Autonomous Agents and Multi-Agent Systems (2024) 38:31
https://doi.org/10.1007/s10458-024-09660-x

Navigating in a space of game views

Michael P. Wellman[1] · **Katherine Mayo**[1]

Accepted: 13 June 2024 / Published online: 6 July 2024

Abstract
Game-theoretic modeling entails selecting the particular elements of a complex strategic situation deemed most salient for strategic analysis. Recognizing that any game model is one of many possible views of the situation, we term this a *game view*, and propose that sophisticated game reasoning would naturally consider multiple views. We introduce a conceptual framework, *game view navigation*, for game-theoretic reasoning through a process of constructing and analyzing a series of game views. The approach is illustrated using a variety of existing methods, which can be cast in terms of navigation patterns within this framework. By formally defining these as well as recently introduced ideas as navigating in a space of game views, we recognize common themes and opportunities for generalization. Game view navigation thus provides a unifying perspective that sheds light on connections between disparate reasoning methods, and defines a design space for creation of new techniques. We further apply the framework by defining and exploring new techniques based on modulating player aggregation in equilibrium search.

Keywords Game theory · Equilibrium computation · Multi-level reasoning

1 Introduction

Advances in game-theoretic reasoning over the past two decades have greatly expanded the scale and scope of multiagent systems subject to analysis with the tools of game theory. Such advances include: (1) compact representations for complex strategic interactions exploiting regularity in structure [1–4], (2) algorithmic improvements, including new techniques for abstraction and approximation [5] and depth-limited search [6, 7], (3) modeling methodologies based on empirical methods such as agent-based simulation [8] and machine learning [9], and (4) powerful new methods for strategy generation, particularly employing deep reinforcement learning [10, 11] and stochastic tree search [12]. Altogether, the game reasoner's toolkit is a great deal larger and varied than it was 20 years ago.

✉ Michael P. Wellman
wellman@umich.edu

Katherine Mayo
kamayo@umich.edu

[1] University of Michigan, Ann Arbor, USA

The common perspective—typically implicit—in game reasoning work is that there is a central object: a *game*, to be solved. This is a convenient fiction. In reality, most agents that we want to reason about live in a strategic environment extended in time, potentially influenced by actions of myriad other agents they or we might not even know about yet. Practically, to make some decisions, we must draw some lines and bound attention—temporally, spatially, socially, topically—focusing on the most salient strategic interactions and trusting that the others are sufficiently unimportant or independent to ignore in the operative context. We might also choose to cover broad swaths of interactions in an aggregated or abstracted way, trading fidelity for tractability.

Such modeling judgments are inescapable. In their classic text, Osborne and Rubinstein [13] express the standard view that "models of game theory are highly abstract representations of classes of real-life situations". Like most (if not all) game theory texts, theirs focuses on the mathematical constructs and logic of the theory, rather than issues of modeling judgment. As they say on page 1, "The art of applying an abstract model to a real-life situation should be the subject of another tome."

Such a tome, if it existed, would be quite interesting. It would by necessity address a host of complex tradeoffs weighing information gathering and specification effort, as well as computation and accuracy, against performance features like decision quality and timeliness. For starters, in a given strategic situation:

- *how far* in the future must an agent plan and reason about behaviors and reactions of others,
- *which others* should be considered, and
- *at what level* of detail?

An ambitious AI developer naturally wishes to make modeling choices explicit and place them under algorithmic control. In some contexts, it is possible to assess tradeoffs quantitatively, for example theoretical bounds on errors due to certain forms of game abstraction [14]. Gilpin and Sandholm [15] used integer programming to select an optimal abstraction scheme for poker game trees, given complexity constraints. More generally, evaluating the tradeoffs to construct an optimal game-theoretic model is beyond current capabilities.

In contrast, generating candidate game models within a defined space is often relatively straightforward. Systematic exploration in this model space can be informative, even if we are unable to determine the "best" model. Different models make different tradeoffs, and the weaknesses of one model may line up with strengths of another. Considering an *ensemble* of models is one way to cover a broad set of perspectives, even when a single model for the union of perspectives is intractable. Page [16] calls this the "many-model thinker" approach, and argues that being informed by a diverse set of logically consistent frames helps us make sense of complexity and contributes to wiser decisions.

Many-model thinking encourages the development of a variety of models, and further raises the question of how to combine insights from the different models. Indeed, one important use of model insight is to drive development of new models. We can envision a scenario where results from model inference leads to ideas about refining or refocusing attention on different factors, captured in new models.

Taking an even more ambitious view, we could aspire to control a strategic agent with game-theoretic reasoning, iteratively scoping their decision problems and generating models with corresponding scopes. The relevant model ensemble continually adapts based on experience and change of circumstance. We are not yet in a position to deliver on this comprehensive vision. However, we can take a first step by framing a slice of the problem.

The step we take in this article is to propose and start to develop a conceptual framework for sequential formulation of game-theoretic models. The key object is what we call a *game view*, which encapsulates a specific set of modeling choices about what aspects of a strategic situation to focus on in a particular instance of game-theoretic analysis. Game-theoretic reasoning overall comprises a series of these instances, through a process we term ***game view navigation***.

Navigating among game views attempts to glean insights about a strategic scenario from multiple perspectives, varying scale and emphasis of different elements. Viewing everything all at once is not generally feasible, even with today's impressive game reasoning toolkit. This is especially true *a priori* when we lack confidence about which of the potentially relevant elements are actually salient.

As we began to develop the framework, we noticed that it actually serves well to describe many existing methods in game reasoning. Some were expressly conceived in these terms, for example iterative abstraction methods that explore alternative coarsenings of the action space [17]. Others have not been explicitly described in this way, but yet they fit the framework quite well. This might simply reflect the general value of recursive problem decomposition in dealing with complex computational tasks. Showing how this framework captures existing methods is instructive, and also helpful in understanding how it might be elaborated to address more speculative applications to multi-perspective game reasoning.

Our key contribution here, therefore, is explicating game view navigation as a unifying framework for iterative game reasoning. We show how numerous and diverse methods can be cast in this framework, drawing interesting connections in the process. We follow by applying the framework for new method design, based on modulating levels of aggregation to guide equilibrium search. The abstraction technique we employ, player reduction [18], approximates many-player games in terms of games with fewer players, coarsening the representation of other players from the perspective of each player. As we demonstrate through computational experiments, navigating across levels of player reduction provides a flexible means to trade off computation and accuracy in the identification of approximate equilibria.

The remainder of the article is organized as follows. Section 2 defines the key elements of our game view navigation framework. Related work is discussed throughout, particularly through description of existing techniques in terms of this framework. In Sect. 3 we present three examples in detail, illustrating how a variety of game-reasoning algorithms can be cast formally as game view navigation. Section 4 shows how this perspective dovetails with simulation-based methods for constructing game models. Algorithms that navigate across game views of multiple scales are the subject of Sect. 5. In Sect. 6 we present a new algorithm inspired by this framework, which navigates across levels of player aggregation. We conclude (Sect. 7) with an assessment and discussion of future directions for game view navigation.

2 Game view navigation

2.1 Conceptual framework

A high-level diagram of the conceptual framework of game view navigation is presented in Fig. 1. ***Game knowledge*** is the underlying source of information about a

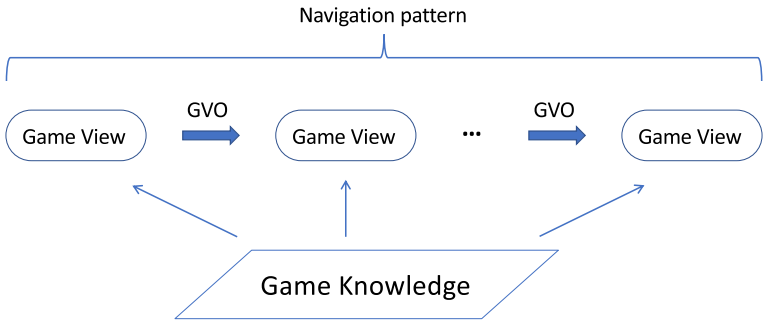

Fig. 1 Game view navigation. A sequence of game views generated from game knowledge by game view operations constitutes a navigation pattern

game. This might include specifications in any well-defined representation language, as well as databases or simulators of various forms. The corpus of knowledge is considered static in this framework, though it may be elaborated or otherwise change form during navigation as a result of reasoning.

For example, in poker, game knowledge comprises rules for how the game is played, including how the cards are dealt out and how betting works. In the course of reasoning, all or part of the game tree could be worked out in a more explicit form, but that would not represent an actual change in game knowledge.

A ***game view*** is a game model intended to capture key elements of a complex strategic situation. Models may be described in a variety of forms, for example standard formal descriptions such as normal-form or extensive-form, as well as special-purpose representations suitable for particular algorithms. Game views are built selectively from game knowledge, and serve as the central object of game-theoretic analysis.

