RESEARCH ARTICLE

COMPUTER SCIENCE

Superhuman AI for multiplayer poker

Noam Brown[1,2]* and Tuomas Sandholm[1,3,4,5]*

In recent years there have been great strides in artificial intelligence (AI), with games often serving as challenge problems, benchmarks, and milestones for progress. Poker has served for decades as such a challenge problem. Past successes in such benchmarks, including poker, have been limited to two-player games. However, poker in particular is traditionally played with more than two players. Multiplayer games present fundamental additional issues beyond those in two-player games, and multiplayer poker is a recognized AI milestone. In this paper we present Pluribus, an AI that we show is stronger than top human professionals in six-player no-limit Texas hold'em poker, the most popular form of poker played by humans.

Poker has served as a challenge problem for the fields of artificial intelligence (AI) and game theory for decades (*1*). In fact, the foundational papers on game theory used poker to illustrate their concepts (*2*, *3*). The reason for this choice is simple: No other popular recreational game captures the challenges of hidden information as effectively and as elegantly as poker. Although poker has been useful as a benchmark for new AI and game-theoretic techniques, the challenge of hidden information in strategic settings is not limited to recreational games. The equilibrium concepts of von Neumann and Nash have been applied to many real-world challenges such as auctions, cybersecurity, and pricing.

The past two decades have witnessed rapid progress in the ability of AI systems to play increasingly complex forms of poker (*4*–*6*). However, all prior breakthroughs have been limited to settings involving only two players. Developing a superhuman AI for multiplayer poker was the widely recognized main remaining milestone. In this paper we describe Pluribus, an AI capable of defeating elite human professionals in six-player no-limit Texas hold'em poker, the most commonly played poker format in the world.

Theoretical and practical challenges of multiplayer games

AI systems have reached superhuman performance in games such as checkers (*7*), chess (*8*), two-player limit poker (*4*), Go (*9*), and two-player no-limit poker (*6*). All of these involve only two players and are zero-sum games (meaning that whatever one player wins, the other player loses). Every one of those superhuman AI systems was generated by attempting to approximate a Nash equilibrium strategy rather than by, for example, trying to detect and exploit weaknesses in the opponent. A Nash equilibrium is a list of strategies, one for each player, in which no player can improve by deviating to a different strategy. Nash equilibria have been proven to exist in all finite games—and many infinite games—though finding an equilibrium may be difficult.

Two-player zero-sum games are a special class of games in which Nash equilibria also have an extremely useful additional property: Any player who chooses to use a Nash equilibrium is guaranteed to not lose in expectation no matter what the opponent does (as long as one side does not have an intrinsic advantage under the game rules, or the players alternate sides). In other words, a Nash equilibrium strategy is unbeatable in two-player zero-sum games that satisfy the above criteria. For this reason, to "solve" a two-player zero-sum game means to find an exact Nash equilibrium. For example, the Nash equilibrium strategy for Rock-Paper-Scissors is to randomly pick Rock, Paper, or Scissors with equal probability. Against such a strategy, the best that an opponent can do in expectation is tie (*10*). In this simple case, playing the Nash equilibrium also guarantees that the player will not win in expectation. However, in more complex games, even determining how to tie against a Nash equilibrium may be difficult; if the opponent ever chooses suboptimal actions, then playing the Nash equilibrium will indeed result in victory in expectation.

In principle, playing the Nash equilibrium can be combined with opponent exploitation by initially playing the equilibrium strategy and then over time shifting to a strategy that exploits the opponent's observed weaknesses (for example, by switching to always playing Paper against an opponent that always plays Rock) (*11*). However, except in certain restricted ways (*12*), shifting to an exploitative nonequilibrium strategy opens oneself up to exploitation because the opponent could also change strategies at any moment. Additionally, existing techniques for opponent exploitation require too many samples to be competitive with human ability outside of small games. Pluribus plays a fixed strategy that does not adapt to the observed tendencies of the opponents.

Although a Nash equilibrium strategy is guaranteed to exist in any finite game, efficient algorithms for finding one are only proven to exist for special classes of games, among which two-player zero-sum games are the most prominent. No polynomial-time algorithm is known for finding a Nash equilibrium in two-player non-zero-sum games, and the existence of one would have sweeping surprising implications in computational complexity theory (*13*, *14*). Finding a Nash equilibrium in zero-sum games with three or more players is at least as hard (because

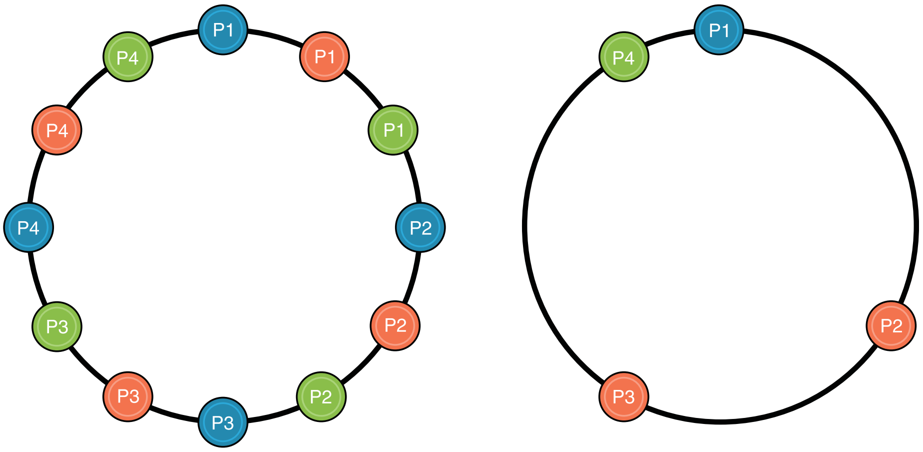

Fig. 1. An example of the equilibrium selection problem. In the Lemonade Stand Game, players simultaneously choose a point on a ring and want to be as far away as possible from any other player. In every Nash equilibrium, players are spaced uniformly around the ring. There are infinitely many such Nash equilibria. However, if each player independently chooses one Nash equilibrium to play, their joint strategy is unlikely to be a Nash equilibrium. (**Left**) An illustration of three different Nash equilibria in this game, distinguished by three different colors. (**Right**) Each player independently chooses one Nash equilibrium. Their joint strategy is not a Nash equilibrium.

