Midgame Solving: A New Weapon for Efficient Large-Scale Equilibrium Approximation

Kailiang Hu
Shenzhen Graduate School
Harbin Institute of Technology
Email: hukailiang@cs.hitsz.edu.cn

Sam Ganzfried
School of Computing and Information Sciences
Florida International University
Miami, Florida
Email: sganzfri@cis.fiu.edu

***Abstract*—Standard subdivision techniques, like backwards induction, are successful in perfect-information games; however, in imperfect-information games they have no theoretical justification. Despite this lack of theory, the strongest agents for large imperfect-information games have employed "endgame solving" and have recently beaten human professional players in poker using this approach. We present an agent that uses a modification, which can be viewed as a generalization of endgame solving, called midgame solving. While endgame solving solves the final portion of a game, assuming strategies prior to the endgame as inputs using Bayes' rule, our approach solves an arbitrary portion of the game, which could be an intermediate portion with further play remaining subsequently. One of the main challenges is determining the payoffs to use for the midgame, while terminal payoffs are used for endgames. We also present a novel approach for interpreting opponents' off-tree actions that is more robust than prior approaches. We describe an agent for no-limit Texas hold 'em utilizing our approaches and preliminary experiments against an agent from the Annual Computer Poker Competition that uses popular state-of-the-art techniques such as CFR+.**

I. Introduction

In theory two-player zero-sum extensive-form games can be solved (for Nash equilibrium) in polynomial time (e.g., [1]). However, standard linear programming techniques only scale to games with around 10^8 states (the algorithms run out of memory for larger games). More recently algorithms have been developed for approximating equilibrium strategies (they converge to equilibrium in the limit) that scale to 10^{15} states [2]. However, even this is a far cry from the size of many interesting games; for example, the version of two-player no-limit Texas hold 'em played in the AAAI Annual Computer Poker Competition has approximately 10^{165} states [3].

Consequently there has been a need for approaches that somehow enable us to approximate full-game solutions despite the fact that we can only solve significantly smaller games. Two main paradigms have been developed for accomplishing this. The first one, called the *abstraction paradigm*, approximates the original game by a smaller game that hopefully retains much of the strategic structure of the initial game. Then the abstract game is solved by an equilibrium-finding algorithm [4], [5], [6], which is then mapped back to a strategy profile in the full game. The first abstractions for two-player Texas hold 'em were manually generated [7], [8], while current abstractions are computed automatically [9], [10], [11]. The abstraction paradigm is depicted in Figure 1.

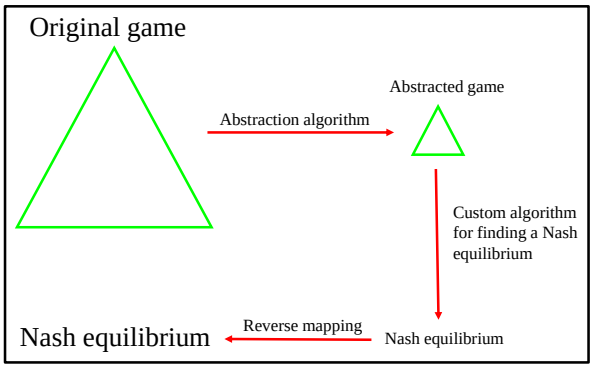

Fig. 1. Abstraction paradigm for solving large games.

Recently a new paradigm has been developed which involves first approximating full solutions in an abstraction offline, then solving the relevant portion of the game that we have reached in real time to a greater degree of accuracy than in the offline computation. This paradigm, called *endgame solving*, is depicted in Figure 2. For example, we can consider endgames in poker where several rounds of betting have taken place and several public cards have already been dealt. We can assume players have a joint distribution of private information from nodes prior to the endgame (i.e., the *trunk*) that are induced from precomputed base trunk strategies using Bayes' rule. Given this distribution as input, we can then solve individual endgames in real time more accurately.[1]

Definition 1. *E is an* endgame *of game G if the following properties hold:*

1) *The set of E's nodes is a subset of the set of G's nodes.*
2) *If s' is a child of s in G and s is a node in E, then s' is also a node in E.*
3) *If s is in the same information set as s' in G and s is a node in E, then s' is also a node in E.*

Unfortunately, this approach has fundamental flaws. It turns out that even if we computed an exact equilibrium in the

[1]We can view the endgames as new games where nature makes an initial move by assigning private information according to the joint distribution induced by the trunk strategies.

