Chapter 6
Selecting Robust Strategies Based on Abstracted Game Models

Oscar Veliz and Christopher Kiekintveld

Abstract Game theory is a tool for modeling multi-agent decision problems and has been used to great success in modeling and simulating problems such as poker, security, and trading agents. However, many real games are extremely large and complex with multiple agent interactions. One approach for solving these games is to use abstraction techniques to shrink the game to a form that can be solved by removing detail and translating a solution back to the original. However, abstraction introduces error into the model. We study ways to analyze games that are robust to errors in the model of the game, including abstracted games. We empirically evaluate several solution methods to examine how robust they are for abstracted games.

Keywords Game theory · Agent based simulation · Agent based modeling · Abstraction

6.1 Introduction

Game theory is widely used for analyzing multi-agent decision problems including auctions [12], security [1, 17], Poker [14], and many others. A game-theoretic analysis begins by specifying a formal model of the decision problem, including the actions players can choose, the information available to the players, and the utilities they receive for different outcomes. Once the model is specified it can be analyzed using Nash Equilibrium or any of the other numerous solution concepts proposed in the literature [16].

There has been extensive research on solution concepts, but less attention has been given to the problem of specifying game models. The model is typically assumed to be given as a starting point, and common knowledge to all players. This is problematic for several reasons. Games that model real world interactions are often complex, with

O. Veliz (✉) · C. Kiekintveld
University of Texas at El Paso, El Paso, USA
e-mail: osveliz@miners.utep.edu

C. Kiekintveld
e-mail: cdkiekintveld@utep.edu

Q. Bai et al. (eds.), *Multi-agent and Complex Systems*, Studies in Computational Intelligence 670, DOI 10.1007/978-981-10-2564-8_6

huge numbers of possible strategies and information states. Formal specifications of these games may be intractable to solve, even for modern supercomputers (or humans). A second problem is that the game itself may be ambiguous and difficult to formalize. Players may not have the same understanding of the possible actions, the payoffs, and the knowledge and beliefs of the other players about the game.

We argue that most game models should be considered *abstractions* that approximate more complex situations. An abstracted game model captures some–but not all–of the relevant detail about the strategic decision. In some cases, abstraction may also be used intentionally as part of the process of analyzing a complex game. For example, work on artificial agents for Texas Hold 'em poker has made extensive use of a methodology that uses abstraction to shrink the size of the game tree before applying an (approximate) equilibrium solver to compute a strategy for the game [14].

We are interested in better understanding the effect of abstraction in game-theoretic analysis. In particular, we focus on the *strategy selection problem*: how should an agent choose a strategy to play in a game, based on an abstracted model of the game? This problem has three interacting components: (1) the method for abstracting the game, (2) the method for selecting a strategy based on the abstraction, and (3) the method for mapping this strategy back to the original game. This approach has been studied fairly extensively for poker, which is a 2-player, zero-sum game. However, much less is known about how abstraction interacts with strategy selection in more general games.

The main contributions of our work are as follows. First, we specify a model of the strategy selection problem when players use asymmetric abstractions as a *meta-game*. In this model players can use different methods for abstracting the game, solving the game, and reverse-mapping the solution. We introduce a collection of specific methods for abstracting and solving games; these are intended to be representative of the most popular methods used in the literature. Finally, we present the results of extensive simulation that evaluate the candidate abstraction and solution methods on different classes of games. Our results lead to several unique observations as well as identifying solution methods that are more robust than others to error introduced by abstraction.

6.2 Abstraction Meta-Games

We first introduce a formal model that can be used to study the situation where players select strategies based on abstracted game models. Our model is based on the meta-game framework introduced by Kiekintveld et al. [11], which focused on situations where players received noisy observations of the same underlying game and had to select strategies based on these observations. The situation where players use abstractions is similar in that the players make strategy choices based on imperfect abstractions of the game. Opposing players may also use different abstractions which may cause problems for solution concepts that rely on coordination (such as Nash equilibrium).