In the poker example, different game views might abstract elements of the state space (e.g., grouping cards into classes) or action space (restricting betting options), or might selectively consider only parts of the game tree.

What counts as a "key element" of a strategic situation is inevitably subjective or speculative, so technically any game model would qualify as a game view. We introduce the redundant terminology expressly to emphasize the fact that any game model represents *just* a particular view of a more complex situation, and there are other views of comparable validity which may provide complementary perspectives.

Game view operations (GVOs) frame new game views based on previous game views and results from analyzing them. The new views are then constructed from the game knowledge. For example, GVOs might frame a new view as an abstraction or refinement of the previous, or as an elaboration or projection of other views. In poker, the domain of betting options could be adapted systematically based on results from prior views [19]. Formally, a GVO maps previous game views plus game knowledge into a new game view. This mapping typically includes explicit reasoning steps, such as solving previous game models according to specified solution concepts. Many specific GVOs are described below.

Finally, a ***navigation pattern*** is a generative sequence of game views, starting from an initial game view, and iterating through game view operations until a termination criterion is satisfied.

2.2 Game-theoretic preliminaries

Formally, a ***game*** $\Gamma = \langle N, (S_i), (u_i) \rangle$ consists of a finite set of players, N, indexed by i; a nonempty set of strategies S_i for each player; and a utility function $u_i : \times_{j \in N} S_j \to \mathbb{R}$ for each player. Let $n = |N|$ be the number of players.

Player i may play a ***pure strategy***, $s_i \in S_i$, or a ***mixed strategy***, $\sigma_i \in \Delta(S_i)$, which is a probability distribution over the pure strategies in S_i. Pure strategies may thus be viewed as special cases of mixed where all the probability is on one element. Let $\sigma_i(s_j)$ denote the probability that strategy s_j is played in σ_i. The ***support*** of a mixed strategy, $supp(\sigma_i)$, is the set of pure strategies played with positive probability: $supp(\sigma_i) = \{s_j \in S_i \mid \sigma_i(s_j) > 0\}$. A ***strategy profile*** $\sigma = (\sigma_1, \ldots, \sigma_n)$, assigns a (generally mixed) strategy to each player. If the assignments are to pure strategies, $s = (s_1, \ldots, s_n)$ is a ***pure profile***. We use notation $\sigma_{-i} = (\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_n)$ to denote a strategy profile for all players excluding i.

The aim of game-theoretic analysis is typically to characterize and identify ***solutions*** of a given game, according to a specified ***solution concept***. Commonly employed solution concepts are based on notions of ***equilibrium*** among strategies. Let $BR_i(\sigma_{-i})$ denote the set of best-response strategies for player i, given that others play σ_{-i}, that is:

$$BR_i(\sigma_{-i}) \triangleq \arg\max_{s_i \in S_i} u_i(s_i, \sigma_{-i}).$$

The classic concept of strategic equilibrium, due to Nash, selects profiles where all players are best-responding to the others. Formally, profile σ^* is a ***Nash equilibrium*** (NE) if and only if (iff) for all i, $\sigma_i^* \in \Delta(BR_i(\sigma_{-i}^*))$. An NE profile that is pure, $\forall i.\ s_i^* \in BR_i(s_{-i}^*)$, is called a ***pure-strategy Nash Equilibrium*** (PSNE).

We can also define approximate solution concepts, in particular ϵ-***Nash equilibrium*** in which no player can gain more than ϵ in payoff by deviating from the profile:

$$\forall s_i' \in S_i.\ \mathbb{E}[u_i(\sigma_i^*, \sigma_{-i}^*)] + \epsilon \geq \mathbb{E}[u_i(s_i', \sigma_{-i}^*)].$$

The possible deviation gain for player i playing strategy σ_i against other-player profile σ_{-i} is termed i's ***player regret***, given by:

$$\epsilon_i(\sigma) = \max_{s_i \in S_i} \mathbb{E}[u_i(s_i, \sigma_{-i})] - \mathbb{E}[u_i(\sigma_i, \sigma_{-i})].$$

The ***profile regret*** of σ is the maximum of player regrets:

$$\epsilon(\sigma) = \max_{i \in N} \epsilon_i(\sigma).$$

Equivalently, $\epsilon(\sigma)$ is the smallest ϵ such that profile σ is an ϵ-NE. Exact NE have zero regret.

Game Γ is ***symmetric*** if all players have the same strategy set ($\forall i, j.\ S_i = S_j = S$) and utility function, and this function $u(s_i, s_{-i})$ is invariant to permutations of the other players ($-i$). In other words, the utility function depends only on how many of the others play each strategy—not which ones. The ***support of a profile*** in a symmetric game is the union over supports of its constituent player strategies: $supp(\sigma) = \bigcup_{i \in N} supp(\sigma_i)$. For symmetric games it is common to focus on ***symmetric mixed-strategy NE*** (SMSNE): Nash equilibria where all players play the same (generally mixed) strategy [20, 21].

All game analysis is with respect to some set of included strategies X_i for each player i. This is typically a strict subset of all *possible* strategies, since that set is generally infeasible

Table 1 A sample normal-form game, for tracing the illustrative examples

	a	b	c	d	e	f
a	3,4	5,3	3,8	7,8	8,6	3,4
b	8,8	1,5	7,3	3,9	5,5	7,0
c	9,5	2,6	7,5	4,7	0,0	6,2
d	2,3	0,8	0,1	8,6	3,3	7,2

$S_1 = \{a, b, c, d\}$; $S_2 = \{a, b, c, d, e, f\}$

to cover. We use the symbol $\Gamma_{\downarrow X}$ to denote a ***restricted game*** with respect to the ***base game*** Γ, where each player i in $\Gamma_{\downarrow X}$ is restricted to playing strategies in $X_i \subseteq S_i$.

As we describe elements of evolving game views, we may extend notation with explicit subscripts indicating the game referenced. For example, ϵ_Γ denotes the regret function with respect to game Γ. When the applicable game is clear from context, we leave it implicit as above.

3 Illustrative examples

We define a navigation pattern by specifying the game knowledge and initial game view Γ^0, along with reasoning steps, GVOs, and termination criteria. A typical process extracts a ***solution*** σ^{*k} for game view Γ^k at each iteration k, according to some solution concept as implemented by a specified ***solver***. The solution σ^{*k} is then employed by the GVO in producing Γ^{k+1} based on the game view history.

To illustrate the framework, we show how a variety of iterative game reasoning methods can be cast in terms of game view navigation. To aid the reader in interpreting the specification, we trace the navigation patterns using the randomly generated two-player game of Table 1.

3.1 Iterative best-response

The IBR method maintains a pure profile $s = (s_1, \ldots, s_n)$. On each iteration, we choose a player i and replace i's strategy with the best-response to the others, $s_i^* \in BR_i(s_{-i}) = \arg\max_{s_i \in S_i} u_i(s_i, s_{-i})$.

	Iterative best response (IBR)
Game knowledge	A base game Γ.
Init	$\Gamma^0 = \Gamma_{\downarrow(X_1,\ldots,X_{n-1},S_n)}$. Players $i \neq n$ are restricted to singleton sets: $X_i = \{s\}$, with $s \in S_i$ arbitrarily chosen.
Solver	Derives $s^{*k} \in \text{PSNE}(\Gamma^k)$.
GVO	$\Gamma^{k+1} = \Gamma_{\downarrow(X_1,\ldots,X_n)}$, with $X_i = \{s_i^{*k}\}$ for $i \not\equiv_n k+1$, and $X_i = S_i$ for $i \equiv_n k+1$.
Termination	On encountering a previously generated game view, that is, $\Gamma^k = \Gamma^{k'}$ for some $k' < k$.

Game knowledge in this instance (and other instances presented as illustrative examples in this section) takes on a simple form, that of a base game. Navigation operates on game views that are restrictions of this base game. Determining which restrictions to solve on each iteration is the essence of the navigation pattern. In subsequent sections we consider forms of game knowledge that do not amount to an encompassing base game.

For IBR, the key idea is that each game view fixes the strategies of all but one player. Note that when the strategy sets for all agents $j \neq i$ are singletons $\{s_j\}$, the PSNE computation in the *Solver* operation reduces to solving for $BR_i(s_{-i})$. Starting from an arbitrary assignment of strategies to $n-1$ players (*Init* step), IBR repeatedly determines a BR for the remaining player in this way, fixing that player's strategy to the BR, and moves on to the next player (*GVO*) who now has all their strategies available. The criterion for *Termination* is generation of a game view at iteration k equivalent to one seen at an earlier iteration k'. A PSNE of Γ^k represents a solution of the base game Γ if $k' = k - n$, which means that the singleton X_i are unchanged on the cycle for each i. Otherwise (i.e., $k' < k - n$) the strategy configurations are changing, and the procedure is cycling rather than converging.