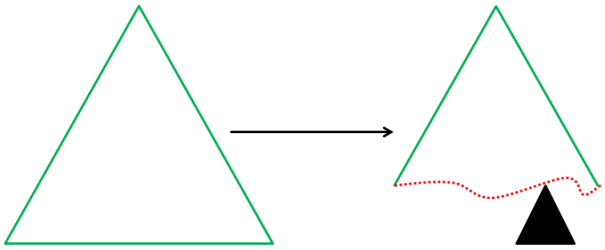

Fig. 2. *Endgame solving* discards the strategies that were precomputed for the endgames, then (re-)solves the relevant endgame that we have actually reached in real time to a greater degree of accuracy than in the offline computation.

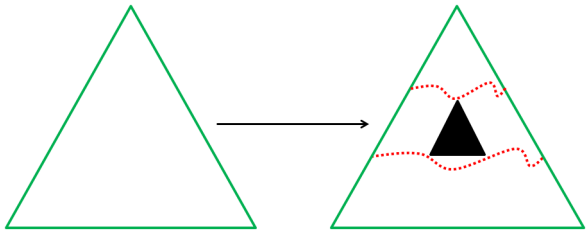

Fig. 3. *Midgame solving* discards the strategies that were precomputed for the midgames, then (re-)solves the relevant midgame that we have actually reached in real time to a greater degree of accuracy than in the offline computation, assuming a given payoff mapping for midgame terminal nodes.

trunk (which is an unrealistically optimistic assumption in large games) and in the endgame, the combined strategies for the trunk and endgame may fail to be an equilibrium in the full game. One obvious reason for this is that the game may contain many equilibria, and we might choose one for the trunk that does not match up correctly with the one for the endgame; or we may compute different equilibria in different endgames that do not balance appropriately. However, Proposition 1 shows that it is possible for this procedure to output a non-equilibrium strategy profile in the full game even if the full game has a unique equilibrium and a single endgame [12]. The proof follows by considering a sequential version of rock-paper-scissors where player 1 acts first, then player 2 acts in a single endgame where has not observed the move made by player 1.

Proposition 1. *There exist games—even with a unique equilibrium and a single endgame—for which endgame solving can produce a non-equilibrium strategy profile.*

Some early research used endgame solving for limit Texas hold 'em agents out of necessity due to limited scalability of existing approaches [9], [13]. But for the next several years it was abandoned in favor of offline approaches that solve abstracted versions of the entire game due to the shortcomings described above. However, despite the adequacy of these holistic approaches, endgame solving was reinvestigated and applied to no-limit Texas hold 'em in 2013 due to the potential benefits of focused computation on the portion of the game tree that has been reached [14], [12]. This approach was used by the agent Claudico that competed in the inaugural Brains vs. AI competition against the strongest human two-player no-limit Texas hold 'em specialists in the world [15]. In fact, the best human player in the world, Doug Polk, has relayed to me in personal communication that the final round strategy of Claudico computed by the endgame solver was the strongest component. Fueled by this promise, there has been a flurry of subsequent research further exploring this new paradigm [16], [17], culminating in two agents DeepStack and Libratus that were recently able to successfully defeat human professional players [18], [19]. These agents both apply endgame solving in very different ways: Libratus used a supercomputer to solve both the turn and river rounds in real time while DeepStack viewed all rounds as independent endgames with leaf payoff values estimated using deep learning [20].

In this paper we introduce a new paradigm, which can be viewed as a generalization of endgame solving, called *midgame solving*. In midgame solving, an intermediate portion of the game tree that we have reached is solved in real time. This paradigm is depicted in Figure 3. We define a midgame in Definition 2. Note that endgame is a special case where the midgame happens to be at the end of the game. An important challenge for midgame solving that does not arise for endgame solving is selecting the payoffs. For endgame solving we are already at the end of the game so we can use the full-game terminal payoffs, but for midgame solving we need a mapping that approximates payoffs at each midgame terminal node.