[1]Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA. [2]Facebook AI Research, New York, NY 10003, USA. [3]Strategic Machine, Inc., Pittsburgh, PA 15213, USA. [4]Strategy Robot, Inc., Pittsburgh, PA 15213, USA. [5]Optimized Markets, Inc., Pittsburgh, PA 15213, USA.
***Corresponding author. Email: noamb@cs.cmu.edu (N.B.); sandholm@cs.cmu.edu (T.S.)**

a dummy player can be added to the two-player game to make it a three-player zero-sum game). Even approximating a Nash equilibrium is hard (except in special cases) in theory (*15*), and in games with more than two players, even the best complete algorithm can only address games with a handful of possible strategies per player (*16*). Moreover, even if a Nash equilibrium could be computed efficiently in a game with more than two players, it is not clear that playing such an equilibrium strategy would be wise. If each player in such a game independently computes and plays a Nash equilibrium, the list of strategies that they play (one strategy per player) may not be a Nash equilibrium and players might have an incentive to deviate to a different strategy. One example of this is the Lemonade Stand Game (*17*), illustrated in Fig. 1, in which each player simultaneously picks a point on a ring and wants to be as far away as possible from any other player. The Nash equilibrium is for all players to be spaced uniformly along the ring, but there are infinitely many ways this can be accomplished and therefore infinitely many Nash equilibria. If each player independently computes one of those equilibria, the joint strategy is unlikely to result in all players being spaced uniformly along the ring. Two-player zero-sum games are a special case where even if the players independently compute and select Nash equilibria, the list of strategies is still a Nash equilibrium.

The shortcomings of Nash equilibria outside of two-player zero-sum games, and the failure of any other game-theoretic solution concept to convincingly overcome them, have raised the question of what the right goal should even be in such games. In the case of six-player poker, we take the viewpoint that our goal should not be a specific game-theoretic solution concept but rather to create an AI that empirically consistently defeats human opponents, including elite human professionals.

The algorithms that we used to construct Pluribus, discussed in the next two sections, are not guaranteed to converge to a Nash equilibrium outside of two-player zero-sum games. Nevertheless, we observe that Pluribus plays a strong strategy in multiplayer poker that is capable of consistently defeating elite human professionals. This shows that even though the techniques do not have known strong theoretical guarantees on performance outside of the two-player zero-sum setting, they are nevertheless capable of producing superhuman strategies in a wider class of strategic settings.

Description of Pluribus

The core of Pluribus's strategy was computed through self-play, in which the AI plays against copies of itself, without any data of human or prior AI play used as input. The AI starts from scratch by playing randomly and gradually improves as it determines which actions, and which probability distribution over those actions, lead to better outcomes against earlier versions of its strategy. Forms of self-play have previously been used to generate powerful AIs in two-player zero-sum games such as backgammon (*18*), Go (*9*, *19*), Dota 2 (*20*), StarCraft 2 (*21*), and two-player poker (*4*–*6*), though the precise algorithms that were used have varied widely. Although it is easy to construct toy games with more than two players in which commonly used self-play algorithms fail to converge to a meaningful solution (*22*), in practice self-play has nevertheless been shown to do reasonably well in some games with more than two players (*23*).

Pluribus's self-play produces a strategy for the entire game offline, which we refer to as the blueprint strategy. Then during actual play against opponents, Pluribus improves upon the blueprint strategy by searching for a better strategy in real time for the situations in which it finds itself during the game. In subsections below, we discuss both of those phases in detail, but first we discuss abstraction, forms of which are used in both phases to make them scalable.

Abstraction for large imperfect-information games

There are far too many decision points in no-limit Texas hold'em to reason about individually. To reduce the complexity of the game, we eliminate some actions from consideration and also bucket similar decision points together in a process called abstraction (*24*, *25*). After abstraction, the bucketed decision points are treated as identical. We use two kinds of abstraction in Pluribus: action abstraction and information abstraction.

Action abstraction reduces the number of different actions the AI needs to consider. No-limit Texas hold'em normally allows any whole-dollar bet between $100 and $10,000. However, in practice there is little difference between betting $200 and betting $201. To reduce the complexity of forming a strategy, Pluribus only considers a few different bet sizes at any given decision point. The exact number of bets it considers varies between 1 and 14 depending on the situation. Although Pluribus can limit itself to only betting one of a few different sizes between $100 and $10,000, when actually playing no-limit poker, the opponents are not constrained to those few options. What happens if an opponent bets $150 while Pluribus has only been trained to consider bets of $100 or $200? Generally, Pluribus