Proposition 1 *If IBR terminates with $k' = k - n$, then s^{*k} is a PSNE of Γ.*

Proof By the termination condition, $\Gamma^k = \Gamma^{k-n}$, and hence by the GVO we have $s_i^{*k} = s_i^{*(k-n)}$ for all $i \not\equiv_n k$. For each of these $n-1$ players, the value s_i^* is re-evaluated exactly once in the iterations between $k-n$ and k, and thus must not have changed. For the one player $i' \equiv_n k$, $s_{i'}^{*k} \in BR(s_{-i'}^{*k})$, and this is unchanged through the interval. Likewise, for each $i \neq i'$, its evaluation in the interval confirms $s_i^{*k} \in BR(s_{-i}^{*k})$. Thus, $s^{*\ell}$ remains constant for iterations $k - n + 1 < \ell \leq k$, and is established as a PSNE of Γ. □

We can trace out the IBR navigation pattern on the sample game of Table 1. The sequence of game views as defined by their restricted strategy sets is (with alphabetical-order tie-breaking):

$$(\{a\}, S_2), (S_1, \{c\}), (\{b\}, S_2), (S_1, \{d\}), (\{d\}, S_2), (S_1, \{b\}), (\{a\}, S_2).$$

The procedure terminates on detecting a cycle, which does not represent a solution as the singleton sets are changing. Indeed, this sample game does not contain a PSNE.[1]

3.2 Double oracle

The idea of the ***double oracle*** (DO) method [23] is to incrementally extend a game model in an informed manner. DO was originally defined for two-player games (hence the "double" of its name), though the idea readily generalizes to n players.

[1] Even when a PSNE exists, IBR is not guaranteed to find it. An exception is the class of *potential games* for which IBR is a complete algorithm [22].

	Double oracle (DO)
Game knowledge	A base game Γ.
Init	$\Gamma^0 = \Gamma_{\downarrow X^0}$. For concreteness, let us assume initialization with singleton strategy sets: $X_i^0 = \{s\}$, with $s \in S_i$ arbitrarily chosen.
Solver	Derives $\sigma^{*k} \in \text{NE}(\Gamma^k)$.
GVO	$\Gamma^{k+1} = \Gamma_{\downarrow X^{k+1}}$, with $X_i^{k+1} = X_i^k \cup \{s_i^{k+1}\}$, for $s_i^{k+1} \in BR_i(\sigma_{-i}^{*k})$.
Termination	On reaching a time or iteration limit, or if no new best responses are found—that is, $\forall i.\ s_i^{k+1} \in X_i^k$.

DO augments players' strategy sets on each iteration, based on BR to other-player choices in the previous NE solution. If the BRs at iteration $k+1$ were already present in the previous game view, $X^{k+1} = X^k$, then the solution σ^{*k} is an NE for Γ.[2]

Tracing DO on our sample game (Table 1), we get[3]:

0 $X_1^0 = \{a\}; X_2^0 = \{a\}; \sigma^{*0} = (a, a)$
1 $X_1^1 = \{a, c\}; X_2^1 = \{a, c\}; \sigma^{*1} = (c, a)$
2 $X_1^2 = \{a, c\}; X_2^2 = \{a, c, d\}; \sigma^{*2} = (a, d)$
3 $X_1^3 = \{a, c, d\}; X_2^3 = \{a, c, d\}; \sigma^{*3} = (d, d)$
4 $X_1^4 = \{a, c, d\}; X_2^4 = \{a, b, c, d\}; \sigma^{*4} = ([\frac{2}{7}a, \frac{5}{7}d], [\frac{1}{6}a, \frac{5}{6}d])$

On the next iteration no new strategies are introduced, as σ^{*4} is indeed an NE of this game.

3.3 Support enumeration

Porter et al. [24] proposed a heuristic algorithm based on search through the space of supports. The idea is to enumerate restricted strategy sets for each player, testing each for the presence of NE supported on exactly those sets. Their algorithm orders the search in favor of small and balanced support sets, which are both easiest to test and most prone to coincide with equilibria. We can cast the support enumeration method in terms of game view navigation as follows.

[2] This claim assumes perfect BR computation. If guaranteed only to a degree of approximation, this would translate to a corresponding approximate equilibrium result.

[3] Results are sensitive to the choice of σ^{*k} in cases where there are multiple equilibria in $\text{NE}(\Gamma^k)$. See discussion of support enumeration below.

	Support enumeration
Game knowledge	A base game Γ.
Init	$\Gamma^0 = \Gamma_{\downarrow X^0}$, where X^0 is the initial support to search. A natural default might be $X^0 = (\{s_{1,1}\}, \ldots, \{s_{n,1}\})$, where $s_{i,j}$ denotes the j^{th} strategy for player i.
Solver	Identify NE(Γ^k). For each $\sigma^{*k} \in$ NE(Γ^k), test whether any deviation outside the support sets $(X_1^k, \ldots, X_n^k)$ is beneficial, that is: $\exists i, s_i \in S_i \setminus X_i^k. \mathbb{E}[u_i(s_i, \sigma^*_{-i})] > \mathbb{E}[u_i(\sigma^*_i, \sigma^*_{-i})]$.
GVO	$\Gamma^{k+1} = \Gamma_{\downarrow(X_1^{k+1}, \ldots, X_n^{k+1})}$, with $X^{k+1} = \mathsf{Next}(X^k)$ determined by the support enumeration ordering.
Termination	A σ^{*k} that survives the test for beneficial deviations in the solver step above is an NE for Γ.

The description above employs the abstraction Next to omit details on the enumeration procedure, which as described by Porter et al. [24] are interesting and consequential for performance. In particular, the enumeration includes dominance checks which can prune much of the support-set space from consideration, thus saving the expense incurred in the solver step for unnecessary game views. For the sample game of Table 1, the algorithm would effectively skip the game views with $|X_1| = |X_2| = 1$ (i.e., those limited to a single pure profile), and proceed to enumerate game views with $|X_1| = |X_2| = 2$. Assuming alphabetical ordering of the strategies, the first two support candidates to survive pruning are $(X_1 = \{a, b\}, X_2 = \{c, d\})$ and $(X_1 = \{a, c\}, X_2 = \{c, d\})$. NE of the former fail the deviation test, but the latter produces the solution $\sigma^* = (a, [\frac{3}{7}c, \frac{4}{7}d])$. Note that this is also a solution to game view Γ^2 in the DO trace above (Sect. 3.2), but we did not terminate there because instead the solver happened to produce a different NE of Γ^2 (i.e., $\sigma^{*2} = (a, d)$), which is not an equilibrium of the base game Γ.

3.4 Illustrative examples: discussion

These first few examples—using existing and to varying extent familiar methods—are meant to illustrate the style of framing game-solving procedures as navigation in a space of game views. Though these methods were conceived in different times, with different motivations and perspectives, writing them down in this way exposes some common structure, and in particular the shared idea of using solutions to a series of smaller games as a way of solving a larger game.

We have also found that writing down these procedures as navigation patterns naturally suggests variations and generalizations. For example, we defined IBR as iterative solution of a series of projected games, each game view fixing strategies for all-but-one player. This immediately suggests the possibility of trying all-but-two (or all-but-m) as a natural alternative, which could prove advantageous. (It has not been tried or proposed to date as far as we know.)

For any navigation pattern, it likely makes sense to examine the solver and GVO elements, and ask whether it could make sense to employ alternate solvers or alternate choice of next game views. As we see in Sect. 4.3 below, generalizing the solver element of DO is a key idea of the Policy-Space Response Oracles algorithm [25]. We could similarly explore varying the GVO element, with alternatives to BR, augmenting strategy sets by more than one new strategy per player (or not all players), forgetting old strategies, etc.

From here we move on to methods where we cannot assume availability of a base game. Game knowledge in these cases is less explicit, thus constructing game views is a potentially involved or expensive process calling for deliberate control.

4 Empirical game-theoretic analysis

Empirical game-theoretic analysis (EGTA) [26] is a broad approach to game reasoning based on simulation. From the standpoint of our framework, the key EGTA feature is that underlying game knowledge is provided by a simulation model, with game models induced from simulation data. We denote the simulator by $\mathcal{O}$, an oracle that generates payoffs for a given strategy profile. Game views defined by simulation are indicated with a hat, $\hat{\Gamma}$, to remind us that payoff values are estimates from noisy samples. EGTA is naturally iterative, with game modeling and analysis interleaved with simulation, to refine payoff estimates and cover broader parts of the strategic space. In this section we consider three distinct kinds of iterative reasoning developed in the EGTA literature, illustrating how they can be cast as game view navigation patterns.

