An "Ethical" Game-Theoretic Solution Concept for Two-Player Perfect-Information Games

Joshua Letchford[1], Vincent Conitzer[1], and Kamal Jain[2]

[1] Department of Computer Science, Duke University, Durham, NC, USA
{jcl, conitzer}@cs.duke.edu
[2] Microsoft Research, Redmond, WA, USA
kamalj@microsoft.com

Abstract. The standard solution concept for perfect-information extensive form games is subgame perfect Nash equilibrium. However, humans do not always play according to a subgame perfect Nash equilibrium, especially in games where it is possible for all the players to obtain much higher payoffs if they place some trust in each other (and this trust is not violated). In this paper, we introduce a new solution concept for two-player perfect-information games that attempts to model this type of trusting behavior (together with the "ethical" behavior of not violating that trust). The concept takes subgame perfect equilibrium as a starting point, but then repeatedly resolves the game based on the players being able to trust each other. We give two distinct algorithmic definitions of the concept and show that they are equivalent. Finally, we give a fast implementation of one of the algorithms for solving the game, and show that it runs in time $O(n \log n + nh \log(n/h))$.

1 Introduction

Under a typical game-theoretic solution concept, the players pursue nothing other than their own interest at every point in the game. Humans, however, do not always behave this way: depending on what happened earlier in the game, they may feel that they "owe" another player something and act accordingly. We propose a solution concept for two-player extensive-information games that attempts to model this phenomenon.

To illustrate the basic idea, consider the example game in Figure 1. The standard game-theoretic approach to solving this game is to simply use backward induction. If player 2 gets to move, he[3] maximizes his utility by moving left, resulting in the utilities $(0, 2)$. Anticipating this, player 1 will choose to move left in the first move, resulting in the utilities $(1, 0)$. This is the unique subgame perfect equilibrium of the game. We note that both players would prefer the rightmost outcome, which has utilities $(2, 1)$, but the strategic structure of the game prevents this outcome from occurring—at least within the standard game-theoretic approach.

Now, we argue that this is not necessarily the most sensible outcome of the game, assuming that the players have some amount of decency. Suppose player 1 does, in fact, move right. In the standard game-theoretic approach, this would be considered a

[3] We use "she" for player 1 and "he" for player 2.

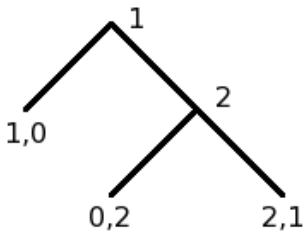

Fig. 1. A simple example.

mistake. However, suppose that it is common knowledge among the players that they understand the game perfectly. Hence, player 2 knows that player 1 did not choose this move by accident, but voluntarily chose to let player 2 pick between the outcome that is better for both of them than the subgame perfect solution, and the outcome that is much better for player 2 but worse for player 1. Player 1 knows very well that she is leaving herself vulnerable to a selfish action by player 2, but chose to move right anyway, with the hope of a better outcome for both. It seems sensible to argue that in this case, it would be unethical for player 2 to move left. Specifically, it seems that player 2 "owes" it to player 1 to give her at least as much utility as she would have received in the subgame perfect equilibrium, especially as player 2 can do so in a way that also gives him at least as much utility as he would have received in the subgame perfect equilibrium. Thus, it seems that the ethical thing to do for player 2 in this situation is to move right; if player 1 believes that player 2 is ethical in this way, then she prefers to move right initially—she "trusts" player 2 to make the "ethical" move. In this paper, we propose a general solution concept corresponding to this ethical type of reasoning.

Incidentally, the simple game above closely resembles a game studied in experimental game theory, called the "trust game." In the trust game, player 1 has an initial budget. She can choose to give any amount not exceeding this budget to player 2; if she does so, the money will be tripled before player 2 receives it. After receiving the money, player 2 can give any amount back to player 1 (this will not be tripled), and the game ends after this. Again, this game can be solved by backwards induction: it is never in player 2's interest to give any money back, and hence player 1 should give player 2 no money at all.[4] Experimentally, however, this is not at all what happens [6, 15, 14]. In an experimental study, 85% of subjects in the player 1 role gave at least some money, and 98% of subjects in the player 2 role that received some money gave some back [14]. Also, on average, subjects in the player 1 role gave $5.52 (out of their initial $10), and subjects in the player 2 role returned $6.96 [15]. We will discuss what our solution concept prescribes for this game in Appendix A.

A few more remarks are in order. We do not wish to argue that the behavior prescribed by our solution concept is the only behavior that can possibly be described as "ethical." For example, in a modified version of the trust game where player 2 does not have the option of giving money back at all, our solution concept prescribes that player

[4] This assumes that a player's utility is simply the amount of money that the player receives.

1 should give no money; but one could perhaps argue that giving money would still be the ethical thing to do, given that the money will be tripled. In fact, under a strict utilitarian framework, one might argue that the ethical thing to do is to transfer all the money. Still, we argue that our solution concept corresponds to a particular, natural (if perhaps limited) type of ethical behavior. For the purposes of this paper, we will avoid discussion of whether our concept is more "rational" than the standard game-theoretic concepts, and hence we will avoid the use of the word "rational."

Also, while there has been an agenda within game theory of justifying cooperative behavior by showing that cooperation can be sustained as an equilibrium of a repeated game (for instance, in the Prisoner's Dilemma [13]), philosophically, this paper does not fall under that agenda. (However, because our solutions always Pareto dominate or are equal to a subgame perfect solution, they can in fact be sustained as an equilibrium of the repeated game as well.)