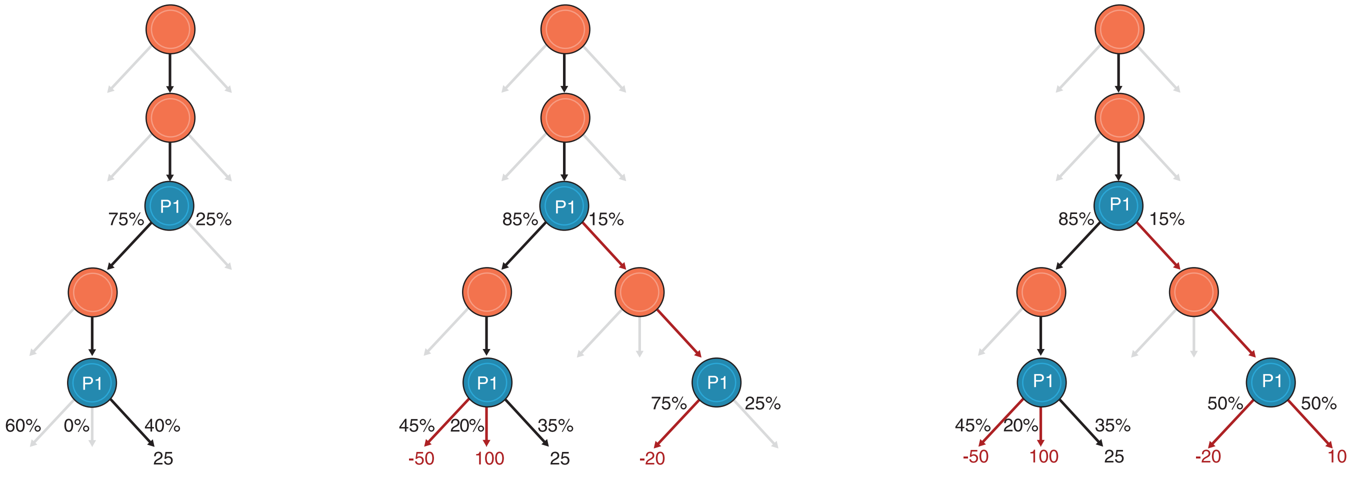

Fig. 2. A game tree traversal via Monte Carlo CFR. In this figure, player P_1 is traversing the game tree. (**Left**) A game is simulated until an outcome is reached. (**Middle**) For each P_1 decision point encountered in the simulation in the left panel, P_1 explores each other action that P_1 could have taken and plays out a simulation to the end of the game. P_1 then updates its strategy to pick actions with higher payoff with higher probability. (**Right**) P_1 explores each other action that P_1 could have taken at every new decision point encountered in the middle panel, and P_1 updates its strategy at those hypothetical decision points. This process repeats until no new P_1 decision points are encountered, which in this case is after three steps but in general may be more. Our implementation of MCCFR (described in the supplementary materials) is equivalent but traverses the game tree in a depth-first manner. (The percentages in the figure are for illustration purposes only and may not correspond to actual percentages that the algorithm would compute.)

will rely on its search algorithm (described in a later section) to compute a response in real time to such "off-tree" actions.

The other form of abstraction that we use in Pluribus is information abstraction, in which decision points that are similar in terms of what information has been revealed (in poker, the player's cards and revealed board cards) are bucketed together and treated identically (*26*–*28*). For example, a 10-high straight and a 9-high straight are distinct hands but are nevertheless strategically similar. Pluribus may bucket these hands together and treat them identically, thereby reducing the number of distinct situations for which it needs to determine a strategy. Information abstraction drastically reduces the complexity of the game, but it may wash away subtle differences that are important for superhuman performance. Therefore, during actual play against humans, Pluribus uses information abstraction only to reason about situations on future betting rounds, never the betting round that it is actually in. Information abstraction is also applied during offline self-play.

Self-play through improved Monte Carlo counterfactual regret minimization

The blueprint strategy in Pluribus was computed using a variant of counterfactual regret minimization (CFR) (*29*). CFR is an iterative self-play algorithm in which the AI starts by playing completely at random but gradually improves by learning to beat earlier versions of itself. Every competitive Texas hold'em AI for at least the past 6 years has computed its strategy using some variant of CFR (*4*–*6*, *23*, *28*, *30*–*34*). We use a form of Monte Carlo CFR (MCCFR) that samples actions in the game tree rather than traversing the entire game tree on each iteration (*33*, *35*–*37*).

On each iteration of the algorithm, MCCFR designates one player as the traverser whose current strategy is updated on the iteration. At the start of the iteration, MCCFR simulates a hand of poker based on the current strategy of all players (which is initially completely random). Once the simulated hand is completed, the AI reviews each decision that was made by the traverser and investigates how much better or worse it would have done by choosing the other available actions instead. Next, the AI reviews each hypothetical decision that would have been made following those other available actions and investigates how much better it would have done by choosing the other available actions, and so on. This traversal of the game tree is illustrated in Fig. 2. Exploring other hypothetical outcomes is possible because the AI knows each player's strategy for the iteration and can therefore simulate what would have happened had some other action been chosen instead. This counterfactual reasoning is one of the features that distinguishes CFR from other self-play algorithms that have been deployed in domains such as Go (*9*), Dota 2 (*20*), and StarCraft 2 (*21*).

The difference between what the traverser would have received for choosing an action versus what the traverser actually achieved (in expectation) on the iteration is added to the counterfactual regret for the action. Counterfactual regret represents how much the traverser regrets having not chosen that action in previous iterations. At the end of the iteration, the traverser's strategy is updated so that actions with higher counterfactual regret are chosen with higher probability.

For two-player zero-sum games, CFR guarantees that the average strategy played over all iterations converges to a Nash equilibrium, but convergence to a Nash equilibrium is not guaranteed outside of two-player zero-sum games. Nevertheless, CFR guarantees in all finite games that all counterfactual regrets grow sublinearly in the number of iterations. This, in turn, guarantees in the limit that the average performance of CFR on each iteration that was played matches the average performance of the best single fixed strategy in hindsight. CFR is also proven to eliminate iteratively strictly dominated actions in all finite games (*23*).