4.1 Incomplete game models

The illustrative examples of Sect. 3 posited a base game as underlying knowledge, with game views essentially projecting the game's payoff function u over the restricted strategy space defining the game view. For games defined by simulation, elaborating the payoff function over even a highly restricted strategy space may be quite expensive, as the number of profiles grows exponentially in $\min(n, |S|)$. Filling out the entire payoff matrix may also be unnecessary. For example, we can verify that σ^{*4} from the DO trace (Sect. 3.2) is an NE by exclusively inspecting profiles where at least one player chooses a or d. This amounts to 16 out of the game's 24 profiles. The search for solutions can entail consideration of more profiles than required for verification, but we observe that equilibrium and BR calculations in the DO trace were not exhaustive. And of course the proportional savings can be far greater for larger games.[4]

The minimum requirement for evaluating and verifying an NE is to consider all the profiles in the support of that equilibrium, plus all of the one-player deviations from those. Support Enumeration is one way of navigating game views, prioritizing smaller support profiles, all else equal. The EGTA context provides even stronger motivation to focus on smaller game views, as the total cost of constructing the view from simulation grows exponentially with the size of strategy sets. The *incremental* cost of constructing a game view depends on the pattern of previous game views analyzed, as the set of profiles evaluated through simulation accumulates over the navigation process.

Wellman et al. [28] introduced a systematic way to search through supports as part of an EGTA process. Their methods were later refined by Brinkman [29]. We describe the navigation pattern for symmetric games; generalization to non-symmetric games is conceptually straightforward. Let $S = \{s_1, \ldots, s_{|S|}\}$ be the common strategy set. Given symmetry, we can represent a profile by a vector $\mathbf{p} = (p_1, \ldots, p_{|S|})$, with p_j indicating the number of

[4] Fearnley et al. [27] study the question from a complexity-theoretic perspective, providing insights on the number of payoff queries required to guarantee identification of approximate solutions, in a variety of game contexts.

players choosing strategy s_j [30]. Let $\mathcal{P}$ be the set of strategy profiles that have been evaluated by simulator $\mathcal{O}$. We say that strategy set X represents a ***complete subgame*** if all profiles over X have been evaluated:

$$\forall \mathbf{p}.\ (\forall j.\ p_j = 0 \vee s_j \in X) \implies \mathbf{p} \in \mathcal{P}.$$

X is further ***maximal*** if there exists no $X' \supset X$ that also represents a complete subgame.

4.1.1 Subgame search

The idea of subgame search is to navigate a series of game views corresponding to maximal complete subgames. The algorithm maintains a record of the maximal subgames already searched, and a priority queue PQ of prospective subgames to explore. Priority is by subgame size, smallest first. The navigation pattern is reminiscent of support enumeration, but with the ordering Next implemented by PQ according to the cost tradeoffs of EGTA.

	Subgame search
Game knowledge	Simulator $\mathcal{O}$ for symmetric game Γ with strategy set S.
Init	Add all singletons $\{s\}, s \in S$, to PQ. $\Gamma^0 = \hat{\Gamma}_{\downarrow X^0}$, where $X^0 \leftarrow get(PQ)$.
Solver	Complete the subgame X^k if necessary by applying $\mathcal{O}$ to all profiles $\mathbf{p}$ such that $(\forall j.\ p_j = 0 \vee s_j \in X) \wedge (\mathbf{p} \notin \mathcal{P})$. Add X^k to the collection of maximal subgames. Identify SMSNE($\hat{\Gamma}^k$). For each $\sigma^{*k} \in$ SMSNE($\hat{\Gamma}^k$), test whether any deviation outside X^k is beneficial (invoking $\mathcal{O}$ as necessary for deviation profiles). If so, identify a most beneficial deviation s' and add $supp(\sigma^{*k}) \cup \{s'\}$ to PQ. If no beneficial deviation would create a new maximal subgame, choose an arbitrary extension to X^k.
GVO	If PQ is empty, identify a smallest strategy set not subsumed by any maximal subgame, and add it to PQ. $\Gamma^{k+1} = \hat{\Gamma}_{\downarrow X^{k+1}}$, where $X^{k+1} \leftarrow get(PQ)$.
Termination	A σ^{*k} that survives the test for beneficial deviations in the solver step above is an SMSNE for $\hat{\Gamma}$.

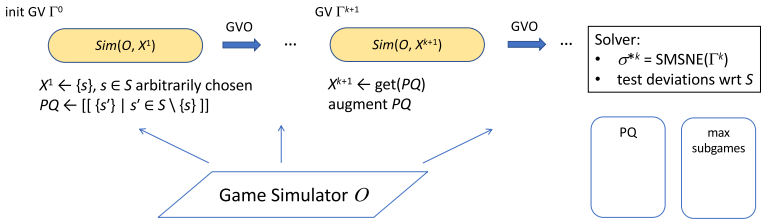

Fig. 2 Subgame Search navigation pattern. $Sim(\mathcal{O}, X)$ represents the game view corresponding to a subgame $\hat{\Gamma}_{\downarrow X}$ estimated by simulation. PQ is a priority queue containing subgames to be evaluated through simulation. The box labeled max subgames denotes a structure maintaining a specification of the set of maximal subgames already evaluated

Figure 2 depicts the SUBGAME SEARCH navigation pattern in the form of the schematic diagram of Fig. 1. Game views at each iteration correspond to empirical game models over a subgame. The subgame may be incompletely evaluated at this point, in which case it is completed and then solved for SMSNE. Solutions of the subgame are considered candidates for solutions of the base game, which are then tested by evaluating deviations outside the subgame. If a candidate passes the deviation test we designate it as ***confirmed***, and the navigation is terminated. Otherwise, the method proceeds to a new subgame according to the queuing policy.

4.2 Structure learning

An alternative way to apply $\mathcal{O}$ in the construction of game views is to *induce* a game model from simulation data [31]. Rather than treating a strategy profile as unevaluated if not explicitly simulated, we can construct a payoff function over the specified domain via a process of regression.

The game-learning approach is especially attractive when there is basis for capturing regularity in game models. For example, *graphical games* [3] support a relatively succinct encoding based on sparse interaction among players. Such compact game representations provide a natural target for game learning, and indeed several works tackle learning graphical game models from payoff data [27, 32, 33].

The method proposed by Li and Wellman [33] expressly interleaves structure learning and payoff regression in a process that iterates through a series of structural hypotheses. This algorithm thus naturally aligns with the game view navigation framework. It is described by the authors generically, and demonstrated with two particular forms of game structure that support scaling to many players. In addition to sparsity (for graphical games), the method applies to symmetry, in an algorithm called *K-Roles* which we cast as a navigation pattern below.

4.2.1 *K*-Roles

A game is ***role symmetric*** if the players can be partitioned into roles $R_1, \ldots, R_K$, such that symmetry (defined in Sect. 2.2) applies within roles. That is, players within each role have the same strategy sets and utility functions, and the utility functions depend only on how many (other) players in each role play each strategy. Role symmetry is without loss of generality, with $K = n$; with $K = 1$, the game is fully symmetric. A *role-symmetric Nash equilibrium* (RSNE) is an NE in role-symmetric strategies, which means that within each role, every player plays the same (generally mixed) strategy.

The idea of the *K*-Roles algorithm is to fit a role-symmetric game model to simulation data. Starting with an arbitrary assignment of players to roles, the algorithm solves the current game view to decide where to focus sampling, and then fits a new game model to the augmented data set. Fitting updates role structure (a mapping of players to roles) by a clustering method reminiscent of *k*-means, and then regresses the payoff data to the new role structure. Let $\mathsf{Reg}(\mathcal{D}, \mathcal{C})$ denote a game model formed by regressing the simulation data $\mathcal{D}$ to role structure $\mathcal{C}$. Li and Wellman [33] find that the method effectively scales to hundreds of players, with modest K and $|S|$.

	K-Roles
Game knowledge	Simulator $\mathcal{O}$ for Γ with strategy sets $S_i = S$. Hyperparameter: number of roles $\hat{K}$.
Init	$\Gamma^0 = \mathsf{Reg}(\mathcal{D}^0, \mathcal{C}^0)$, where $\mathcal{D}^0$ is the data buffer initialized with uniform sampling, and $\mathcal{C}^0$ is an initial role assignment.
Solver	Derives $\sigma^{*k} \in \text{RSNE}(\Gamma^k)$. Collect samples $\mathcal{D}^*$ in the neighborhood of σ^{*k}.
GVO	$\Gamma^{k+1} = \mathsf{Reg}(\mathcal{D}^{k+1}, \mathcal{C}^{k+1})$, where $\mathcal{D}^{k+1} = \mathcal{D}^k \cup \mathcal{D}^*$ and $\mathcal{C}^{k+1}$ is the result of clustering the players into new roles based on Γ^k and $\mathcal{D}^{k+1}$.
Termination	Based on time or convergence criteria.

4.3 Strategy exploration

Both the subgame-search and structure-learning approaches above operate with an enumerated strategy set S. In EGTA we often wish to explore very large strategy spaces $\mathcal{S}$ (say defined by a general language or parametrized policy description), beyond which we could search subgames or directly learn a payoff function over. The *strategy exploration* approach is as described in Sect. 3 for double oracle: to incrementally extend a game model in an informed manner.