Solution concepts that model this type of ethical behavior potentially have a number of applications. They can be used to predict human behavior. Also, when combined with algorithms for computing the ethical solution, such concepts can be used in artificially intelligent agents, for interacting either with humans or with each other. Indeed, it has been argued that standard game-theoretic solutions do not always perform well in settings where artificially intelligent agents interact with humans [4, 3, 12]. The design of artificial intelligence that behaves ethically has previously received attention [1, 10]. Much of this work relies on humans specifying examples of ethical behavior, which the agent then tries to generalize into more general rules [5, 9]. Other work specifies certain *prima facie* duties, and the agent needs to learn from labeled examples how to trade off these duties when they conflict [2]. Our work differs from this prior work in that we define a single concept that is intended to capture a subset of ethical behavior, and all that remains to be done is to find the corresponding solution (no learning is needed).

The rest of this paper is laid out as follows. In Section 2, we study some more complex examples to get some intuition about our solution concept. In Section 3, we give a first definition of our solution concept, which relies on iteratively modifying the agents' preferences and re-solving for the subgame perfect equilibrium. In Section 4, we give another definition of the concept, which relies on iteratively removing nodes from the game tree and re-solving for the subgame perfect equilibrium; we show that this definition is equivalent to the one from Section 3. Finally, in Section 5, we give a fast algorithm for computing a solution according to our concept.

2 Introductory examples

In this section, we study two additional example games. The first example shows a seemingly more complex game that can be simplified to be similar to the example in Figure 1. The second example is inherently more complex; however, understanding this example will help significantly to understand the general definition.

Example: a game with moves by Nature. Alice and Bob are sitting next to each other on a plane, and there are not enough pillows on the plane. Alice has a pillow (it was sitting in her seat), and Bob does not. Alice is currently not tired, and Bob is (and, from their demeanors, this is common knowledge). Alice could give the pillow

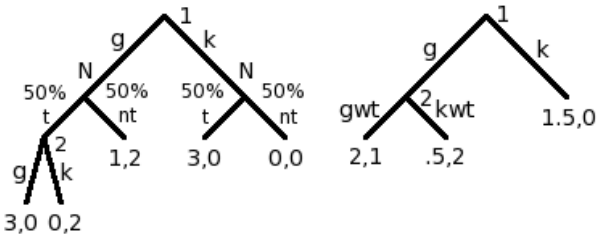

Fig. 2. Example: airplane pillows. Key: give pillow, keep pillow, Alice becomes tired, Alice does not become tired, give when Alice becomes tired, keep when Alice becomes tired

to Bob, but she might regret it if she gets tired later on. Of course, Bob could give the pillow back in that case, if he chooses to do so.[5] A possible interpretation of this game is shown in Figure 2. On the left side is the full game tree: Alice first decides whether to give the pillow, then Nature decides whether Alice gets tired, and finally Bob decides whether to return the pillow. (We note that if Alice is not tired, she slightly prefers not having the pillow, to have some more space.) We emphasize that this is a perfect-information game. Because of that, we can remove Nature from the game by taking expectations, resulting in the game on the right-hand side. By similar reasoning as that for the example in Figure 1, Alice should give the pillow, and Bob should give it back if Alice is tired. This contrasts with the subgame perfect solution in which Bob would not return the pillow, so that Alice keeps the pillow to herself; the subgame perfect solution is worse for both players.

Example: a more complex game with 6 leaves. We now move on to an example that is fundamentally more complex and that will require some more reflection on what is ethical. Consider the example in Figure 3.

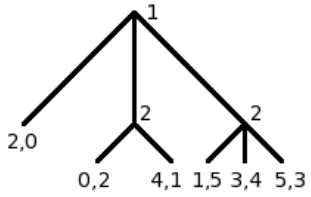

Fig. 3. A more complex example with 6 leaves.

Backward induction would tell us that player 2 will move left in each subtree, and hence player 1 should move left, resulting in the unique subgame perfect equilibrium with utilities $(2, 0)$. However, again, we may argue that if player 1 chooses middle or right, then player 2 owes it to player 1 to give her at least 2 (since she could have guaranteed herself this much, and to give her this much player 2 does not need to accept

[5] This is ignoring the potential complication that Bob may have fallen asleep on the pillow.

a utility less than the 0 that he would receive in the subgame perfect solution). That is, if player 1 plays middle, player 2 should play right (resulting in utilities $(4, 1)$); and if player 1 plays right, player 2 should play middle or right—but middle will give him a higher utility, resulting in utilities $(3, 4)$. Hence, at this level of analysis, the best move for player 1 is to move to the middle, resulting in utilities $(4, 1)$. However, we can take this analysis one step further. Now suppose that player 1 moves right anyway. Since (given ethical behavior by player 2) she could have guaranteed herself 4 by choosing middle, it can be argued that player 2 owes her at least 4 (especially because player 2 can do so while still getting at least the 1 that he received at the previous level of analysis). So, at this level, the only ethical thing for player 2 to do is to move right; middle is no longer ethical. Hence, the final solution is for both players to move right.

3 A definition of ethical behavior based on iterated solutions

We now give the general definition of our ethical solution concept. In the example in Figure 3, in a sense, we "solved" the game three times: first, we found the subgame perfect solution; second, we modified the solution based on the notion that player 2 should give player 1 what she could have guaranteed herself in the first (subgame perfect) solution (as long as doing so does not make player 2 worse off than he would have been in the first solution); third, we modified the solution again based on the notion that player 2 should give player 1 what she could have guaranteed herself in the second solution (as long as doing so does not make player 2 worse off than he would have been in the second solution). Furthermore, it is easy to construct examples in which even more levels of analysis are required.