Definition 2. *M is a* midgame *of game G if the following properties hold:*

1) *The set of M's nodes is a subset of the set of G's nodes.*
2) *If s is in the same information set as s' in G and s is a node in M, then s' is also a node in M.*
3) *If s and s' are nodes in M and there is a path from s to s', then all nodes along that path are also in M.*

Clearly midgame solving will suffer the same theoretical flaws as endgame solving, as highlighted by Proposition 1. However, it could also similarly experience significant benefits. Note that this approach was initially proposed in the author's thesis proposal [21]. The recent decomposed resolving approach used by the DeepStack agent can be viewed as one instantiation of midgame solving [19], although that approach was offline while we propose to solve the relevant midgame that we have reached in real time. DeepStack used a complex deep learning technique to compute the midgame payoffs. This training required approximately 175 core years, or several hundred computers for several months. Another recent approach called nested endgame solving was used by agent Libratus [18]. For this approach one larger endgame is solved in real time, followed by a smaller one; this differs from our approach and can just use terminal payoffs for all endgames. Furthermore it required around 25 million core hours on a supercomputer and took around 20 seconds per endgame. By contrast we have created an agent based on our approach that runs in seconds on one machine. It uses a new value mapping function we have developed for evaluating midgame terminal nodes in the poker instantiation.

II. General midgame-solving algorithm

Our general midgame solving algorithm is depicted in Figure 1. The algorithm assumes we have precomputed a trunk strategy for the portion of the game prior to the midgames, which can be computed using any approach. We follow the trunk strategy in real time until we arrive at a midgame m_i, which we then solve using some algorithm a_i with payoff mapping v_i. The midgame is defined by using Bayes' rule for the input distributions of private information for both players entering the midgame assuming they had followed the trunk strategy. The solution to m_i is then played until we reach a new midgame m_j, and so on. Eventually we will compute a strategy at a terminal midgame (i.e., an endgame), after which point the game will conclude. Note that for the concluding midgames a payoff mapping function is not needed. Note that this approach is very general, and could allow for a wide variety of subdivisions of the game into a trunk and midgames, different payoff functions for different midgames, and different equilibrium-solving algorithms for different midgames. For the poker application, we will assume the trunk strategy for the first round (called *preflop*), and solve for new midgame strategies each time we enter one of the new postflop rounds (the *flop*, *turn*, and *river*). Thus, during a single path of play we will be solving at most three midgames. In comparison, the agent Claudico solved for just the river endgame in real time, and the agent Libratus applied nested endgame solving to solve for endgames in both the turn and river, sometimes solving for multiple endgames starting within a single round depending on the actions taken by the opponent.

Algorithm 1 Algorithm for midgame solving

Inputs: Game G, trunk strategy t, set of midgames $\{m_i\}$ for $1 \leq i \leq K$, midgame mapping m assigning each node n to a midgame m_i, payoff mapping $v_i(j)$ which maps the j'th terminal node of midgame m_i to a vector of real payoffs one per player, algorithms $\{a_i\}$ for $1 \leq i \leq K$ for computing Nash equilibrium in midgame i given payoff mapping v_i.

Follow relevant component of t for the trunk strategy until a leaf node or a node from some midgame m_{k1} is reached.
if a leaf node has been reached **then**
  return
end if
$i \leftarrow 1$
while true do
  Compute the solution s_i^* to game defined by midgame $m'_i = m_{ki}$ and v_{ki} using algorithm a_{ki}.
  Play according to s_i^* until we reach either a leaf node or a node from some midgame $m_{k,i+1}$ that is not m'_i.
  if a leaf node has been reached **then**
    return
  end if
  $i \leftarrow i + 1$
end while

III. Interpolated action translation for robust midgame input distribution construction

In order to apply the midgame-solving algorithm, we need a technique to construct the input hand distributions entering the midgame by applying Bayes' rule on the trunk strategy. Often this is relatively straightforward. If we have a base strategy for the trunk, we can assume both ourselves and the opponent followed the base strategy, and use the induced distributions. It turns out that the naïve approach for doing this requires O(n^2) strategy table lookups making the computation too slow; however, an improved approach is able to do this with just O(n) strategy table lookups [12]; pseudocode for this approach from that paper is given in Algorithm 2. For more details of the specific indexing schemes used see that paper. Note that this approach was for endgame solving, but the same approach is applicable to constructing midgame input distributions.