A variation on DO geared for EGTA is the ***Policy-Space Response Oracles*** (PSRO) framework defined by Lanctot et al. [25] and extended in a rich thread of subsequent work [34]. PSRO leverages the availability of the simulator $\mathcal{O}$ in positing that best-response computations can be performed by training a strategy using deep reinforcement learning (DRL). PSRO further provides flexibility in the selection training targets via the ***meta-strategy solver*** (MSS), which may implement a variety of solution concepts or really any way of extracting a target profile from the already-explored strategy space.

4.3.1 PSRO

Comparing the navigation patterns for PSRO and DO, we can see the former is more specific in using DRL for best-response, and the latter is more specific in adopting NE as its MSS.

	Policy-Space Response Oracles (PSRO)
Game knowledge	Simulator $\mathcal{O}$ for Γ over strategy space $\mathcal{S}$.
Init	$\Gamma^0 = \Gamma_{\downarrow X^0}$: $X_i^0 = \{s\}$, $s \in S_i$ arbitrarily chosen.
Solver	Derives $\sigma^{*k} \in \text{MSS}(\Gamma^k)$.
GVO	$\Gamma^{k+1} = \Gamma_{\downarrow X^{k+1}}$, with $X_i^{k+1} = X_i^k \cup \{s_i^{k+1}\}$, for $s_i^{k+1} = \text{DRL}(\sigma_{-i}^{*k})$.
Termination	On reaching a time or iteration limit, or if the degree of best-response improvement is below a specified threshold.

Certain degenerate MSSs instantiate classic methods. If the MSS extracts a uniform distribution over strategies encountered, PSRO implements ***fictitious play*** [35]. If it extracts

the most recent other-agent profile, the procedure corresponds to learning from ***self play*** [12].

Starting with the original PSRO paper [25], researchers have invented and evaluated a variety of MSSs, based on well known or novel solution concepts [34]. Experience has shown that choice of MSS can have significant influence on the efficacy of strategy exploration in a wide range of game contexts [36–40].

5 Multi-scale game reasoning

In the examples above, game views are varied primarily through changes in strategy sets. The *K*-Roles example also included change of structural assumptions in the payoff function. As suggested in our motivating introduction, we would like to consider a broader diversity of perspectives, with all manner of game elements subject to variation. An obvious example is the set of players. Different game views may select different agents as strategically salient, or may even draw different abstractions of agency by associating playerhood in the game with aggregations of agent entities.

The defining feature of *multi-scale* game reasoning is the ability to consider the set of players at multiple levels of abstraction. For instance, one game view may treat a set of agents as individual players, and another may aggregate the same set as one group player. Aggregation can yield computational economy, at the cost of fidelity in comparison with the finest-grain view. A multi-scale game-reasoning algorithm manages this tradeoff by using results from coarse-grain views to select elements for more detailed attention in subsequent game views.

5.1 Hierarchical graphical games

Jin et al. [41] considered a class of *hierarchical graphical games*, where agents are organized in a tree of groups, with group-level strategies defined as aggregates of individual strategies. At each level l of the tree the players are engaged in a kind of graphical game, where utility depends on one's own strategy, the strategies of one's neighbors at level l, and the group-level strategies of one's own group and its neighbors at higher levels.

5.1.1 Separated hierarchical IBR

The authors define a multi-scale game-solving algorithm, which they describe as a *separated hierarchical* application of iterative best-response. The navigation pattern has two layers: a top layer described below, and a second layer where we invoke the IBR navigation pattern as described in Sect. 3. At the top layer, game views focus on one level at a time, constructing the graphical game corresponding to the designated level of the multi-scale game representation. Let us denote this level-l graphical game by $\Gamma_{[l]} = \langle N_l, (S_{[l],i}), (u_{[l],i}) \rangle$, with $n_l = |N_l|$ the number of applicable players at level l. The algorithm operates by rotating through the game levels, approximately solving the corresponding game for each using IBR.

	Separated hierarchical IBR
Game knowledge	A hierarchical graphical game Γ, with L levels.
Init	Initialize level-1 actions arbitrarily; propagate to group-level actions up the tree. $\Gamma^0 = \Gamma_{[1]}$, the level-1 graphical game.
Solver	Invokes the IBR navigation pattern for Γ^k at level l, starting with strategies for players N_l at their most recently set values, and applied for n_l game-view steps. Let s^{*k} denote the strategy assignment after the last step.
GVO	Set new strategies based on s^{*k}. Update penalty values based on inconsistencies in group and constituent strategy values at adjacent levels. $\Gamma^{k+1} = \Gamma_{[(k+1 \mod L)+1]}$.
Termination	On convergence: L consecutive game views with unchanged strategies.

5.2 Player reduction

The multi-scale reasoning of the previous section exploits group structure arranged in a hierarchy. We can also consider aggregation as a means of summarization and approximation, apart from any presumption about agent organization in groups. The idea of ***player reduction*** [42] is to approximate a many-player ***full game*** by one with considerably fewer players. Here we describe one particular technique, ***deviation-preserving reduction*** (DPR), introduced by Wiedenbeck and Wellman [18] as a generalization of *twins reduction* [43] from two to p players, $p \ll n$. The idea of DPR is to consider each reduced player as controlling one player in the full game, but to treat the other reduced players as full-game aggregates.

Formally, a ***DPR game*** $\Gamma^{(p)} = \langle P, S, (u^{(p)}) \rangle$, $|P| = p$, is a p-player symmetric game[5] derived from an n-player symmetric full game $\Gamma = \langle N, S, u \rangle$. The key to defining the reduction is specifying how the utility function $u^{(p)}$ for the reduced game can be derived from that of the full game. As the game is symmetric, consider $i = 1$ without loss of generality. The DPR mapping is given by $u^{(p)}(s_1, s_{-1}^{(p)}) = u(s_1, s_{-1}^{(n)})$, where the full-game other-agent profile $s_{-1}^{(n)}$ contains $(n-1)/(p-1)$ copies of $s_{-1}^{(p)}$.

For example, suppose $n = 100$ and $p = 4$. The 4-player reduced game comprises profiles of the form $(s_1, s_{-1}^{(4)}) = (s_1, (s_2, s_3, s_4))$. DPR defines the reduced-game utility

$$u_1^{(4)}(s_1, (s_2, s_3, s_4)) = u(s_1, (s_2, \ldots, s_3, \ldots, s_4, \ldots)),$$

where each "$s_j, \ldots$" stands for 33 players playing strategy s_j. This reduction is called *deviation-preserving* because player 1 changing strategy in the reduced game equates to a corresponding deviation by player 1 in the full game. From player 1's perspective, the other players are aggregated but its own effect remains as just one-in-n.

Reductions of this sort mesh well with EGTA since payoff data for estimating the reduced game can be taken from simulations of the full game. For example, we can run the Subgame Search navigation pattern essentially as written in Sect. 4.1 to analyze DPR

[5] The definitions can be straightforwardly generalized to role symmetry. We further assume for simplicity that $p - 1$ divides $n - 1$.

games for any p. Whenever the algorithm asks for simulation of a reduced-game profile, we instead simulate the profiles in the full game needed to get the payoff data for that profile, based on the mapping above.

6 Modulating player aggregation

A typical way player reduction has been used to scale EGTA to many-player games is simply to follow the procedure mentioned above: run subgame search with DPR for a specified p. The game view navigation framework suggests a more flexible approach, where different settings of p may be considered as part of a series of game views. By modulating the level of player aggregation, we can trade off accuracy for speed, using the "quick-and-dirty" solutions at coarse levels of aggregation to focus a higher-fidelity search at finer reduction levels.

6.1 Navigation pattern

A navigation pattern realizing this approach operates at two layers (as in Separated Hierarchical IBR from Sect. 5.1). The first layer, described below, defines a variety of game views based on reduction level (p) and restricted strategy set (X). It may also apply alternative solution concepts, realized by a meta-strategy solver (MSS) as defined for PSRO (Sect. 4.3). The second layer performs a version of Subgame Search to derive a solution for the first-level game view. subgame Search is as described in Sect. 4.1, except that it accommodates a reduction parameter p as discussed above.

6.1.1 Modulated player aggregation

A schematic version of modulating player aggregation—allowing for flexibly specified policies of modulation—is presented below.

	Modulated player aggregation
Game knowledge	Simulator $\mathcal{O}$ for symmetric game Γ with strategy set S.
Init	$\Gamma^0 = \Gamma^{(p_0)}$, where p_0 is the initial reduction level.
Solver	Derives $\sigma^{*k} \in \text{MSS}(\Gamma^k)$.
GVO	$\Gamma^{k+1} = \Gamma^{(p_{k+1})}_{\downarrow X^{k+1}}$, with p_{k+1} and X^{k+1} selected based on results from previous game views.
Termination	After solving a game view with $p_k = n$.

Note that the computation performed in the solver step depends pivotally on the MSS specified. When the MSS is NE, the method invokes Subgame Search on the current Γ^k. This subroutine (itself a navigation pattern) drives elaboration of the game view through simulation of profiles using $\mathcal{O}$.