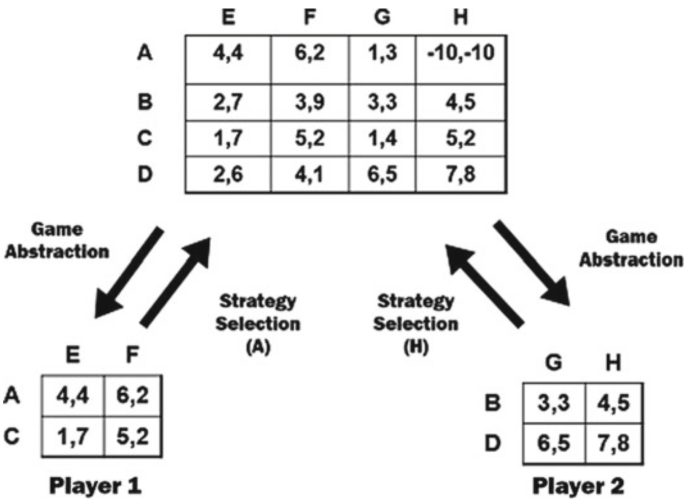

Fig. 6.1 2-players asymmetric abstractions

An example of an *abstraction meta-game* is shown in Fig. 6.1. In this example, we have two players who are playing the one-shot normal form game shown at the top of the figure; this is the *base game*. Each player has four possible actions in the original game, and the payoffs are listed in the the matrix. Each player uses a different (unspecified) abstraction method to reduce the size of the game to only two actions for each player, as shown. Now the players analyze these smaller games to select a strategy to play. Here, both of the small games can be solved using dominance to find a unique Nash equilibrium. The players select these strategies (A and H) to play. However, when these strategies are played in the base game they result in the outcome $-10, -10$, which is the worst possible payoff for both players!

In addition to illustrating the structure of the abstraction meta-game, this example shows one of the possible problems with using abstraction in game analysis. Note that we could replace the payoffs $-10, -10$ with arbitrarily small values. By playing according to a (dominant strategy) Nash equilibrium of the smaller abstracted game, the payoffs the players actually receive are less than the payoffs they expect to received by an unbounded amount. They can also be lower than the equilibrium payoffs in the original game, or even the maxmin payoffs by an arbitrary amount.

This emphasizes the problem of selecting strategies based on abstracted games. Many existing applications of automated abstraction (e.g., in Poker [8, 10, 14, 15]) still analyze the abstracted games to find an (approximate) Nash equilibrium, and play the original game using the equilibrium strategy for the abstracted game. Much of this work focuses on zero-sum games where Nash equilibrium is equivalent to the pessimistic maxmin solution concept, and has a unique value. However, for general-sum games it is clear from our example that the simple approach of analyzing abstracted games using solution concepts that assume perfect knowledge of the game model is highly problematic. Even in the case of poker, there have been pathologies shown in using abstraction [18], which has led to the exploration of alternative methods for playing games using abstraction that are not based on playing according to an approximate Nash equilibrium [7, 9].

We refer to the full model as a *meta-game*, and strategies in the meta-game as *meta-strategies*. The general strategy selection algorithms we are interested in map directly to meta-strategies in this model. We can view players as selecting meta-strategies (solution algorithms) that act on their behalf once game observations are made. Modeling solution algorithms as strategies in a larger game motivates evaluation of the algorithms with game-theoretic concepts, such as regret and equilibrium.

We now present a more formal description of abstraction meta-games. Both the base game and abstracted games are instances of normal-form games, which can be defined by a tuple $\{I, \{S_i\}, \{u_i(s)\}\}$. The set I represents the players, and the sets S_i are the pure strategies available for each player. The utility function u_i maps each outcome to a real number representing the payoff for the player if this outcome is played. We extend the model to allow mixed strategies in the standard way, using the notation σ_i to refer to a mixed strategy that is a probability distribution over the pure strategies. Payoffs for mixed strategies are defined using the expected utilities based on the pure-strategy outcomes.

A *Nash equilibrium* is a strategy profile in which every player is playing a best-response to the opponent's strategies. That is, for all players i and all strategies $s_i \in S_i$, $u_i(\Sigma_i, \Sigma_{-i}) \geq u_i(s_i, \Sigma_{-i})$. We also define the *regret* for a strategy profile to the maximum benefit to be the maximum gain that any player can gain by deviating to a different pure strategy. Any strategy profile with zero regret is Nash equilibrium.

An abstraction meta-game has a base game G in the normal form. This is the game that the players will actually play, and their strategy choices in this game define the payoffs they receive. The meta-game extends the description of the base game to include a description of how players select their strategies in the base game. This description has three components. The *abstraction function* (ω_i) for a player maps the base game G into a smaller abstracted game G'_i that the player will analyze. The *selection function* for a player maps from the abstracted game G'_i into a strategy for the player to play in the abstracted game, σ'_i. The *reverse mapping function* for a player maps from the strategy in the abstracted game σ'_i back to a strategy in the original game σ_i. Collectively, we refer to these three functions as a players *meta-strategy*. The meta-strategy represents a detailed description of how a player analyzes the base game to select a strategy to play.

6.3 Abstraction Methods

We define an abstraction method as a function that maps one normal-form game into a second (smaller) normal-form game. A good abstraction should be simpler than the original, but also retain as much information about the strategic interaction as possible. We identify two broad categories of abstractions that are common in the literature: *strategy elimination* and *strategy combination*.