Algorithm 2 Algorithm for computing hand distributions

Inputs: Public board B; number of private hands H; betting history of current hand h; array of index conflicts IC[][]; base strategy s^*

$D_1, D_2 \leftarrow$ array of dimension H of zeroes
for $p_1 = 0$ to 50, p_1 not already on B **do**
  for $p_2 = p_1 + 1$ to 51, p_2 not already on B **do**
    $I \leftarrow$ IndexFull(B, p_1, p_2)
    IndexMap[I] $\leftarrow$ IndexHoles(p_1, p_2)
    $P_1 \leftarrow$ probability P1 would play according to h with p_1, p_2 in s^*
    $P_2 \leftarrow$ probability P2 would play according to h with p_1, p_2 in s^*
    $D_1[I]$ += P_1, $D_2[I]$ += P_2
  end for
end for
Normalize D_1 and D_2 so all entries sum to 1
for $i = 0$ to H **do**
  for $j = 0$ to H **do**
    if !IC[IndexMap[i]][IndexMap[j]] **then**
      $D[i][j] \leftarrow D_1[i] \cdot D_2[j]$
    else
      $D[i][j] \leftarrow 0$
    end if
  end for
end for
Normalize D so all entries sum to 1 **return** D

A key step is when the strategy probabilities P_1, P_2 are computed. This is straightforward when the opponent's actions are within the action abstraction we have used. However, it is more complex when the opponent has made an off-tree bet size. The standard approach is to apply an action translation mapping to map the opponent's action to one of the neighboring actions in the abstraction (Figure 4). The most successful recent mapping is the pseudoHarmonic mapping [22], which was derived from analytical solutions of simplified poker games. Assuming A, B are the two closest sizes to bet x and $A < x < B$, then this will map x to A with probability $f_{A,B}(x) = \frac{(B-x)(1+A)}{(B-A)(1+x)}$.

Assume we are constructing the hand distributions for the flop midgame. For our play during preflop, each time the opponent takes an off-tree action, we apply the translation mapping we have chosen to interpret his action as an abstract

Fig. 4. Action translation paradigm for large games. Off-tree action x is mapped probabilistically to closest actions in abstraction A, B.

action, and play as if the opponent had taken that abstract action. The standard endgame-solving approach from Algorithm 2 will then also assume the opponent has taken the same deterministic abstract action for the construction of P_i. However, during the course of our testing we uncovered a shortcoming of this approach. Using initial blinds of small blind (SB) = 50, big blind (BB) = 100, the opponent raised to 800 (an unusually large size—more common would be 200 or 300). The two neighboring actions in our abstraction in the situation are $A = 400, B = 1000$; however, it turns out that the trunk strategy did not actually raise to 1000 with any hands at all, though he raised to 400 with many hands. Therefore, had our translation mapping interpreted his bet as 1000, we would have assigned the opponent probability zero for all hands, leading the midgame to be poorly defined and solution would be meaningless (and this would propagate down to the midgames for future rounds as well). Note that our *strategy* for the trunk is still well defined against a bet of 1000; it is just the values for the midgame construction that are problematic.

In order to address this issue, we have developed a new approach that will ensure that the midgame distributions are robust to such "degenerate" behavior in portions of the trunk strategy. Rather than interpret the opponent's off-tree bet as a single deterministic action in the computation of P_i, we will instead take the weighted average of his strategy had we mapped it either to A or B. Our approach, depicted in Algorithm 3, takes an arbitrary trunk strategy and action translation mapping as input. In the example described above, the distributions would no longer be degenerate because we will be interpolating between the nonzero probabilities of raising to 400 in addition to the zeroes for 1000.[2]

Algorithm 3 Interpolated action translation algorithm

Inputs: Opponent's private information state x_{-i}, trunk action history h, trunk strategy for opponent s_{-i}, action translation mapping m, nearest action mapping n.