6.1.2 Minimum regret constrained profile

The instances of Modulated Player Aggregation described and evaluated below employ NE as MSS, except on the last iteration where most of them invoke an alternative solution concept called ***minimum regret constrained profile*** (MRCP) [44]. MRCP is defined as the strategy profile within a constrained set of profiles that minimizes regret with respect to a base game. More formally,

$$\text{MRCP}(\Gamma, \mathcal{P}) = \arg\min_{\sigma \in \mathcal{P}} \epsilon_\Gamma(\sigma).$$

By definition, if $\mathcal{P}$ contains all profiles of Γ, then MRCP and NE coincide. The interesting cases are when $\mathcal{P}$ reflects a constraint, for example profiles in a restricted, reduced, and/or incompletely evaluated game. Some recent authors [40, 45] have proposed using MRCP for conducting or evaluating strategy exploration in PSRO settings.

To identify the MRCP, we compute regret for all profiles in $\mathcal{P}$, which entails evaluating all one-player deviations. This may entail further simulation of profiles using $\mathcal{O}$.

6.2 Variations

We illustrate the schematic Modulated Player Aggregation by defining several instances, which we have implemented and evaluated experimentally. Method 1 (M1) is the baseline, where we just perform subgame search on the full (unreduced) game. M2 starts with a reduced game, and uses the result to prune the strategy space for analysis of the full game.

6.2.1 M1

$p_0 = n$. MSS is NE.

6.2.2 M2

$p_0 = p$. $p_1 = n$. $X^1 = supp(\sigma^{*0})$. MSS is NE.

In other words, M2 performs a subgame-search for NE in the full game, but restricted to strategies that are supported in a solution to the p-player reduced game.

The remaining methods we define invoke MRCP as MSS on their last (full-game) iteration. In the following, let $\mathcal{P}$ denote the set of profiles evaluated by $\mathcal{O}$ up to the current game view. Let $\mathcal{E}$ denote the set of candidates generated during the previous subgame search, that is, all profiles identified as subgame equilibria, whether confirmed or refuted in their containing game by subsequent deviation tests.

6.2.3 M3

$p_0 = p$. $p_1 = n$. MSS is NE for Γ^0 and MRCP$(\Gamma, \mathcal{P})$ for Γ^1.

6.2.4 M4

$p_0 = p.\ p_1 = n.$ MSS is NE for Γ^0 and MRCP(Γ, $\{\mathbf{p} \mid \mathbf{p} \in \mathcal{P} \wedge supp(\mathbf{p}) \subseteq supp(\sigma^{*0})\}$) for Γ^1.

6.2.5 M5

$p_0 = p.\ p_1 = n.$ MSS is NE for Γ^0 and MRCP(Γ, $\mathcal{E}$) for Γ^1.

6.2.6 M6

$p_0 = p.\ p_1 = n.$ MSS is NE for Γ^0 and MRCP(Γ, $\{\tau \mid \tau \in \mathcal{E} \wedge supp(\tau) \subseteq supp(\sigma^{*0})\}$) for Γ^1.

In other words, methods M3–M6 perform a subgame search for NE in the p-player reduced game, then use this result to define a set of profiles for computing an MRCP in the full game. The methods differ in whether they consider pure profiles evaluated versus equilibrium candidates, and whether they further filter profiles based on support of the reduced-game solution.

All the methods above include at most one reduced-game analysis, and thus barely scratch the surface of modulation approaches that could be implemented under the navigation pattern above. Trying these two-step methods provides a first exploration of whether there is potential benefit to any multi-scale approach combining subgame-search with player reduction. Based on our experiments with these (described below), we introduce another method, which scratches just a bit deeper by inserting one more level of player aggregation.

6.2.7 M7

$p_0 = p.\ p_1 = p'.\ p_2 = n.\ X^1 = \textsc{Top}(S, m_1),\ X^2 = \textsc{Top}(X^1, m_2)$. MSS is NE for Γ^0 and Γ^1, and MRCP(Γ, $\{\mathbf{p} \mid \mathbf{p} \in \mathcal{P} \wedge supp(\mathbf{p}) \subseteq X^2\}$) for Γ^2.

The function $\textsc{Top}(S, m)$ in M7 selects the best m strategies from set S, ranking strategies by ***NE-regret*** [46]. The NE-regret of a strategy s_i is defined as $\epsilon_i(s_i, \sigma^*_{-i})$, for σ^* an NE profile. Jordan [47] showed how to use NE-regret for ranking strategies as part of an EGTA process. The proposal by Balduzzi et al. [48] to rank strategies by ***Nash-averaging*** is technically equivalent to ranking by NE-regret.

6.3 Experiments

6.3.1 Five-strategy games

We evaluated the methods experimentally using two classes of random symmetric games with many players. The first are matrix games (MG), with payoffs drawn uniformly on the unit interval. For class MG, we set $n = 25$ and $|S| = 5$.

The second game class (AGG) comprises bipartite ***action-graph games*** with additive function nodes [1]. We set $n = 49$ and $|S| = 5$, and include seven function nodes. Edges from the five action nodes to the function nodes are generated independently with probability 0.2 and every function node has an edge to every action node. Action weights in

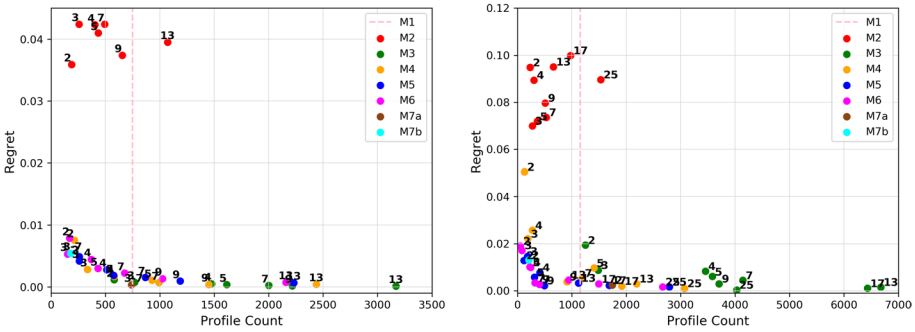

Fig. 3 Average profiles searched and regret achieved for methods M1–M7, various player aggregation levels (DPR player reductions, p indicated by point labels). Left: MG; Right: AGG

the graph are randomly generated $\sim N(0, 1)$. The function nodes are defined as in the game class employed by Sokota et al. [9] for a game-learning study.

For each game class, we draw 100 random game instances and run each of the methods M1–M6, plus M7a and M7b, on all these games. For MG, methods M2–M6 are run with all DPR levels $p \in \{13, 9, 7, 5, 4, 3, 2\}$. For AGG, these methods are run with $p \in \{25, 17, 13, 9, 7, 5, 4, 3, 2\}$. We consider two settings of the m parameters in M7.

- M7a. $m_1 = 4; m_2 = 3$
- M7b. $m_1 = 3; m_2 = 2$

M7a and M7b are run with p-p' 3–7 and 4–9.

The results for all runs are plotted in Fig. 3.[6] The dashed vertical line indicates the number of profiles evaluated by the baseline M1, which performs subgame search on the full game. For MG the baseline searched 749 profiles on average (out of 118,755 distinct profiles in the game), and for AGG the average was 1153 (out of 1,906,884). M1 finds exact equilibria, so anything to the right of the vertical line is dominated. The other method to highlight here is M2 (red points in upper left), which often searches significantly less but has the worst accuracy in terms of full-game regret. What we see overall is that the methods offer a range of performance, with a clear efficient frontier of tradeoffs in the lower-left corner.

Figure 4 (rescaled) focuses on method M6, which demonstrates a promising performance tradeoff. Increasing player aggregation (i.e., decreasing p) tends to improve efficiency (fewer profiles evaluated), at the cost of worse accuracy (higher regret). The method provides reasonably low-regret solutions for all p, and when aggregation is sufficient ($p \leq 7$ for MG and $p \leq 13$ for AGG), with savings in search effort.

Figure 5 zooms in on the data of Fig. 3, by truncating points that evaluate more profiles than required for method M1. We are interested in the *Pareto frontier*, comprising points that are undominated (i.e., no other setting is better in both accuracy and profiles evaluated). For MG, we find that the Pareto frontier includes points from all methods except for M2:

[6] This plot is too cluttered to tease apart all the methods, but provides a helpful overview. We zoom in on particular methods in subsequent figures.