In fact, the second and third solutions can be seen as subgame perfect solutions of a game in which the preferences have been modified based on the payoffs in the previous solution. In particular, let us call the utilities (b_1, b_2) from the previous solution the *base utilities*. Then, player 1's primary goal is to obtain at least utility b_1; player 1's secondary goal is for player 2 to obtain at least utility b_2; her tertiary goal is to maximize her own utility; and her quaternary goal is to maximize player 2's utility. [6] That is, given that she achieves her own base utility, player 1 temporarily sets her own interest aside and attempts to ensure that player 2 obtains his base utility; once that has been done, she pursues her own utility again. Player 2's modified preferences are defined similarly. Formally, we have:

Definition 1. *Given base utilities (b_1, b_2), we define player 1's* ethical preference relation $\succ^1_{(b_1,b_2)}$ *as follows: $(u_1, u_2) \succ^1_{(b_1,b_2)} (u'_1, u'_2)$ if and only if at least one of the following three conditions applies:*

- $u'_1 < b_1$, *and: either* $u'_1 < u_1$, *or both* $u'_1 = u_1$ *and* $u'_2 < u_2$.
- $u_1 \geq b_1, u'_1 \geq b_1, u'_2 < b_2$, *and:* $u_2 > u'_2$.
- $u_1 \geq b_1, u_2 \geq b_2$, *and: either* $u_1 > u'_1$, *or* $u_1 = u'_1$ *and* $u_2 > u'_2$.

[6] The quaternary goal is relevant only for breaking ties and is not essential to our concept; we add it for completeness.

Player 2*'s ethical preference relation* $\succ^2_{(b_1,b_2)}$ *is defined similarly (with the roles of* 1 *and* 2 *reversed).*

In the special case in which b_1 and b_2 are smaller than any utility in the game, the players simply maximize their own utility (and break ties in favor of the other's utility).

Now, we obtain a solution as follows: we solve the game, then update the base utilities to be the utilities in that solution, solve the game again with the modified utilities, modify the utilities again, *etc.*, until the solution stops changing.[7] Formally, we have the following algorithm:

Iterated Backward Induction with Modified Preferences (IBIMP)

1. initialize $b_1 \leftarrow -\infty$
2. initialize $b_2 \leftarrow -\infty$
3. repeat until convergence:
   (a) solve the game by backward induction with respect to $\succ^1_{(b_1,b_2)}, \succ^2_{(b_1,b_2)}$
   (b) update b_1, b_2 to be the final utilities in this solution

We have not yet shown that this process will in fact converge, but this will become clear from the alternative characterization in the next section.

For example, for the game in Figure 3, we have the following three solutions:

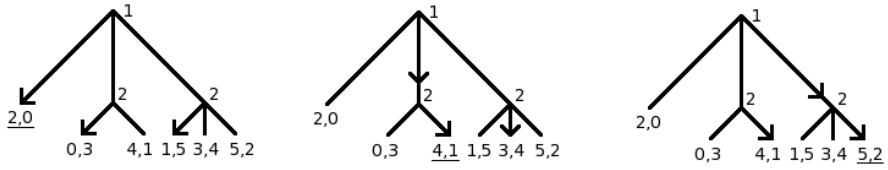

Fig. 4. IBIMP solves the example game in three iterations. At each nonleaf vertex, an arrow indicates the player's move in the subgame perfect solution for the modified preferences, and the leaf corresponding to the solution is underlined.

Another way to interpret this process is as follows: a third party repeatedly proposes strategy profiles for both players; the players accept the new proposal if and only if every move is consistent with their ethical preference relation (with respect to the base utilities from the currently accepted proposal). Then, the only sequence of solutions that the third party can successfully propose is the sequence of solutions that results from the algorithm above.

We emphasize again that breaking ties in favor of the other player is not essential to the concept, but it seems natural. (Incidentally, if ties are broken in this way, then the

[7] It should be noted that in general, a perfect-information game can have multiple subgame perfect Nash equilibria due to ties; finding the optimal one is nontrivial, but can be done in polynomial time [7]. Because we specified a tie-breaking mechanism—breaking ties in the other agent's favor—we do not need to deal with these issues.

airline pillow example (Figure 2) has the same solution even if player 1 is indifferent between having the pillow or not when she is not tired.)

4 An alternative characterization based on global pruning

In this section, we present an alternative definition of the solution concept, and show the equivalence between the two definitions. The alternative definition is also algorithmic, and also relies on repeatedly solving games. The difference is in how we modify the game. Instead of modifying the preferences based on the base utilities, we now remove all the leaf nodes for which at least one player's utility is lower than the base utility.

Iterated Backward Induction with Pruned Leaves (IBIPL)

1. repeat until convergence:
   (a) solve the game by backward induction (breaking ties in favor of the other player)
   (b) let b_1, b_2 be the final utilities in this solution
   (c) remove all the leaves with utilities (u_1, u_2) such that $u_1 < b_1$ or $u_2 < b_2$, and all intermediate nodes that have no children left

For example, for the game in Figure 3, we have the following three solutions:

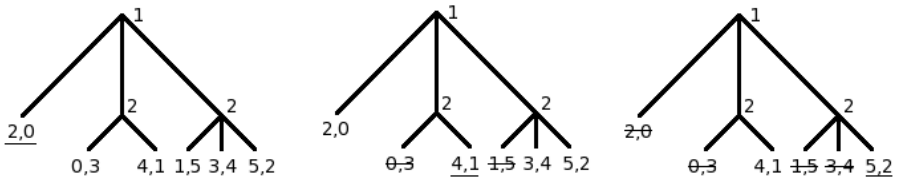

Fig. 5. IBIPL solves the example game in three iterations. The leaf corresponding to the solution in each iteration is underlined, and removed leaves are crossed out.