Strategy elimination abstractions remove some strategies completely to reduce the size of the game. The well-known method of iterated elimination of dominated strategies [16] is one example of this approach. Weakening the criteria for dominance so that weakly or approximately dominated strategies are also removed leads to even smaller games at the cost of potentially removing some equilibria of the original game [3].

Abstractions methods based on strategy combination simplify games by merging multiple strategies into a single representative strategy. In the extreme case where two strategies have exactly identical payoffs for a player in every outcome merging these strategies results in a lossless abstraction. The idea can be extended to merge strategies that are only similar rather than identical, but the result is a lossy abstraction. A common approach in the literature is to use clustering algorithms to merge similar strategies into clusters. In extensive form games, clustering can also be applied to information states [8]. A different type of similarity between strategies can be used to solve a game by reducing it to sub games [4].

Our goal in this paper is not to develop novel abstraction methods, or to exhaustively evaluate the many existing techniques. We are interested in the interaction between abstractions and the solution methods applied to abstracted games. Therefore, we selected two simple abstraction methods for our initial study that are representative of the broad categories described above. We also do not try to guarantee any bounds on these general abstractions as shown in Fig. 6.1 with potentially unbounded error. It is also the case that the simulated agents may not be using an equilibrium or convergent strategy.

6.3.1 TopN Abstraction

The first abstraction method we consider is *TopN*. This method creates a smaller game by selecting a subset of size N of the strategies for each player to form the abstracted game. For each strategy in the game, we first calculate the expected payoff of the strategy against a uniform random opponent strategy. We then select the N strategies with the highest expected payoffs, breaking ties randomly. The abstracted game is simply the game where players are restricted to playing only the N selected strategies; the payoffs are unchanged. Since each strategy in the abstracted games is also a strategy in the original game the reverse mapping of the strategies back to the original game is trivial.

6.3.2 K-Means Clustering Abstraction

The second abstraction method we use is *KMeans*, which is representative of strategy combination methods. This method uses the k-means clustering algorithm to group strategies into clusters based on similarity of their payoffs. Each strategy is represented in the clustering algorithm by a vector of the payoffs for the strategy in every outcome that can result when the strategy is played. We cluster the strategies for each player separately, following the standard k-means algorithm. Each strategy is initially assigned randomly to one of the k clusters. The centroid is calculated for each cluster, and then the Euclidian distance between every strategy and every cluster centroid is calculated. Strategies are re-assigned to the closest cluster, ensuring that no cluster becomes empty. This process iterates until there are no further changes in the clusters. We run the clustering 100 times with different initial clusterings, and select the one with the smallest maximum distance between any strategy in a cluster and the centroid.

Once the strategies are clustered we create the abstracted game as follows. Each cluster maps to a pure strategy in the abstracted game (so the number of strategies is controlled by the k parameter used in k-means). The payoffs for each outcome in the abstracted game are computed by averaging the payoffs for all of the outcomes in the cluster. In other words, we assume that players will play each strategy in a given cluster with equal probability. The reverse mapping also assumes that players play strategies in the same with uniform random probability. For example, if a strategy in the abstracted game places a probability of 0.5 on playing a given pure strategy, this probability would be distributed uniformly over all of the strategies that comprise that cluster in the strategy used to play the original game.

6.4 Candidate Solution Methods

The task of our solution methods is to select a strategy to play in a game. We consider several candidate solution methods for selecting strategies in abstracted games. All of these are based on known solution concepts or simple heuristics for playing games, and they are intended to provide a diverse pool of plausible strategies to evaluate. Little is known of the interactions among these commonly used solution techniques in the presence of abstraction or even in different classes of games. Of particular interest are several solution concepts that originate in behavioral game theory and have been shown to predict human behavior better than Nash equilibrium and related concepts. We hypothesize that humans may be adopting these types of strategies in part because they are more robust to ambiguity and uncertainty.
Uniform Random (UR): Play each pure strategy with equal probability.
Best Response to Uniform Random (BRU): Play the pure-strategy best-response to UR. It is equivalent to a level-1 strategy in the cognitive hierarchies model [2].
Nash Equilibrium (MSNE): We use Gambit [5] logit solver to calculate a sample Nash equilibrium in mixed strategies and play according to this strategy.
Epsilon-Nash Equilibrium (ENE): For every pure-strategy profile, we first calculate the maximum that value (ε) that any player can gain by deviating to a different pure strategy. We select the profile with the smallest value of ε (the best approximate Nash equilibrium in pure strategies) and play the associated pure strategy.
MaxMin: Play the strategy that maximizes the worst-case payoff for the player.

Fair: This heuristic strategy focuses on outcomes that are "fair" in that there is a small difference between the payoffs for the players. For every strategy profile we calculate the difference between the payoffs and select an outcome that minimizes this difference. Ties are broken in favor of outcomes with a higher sum of payoffs, and then randomly.