Because the difference between counterfactual value and expected value is added to counterfactual regret rather than replacing it, the first iteration in which the agent played completely randomly (which is typically a very bad strategy) still influences the counterfactual regrets, and therefore the strategy that is played, for iterations far into the future. In the original form of CFR, the influence of this first iteration decays at a rate of $\frac{1}{T}$, where T is the number of iterations played. To more quickly decay the influence of these early "bad" iterations, Pluribus uses a recent form of CFR called Linear CFR (*38*) in early iterations. (We stop the discounting after that because the time cost of doing the multiplications with the discount factor is not worth the benefit later on.) Linear CFR assigns a weight of T to the regret contributions of iteration T. Therefore, the influence of the first iteration decays at a rate of $\frac{1}{\sum_{t=1}^{T} t} = \frac{2}{T(T+1)}$. This leads to the strategy improving more quickly in practice while still maintaining a near-identical worst-case bound on total regret. To speed up the blueprint strategy computation even further, actions with extremely negative regret are not explored in 95% of iterations.

The blueprint strategy for Pluribus was computed in 8 days on a 64-core server for a total of 12,400 CPU core hours. It required less than 512 GB of memory. At current cloud computing spot instance rates, this would cost about $144 to produce. This is in sharp contrast to all the other recent superhuman AI milestones for games, which used large numbers of servers and/or farms of graphics processing units (GPUs). More memory

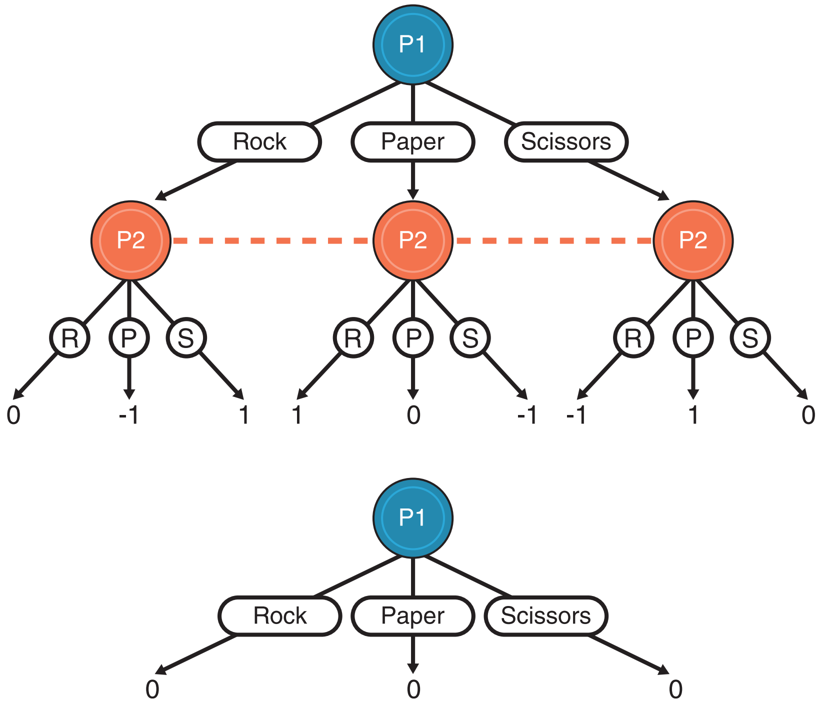

Fig. 3. Perfect-information game search in Rock-Paper-Scissors. (**Top**) A sequential representation of Rock-Paper-Scissors in which player 1 acts first but does not reveal her action to player 2, who acts second. The dashed lines between the player 2 nodes signify that player 2 does not know which of those nodes he is in. The terminal values are shown only for player 1. (**Bottom**) A depiction of the depth-limited subgame if player 1 conducts search (with a depth of one) using the same approach as is used in perfect-information games. The approach assumes that after each action, player 2 will play according to the Nash equilibrium strategy of choosing Rock, Paper, and Scissors with $\frac{1}{3}$ probability each. This results in a value of zero for player 1 regardless of her strategy.

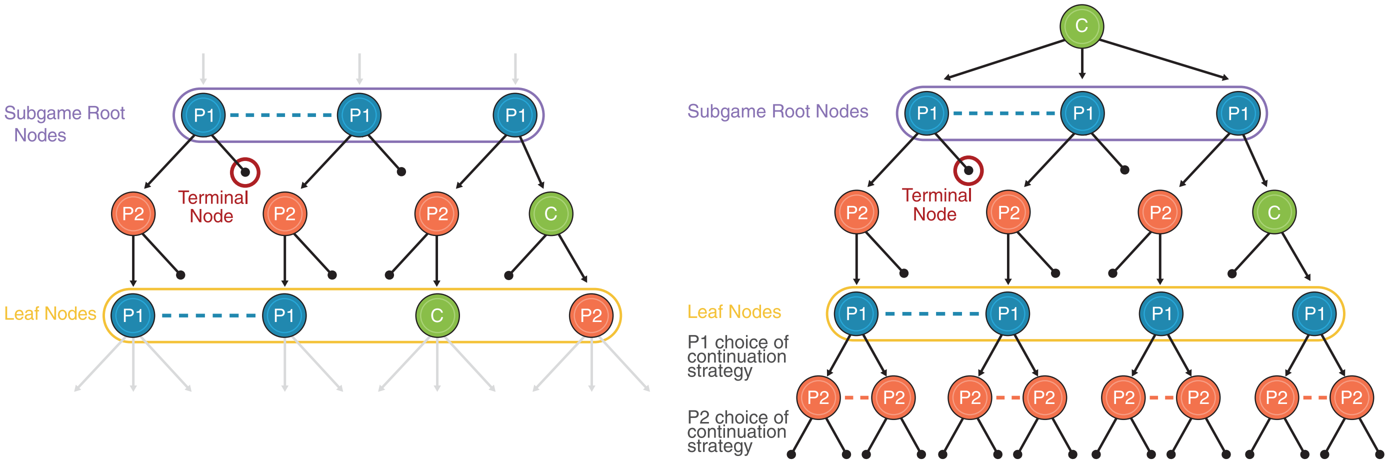

Fig. 4. Real-time search in Pluribus. The subgame shows just two players for simplicity. A dashed line between nodes indicates that the player to act does not know which of the two nodes she is in. (**Left**) The original imperfect-information subgame. (**Right**) The transformed subgame that is searched in real time to determine a player's strategy. An initial chance node reaches each root node according to the normalized probability that the node is reached in the previously computed strategy profile (or according to the blueprint strategy profile if this is the first time in the hand that real-time search is conducted). The leaf nodes are replaced by a sequence of new nodes in which each player still in the hand chooses among k actions, with no player first observing what another player chooses. For simplicity, $k = 2$ in the figure. In Pluribus, $k = 4$. Each action in that sequence corresponds to a selection of a continuation strategy for that player for the remainder of the game. This effectively leads to a terminal node (whose value is estimated by rolling out the remainder of the game according to the list of continuation strategies that the players chose).