$p(x_{-i}) \leftarrow 1$
for opponent actions a_{-j} in h **do**
  $a', a'' \leftarrow n(a_{-j})$
  $p(x_{-i}) \leftarrow p(x_{-i}) \cdot [s_{-i}(a', h)m(a', h) + s_{-i}(a'', h)m(a'', h)]$
end for
return $p(x_{-i})$

Note that our new interpolated approach does not completely solve the degeneracy issue; for example, it is possible that the opponent would never raise to both 400 and 1000 in the situation described above, in which case even the new approach would give all hands probability zero. However, in general we would expect it to be much more robust than the prior approach, which would potentially fail each time there is even a single action with probability zero for all hands. We expect the benefit to be particularly high as more refined abstractions are used, which include a larger number of actions. An alternative approach could be to remove the action from the abstraction altogether *ex post* after we observe all hands have probability zero; however, this would also not always solve the problem, and could also lead to huge gaps between actions. For example, the next size beyond 1000 could be 20000 (for all-in), which would likely also have probability zero on all hands as well (it does for our trunk strategies).

[2]Note that the recent agent Libratus precluded the need for action translation within endgames by adding off-tree opponent actions and solving several new endgames [18]. However, this approach would not be applicable to the portion of the game prior to the endgame, and translation would still be required.

IV. No-limit Texas hold 'em agent

We have created an agent for two-player no-limit Texas hold 'em using our approach. For the trunk preflop round we use a fixed strategy that was precomputed. Then once the flop, turn, and river rounds are entered, we solve for the corresponding midgame assuming the trunk strategy. We use our new interpolated translation algorithm to construct the midgame input hand distributions for the opponent, as described in Section III. To solve the midgames we apply a version of CFR+ [23], which was the algorithm used to compute a near-optimal strategy for limit Texas hold 'em [24]. For the payoff mapping we assume that a player wins the full pot whenever the other player folds, and if the midgame ends with neither player folding then we assume both players achieve their equity in the pot (i.e., probability of winning plus one half probability of tying) assuming the remaining public cards are dealt uniformly at random. These payoffs were precomputed offline for the flop round, and computed in real time for the turn (for the river the payoffs are terminal and there are no future midgames to be solved).

The runtime of the midgame-solving algorithm would depend on the number of bet sizes used in the midgame and the specific implementation. Our current version runs in around 30 seconds per midgame (it is several seconds longer for the turn than the other rounds because we are computing the table for the value payoff mapping in real time), though we know that it can be significantly optimized. For instance, commercial software PIOSolver runs in just milliseconds using a gradient-based algorithm to solve for a river endgame using multiple bet sizes [25], and therefore would also run at a similar speed for single-round midgames with a payoff function. Unfortunately their code is not publicly available (nor is code for CFR+ for extensive-form games), so we have implemented our own version. In the future we plan to optimize our algorithm so performance is competitive with the state-of-the-art solvers.

We have run experiments against a previous agent from the AAAI Annual Computer Poker Competition [26] called HITSZ-HKL. HITSZ-HKL, which was submitted to the 2017 competition, was trained by an experimental distributed implementation of the Pure CFR algorithm. In incorporated additional techniques for pruning, reducing the time spent per training iteration. It use card abstraction (public card

clustering) and betting abstraction. It uses the vector form for CFR+ with alternating updates and uses an efficient O(n) technique for efficient terminal node reward evaluation, assuming n information sets per player (as opposed to the standard technique which is O(n^2)). In the experiments, HITSZ-HKL won at a rate of 0.288 big blinds per hand (or 288 milli big blinds per hand), over a sample of 23 duplicate matches each with 3000 hands. The 95% confidence interval was 0.097 big blinds per hand, so these results are statistically significant.

V. Conclusion

We are in the process of improving our agent in various dimensions and plan to run more comprehensive experiments against HITSZ-HKL as well as several other previous agents from the AAAI Annual Computer Poker Competition [26]. We admit that this research is in a very preliminary phase and that the agent can be significantly improved on in many ways. The most obvious improvement would be to develop other more sophisticated value functions. While we have selected a particular simple value mapping function to use for our agent, our approach applies to arbitrary value functions, and could be integrated with significantly more complex functions, such as those computed with deep learning. We also plan to obtain a stronger preflop strategy to use for the trunk strategy; this is very important, because the preflop hand probabilities are propagated down as the basis of all the midgame input distributions, and poor preflop strategies would likely serve as a very inaccurate model for the opponent's distribution. Furthermore we would like the computed preflop strategies to have more bet sizes and fewer "probability zero" situations, in order to make the hand distributions for the midgame more robust. As described in Section IV, we plan to significantly optimize our agent's runtime, which we know is possible based on the efficiency of commercial solvers. We also plan to run comprehensive experiments to isolate out the improvement obtained using our new interpolated action translation approach as opposed to the standard paradigm.