Social: This strategy plays according to the outcome that maximizes the sums of the payoffs for all players. If there are ties the strategy plays a uniform random strategy over the strategies in the tied outcomes.

Quantal Response Equilibria (QRE): Quantal Response Equilibrium [13] originated in behavioral game theory. It incorporates a model of *noisy best-response* where players use a softmax decision rule instead of strict best-response. A logistic function is normally used to specify this response, and it has a parameter λ that interpolates between a uniform random strategy when $\lambda = 0$ and a best response as $\lambda \rightarrow \infty$. A QRE is defined similarly to a Nash equilibrium, except that both players play noisy best-responses to each other for some value of λ. QRE has been shown to provide a better fit for experimental data on human behavior in some games [13]. QRE has also been shown to have more robust strategy choices that NE in situations where players make choices based on noisy observations of an underlying game [11]. We compute a QRE using Gambit [5] and play a best-response to the predicted QRE strategy of the opponent.

Cognitive Hierarchies (CH): Cognitive Hierarchies [2] also originates in behavioral game theory. It models a recursive style of reasoning where level-0 agents play a uniform random strategy, level-1 agents play a best response to the level-0 agents, level-2 agents play a best response to a mixture over level 0 and 1 agents, etc. Agents at each level use a Poisson distribution, based on a parameter τ, to predict the probabilities of playing agents at lower levels, and play a best response to this mixture.

Quantal Level-k (QLK): This method combines the features of QRE and CH. It uses a recursive reasoning model identical to CH, except that in place of the best-response at each level agents play a noisy best-response using the same logit function used in QRE. It has parameters for both λ and τ. We play a best-response to the predicted strategy of the opponent, based on the specified level of reasoning.

6.5 Experimental Methodology

We run simulations using the meta-game framework from Sect. 6.2. Our testbed allows us to run tournaments with different combinations of (1) classes of base games, (2) abstraction methods, and (3) solution methods. The simulation first generates a random game from the class as the base game. Each meta-strategy is a combination of an abstraction method and a solution algorithm. We calculate the strategies selected by each meta-strategy and then play a round-robin tournament among these strategies. The payoffs are based on the strategy profile that is played in the original game.

The result of the tournament is a payoff matrix for the meta-game where each meta-strategy can be played by either player. To estimate this matrix for a class of base games we average the payoffs over a large number of sampled games. We can analyze this payoff matrix to generate several performance metrics for the meta-strategies. Average payoffs are quite biased, in that they can reward strategies that only do well against other weak strategies. Instead, we present our results using *stability* and *average regret* measures. Stability is a measure of how close the pure strategy profile where all players use the same meta-strategy is to being a pure Nash equilibrium. It is calculated by finding the maximum gain for deviating to another meta-strategy from this profile. Any value less than or equal to zero indicates that the profile is a pure equilibrium of the estimated meta-game. Average regret is the average of the regret for playing a given meta-strategy against each other meta-strategy in the estimated game.

We use three classes of randomly-generated games: general sum, zero sum, and logical games. General sum and zero sum are generated using GAMUT [6]. Players have 20 pure strategies, and payoffs are real numbers in the range [–100, 100]. Logical games have more structure and should be more amenable to abstraction. Pure strategies are based on choosing either 0 or 1 for a set of boolean variables. Payoffs are based on randomly generated oppositions with varying numbers of clauses, literals, and values. If a proposition is true for a given outcome in the game, the payoff associated with the proposition is added to the payoff for that player. The payoffs are additive over all true propositions for a given outcome. Finally, payoffs are normalized to the range [0,100]. We also could not possibly evaluate every level of abstraction so we picked several interesting abstraction levels. An abstraction level of 10 means that there are now 10 actions in the game, Top10 or 10 clusters, and 20 indicates these are the original 20 pure strategies with no abstraction.

6.6 Results

Our experiments are designed to test the relationships between abstraction methods and solution concepts across different classes of games, and with varying levels of abstraction. In total, we experiment with three different classes of games, two types of abstraction, four levels of abstraction, and more than 30 different solution methods (including multiple parameter settings for QRE, CH, and QLK). The main results span 24 different tournaments with 500 sample games played in each one.

The full data set is difficult to visualize, so we will present results focusing on selected subsets of agents to illustrate key points. We focus on measures of stability and regret here as we believe they are the most informative, but we have also looked at measures of average payoffs and exploitability. Average payoffs can be misleading for agents that exploit weak players but perform poorly against strong players. Exploitability does not differentiate well among the meta-strategies, since almost all of them are highly exploitable in the worst case.

We first present a set of results that demonstrates how we compare the different parameter settings for QRE, CH, and QLK. The purpose of this analysis is to discover which parameter settings are strongest. In later results we will include only best parameter settings for each solution method to aid in visualization.