and computation would enable a finer-grained blueprint that would lead to better performance but would also result in Pluribus using more memory or being slower during real-time search. We set the size of the blueprint strategy abstraction to allow Pluribus to run during live play on a machine with no more than 128 GB of memory while storing a compressed form of the blueprint strategy in memory.

Depth-limited search in imperfect-information games

The blueprint strategy for the entire game is necessarily coarse-grained owing to the size and complexity of no-limit Texas hold'em. Pluribus only plays according to this blueprint strategy in the first betting round (of four), where the number of decision points is small enough that the blueprint strategy can afford to not use information abstraction and have a lot of actions in the action abstraction. After the first round (and even in the first round if an opponent chooses a bet size that is sufficiently different from the sizes in the blueprint action abstraction), Pluribus instead conducts real-time search to determine a better, finer-grained strategy for the current situation it is in. For opponent bets on the first round that are slightly off the tree, Pluribus rounds the bet to a nearby on-tree size [using the pseudoharmonic mapping (*39*)] and proceeds to play according to the blueprint as if the opponent had used the latter bet size.

Real-time search has been necessary for achieving superhuman performance in many perfect-information games, including backgammon (*18*), chess (*8*), and Go (*9*, *19*). For example, when determining their next move, chess AIs commonly look some number of moves ahead until a leaf node is reached at the depth limit of the algorithm's lookahead. An evaluation function then estimates the value of the board configuration at the leaf node if both players were to play a Nash equilibrium from that point forward. In principle, if an AI could accurately calculate the value of every leaf node (e.g., win, draw, or loss), this algorithm would choose the optimal next move.

However, search as has been done in perfect-information games is fundamentally broken when applied to imperfect-information games. For example, consider a sequential form of Rock-Paper-Scissors, illustrated in Fig. 3, in which player 1 acts first but does not reveal her action to player 2, followed by player 2 acting. If player 1 were to conduct search that looks just one move ahead, every one of her actions would appear to lead to a leaf node with zero value. After all, if player 2 plays the Nash equilibrium strategy of choosing each action with $\frac{1}{3}$ probability, the value to player 1 of choosing Rock is zero, as is the value of choosing Scissors. So player 1's search algorithm could choose to always play Rock because, given the values of the leaf nodes, this appears to be equally good as any other strategy.

Indeed, if player 2's strategy were fixed to always playing the Nash equilibrium, always playing Rock would be an optimal player 1 strategy. However, in reality, player 2 could adjust to a strategy of always playing Paper. In that case, the value of always playing Rock would actually be -1.

This example illustrates that in imperfect-information subgames (the part of the game in which search is being conducted) (*40*), leaf nodes do not have fixed values. Instead, their values depend on the strategy that the searcher chooses in the subgame (that is, the probabilities that the searcher assigns to his actions in the subgame). In principle, this could be addressed by having the value of a subgame leaf node be a function of the searcher's strategy in the subgame, but this is impractical in large games. One alternative is to make the value of a leaf node conditional only on the belief distribution of both players at that point in the game. This was used to generate the two-player poker AI DeepStack (*5*). However, this option is extremely expensive because it requires one to solve huge numbers of subgames that are conditional on beliefs. It becomes even more expensive as the amount of hidden information or the number of players grows. The two-player poker AI Libratus sidestepped this issue by only doing real-time search when the remaining game was short enough that the depth limit would extend to the end of the game (*6*). However, as the number of players grows, always solving to the end of the game also becomes computationally prohibitive.

Pluribus instead uses a modified form of an approach that we recently designed—previously only for two-player zero-sum games (*41*)—in which the searcher explicitly considers that any or all players may shift to different strategies beyond the leaf nodes of a subgame. Specifically, rather than assuming that all players play according to a single fixed strategy beyond the leaf nodes (which results in the leaf nodes having a single fixed value), we instead assume that each player may choose between k different strategies, specialized to each player, to play for the remainder of the game when a leaf node is reached. In the experiments in this paper, $k = 4$. One of the four continuation strategies that we use in the experiments is the