We note that for this game, IBIPL's solution in each stage is the same as in IBIMP. This is true in general, as we will see shortly. First, we note:

Lemma 1. *Under IBIPL, b_1 and b_2 monotonically (weakly) increase.*

Proof. b_1 and b_2 always correspond to a solution, and any leaf l with either $u_1^l < b_1$ or $u_2^l < b_2$ is immediately removed and can hence never be a future solution. ■

Theorem 1. *In each iteration, IBIMP and IBIPL find the same solution. That is, b_1 and b_2 are the same at each stage, and the values at each intermediate node are the same in the solution at each stage (if the intermediate node still exists under IBIPL).*

Proof. The first solutions are the same (both find the backward induction solution in which ties are broken in favor of the other player). We show that if the claim is true for the first k solutions, it is true for the $k+1$th solution, proving the claim by induction.

Given an intermediate node v (without loss of generality, one at which player 1 moves) that still occurs in both games, suppose that the utilities for all its children (that still occur in both games) are the same. We will show that for any child c that still occurs in the IBIMP game but not in the IBIPL tree, that child will not be the most preferred under player 1's ethical preferences $\succ^1_{(b_1,b_2)}$ (where b_1 and b_2 are the solution values for the kth solution, under both IBIMP and IBIPL by the induction assumption). Because v still occurs in the IBIPL game, it must have at least one child c' left in the IBIPL tree; since it has not been removed, it must have utilities $u_1^{c'} \geq b_1, u_2^{c'} \geq b_2$. (We emphasize again that the utilities are the same under both trees, by assumption.) On the other hand, because c was removed, every leaf l that is a child of c must have been removed; therefore, using the monotonicity property in Lemma 1, l has either $u_1^l < b_1$ or $u_2^l < b_2$. It follows that the utilities at c in the current iteration of IBIMP must have the same property: either $u_1^c < b_1$ or $u_2^c < b_2$. But then, it follows that $(u_1^{c'}, u_2^{c'}) \succ^1_{(b_1,b_2)} (u_1^c, u_2^c)$. Therefore, under IBIMP, player 1 will not choose c from v.

Hence, both IBIMP and IBIPL choose from the children c' of v for which $u_1^{c'} \geq b_1, u_2^{c'} \geq b_2$. Both of them will choose a child with the highest $u_1^{c'}$, breaking ties to maximize $u_2^{c'}$. It follows that the utilities for v are the same under both IBIMP and IBIPL, and we can repeat this process to show this for all the vertices up to and including the root, thereby establishing that the new b_1 and b_2 will be the same. ■

While the definition of the ethical solution concept corresponding to IBIMP is perhaps more natural and easier to motivate, the equivalent definition corresponding to IBIPL is often easier to work with and prove properties about. The following propositions illustrate this.

Proposition 1. *IBIPL and IBIMP always terminate.*

Proof. IBIPL clearly must terminate, because the tree shrinks in each step (other than the last one). By Theorem 1, it follows that IBIMP must also terminate. ■

Lemma 2. *When IBIPL terminates, all remaining leaves have the same utilities* (b_1, b_2).

Proof. For each remaining leaf l, we must have $u_1^l \geq b_1$ and $u_2^l \geq b_2$ (otherwise, the leaf would have been eliminated). So, if there is a remaining leaf l with utilities other than (b_1, b_2), it must Pareto dominate the current solution ($u_1^l > b_1$ and $u_2^l \geq b_2$, or $u_1^l \geq b_1$ and $u_2^l > b_2$). For the sake of contradiction, suppose that such a leaf l exists. Both players break ties in favor of the other, so the utilities (u_1^l, u_2^l) will always be preferred to (b_1, b_2). Hence, the utilities at the parent of l will be (u_1^l, u_2^l), or something else that Pareto dominates (b_1, b_2). The same is true for its parent, *etc.*, up to and including the root. This contradicts (b_1, b_2) being the backward induction solution. ■

Proposition 2. *IBIPL and IBIMP always return a Pareto optimal solution.*

Proof. For the sake of contradiction, suppose that the final solution under IBIPL is not Pareto optimal—that is, there exists a leaf that Pareto dominates the solution. This leaf cannot have been eliminated, based on Lemma 1. But then, we have two remaining leaves with different utilities, which contradicts Lemma 2. By Theorem 1, it follows that IBIMP also returns a Pareto optimal solution. ■

Proposition 3. *IBIPL and IBIMP always return a solution in which both players' utilities are at least as high as in any subgame perfect equilibrium where players break ties in each other's favor.*

Proof. Under IBIPL, after the first iteration, b_1 and b_2 are equal to the utilities from such a subgame perfect equilibrium; by Lemma 1, the utilities in later iterations can only be higher. By Theorem 1, the property also holds for IBIMP. ■

5 A fast implementation of IBIPL

The pseudocodes for IBIMP and IBIPL give us some basic (albeit not fully specified) algorithms for finding the ethical solution. In this section, we present a fast implementation of IBIPL with a runtime of $O(n \log n + nh \log(n/h))$, where n is the number of leaf nodes, and h is the height of the tree.

The algorithm and analysis assume, without loss of generality, a game tree in which all nonleaf nodes have at least two children. The basic idea is to maintain a data structure corresponding to the game tree, which maintains the optimal action at each nonleaf node. When in successive iterations, leaves are deleted, we only need to update the ancestors of those leaves (in fact, we may not need to update all of them).