Figures 6.2 and 6.3 show results for QRE with different values of the λ parameter. Each plot shows results for all three classes of games. There are separate plots for each type of abstraction. The x-axis shows the level of abstraction where the values are the number of actions in the abstracted game. The point on the far left corresponds to no abstraction, and more information is lost for data points further to the right. The y-axis represents the stability (ε) value for each solution method.

Lower stability values are better. In particular, any value less than or equal to 0 indicates that a solution method is a pure-strategy Nash equilibrium, and if all players were using this method none of them would benefit by deviating to any other solution method. These stabilities are calculated with respect to the full data set, meaning that players are allowed to deviate to *any* solution method, not just the ones in the figure.

In general sum and logical games, the best QRE parameter settings are closer to playing a best-response to a more random opponent than playing a best-response to an opponent closer to a Nash equilibrium. For example, the very low parameter setting of $\lambda = 0.05$ performs very well, and this setting is the closest to uniform

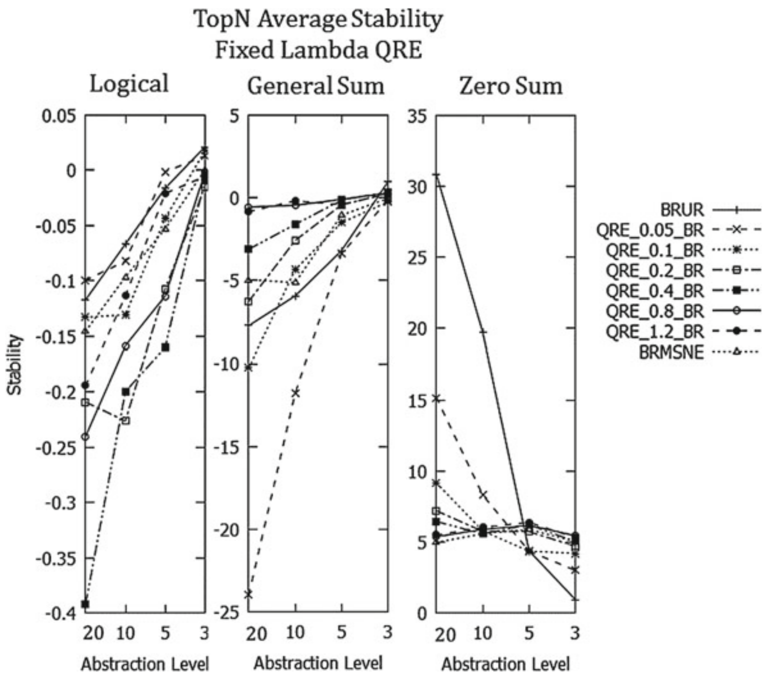

Fig. 6.2 TopN stability fixed lamda QRE

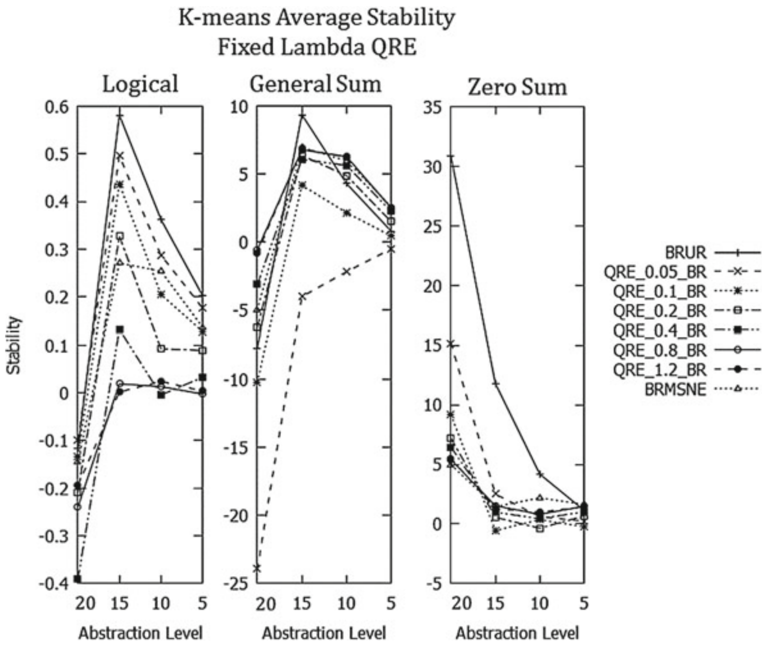

Fig. 6.3 K-means stability for fixed lambda QRE

random. Zero-sum games appear to behave quite differently where playing closer to an equilibrium strategy performs better.