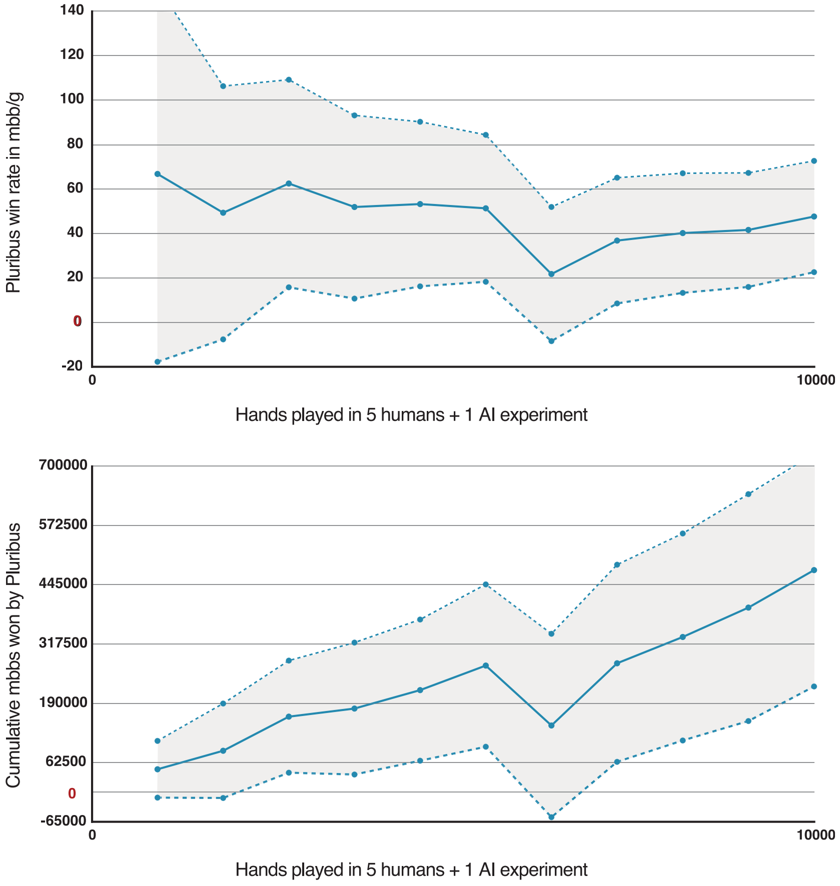

Fig. 5. Performance of Pluribus in the 5 humans + 1 AI experiment. The dots show Pluribus's performance at the end of each day of play. (**Top**) The lines show the win rate (solid line) plus or minus the standard error (dashed lines). (**Bottom**) The lines show the cumulative number of mbbs won (solid line) plus or minus the standard error (dashed lines). The relatively steady performance of Pluribus over the course of the 10,000-hand experiment also suggests that the humans were unable to find exploitable weaknesses in the bot.

precomputed blueprint strategy; another is a modified form of the blueprint strategy in which the strategy is biased toward folding; another is the blueprint strategy biased toward calling; and the final option is the blueprint strategy biased toward raising. This technique results in the searcher finding a strategy that is more balanced because choosing an unbalanced strategy (e.g., always playing Rock in Rock-Paper-Scissors) would be punished by an opponent shifting to one of the other continuation strategies (e.g., always playing Paper).

Another major challenge of search in imperfect-information games is that a player's optimal strategy for a particular situation depends on what the player's strategy is for every situation the player could be in from the perspective of her opponents. For example, suppose the player is holding the best possible hand. Betting in this situation could be a good action. But if the player bets in this situation only when holding the best possible hand, then the opponents would know that they should always fold in response.

To cope with this challenge, Pluribus keeps track of the probability it would have reached the current situation with each possible hand according to its strategy. Regardless of which hand Pluribus is actually holding, it will first calculate how it would act with every possible hand, being careful to balance its strategy across all the hands so as to remain unpredictable to the opponent. Once this balanced strategy across all hands is computed, Pluribus then executes an action for the hand it is actually holding. The structure of a depth-limited imperfect-information subgame as used in Pluribus is shown in Fig. 4.

Pluribus used one of two different forms of CFR to compute a strategy in the subgame, depending on the size of the subgame and the part of the game. If the subgame is relatively large or it is early in the game, then Monte Carlo Linear CFR is used just as it was for the blueprint strategy computation. Otherwise, Pluribus uses an optimized vector-based form of Linear CFR (*38*) that samples only chance events (such as board cards) (*42*).

When playing, Pluribus runs on two Intel Haswell E5-2695 v3 CPUs and uses less than 128 GB of memory. For comparison, AlphaGo used 1920 CPUs and 280 GPUs for real-time search in its 2016 matches against top Go professional Lee Sedol (*43*), Deep Blue used 480 custom-designed chips in its 1997 matches against top chess professional Garry Kasparov (*8*), and Libratus used 100 CPUs in its 2017 matches against top professionals in two-player poker (*6*). The amount of time that Pluribus takes to conduct search on a single subgame varies between 1 and 33 s, depending on the particular situation. On average, Pluribus plays at a rate of 20 s per hand when playing against copies of itself in six-player poker. This is roughly twice as fast as professional humans tend to play.

Experimental evaluation

We evaluated Pluribus against elite human professionals in two formats: five human professionals playing with one copy of Pluribus (5H+1AI), and one human professional playing with five copies of Pluribus (1H+5AI). Each human participant has won more than $1 million playing poker professionally. Performance was measured by using the standard metric in this field of AI, milli big blinds per game (mbb/game). This measures how many big blinds (the initial money the second player must put into the pot) were won on average per thousand hands of poker. In all experiments, we used the variance-reduction technique AIVAT (*44*) to reduce the luck factor in the game (*45*) and measured statistical significance at the 95% confidence level using a one-tailed *t* test to determine whether Pluribus is profitable.

The human participants in the 5H+1AI experiment were Jimmy Chou, Seth Davies, Michael Gagliano, Anthony Gregg, Dong Kim, Jason Les, Linus Loeliger, Daniel McAulay, Greg Merson, Nicholas Petrangelo, Sean Ruane, Trevor Savage, and Jacob Toole. In this experiment, 10,000 hands of poker were played over 12 days. Each day, five volunteers from the pool of professionals were selected to participate on the basis of availability. The participants were not told who else was participating in the experiment. Instead, each participant was assigned an alias that remained constant throughout the experiment. The alias of each player in each game was known, so that players could track the tendencies of each player throughout the experiment. $50,000 was divided among the human participants on the basis of their performance to incentivize them to play their best. Each player was guaranteed a minimum of $0.40 per hand for participating, but this could increase to as much as $1.60 per hand on the basis of performance. After applying AIVAT, Pluribus won an average of 48 mbb/game (with a standard error of 25 mbb/game). This is considered a very high win rate in six-player no-limit Texas hold'em poker, especially against a collection of elite professionals, and implies that Pluribus is stronger than the human opponents. Pluribus was determined to be profitable with a p value of 0.028. The performance of Pluribus over the course of the experiment is shown in Fig. 5. (Owing to the extremely high variance in no-limit poker and the impossibility of applying AIVAT to human players, the win rate of individual human participants could not be determined with statistical significance.)