A fast implementation of IBIPL

1. Initialize two arrays A_1, A_2 of pointers to the leaf nodes
2. Sort A_1 by the first player's utility, and A_2 by the second player's utility (ties can be broken arbitrarily)
3. Intialize index pointers i_1, i_2 to the first element of A_1 and A_2, respectively
4. Using A_1, compute the smallest value by which any two distinct values of u_1 differ, divide this number by twice the largest value of u_2, and call the result ϵ_1; compute ϵ_2 similarly
5. Solve the game by backward induction (breaking ties in favor of the other player); in the process, at each node v, create a Fibonacci max heap whose elements are v's children, ordered by their values for $u_1 + \epsilon_1 u_2$ if player 1 controls v, and by $u_2 + \epsilon_2 u_1$ if player 2 controls v (the ϵ terms are used to break ties in the other player's favor); the top child's u_1, u_2 values become v's values
6. Let b_1, b_2 be the values at the root
7. Repeat until convergence:
   (a) In array A_1 use binary search to find the first element for which $u_1 \geq b_1$; let its location be ${i_1}'$
   (b) For every element in A_1 in a location $\{i_1, i_1 + 1, \ldots, {i_1}' - 1\}$ do:
      i. If the corresponding leaf v has not been marked deleted, mark it deleted and call **Update**$_1(P(v), u_1(v), u_2(v))$, where $P(v)$ is v's parent
   (c) Repeat the previous two steps with array A_2
   (d) Let $i_1 \leftarrow {i_1}'$ and $i_2 \leftarrow {i_2}'$
   (e) Update b_1, b_2 to the new values of u_1, u_2 at the root

Update$_1(w, u_1, u_2)$

1. From w's Fibonacci max heap, remove (u_1, u_2)
2. If the max heap has become empty, call **Update**$_1(P(w), u_1, u_2)$
3. Otherwise, if the values at the top of the heap have changed, update w's values $u_1(w), u_2(w)$; if w is not the root, then call **Update**$_2(P(w), u_1, u_2, u_1(w), u_2(w))$

Update$_2(w, u_1, u_2, {u_1}', {u_2}')$

1. Let ${u_1}'', {u_2}''$ be the current values of w
2. From w's Fibonacci max heap, remove (u_1, u_2), and insert $({u_1}', {u_2}')$
3. If the values at the top of the heap have changed, update w's values $u_1(w), u_2(w)$; if w is not the root, then call **Update**$_2(P(w), {u_1}'', {u_2}'', u_1(w), u_2(w))$

Theorem 2. *The fast implementation of IBIPL runs in $O(n \log n + nh \log(n/h))$ time.*

Proof. Creating the sorted arrays will take $O(n \log n)$ time.

The first subgame perfect solution takes $O(n)$ time to generate; this includes creating and populating all of the Fibonacci max heaps (for which the amortized insertion time is $O(1)$). Calculating ϵ_1 and ϵ_2 also takes $O(n)$ time, because we have sorted arrays and thus only need to compare n adjacent pairs to find the smallest difference.

There are at most n iterations of the loop: each iteration other than the last must delete at least one leaf node. Within each iteration, we must find ${i_1}'$ and ${i_2}'$, which takes $O(\log n)$ time using binary search. Thus, this takes a total of $O(n \log n)$ time.

We still need to consider the time needed for the deletions and updates. Each leaf node can be deleted at most once, so we have $O(n)$ deletions in total. Finding the leaves that need to be deleted only requires us to advance through the arrays from i_1 to i_1' and from i_2 to i_2'. Hence, finding the leaf nodes to delete requires $O(n)$ time in total.

Each individual leaf deletion can result in a number of updates (including both **Update**$_1$s and **Update**$_2$s); however, it can result in at most h updates, because a node can only call **Update**$_1$ or **Update**$_2$ on its parent. Letting b_v be the branching factor (number of children) of v, updating node v requires $O(\log(b_v))$ time for (at most) an insertion and a deletion into a Fibonacci max heap. At worst, we have h nodes $v_1, \ldots, v_h$ that require updating as a result of a single leaf deletion, resulting in a total update time of $\log(b_{v_1}) + \ldots + \log(b_{v_h})$. We know, however, that $b_{v_1} + \ldots + b_{v_h} \leq 2n$: this is because there are at most $2n$ nodes in the tree in total (because we assume that each node has a branching factor of at least 2), and the children of different nodes do not overlap. The optimization problem **maximize** $\log(b_{v_1}) + \ldots + \log(b_{v_h})$ **subject to** $b_{v_1} + \ldots + b_{v_h} \leq 2n$ is solved by setting $b_{v_i} = 2n/h$ for every i, because the $\log$ function is concave. It follows that the total time required for updates as a result of a single leaf deletion is $O(h \log(n/h))$, resulting in a bound of $O(nh \log(n/h))$ for the total time for updates.

Adding everything together, our total runtime bound is $O(n \log n + n + n \log n + n + nh \log(n/h)) = O(n \log n + nh \log(n/h))$. ∎

For the purpose of reducing the runtime bound (and in its own right), it is interesting to consider how many iterations a particular type of tree can require. In the proof of the runtime bound above, we only used the fact that there are at most n iterations. We already know that the centipede game requires $\Omega(n)$ iterations, but of course the centipede game tree is extremely unbalanced. In Appendix B, we show how to construct games with balanced binary trees that require $\Omega(\sqrt{n})$ iterations, as well as games with depth 2 (not binary) that require $\Omega(\sqrt{n})$ iterations. We also construct a game in which the solution in the first iteration is to move left at the root, in the second iteration it is to move right, and in the third it is once again to move left—that is, we cannot eliminate a move/subtree when it stops being used (see Appendix C).