We perform a similar set of experiment to determine the parameterizations of CH and QLK agents for the values of k, τ, and λ. We omit them due to space considerations, but use a similar procedure to identify the most effective parameter settings to visualize in the later plots. The values of τ considered are 1.5, 2.5, 5, 7.5, and 9, the values of k are 0, 1, 2, 3, 5, 10, and 20, and the values of λ are 0.8, 1.5, 2.0, 5.0, and 20. For QLK agents we also consider both version that play the raw QLK strategy and ones that play a best response to the strategy of the opponent for a particular level. One results to note from this analysis is that for CH agents with the same τ but different levels of reasoning there is not much difference between agents with mid-level k versus high level k. The main variations between these agents occur at lower levels. However, higher levels of k typically have better, more robust performance.

We now turn to analysis of the complete set of agents. The following results are from a fully symmetric tournament of 500 games for each class of games and 30 agents which include the best parameter settings for QRE, CH, QLK, and the other main solution concepts. We selected the best parameter settings for each of the QRE, CH, and QLK variants based on the results of the previous analysis to visualize in these results along with the other agents.

Figures 6.4, 6.5 and 6.6 show the average stability and regret for the different solution methods (excluding the lines for all but the best QRE, CH, and QRE settings) when using the KMeans and TopN abstractions in different classes of games. We begin by noting some general trends in the results. The results for logical games in Fig. 6.6 shows how all of the stability and regret values are very low compared to both general-sum and zero-sum games. There is also very little variation among each of the strategies. This indicates that these games are much easier to play well in, even when they are abstracted. In most cases, agents are more stable for cases without any abstraction, and less stable but converging to the same range as abstraction increases. However, regrets are very high for cases without abstraction and lower for cases with abstraction. Zero-sum games behave somewhat differently; the overall stability is worse, and the stability of many of the agents actually improves (surprisingly) as abstraction increases.

There is no solution concept that is clearly the best across all of the settings tested. Instead, the best strategy depends heavily on both the class of games and the type of abstraction being used. The UR and Fair agents do poorly in almost all cases. The ENE and Social agents perform surprisingly well, particularly when using the TopN abstraction and in the general-sum and logical games. These agents focus on coordinating on high payoffs, so it is interesting that even these simple strategies are able to do so well even in cases with abstraction error. For the cases with KMeans abstraction and for zero-sum games, approaches close to equilibrium solvers perform better particularly MSNE and the QRE and QLK.