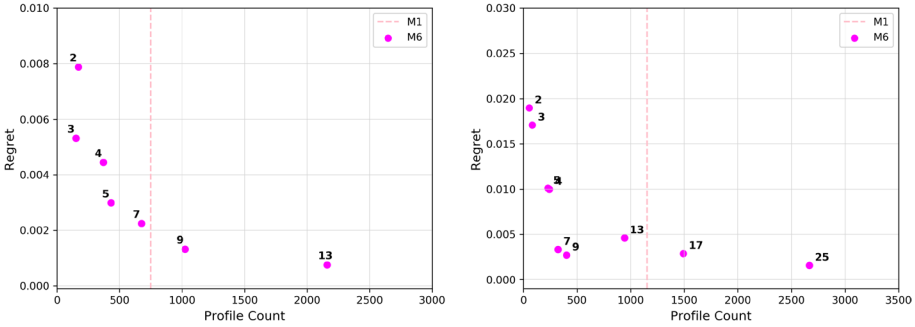

Fig. 4 M6, various DPR levels. Left: MG; Right: AGG

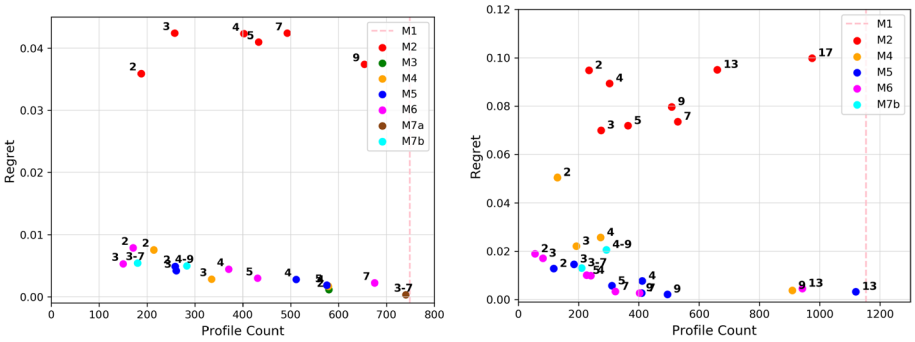

Fig. 5 Average profiles searched and regret achieved for methods M1–M7, focused on instances evaluating fewer profiles than M1. DPR levels are shown as point labels. Left: MG; Right: AGG

M6(3), M5(3), M4(3), M5(5), M3(2), M7a(3-7), M1.

For AGG, M5 and M6 (MRCP over $\mathcal{E}$) appear to be the most effective methods. The Pareto frontier points are:

M6(2), M6(3), M5(2), M6(5), M6(4), M5(5), M6(7), M6(9), M5(9), M1.

6.3.2 Seven-strategy games

We further evaluated these methods on a somewhat larger version of MG, with $|S| = 7$. Though we added only two more strategies, the number of full-game profiles in this game is 22 times that of the smaller game. Our baseline M1 (SUBGAME SEARCH on the full game) evaluates 1988 profiles on average for the seven-strategy game, which is 2.7 times that of the five-strategy game.

For this experiment, we tried methods M2–M6 with all of the DPR settings that performed better than baseline on the 5 strategy version (i.e., those shown in Fig. 5). For M7, we defined M7a by $m_1 = 5$; $m_2 = 4$, and M7b by $m_1 = 4$; $m_2 = 3$. Results are shown in Fig. 6. The Pareto frontier comprises:

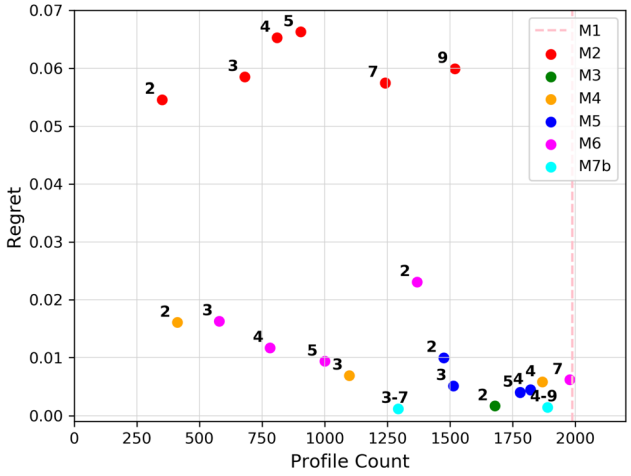

Fig. 6 Performance on MG with seven strategies. Average profiles searched and regret achieved for methods M1–M7, showing instances evaluating fewer profiles than M1. Point labels indicate DPR levels

M2(2), M4(2), M4(3), M6(4), M6(5), M7a(3-7), M1,

which actually includes an instance of M2, but otherwise is qualitatively similar to what we found in the smaller version of MG.

6.4 Discussion

Our investigations provide some initial evidence that modulating player aggregation provides a flexible way to trade off small amounts of accuracy for efficiency in EGTA for symmetric games. Whereas DPR had previously been employed as an approximation heuristic, this is the first exploration of its use as a means to direct a process of search in incompletely evaluated game models. Our experiments cover only a couple of abstract game classes, but even these were sufficient to reveal some interesting observations. First, relative performance of our methods differed in the two game classes, suggesting that presence of game structure (in our case action-graph structure) differentially effects performance of different approaches. Second, only one of the methods in the initial set we tried (M2) was apparently dominated. This speaks to the richness of tradeoffs available by considering a portfolio of techniques. Finally, multi-level aggregation (i.e., method M7) proved advantageous over two-level aggregation in some cases, suggesting that further exploration of that direction may be fruitful.

The study has limited scope, so we do not consider that it can support very confident or robust conclusions about the precise relative performance of the different techniques. Many of our method instances just missed being on the Pareto frontier, and we expect that for slightly different problems (or even larger experiments) they might displace points that comprised the frontier in our experiments. Nevertheless, the higher-level observations as expressed above seem well supported as provisional conclusions.

7 Conclusion

Game view navigation provides a conceptual framework for understanding a variety of existing iterative methods for game reasoning, for exploring variations, and for designing new approaches. We showed that a diverse set of techniques can be cast in this framework in a relatively natural way. Doing so helps to clarify relations between methods, for example how PSRO generalizes DO and some other algorithms with its MSS abstraction, and how the structure of Subgame Search evokes that of Support Enumeration.

Casting existing methods in this way may facilitate exploration of new techniques. For example, we discussed above how novel variants of IBR and DO are directly suggested by tweaks in elements of the navigation pattern.

Another natural source of new methods is *composition*, leveraging the ability to layer navigation patterns (e.g., as Separated Hierarchical IBR invokes IBR). The solver component in navigation patterns is a natural place for composing methods, as it represents a recursive invocation of game solution. Solvers may be amenable to alternate implementations, including methods that produce approximate solutions or incorporate equilibrium selection criteria or biases. The orthogonality of solving algorithms (or even choice of solution concepts) provides a wealth of mix-and-match options for navigation patterns. Composition also enables broader generalization of existing methods. For example, PSRO could naturally accommodate incomplete models or employ game model learning, using the navigation patterns defined here or others.

We demonstrated use of the framework for developing new approaches, with a first exploration of Modulated Player Aggregation. We found in experiments that even a simple two-step process can leverage player reduction to trade off small amounts of accuracy for performance. Our exploration also provides some evidence for MRCP as a promising solution concept for deriving approximate equilibria from results of partial search. It further suggests that multi-level extensions may also be fruitful, filtering strategy sets as the reduction is refined. Much more substantial experimentation and exploration of the space of methods will be necessary to establish confidence about what exactly is the best way to modulate player aggregation for game classes of interest.

Recent advances in game reasoning with machine learning may provide special benefits for algorithms defined by navigation patterns. For example, Smith et al. [49] introduce methods for *strategic transfer learning* whereby solutions from one context (i.e., what we term a game view) can be leveraged to reduce computation required in related contexts. Many of the navigation patterns described here entail solution of related game views. Similarly, techniques for representing and learning parametrized *families* of games [50] provide a source of game knowledge that readily supports a multiplicity of views about related games.

Further work developing the framework will enrich the space of methods cast as navigation patterns, and in the process we hope refine and clarify the formal constructs employed in describing game view navigation. Ideally, such development will lead to specification languages that enable both the implementation and theoretical analysis of methods for game reasoning based on navigating a space of game views.

Author Contributions MPW conceived and developed the framework and wrote the manuscript. MPW and KM designed the new methods for modulating player aggregation. KM implemented these methods and performed the experiments. Both authors reviewed the manuscript.

Funding This work was supported in part by the US Army Research Office under MURI Grant # W911NF-18-1-0208.

Data availability Not applicable.

Declarations

Conflict of interest The authors declare no conflict of interests.

Ethical approval Not applicable.