6 Conclusions

In this paper, we introduced a new solution concept for two-player perfect-information games that attempts to model a type of trusting behavior (together with the "ethical" behavior of not violating that trust). The concept takes subgame perfect Nash equilibrium as a starting point, but then repeatedly resolves the game based on the players being able to trust each other. We gave two distinct algorithmic definitions of the concept and showed that they are equivalent. Finally, we gave a fast implementation of one of the algorithms for solving the game, and showed that it runs in time $O(n \log n + nh \log(n/h))$.

There exist a large number of directions for future research. First, the validity of the concept should be evaluated. While we believe that the two equivalent definitions provide a strong normative justification of our concept, there may be other axiomatizations of the concept that make it even more convincing. However, as we have said previously, our concept only considers one particular type of ethical behavior, and other types of ethical behavior may lead to other natural solution concepts. It would also be interesting to investigate in more detail to what extent our solution concept models human behavior, taking a more descriptive approach rather than the normative approach discussed so far.

Another interesting direction is to try to generalize the concept to 3+-player games and/or games with imperfect information. Neither of these generalizations seem trivial. For example, if there is third player that barely affects the outcome of the game at all, then are the first two players still obliged to maintain player 3's utility at at least the same level across iterations? And, if (due to imperfect information) it is not clear to player 1 whether player 2 took a "trusting" move, is player 1 obliged to assume that player 2 took such a move or not? Hence, it is not clear whether the 3+-player and/or imperfect-information cases admit as clean of a concept as the 2-player perfect-information case. Another issue is that our concept in some sense assumes that it is common knowledge that both players will behave ethically, and it is not clear what should be done if this is not the case. (One might model this as a game of imperfect information in which Nature first decides which players are ethical.)

Finally, how can we use this concept to approximately solve games that are so large that it is not possible to write down the entire tree? AI techniques for such games are usually based on limited-depth lookahead and heuristics to evaluate the nodes at this

limited depth. For our concept, it is not clear whether the correct approach is to use this type of limited-depth search on the full remaining tree within each iteration of the algorithm; or, to run the algorithm (all the iterations) on a limited-depth tree; or to do something entirely different. It also seems that if the two players do not use the same heuristics or depths, this can cause significant difficulties, because from one player's perspective the other may not be acting ethically.

7 Acknowledgments

We thank the National Science Foundation and the Alfred P. Sloan Foundation for support (through award number IIS-0812113 and a Research Fellowship, respectively).

References

1. Michael Anderson and Susan Leigh Anderson. The status of machine ethics: a report for the AAAI symposium. *Minds and Machines*, 17:1–10, 2007.
2. Michael Anderson, Susan Leigh Anderson, and Chris Armen. Medethex: A prototype medical ethics advisor. *AAAI*, pages 1759–1765, 2006.
3. Ya'akov Gal and Avi Pfeffer. Modeling reciprocity in human bilateral negotiation. *AAAI*, 2007.
4. Barbara J. Grosz, Sarit Kraus, Shavit Talman, Boaz Stossel, and Moti Havlin. The influence of social dependencies on decision-making: Initial investigations with a new game. *AAMAS*, pages 782–789, 2004.
5. Marcello Guarini. Particularism and the classification and reclassification of moral cases. *IEEE Intelligent Systems*, 21(4):22–28, 2006.
6. Brooks King-Casas, Damon Tomlin, Cedric Anen, Colin F. Camerer, Steven R. Quartz, and P. Read Montague. Getting to know you: Reputation and trust in a two-person economic exchange. *Science*, 308(5718):78–83, 2005.
7. Michael L. Littman, Nishkam Ravi, Arjun Talwar, and Martin Zinkevich. An efficient optimal-equilibrium algorithm for two-player game trees. *UAI*, 2006.
8. Richard D. McKelvey and Thomas R. Palfey. An experimental study of the centipede game. *Econometrica*, 60(4):803–836, 1992.
9. Bruce M. McLaren. Computational models of ethical reasoning: Challenges, initial steps, and future directions. *IEEE Intelligent Systems*, 21(4):29–37, 2006.
10. James H. Moor. The nature, importance, and difficulty of machine ethics. *IEEE Intelligent Systems*, 21(4):18–21, 2006.
11. Rosemarie Nagel and Tang Fang Fang. An experimental study on the centipede game in normal form—an investigation on learning. *Journal of Mathematical Psychology*, 42:356–382, June 1998.
12. Shavit Talman, Ya'akov Gal, Meirav Hadad, and Sarit Kraus. Adapting to agents' personalities in negotiation. *AAMAS*, 2005.
13. Albert Tucker. A two-person dilemma. In Eric Rasmusen, editor, *Readings in Games and Information*, pages 7–8. Blackwell Publishing, 2001. Originally written in 1950 (unpublished).
14. Paul J. Zak. The neurobiology of trust. *Scientific American*, pages 88–95, June 2008.
15. Paul J. Zak, Robert Kurzban, and William T. Matzner. Oxytocin is associated with human trustworthiness. *Hormones and Behavior*, 48:522–527, December 2005.