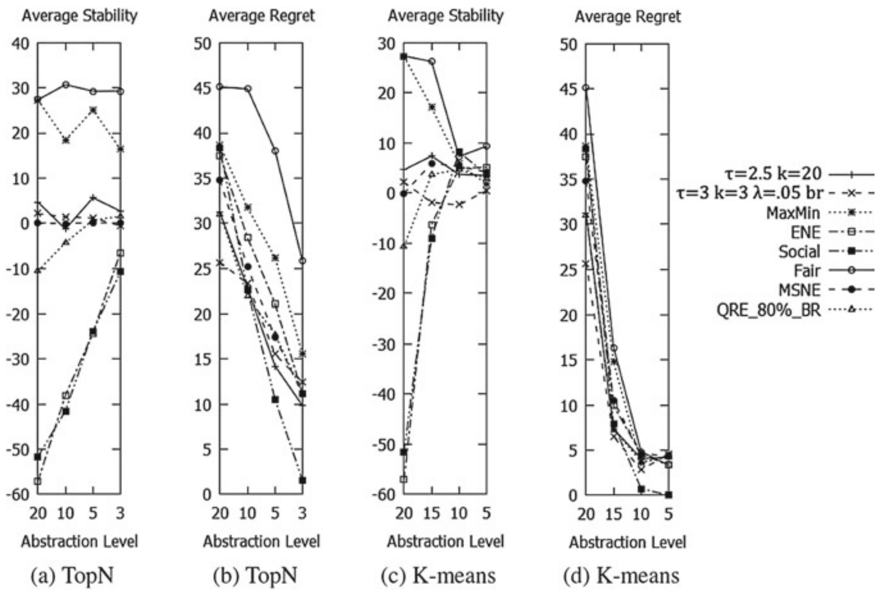

Fig. 6.4 Average stability and regrets for general sum games

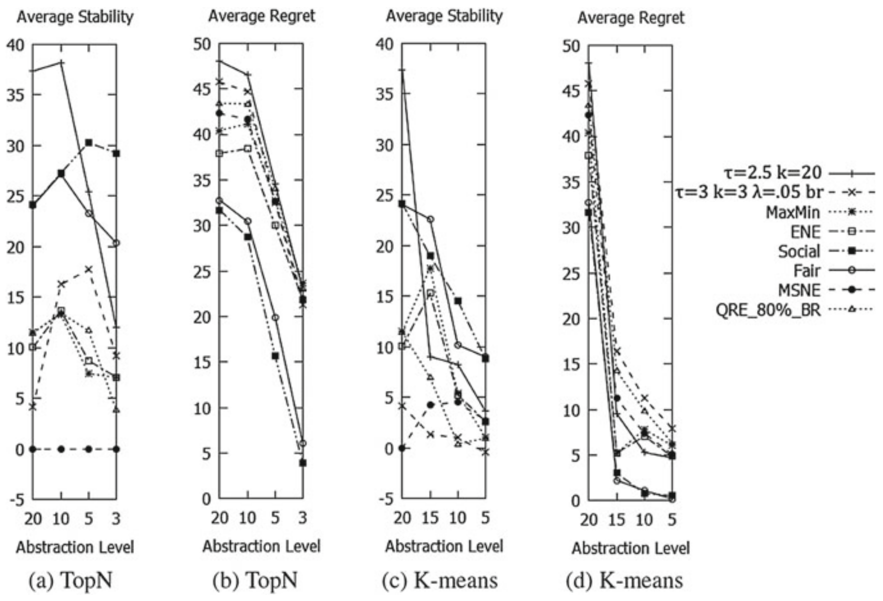

Fig. 6.5 Average stability and regrets for zero-sum games

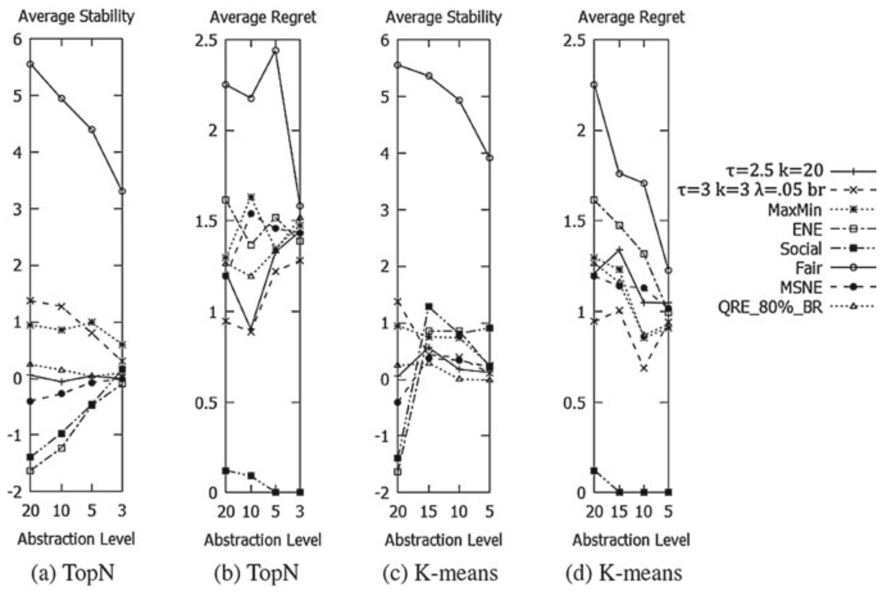

Fig. 6.6 Average stability and regrets for logical games

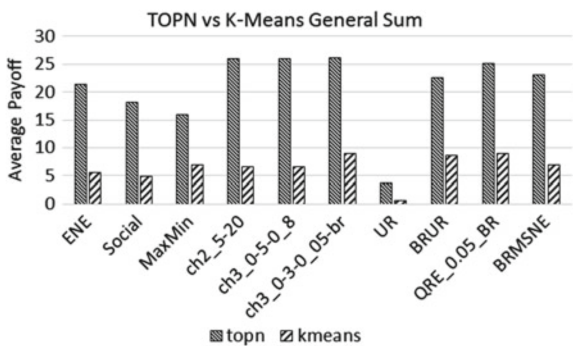

Fig. 6.7 TopN versus Kmeans in general sum—abstraction level 5. The third CH is QLK

While this pattern of results fails to identify a clear winner among the common solution methods for how to select strategies in abstracted games, the results are still quite interesting. They clearly show that both the game class and abstraction method interact strongly with the applied solution concept, and simply finding a NE is not the best approach. There is also an opportunity to develop significantly improved solution concepts that account for these factors unlike current practices.

The final experiment we present directly compare the performance of the two different types of abstractions. In this experiment, players can use either abstraction, which leads to even more asymmetry in the abstractions players are using.

In Fig. 6.7 we show the average payoffs for the different solution methods in a tournament that combines agents using both abstractions. We only use the best parameter settings for the QRE, CH, and QLK agents. This tournament was run on general-sum games, using the highest level of abstraction (from a game size of 20 down to a game size of 5). For each solution method we show the average payoffs achieved when using the two abstraction methods side by side. Interestingly, the TopN abstraction method outperforms KMeans in combination with *every one* of the solution methods, often quite substantially. The TopN abstraction plays a role in coordinating agents on high-payoff outcomes independent the solution algorithm.

