that cross. We use the crossover widget (Figure 14) to accomplish this. Regardless of the flow of play, any series of plays entering this widget must exit from the opposite side. For example, entering from the upper-left path will result in play continuing out through the lower-right path (the careful reader can verify this for all entrance options). Since any game will use exactly one path to investigate a variable, each crossover will be used at most once.

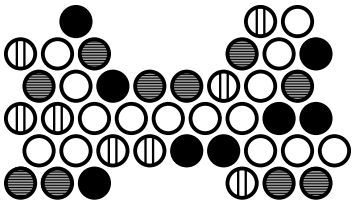

Figure 14. This widget crosses two paths.

These tools provide enough to construct Atropos boards equivalent to TQBF instances. For brevity, we do not provide an example reduction here.

6. An Open Question and a Conjecture

Unrestricted Atropos is an interesting variation for beginners and can be useful for gaining intuition about the parity underlying Sperner's lemma. Although its complexity is still open, we conjecture that this parity structure will lead to a polynomial-time solution for games played from the starting configuration.

Conjecture 6.1. (Initial Unrestricted Atropos is easy.) *The player who would not play in the last uncolored circle if the entire board were to be filled has a winning strategy for Unrestricted Atropos from the starting configuration.*

This conjecture arises from both play-testing and an illuminating property of Sperner's lemma. Recall that Sperner dictates that there will be an odd number of triangles if the board is filled up. At the end of an unrestricted game, just before the last player makes a losing play, only doomed circles remain. Coloring any of these will create either an odd or an even number of bad triangles (regardless of which color is chosen).

It turns out that the structure required to force a doomed circle to represent an even number of bad triangles is very fragile. As a piece of the game board approaches such a circle, it is very simple for a player to break the pattern, and ensure that only odd-weighted doomed circles will appear. Unfortunately, this approach is not quite as simple when one is dealing with many connected and possibly overlapping such structures.

If one of the players can either influence or predict the number of doomed circles representing an even number of bad triangles, then that player can determine the number of doomed circles at the end of the game. With this knowledge, the number of nonlosing plays is apparent, and it will be clear which player will be the last to make a nonlosing play. Although it is not yet clear, we believe that correct predictions can be made in polynomial time.

7. Conclusion

With the recent amount of attention paid to the implications of Sperner's lemma in the theoretical computer science community, the study of Atropos and other similar fixed-point games might enhance our understanding of the complexity

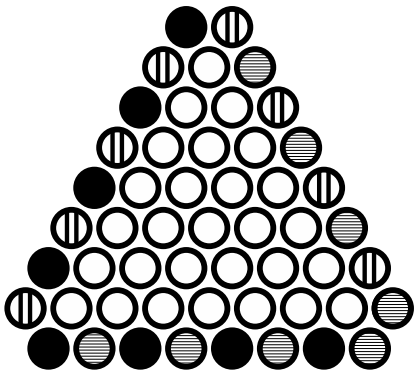

Figure 15. A board of size 7. Boards of other sizes are available at http://cs-people.bu.edu/paithan/atropos/.

of fixed-point computation. Indeed, with the newfound relationship between the complexity class PPAD, fixed points, and Nash equilibria [Papadimitriou 94b, Daskalakis et al. 06, Chen et al. 06, Chen and Deng 06], this is a promising avenue for continuing study.

Assuming that Conjecture 6.1 holds, we will have generated an interesting environment that contains the boundary between P and PSPACE (should one exist). The solution of our open question could lead immediately to another: How much distance do we allow between plays before efficient strategies exist for the game? For example, what happens to the complexity if we allow one player to color in a circle with distance *two* from the last-colored circle? What if we allow a nonconstant amount of space between plays? Answers to these questions may reveal additional information about the properties of complexity class boundaries.

Aside from the ramifications of our conjecture, Atropos is a simple, PSPACE-complete game arising from a mathematically significant construct. It inherently carries a fixed-point awareness (a loss in the game corresponds to the creation of a fixed point) and simultaneously avoids the first-player-wins dilemma faced by other related games. Most exciting, perhaps, is that it accomplishes both these benchmarks without sacrificing any elegance. For beginning players, we suggest starting on a game board with seven circles on a side, as appears in the board shown in Figure 15.

Acknowledgments. Kyle wishes to thanks all his friends at Boston University; his mother, Janelle; Silvia Montarani and Katelyn Mann for being good initial Atropos opponents; as well as everyone who has played with him since. Notably, he would like to thank

Olivia George for testing out all the different versions that have come out of the initial design and putting up with attempts to make the game more fun.

We also thank Xi Chen for teaching us many cool fixed-point theorems and Scott Russell for proofreading our paper, as well as all the anonymous reviewers who provided good feedback. Since the Workshop on Internet and Network Economics in 2007, we have found considerable interest in Atropos. Thanks to Xiaotie Deng and Fan Chung Graham for running that wonderful meeting and to Herbert Scarf for later showing off Atropos at a lecture in honor of Arrow.

This work was partially supported by the National Science Foundation under Grant No. DGE-0221680.

References

[Brouwer 10] L. Brouwer. "Über Abbildung von Mannigfaltigkeiten." *Mathematische Annalen* 71 (1910), 97–115.

[Chen and Deng 06] X. Chen and X. Deng. "Settling the Complexity of Two-Player Nash Equilibrium." In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 261–272. Los Alamitos, CA: IEEE Press, 2006.

[Chen et al. 06] X. Chen, X. Deng, and S.-H. Teng. "Computing Nash Equilibria: Approximation and Smoothed Complexity." In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 603–612. Los Alamitos, CA: IEEE Press, 2006.

[Daskalakis et al. 06] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. "The Complexity of Computing a Nash Equilibrium." In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 71–78. New York: ACM Press, 2006.

[Eppstein 06] D. Eppstein. "Computational Complexity of Games and Puzzles." Available at http://www.ics.uci.edu/~eppstein/cgt/hard.html, 2006.

[Gale 79] David Gale. "The Game of Hex and the Brouwer Fixed-Point Theorem." *American Mathematical Monthly* 10 (1979), 818–827.

[Lichtenstein and Sipser 80] David Lichtenstein and Michael Sipser. "Go is Polynomial-Space Hard." *J. ACM* 27:2 (1980), 393–401.

[Papadimitriou 94a] C. H. Papadimitriou. *Computational Complexity.* Reading, MA: Addison Wesley, 1994.

[Papadimitriou 94b] C. H. Papadimitriou. "On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence." *Journal of Computer and System Sciences* 48:3 (1994), 498–532.

[Reisch 81] S. Reisch. "Hex ist PSPACE-vollständig." *Acta Inf.* 15 (1981), 167–191.

[Sperner 28] E. Sperner. "Neuer Beweis für die Invarianz der Dimensionszahl und des Gebietes." *Abhandlungen aus dem Mathematischen Seminar Universität Hamburg* 6 (1928), 265–272.

Kyle W. Burke, Department of Mathematics and Computer Science, Wittenberg University, P. O. Box 720, Springfield, OH 45501 (kburke@wittenberg.edu)

Shang-Hua Teng, Viterbi School of Engineering, University of Southern California, 941 Bloom Walk, Los Angeles, CA 90089 (shanghua@usc.edu)

Received March 31, 2008; accepted January 27, 2009.