References

1. Jiang, A. X., Leyton-Brown, K., & Bhat, N. A. R. (2011). Action-graph games. *Games and Economic Behavior, 71*, 141–173.
2. Jiang, A.X., Chan, H., & Leyton-Brown, K. (2017). Resource-graph games: A compact representation for games with structured strategy spaces. In *Thirty-first AAAI conference on artificial intelligence* (pp. 572–578).
3. Kearns, M. (2007). Graphical games. In N. Nisan, T. Roughgarden, E. Tardos, & V. V. Vazirani (Eds.), *Algorithmic Game Theory* (pp. 159–180). Cambridge: Cambridge University Press.
4. Li, Z., Jia, F., Mate, A., Jabbari, S., Chakraborty, M., Tambe, M., & Vorobeychik, Y. (2022). Solving structured hierarchical games using differential backward induction. In *Thirty-eighth conference on uncertainty in artificial intelligence* (pp. 1107–1117).
5. Sandholm, T. (2015). Solving imperfect-information games. *Science, 347*(6218), 122–123.
6. Brown, N., & Sandholm, T. (2019). Superhuman AI for multiplayer poker. *Science, 365*(6456), 885–890.
7. Moravčik, M., Schmid, M., Burch, N., Lisý, V., Morill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., & Bowling, M. (2017). DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science, 356*(6337), 508–513.
8. Wellman, M. P. (2016). Putting the agent in agent-based modeling. *Autonomous Agents and Multi-Agent Systems, 30*, 1175–1189.
9. Sokota, S., Ho, C., & Wiedenbeck, B. (2019). Learning deviation payoffs in simulation-based games. In *Thirty-third AAAI conference on artificial intelligence* (pp. 2173–2180).
10. Perolat, J., De Vylder, B., Hennes, D., Tarassov, E., Strub, F., Boer, V., Muller, P., et al. (2022). Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science, 378*(6623), 990–996.
11. Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., … Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature, 575*, 350–354.
12. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science, 362*(6419), 1140–1144.
13. Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory*. Cambridge, MA: MIT Press.
14. Sandholm, T., & Singh, S. (2012). Lossy stochastic game abstraction with bounds. In *Thirteenth ACM conference on electronic commerce* (pp. 880–897).
15. Gilpin, A., & Sandholm, T. (2007). Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In *Sixth international joint conference on autonomous agents and multi-agent systems* (pp. 1168–1175).
16. Page, S. E. (2018). *The model thinker*. New York: Basic Books.
17. Sandholm, T. (2015). Abstraction for solving large incomplete-information games. In *Twenty-ninth AAAI conference on artificial intelligence, Austin* (pp. 4127–4131).
18. Wiedenbeck, B., & Wellman, M.P. (2012). Scaling simulation-based game analysis through deviation-preserving reduction. In *Eleventh international conference on autonomous agents and multi-agent systems* (pp. 931–938).
19. Hawkin, J., Holte, R.C., & Szafron, D. (2012). Using sliding windows to generate action abstractions in extensive-form games. In *Twenty-sixth AAAI conference on artificial intelligence* (pp. 1924–1930).

20. Cheng, S.-F., Reeves, D.M., Vorobeychik, Y., & Wellman, M.P. (2004). Notes on equilibria in symmetric games. In *AAMAS-04 workshop on game-theoretic and decision-theoretic agents, New York*.
21. Kreps, D. M. (1990). *Game theory and economic modelling*. Oxford: Oxford University Press.
22. Tardos, E., & Wexler, T. (2007). Network formation games and the potential function method. In N. Nisan, T. Roughgarden, E. Tardos, & V. V. Vazirani (Eds.), *Algorithmic game theory* (pp. 487–516). Cambridge: Cambridge University Press.
23. McMahan, H.B., Gordon, G.J., & Blum, A. (2003). Planning in the presence of cost functions controlled by an adversary. In *Twentieth international conference on machine learning* (pp. 536–543).
24. Porter, R., Nudelman, E., & Shoham, Y. (2008). Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior, 63*, 642–662.
25. Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., & Graepel, T. (2017). A unified game-theoretic approach to multiagent reinforcement learning. In *Thirty-first annual conference on neural information processing systems* (pp. 4190–4203).
26. Wellman, M.P., Tuyls, K., & Greenwald, A. (2024). Empirical game-theoretic analysis: a survey. Technical Report. *Journal of Artificial Intelligence* Research (to appear).
27. Fearnley, J., Gairing, M., Goldberg, P., & Savani, R. (2015). Learning equilibria of games via payoff queries. *Journal of Machine Learning Research, 16*, 1305–1344.
28. Wellman, M.P., Kim, T.H., & Duong, Q. (2013). Analyzing incentives for protocol compliance in complex domains: A case study of introduction-based routing. In *Twelfth workshop on the economics of information security*.
29. Brinkman, E. (2018). Understanding financial market behavior through empirical game-theoretic analysis. PhD thesis, University of Michigan.
30. Wiedenbeck, B., & Brinkman, E. (2023). Data structures for deviation payoffs. In *Twenty-second international conference on autonomous agents and multi-agent systems* (pp. 670–678).
31. Vorobeychik, Y., Wellman, M. P., & Singh, S. (2007). Learning payoff functions in infinite games. *Machine Learning, 67*, 145–168.
32. Duong, Q., Vorobeychik, Y., Singh, S., & Wellman, M.P. (2009). Learning graphical game models. In *Twenty-first international joint conference on artificial intelligence, Pasadena* (pp. 116–121).
33. Li, Z., & Wellman, M.P. (2020). Structure learning for approximate solution of many-player games. In *Thirty-fourth AAAI conference on artificial intelligence* (pp. 2119–2127).
34. Bighashdel, A., Wang, Y., McAleer, S., Savani, R., & Oliehoek, F.A. (2024). Policy space response oracles: A survey. In *Thirty-third international joint conference on artificial intelligence*.
35. Brown, G. W. (1951). Iterative solution of games by fictitious play. In T. C. Koopmans (Ed.), *Activity analysis of production and allocation* (pp. 374–376). New York: Wiley.
36. Balduzzi, D., Garnelo, M., Bachrach, Y., Czarnecki, W.M., Perolat, J., Jaderberg, M., & Graepel, T. (2019). Open-ended learning in symmetric zero-sum games. In *Thirty-sixth international conference on machine learning* (pp. 434–443).
37. Marris, L., Muller, P., Lanctot, M., Tuyls, K., & Graepel, T. (2021). Multi-agent training beyond zero-sum with correlated equilibrium meta-solvers. In *Thirty-eighth international conference on machine learning* (pp. 7480–7491).
38. Muller, P., Omidshafiei, S., Rowland, M., Tuyls, K., Perolat, J., Liu, S., Hennes, D., Marris, L., Lanctot, M., Hughes, E., *et al.* (2020). A generalized training approach for multiagent learning. In *Eighth international conference on learning representations*.
39. Wang, Y., Shi, Z.R., Yu, L., Wu, Y., Singh, R., Joppa, L., & Fang, F. (2019). Deep reinforcement learning for green security games with real-time information. In *Thirty-third AAAI conference on artificial intelligence* (pp. 1401–1408).
40. Wang, Y., Ma, Q., & Wellman, M.P. (2022). Evaluating strategy exploration in empirical game-theoretic analysis. In *Twenty-first international conference on autonomous agents and multi-agent systems* (pp. 1346–1354).
41. Jin, K., Vorobeychik, Y., & Liu, M. (2021). Multi-scale games: Representing and solving games on networks with group structure. In *Thirty-fifth AAAI conference on artificial intelligence* (pp. 5497–5505).
42. Wellman, M.P., Reeves, D.M., Lochner, K.M., Cheng, S.-F., & Suri, R. (2005). Approximate strategic reasoning through hierarchical reduction of large symmetric games. In *Twentieth national conference on artificial intelligence* (pp. 502–508).
43. Ficici, S.G., Parkes, D.C., & Pfeffer, A. (2008). Learning and solving many-player games through a cluster-based representation. In *Twenty-fourth conference on uncertainty in artificial intelligence* (pp. 187–195).
44. Jordan, P.R., Schvartzman, L.J., & Wellman, M.P. (2010). Strategy exploration in empirical games. In *Ninth international conference on autonomous agents and multi-agent systems* (pp. 1131–1138).

45. McAleer, S., Wang, K.A., Lanier, J., Lanctot, M., Baldi, P., Sandholm, T., & Fox, R. (2022). Anytime PSRO for two-player zero-sum games. In *AAAI-22 workshop on reinforcement learning in games*.
46. Jordan, P.R., Kiekintveld, C., & Wellman, M.P. (2007). Empirical game-theoretic analysis of the TAC supply chain game. In *Sixth international joint conference on autonomous agents and multi-agent systems* (pp. 1188–1195).
47. Jordan, P.R. (2010). Practical strategic reasoning with applications in market games. PhD thesis, University of Michigan.
48. Balduzzi, D., Tuyls, K., Perolat, J., & Graepel, T. (2018). Re-evaluating evaluation. In *Thirty-second annual conference on neural information processing systems* (pp. 3272–3283).
49. Smith, M., Anthony, T., & Wellman, M.P. (2021). Iterative empirical game solving via single policy best response. In *Ninth international conference on learning representations*.
50. Gatchel, M., & Wiedenbeck, B. (2023). Learning parameterized families of games. In *Twenty-second international conference on autonomous agents and multi-agent systems* (pp. 1044–1052).
51. Nisan, N., Roughgarden, T., Tardos, E., & Vazirani, V. V. (Eds.). (2007). *Algorithmic game theory*. Cambridge: Cambridge University Press.