6.7 Conclusion

Our results demonstrate that using abstraction to solve games is a complex endeavor, and the type of abstraction, the solution methods used to analyze the abstracted games, and the class of games all have a strong influence on the results. Many of the strongest results using abstraction to analyze large games (e.g., Poker) have focused on zero-sum games. One of the most interesting observations from our results is that abstraction often works very differently in zero-sum games than it does general-sum games or the more structured logical games. In particular, solution methods based on finding Nash equilibrium seem to work much better in zero-sum games than they do in the other classes of games in our experiments. Another important observation from our experiments is that Nash equilibrium often does not perform well in cases where abstraction is used as part of the solution process. It is still effective when the games are zero-sum, but in the other cases it was not robust to the introduction of error based on game abstraction. The two solution methods that were the most robust to abstraction error across the many different settings were BRU and QRE. Of the two, QRE was more consistent and had stronger performance in most cases, but it was not dominant in all situations.

We also found that the specific method used for generating abstractions has a strong impact on the results. One very interesting result was that TopN was sometimes able to *increase* the payoffs for the agents in comparison to the case without any abstraction. However, TopN is a symmetric abstraction when both players use it, while KMeans is asymmetric. Some methods like the social agent performed much better when using the symmetric TopN abstraction than when using the asymmetric KMeans abstraction. This kind of interaction is very important to understand in greater depth if we are to make effective use of abstraction as part of game-theoretic analysis. Our model of abstraction meta-games provides a formal model for studying this type of interaction, and our simulations have resulted in several interesting observations that provoke many additional questions about the use of abstraction in solving games.

Future work will include other types of abstraction and tests on even larger games. We plan to also develop new robust solution techniques that can perform well in the presence of abstraction. It is also important to test on games with more realism like a simplified version of poker or a security game to examine how general abstraction and robust strategy selection will perform in these kinds of games.

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant No. IIS-1253950.

References

1. Alpcan, T., Basar, T.: Network Security: A Decision and Game Theoretic Approach. Cambridge University Press (2011)
2. Camerer, C.F., Ho, T.H., Chong, J.K.: A cognitive hierarchy model of games. Q. J. Econ. **119**(3), 861–898 (2004)
3. Cheng, S., Wellman, M.: Iterated weaker-than-weak dominance. In: IJCAI, pp. 1233–1238 (2007)
4. Conitzer, V., Sandholm, T.: A technique for reducing normal-form games to compute a nash equilibrium. In: Proceedings of the 5th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-06), pp. 537–544 (2006)
5. Gambit: Gambit. http://gambit.sourceforge.net/ (2014). Accessed 01 Aug 2014
6. GAMUT: Gamut. http://gamut.stanford.edu/ (2014). Accessed 01 Aug 2014
7. Ganzfried, S., Sandholm, T., Waugh, K.: Strategy purification and thresholding: effective non-equilibrium approaches for playing large games. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems. AAMAS'12, vol. 2, pp. 871–878 (2012)

8. Gilpin, A., Sandholm, T.: Expectation-based versus potential-aware automated abstraction in imperfect information games: an experimental comparison using poker. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008), pp. 1454–1457 (2008)
9. Johanson, M., Bard, N., Burch, N., Bowling, M.: Finding optimal abstract strategies in extensive-form games. In: AAAI (2012)
10. Johanson, M., Burch, N.: Evaluating state-space abstractions in extensive-form games. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 6–10 (2013)
11. Kiekintveld, C., Wellman, M.: Selecting strategies using empirical game models: an experimental analysis of meta-strategies. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS'08, vol. 2, pp. 1095–1101 (2008)
12. Krishna, V.: Auction Theory. Academic Press (2002)
13. McKelvey, R., Palfrey, T.: Quantal response equilibria for normal form games. Games Econ. Behav. **10**(1), 6–38 (1995)
14. Sandholm, T.: The state of solving large incomplete-information games, and application to Poker. AI Mag. Spec. Issue Algorithmic Game Theor. 13–32 (2010)
15. Sandholm, T., Singh, S.: Lossy stochastic game abstraction with bounds. In: Proceedings of the 13th ACM Conference on Electronic Commerce. EC'12, vol. 1, pp. 880–897 (2012)
16. Shoham, Y., Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press (2009)
17. Tambe, M.: Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned. Cambridge University Press (2011)
18. Waugh, K., Schnizlein, D., Bowling, M., Szafron, D.: Abstraction pathologies in extensive games. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 781–788 (2009)