A Some example games

In this section, we will consider some classic games for which our solution concept provides interesting solutions. First, we consider the trust game described in the introduction. To keep the game at a manageable size, the first player has one dollar, anything she gives to player 2 will be tripled, and only integer donations are allowed. This game appears in Figure 6. As we noted earlier, the subgame perfect (backward induction) so-

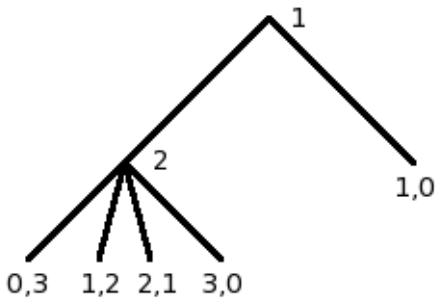

Fig. 6. A version of the trust game.

lution is for player 1 to give no money (move right). As for our solution concept, let us consider the IBIPL interpretation. First, it eliminates $(0, 3)$ (because this has a lower utility for player 1 than the subgame perfect solution $(1, 0)$. Then, the next solution is for player 1 to give 1, and for player 2 to give 1 back. This is the final solution (nothing Pareto dominates it). In general trust games, our solution concept always results in 1 giving all her money, and 2 giving her back exactly that much. This solution is always reached in the second iteration of IBIMP/IBIPL.

The second example game is the *centipede game* (really, a family of games). In this game, both players have some amount of money. They alternate turns, and at each turn, the current player can either end the game, or pass to the other player. If she passes, then her amount decreases, but the other player's amount increases. If the other player then passes back, then both players will be better off than they were before the two passes. (A typical example is that the player who passes loses half of her money, but the other player's money quadruples.) However, there is a fixed deadline. An example centipede game can be seen in Figure 7. The subgame perfect (backward induction) solution to game is for player 1 to end the game at the first step. As for our solution concept, let us consider the IBIPL interpretation. First, it eliminates $(1, 8)$ (because it has a lower utility for player 1 than the subgame perfect solution $(2, 2)$), resulting in $(4, 4)$ becoming the new solution. Then, it eliminates $(2, 16)$ (and $(2, 2)$), resulting in the final solution $(8, 8)$. In general, in centipede games, our solution concept will always choose the bottom-most leaf that is more advantageous to player 1. That is, the players pass until the very last move, which may be either a pass or an ending move. This solution is reached as follows: in each iteration of IBIMP/IBIPL, the current solution moves two levels down, as in the above example.

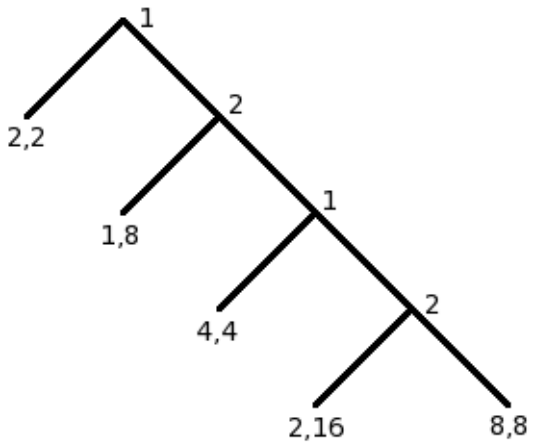

Fig. 7. A version of the centipede game.

In empirical studies of the centipede game, humans rarely play the subgame perfect equilibrium; rather, they usually continue to pass until a little after the middle of the game (but usually they do not continue all the way to the end, either) [8, 11].

B Lower bounds on the number of iterations

We now introduce some example games that require a large number of iterations to solve using our concept. In these games, all utilities are strictly positive and have no ties. We recall that n is the total number of leaf nodes.

We first introduce a game that will give us a lower bound on the number of iterations required in a complete alternating-move binary tree (in the worst case).

Definition 2. *We define the game G_k recursively as follows:*

- *G_1 is a modified version of the game in Figure 1: all its utilities are increased by 1 (to make them positive).*
- *G_{k+1} is defined as follows:*
  - *If player 1 moves left at the root, then player 2 makes a dummy move after that, after which the players play G_k.*
  - *If player 1 moves right at the root, and player 2 subsequently takes his rightmost move, then the players play a modified version of G_k (which we will call G'_k) in which all of player 1's (player 2's) utilities have been increased by the largest utility that player 1 (player 2) gets in G_k.*
  - *If player 1 moves right at the root, then player 2 has a move (not the rightmost one) for every leaf in G_k that is the solution for G_k in some iteration. This move results in a utility for player 1 that is worse than what she receives in the corresponding solution for G_k, but better than what she received in the solution in the previous iteration for G_k (if there is a previous iteration). It results in*

a utility for player 2 that is larger than any utility in G'_k (and the larger the iteration that the solution corresponds to, the lower player 2's utility).

Figure 8 shows G_2.

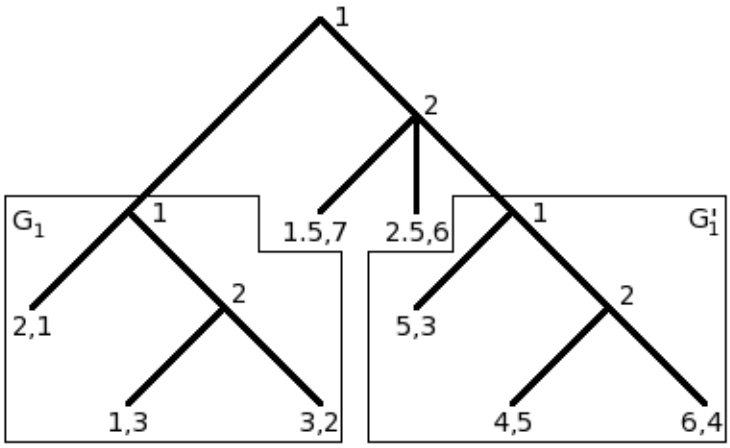

Fig. 8. G_2

Theorem 3. *G_k requires 2^k iterations to solve.*

Proof. We will show this by induction. G_1 requires 2 iterations. Suppose that we have shown that G_k requires 2^k iterations; we will show that G_{k+1} requires 2^{k+1} iterations. Specifically, we will show that the first 2^k iterations will correspond to the solutions of G_k, and the next 2^k iterations will correspond to the solutions of G'_k.