The human participants in the 1H+5AI experiment were Chris "Jesus" Ferguson and Darren Elias. Each of the two humans separately played 5000 hands of poker against five copies of Pluribus. Pluribus does not adapt its strategy to its opponents and does not know the identity of its opponents, so the copies of Pluribus could not intentionally collude against the human player. To incentivize strong play, we offered each human \$2000 for participation and an additional \$2000 if he performed better against the AI than the other human player did. The players did not know who the other participant was and were not told how the other human was performing during the experiment. For the 10,000 hands played, Pluribus beat the humans by an average of 32 mbb/game (with a standard error of 15 mbb/game). Pluribus was determined to be profitable with a p value of 0.014. (Darren Elias was behind Pluribus by 40 mbb/game with a standard error of 22 mbb/game and a p value of 0.033, and Chris Ferguson was behind Pluribus by 25 mbb/game with a standard error of 20 mbb/game and a p value of 0.107. Ferguson's lower loss rate may be a consequence of variance, skill, and/or his use of a more conservative strategy that was biased toward folding in unfamiliar difficult situations.)

Because Pluribus's strategy was determined entirely from self-play without any human data, it also provides an outside perspective on what optimal play should look like in multiplayer no-limit Texas hold'em. Pluribus confirms the conventional human wisdom that limping (calling the "big blind" rather than folding or raising) is suboptimal for any player except the "small blind" player who already has half the big blind in the pot by the rules, and thus has to invest only half as much as the other players to call. Although Pluribus initially experimented with limping when computing its blueprint strategy offline through self-play, it gradually discarded this action from its strategy as self-play continued. However, Pluribus disagrees with the folk wisdom that "donk betting" (starting a round by betting when one ended the previous betting round with a call) is a mistake; Pluribus does this far more often than professional humans do.

Conclusions

Forms of self-play combined with forms of search have led to a number of high-profile successes in perfect-information two-player zero-sum games. However, most real-world strategic interactions involve hidden information and more than two players. This makes the problem very different and considerably more difficult both theoretically and practically. Developing a superhuman AI for multiplayer poker was a widely recognized milestone in this area and the major remaining milestone in computer poker. In this paper we described Pluribus, an AI capable of defeating elite human professionals in six-player no-limit Texas hold'em poker, the most commonly played poker format in the world. Pluribus's success shows that despite the lack of known strong theoretical guarantees on performance in multiplayer games, there are large-scale, complex multiplayer imperfect-information settings in which a carefully constructed self-play-with-search algorithm can produce superhuman strategies.