While no-limit Texas hold 'em agents have already been developed that defeat human professional players ([18], [19]), these approaches appear to be heavily reliant on access to massive computational resources. Despite the relatively poor performance of our agent so far in preliminary experiments, we think that eventually it could have several advantages over these prior agents after significant improvements are made, since our approach runs in seconds on a single machine and can potentially be optimized to run in only milliseconds.

References

[1] D. Koller and N. Megiddo, "The complexity of two-person zero-sum games in extensive form," *Games and Economic Behavior*, vol. 4, no. 4, pp. 528–552, Oct 1992.

[2] N. Brown, S. Ganzfried, and T. Sandholm, "Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit Texas Hold'em agent," in *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2015.

[3] M. Johanson, "Measuring the size of large no-limit poker games," University of Alberta, Tech. Rep., 2013.

[4] S. Hoda, A. Gilpin, J. Peña, and T. Sandholm, "Smoothing techniques for computing Nash equilibria of sequential games," *Mathematics of Operations Research*, vol. 35, no. 2, pp. 494–512, 2010.

[5] M. Zinkevich, M. Bowling, M. Johanson, and C. Piccione, "Regret minimization in games with incomplete information," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2007.

[6] M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling, "Monte Carlo sampling for regret minimization in extensive games," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2009.

[7] J. Shi and M. Littman, "Abstraction methods for game theoretic poker," in *CG '00: Revised Papers from the Second International Conference on Computers and Games*. London, UK: Springer-Verlag, 2002.

[8] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron, "Approximating game-theoretic optimal strategies for full-scale poker," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[9] A. Gilpin and T. Sandholm, "A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

[10] K. Waugh, M. Zinkevich, M. Johanson, M. Kan, D. Schnizlein, and M. Bowling, "A practical use of imperfect recall," in *Proceedings of the Symposium on Abstraction, Reformulation and Approximation (SARA)*, 2009.

[11] M. Johanson, N. Burch, R. Valenzano, and M. Bowling, "Evaluating state-space abstractions in extensive-form games," in *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2013.

[12] S. Ganzfried and T. Sandholm, "Endgame solving in large imperfect-information games," in *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2015.

[13] A. Gilpin and T. Sandholm, "Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker," in *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2007.

[14] S. Ganzfried and T. Sandholm, "Improving performance in imperfect-information games with large state and action spaces by solving endgames," in *AAAI Workshop on Computer Poker and Incomplete Information*, 2013.

[15] S. Ganzfried, "Reflections on the first man versus machine no-limit Texas hold 'em competition," *AI Magazine*, vol. 38, no. 2, 2017.

[16] N. Burch, M. Johanson, and M. Bowling, "Solving imperfect information games using decomposition," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[17] M. Moravcik, M. Schmid, K. Ha, M. Hladik, and S. J. Gaukrodger, "Solving imperfect information games using decomposition," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[18] N. Brown and T. Sandholm, "Safe and nested endgame solving in imperfect-information games," in *AAAI Workshop on Computer Poker and Imperfect Information Games*, 2017.

[19] M. Moravcík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. H. Bowling, "Deepstack: Expert-level artificial intelligence in no-limit poker," *CoRR*, vol. abs/1701.01724, 2017.

[20] S. Ganzfried, "Endgame solving: The surprising breakthrough that enabled superhuman two-player no-limit Texas hold 'em play," in *International Conference on Game Theory*, 2017.

[21] ——, "Computing strong game-theoretic strategies and exploiting suboptimal opponents in large games," 2013, thesis Proposal, Carnegie Mellon University, Computer Science Department.

[22] S. Ganzfried and T. Sandholm, "Action translation in extensive-form games with large action spaces: Axioms, paradoxes, and the pseudo-harmonic mapping," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[23] O. Tammelin, "Solving large imperfect information games using CFR+," *CoRR*, vol. abs/1407.5042, 2014.

[24] M. Bowling, N. Burch, M. Johanson, and O. Tammelin, "Heads-up limit hold'em poker is solved," *Science*, vol. 347, no. 6218, pp. 145–149, January 2015.

[25] "Piosolver," https://piosolver.com/.

[26] Http://www.computerpokercompetition.org/.