First, we argue by induction that in iteration i ($i < 2^k$), we have:

- The solution is the leaf in the subtree G_k that is also the solution in iteration i for G_k.
- Of the moves that player 2 has after player 1 moves right at the root, the first $i - 1$ have been removed.

It is easy to see that this is true for $i = 1$. If it is true for i, then in the next iteration, we will remove the leftmost move for player 2 (at the vertex after player 1 has moved right at the root); the most preferred move for player 2 at this vertex will then become the next move. The solution for the remaining subtree on the left side will correspond to the $i + 1$th solution of G_k, and player 1 will prefer this at the root, based on how we defined the utilities. This establishes the claim above, so that the first 2^k iterations will correspond to the solutions in the subtree G_k.

After the first 2^k iterations, all of the game tree has been eliminated, with the exception of G'_k. Since the utilities in G'_k differ from those in G_k by a constant, by the induction assumption, it requires 2^k additional iterations.

Corollary 1. *A complete alternating-move binary tree can require $\Omega(\sqrt{n})$ iterations to solve.*

Proof. G_k can be turned into a complete binary tree of depth $2k$ by adding dummy moves; this tree will have $n = 2^{2k}$ leaves and still require $2^k = \sqrt{n}$ iterations.

Now, we introduce a game that will allow us to show a lower bound on the number of iterations required to solve a game of depth 2 (in the worst case).

Definition 3. *We define the game H_k recursively as follows:*

- *H_1 consists of a single leaf.*
- *H_k consists of H_{k-1}, except there is a single additional move for player 1 at the root. If player 1 takes this move, then, for every iteration of H_{k-1}, there is a corresponding move for player 2. This move gives player 1 a utility that is less than the utility that player 1 receives in the solution to the corresponding iteration of H_{k-1}, but more than the utility that player 1 receives in the previous iteration of H_{k-1} (if there is a previous iteration). There is one additional move for player 2 that gives player 1 a utility that is greater than any other utility in the game. As for player 2's utility, the earlier moves give player 2 a larger utility, and all these moves give player 2 a utility that is greater than anything that player 2 receives in H_{k-1}.*

For example, the game in Figure 1 is H_2, and the game in Figure 3 is H_3.

Theorem 4. *H_k requires k iterations to solve.*

Proof. We will prove this by induction on k. Clearly this is true for H_1. We will show, by induction on i, that the solution in the ith iteration of H_{k+1} corresponds to the solution in the ith iteration of H_k, for $i \leq k$. If this is true for i, then in the next iteration, the ith move for player 2 (at the node after player 1 plays the new, rightmost move at the root) is removed, so that the $i+1$th move becomes optimal at this node. The next solution for the part of H_{k+1} corresponding to H_k will correspond to the $i+1$th solution of H_k, and player 1 will prefer this at the root, based on how we defined the utilities. This establishes that the solution in the ith iteration of H_{k+1} corresponds to the solution in the ith iteration of H_k, for $i \leq k$. Finally, there will be one last iteration, in which we move to the rightmost leaf, which both players prefer to the solution corresponding to the kth iteration of H_k.

Corollary 2. *A game tree of depth 2 can require $\Omega\sqrt{n}$ iterations to solve.*

Proof. Because we have shown that H_k requires k iterations, it follows that H_{k+1} has $k+1$ more leaves than H_k. Hence, H_k has $n = 1 + \ldots + k = k(k+1)/2$ leaves, which is $O(k^2)$; and H_k requires k iterations to solve.

C An example where a move switches back and forth

In all of the example games above, when the move taken at a node changes in an iteration, it never changes back to the original move in later iterations. One might wonder

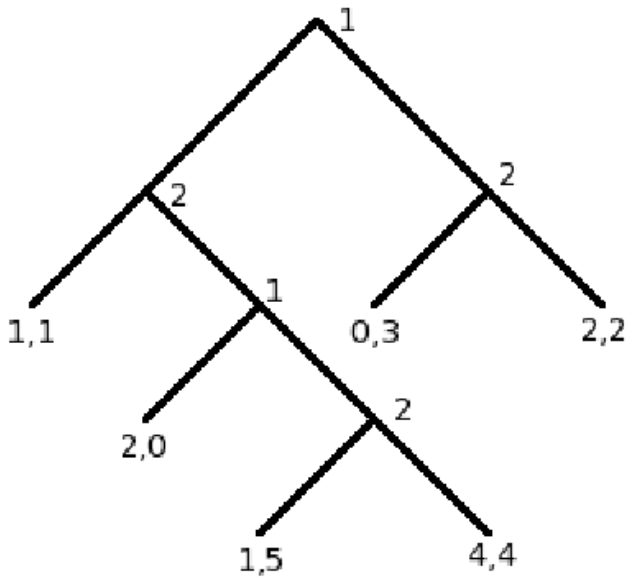

Fig. 9. An example where the move at the root switches back and forth.

whether this is in fact impossible; if that were the case, that might facilitate the design of a more efficient algorithm for computing solutions (because we would be able to remove subtrees that were used earlier). However, the example in Figure 9 shows that it is in fact possible for the move to change back to the original move.

In iteration 1, the (intermediate) solution is (1,1), using the left branch of the root. In iteration 2, the solution is (2,2), using the right branch of the root. Finally, in iteration 3 (the final iteration), the solution is (4,4), using the left branch of the root again.