REFERENCES AND NOTES

1. D. Billings, A. Davidson, J. Schaeffer, D. Szafron, *Artif. Intell.* **134**, 201–240 (2002).
2. J. von Neumann, *Math. Ann.* **100**, 295–320 (1928).
3. J. Nash, *Ann. Math.* **54**, 286 (1951).
4. M. Bowling, N. Burch, M. Johanson, O. Tammelin, *Science* **347**, 145–149 (2015).
5. M. Moravčík *et al.*, *Science* **356**, 508–513 (2017).
6. N. Brown, T. Sandholm, *Science* **359**, 418–424 (2018).
7. J. Schaeffer, *One Jump Ahead: Challenging Human Supremacy in Checkers* (Springer-Verlag, New York, 1997).
8. M. Campbell, A. J. Hoane Jr., F.-H. Hsu, *Artif. Intell.* **134**, 57–83 (2002).
9. D. Silver *et al.*, *Nature* **529**, 484–489 (2016).
10. Recently, in the real-time strategy games Dota 2 (*20*) and StarCraft 2 (*21*), AIs have beaten top humans, but as humans have gained more experience against the AIs, humans have learned to beat them. This may be because for those two-player zero-sum games, the AIs were generated by techniques not guaranteed to converge to a Nash equilibrium, so they do not have the unbeatability property that Nash equilibrium strategies have in two-player zero-sum games. (Dota 2 involves two teams of five players each. However, because the players on the same team have the same objective and are not limited in their communication, the game is two-player zero-sum from an AI and game-theoretic perspective.)
11. S. Ganzfried, T. Sandholm, in *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)* (2011), pp. 533–540.
12. S. Ganzfried, T. Sandholm, *ACM Trans. Econ. Comp. (TEAC)* **3**, 8 (2015). Best of EC-12 special issue.
13. C. Daskalakis, P. W. Goldberg, C. H. Papadimitriou, *SIAM J. Comput.* **39**, 195–259 (2009).
14. X. Chen, X. Deng, S.-H. Teng, *J. Assoc. Comput. Mach.* **56**, 14 (2009).
15. A. Rubinstein, *SIAM J. Comput.* **47**, 917–959 (2018).
16. K. Berg, T. Sandholm, *AAAI Conference on Artificial Intelligence (AAAI)* (2017).
17. M. A. Zinkevich, M. Bowling, M. Wunder, *ACM SIGecom Exchanges* **10**, 35–38 (2011).
18. G. Tesauro, *Commun. ACM* **38**, 58–68 (1995).
19. D. Silver *et al.*, *Nature* **550**, 354–359 (2017).
20. OpenAI, OpenAI Five, https://blog.openai.com/openai-five/ (2018).
21. O. Vinyals *et al.*, AlphaStar: Mastering the Real-Time Strategy Game StarCraft II, https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/ (2019).
22. L. S. Shapley, *Advances in Game Theory*, M. Drescher, L. S. Shapley, A. W. Tucker, Eds. (Princeton Univ. Press, 1964).
23. R. Gibson, Regret minimization in games and the development of champion multiplayer computer poker-playing agents, Ph.D. thesis, University of Alberta (2014).
24. T. Sandholm, in *AAAI Conference on Artificial Intelligence (AAAI)* (2015), pp. 4127–4131. Senior Member Track.
25. T. Sandholm, *Science* **347**, 122–123 (2015).
26. M. Johanson, N. Burch, R. Valenzano, M. Bowling, in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2013), pp. 271–278.
27. S. Ganzfried, T. Sandholm, in *AAAI Conference on Artificial Intelligence (AAAI)* (2014), pp. 682–690.
28. N. Brown, S. Ganzfried, T. Sandholm, in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2015), pp. 7–15.
29. M. Zinkevich, M. Johanson, M. H. Bowling, C. Piccione, in *Neural Information Processing Systems (NeurIPS)* (2007), pp. 1729–1736.
30. E. G. Jackson, *AAAI Workshop on Computer Poker and Imperfect Information* (2013).
31. M. B. Johanson, Robust strategies and counter-strategies: from superhuman to optimal play, Ph.D. thesis, University of Alberta (2016).
32. E. G. Jackson, *AAAI Workshop on Computer Poker and Imperfect Information* (2016).
33. N. Brown, T. Sandholm, in *International Joint Conference on Artificial Intelligence (IJCAI)* (2016), pp. 4238–4239.
34. E. G. Jackson, *AAAI Workshop on Computer Poker and Imperfect Information Games* (2017).
35. M. Lanctot, K. Waugh, M. Zinkevich, in M. Bowling, *Neural Information Processing Systems (NeurIPS)* (2009), pp. 1078–1086.
36. M. Johanson, N. Bard, M. Lanctot, R. Gibson, M. Bowling, in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2012), pp. 837–846.
37. R. Gibson, M. Lanctot, N. Burch, D. Szafron, M. Bowling, in *AAAI Conference on Artificial Intelligence (AAAI)* (2012), pp. 1355–1361.
38. N. Brown, T. Sandholm, *AAAI Conference on Artificial Intelligence (AAAI)* (2019).
39. S. Ganzfried, T. Sandholm, in *International Joint Conference on Artificial Intelligence (IJCAI)* (2013), pp. 120–128.
40. Here we use the term "subgame" the way it is usually used in AI. In game theory, that word is used differently by requiring a subgame to start with a node where the player whose turn it is to move has no uncertainty about state—in particular, no uncertainty about the opponents' private information.
41. N. Brown, T. Sandholm, B. Amos, in *Neural Information Processing Systems (NeurIPS)* (2018), pp. 7663–7674.
42. M. Johanson, K. Waugh, M. Bowling, M. Zinkevich, in *International Joint Conference on Artificial Intelligence (IJCAI)* (2011), pp. 258–265.
43. E. P. DeBenedictis, *Computer* **49**, 84–87 (2016).
44. N. Burch, M. Schmid, M. Moravcik, D. Morill, M. Bowling, in *AAAI Conference on Artificial Intelligence (AAAI)* (2018), pp. 949–956.
45. Owing to the presence of AIVAT and because the players did not know each others' scores during the experiment, there was no incentive for the players to play a risk-averse or risk-seeking strategy to outperform the other human.

ACKNOWLEDGMENTS

We thank P. Ringshia for building a graphical user interface and thank J. Chintagunta, B. Clayman, A. Du, C. Gao, S. Gross, T. Liao, C. Kroer, J. Langas, A. Lerer, V. Raj, and S. Wu for playing against Pluribus as early testing. **Funding:** This material is based on Carnegie Mellon University research supported by the National Science Foundation under grants IIS-1718457, IIS-1617590, IIS-1901403, and CCF-1733556 and by the ARO under award W911NF-17-1-0082, as well as XSEDE computing resources provided by the Pittsburgh Supercomputing Center. Facebook funded the player payments. **Author contributions:** N.B. and T.S. designed the algorithms. N.B. wrote the code. N.B. and T.S. designed the experiments and wrote the paper. **Competing interests:** The authors have ownership interest in Strategic Machine, Inc., and Strategy Robot, Inc., which have exclusively licensed prior game-solving code from the Carnegie Mellon University laboratory of T.S., which constitutes the bulk of the code in Pluribus. **Data and materials availability:** The data presented in this paper are shown in the main text and supplementary materials. Because poker is played commercially, the risk associated with releasing the code outweighs the benefits. To aid reproducibility, we have included the pseudocode for the major components of our program in the supplementary materials.

SUPPLEMENTARY MATERIALS

science.sciencemag.org/content/365/6456/885/suppl/DC1
Supplementary Text
Table S1
References (*46*–*52*)
Data File S1

31 May 2019; accepted 2 July 2019
Published online 11 July 2019
10.1126/science.aay2400

Science

Superhuman AI for multiplayer poker

Noam Brown and Tuomas Sandholm

Science **365** (6456), 885-890.
DOI: 10.1126/science.aay2400originally published online July 11, 2019

AI now masters six-player poker

Computer programs have shown superiority over humans in two-player games such as chess, Go, and heads-up, no-limit Texas hold'em poker. However, poker games usually include six players––a much trickier challenge for artificial intelligence than the two-player variant. Brown and Sandholm developed a program, dubbed Pluribus, that learned how to play six-player no-limit Texas hold'em by playing against five copies of itself (see the Perspective by Blair and Saffidine). When pitted against five elite professional poker players, or with five copies of Pluribus playing against one professional, the computer performed significantly better than humans over the course of 10,000 hands of poker.

Science, this issue p. 885; see also p. 864

Science (print ISSN 0036-8075; online ISSN 1095-9203) is published by the American Association for the Advancement of Science, 1200 New York Avenue NW, Washington, DC 20005. The title *Science* is a registered trademark of AAAS.