CI in General Game Playing - to date achievements and perspectives

Karol Walędzik and Jacek Mańdziuk

Faculty of Mathematic and Information Science,
Warsaw University of Technology,
Pl. Politechniki 1, 00-661 Warsaw, Poland,
{k.waledzik,j.mandziuk}@mini.pw.edu.pl

Abstract. Multigame playing agents are programs capable of autonomously learning to play new, previously unknown games. In this paper, we concentrate on the General Game Playing Competition which defines a universal game description language and acts as a framework for comparison of various approaches to the problem. Although so far the most successful GGP agents have relied on classic Artificial Intelligence approaches, we argue that it would be also worthwhile to direct more effort to construction of General Game Players based on Computational Intelligence methods. We point out the most promising, in our opinion, directions of research and propose minor changes to GGP in order to make it a common framework suited for testing various aspects of multigame playing.

1 Introduction

One of the most interesting areas of contemporary research on application of Artificial Intelligence (AI) and Computational Intelligence (CI) to mind games is the topic of multigame playing, i.e. development of agents able to effectively play any game within some general category being informed only about the rules of each of the games played. This poses a unique challenge to the AI community, as all the most successful game playing agents to date have been developed to achieve master level of play only in their specific games. Creating a system exhibiting high playing competency across a variety of previously unknown games would be a significant step in CI/AI research.

In the remainder of this paper we introduce the General Game Playing (GGP) framework and deal with the CI perspectives in GGP. We devote chapter 4 to analysis of possible machine learning approaches to GGP – identifying elements of existing programs (mainly AI-based) that can be incorporated into soft learning solutions and proposing a number of possible research directions. Finally, in chapter 5 we argue that GGP can easily become a universal multigame playing platform, useful in many research areas, even outside the context of the GGP tournament, as long as some necessary extensions are introduced into the standard.

2 General Game Playing Competition

General Game Playing (GGP) [3] is one of several approaches to the multigame playing topic. It was proposed at Stanford University in 2005 in the form of General Game Playing Competition held annually at the National Conference for Artificial Intelligence [4]. General Game Players are agents able to interpret game rules described as a set of Game Description Language (GDL) [6] statements in order to devise a strategy allowing them to play those games effectively without human intervention.

The competition always includes a wide variety of games, both known previously and devised specifically for the tournament. Contestants should be prepared to deal with games of various complexity, varied branching factors and numbers of players, both cooperative and competitive.

2.1 Game Description Language

Game Description Language (GDL) [6] is used to describe the rules of the class of games playable within the GGP framework, i.e. finite, discrete, deterministic multi-player games of complete information. GDL describes games in a variant of Datalog. Game states are defined in terms of facts and algorithms for computing legal moves, subsequent game states, termination conditions and final scores for players are represented as logical rules.

3 GGP competition winners

In this section, selected most notable achievements in the field of GGP agents development are described. As it will become evident in the following sections, the winners of the first four editions of the contest relied on AI rather than CI methods. We believe, however, that elements of these successful AI solutions may be transferable to more CI-focused approaches, and these transferable aspects will be described in more detail in further chapters.

3.1 Cluneplayer

Cluneplayer [1], developed by James Clune, was the champion of the first and vice-champion of the second GGP tournament. It relies heavily on game domain specific observation that most mind games share a number of common concepts important in close-to-optimal play. Extracting definition of these crucial game features from game description should allow construction of a new simplified game in the form of a compound lottery based on the three core aspects of the original game: *expected payoff*, *control* (or mobility) measure and *game termination* probability (or game longevity). The expected outcome of thus created model approximates original game state evaluation.

3.2 Fluxplayer

Fluxplayer [9] was developed by S. Schiffel and M. Thielscher and proved superior to Cluneplayer in the second GGP championship. Similarly to Cluneplayer it depends on depth-first game tree search algorithm with widely known enhancements and automatically generated evaluation function based on fuzzy logic concepts. In each analyzed state approximate truth values of terminal and goal formulas are calculated and the program attempts to end the game whenever its goal is attained and avoid reaching terminal states when it would mean its loss.

3.3 CadiaPlayer

Cadiaplayer, developed by H. Finnsson and Y. Björnsson [2], is the first program that managed to win the GGP tournament two times in a row – in 2007 and 2008. Unlike earlier champions, it does not concentrate on advanced analysis of game description, relying, instead, on strong simulation-based game tree evaluation algorithm. Its operation is, in general case, based on Monte Carlo simulations enhanced by the UCT (Upper Confidence bounds applied to Trees) method. Cadiaplyer repeatedly plays partially random matches, gathering statistical data on each moves' relative strength. The move choice routine in each of the simulated matches is partially guided by the move quality data gathered so far.

4 Computational Intelligence perspectives in GGP

So far all GGP champions relied on traditional AI approaches in the form of deterministic analysis of game description (in order to identify some well known patterns) and sophisticated game tree search algorithms. This is probably, to a large extent, caused by severely limited time allotted for learning before the actual match starts, while most CI learning processes require repetitive and time-consuming learning patterns presentation or environment sampling.

Even though current tournament rules make it difficult for an agent relying on Computational Intelligence methods to compete against deterministic symbolic approaches and/or brute-force game tree analysis algorithms in the competition itself, GGP-like environment with increased learning time limits can be used as a common test bed for various approaches to multigame playing agent implementation. In the following sections we intend to present our view on the promising research directions and concepts in this area.

Most perfect-information deterministic game playing agents make use of various minimax algorithms, and so far there is no evidence that this approach should be unsuitable for multigame playing. The most difficult (and interesting) part of minimax search-based agent development is construction of its game state evaluation function. There are many possible representations of this heuristic, but two of them have gained most popularity: linear combination of selected state features and artificial neural networks. In any case, the input game features should first be identified. Before presenting our solutions to those questions, we want, however, to take a peek at a CI-based GGP agent, which we think may provide inspiration for further research in the field of multigame playing with machine learning methods.

4.1 nnrg.hazel

nnrg.hazel [8] is one of the applications that suggest that CI-based approaches may yet prove successful in GGP-like problems. It was developed by J. Reisinger, E. Bahçeci, I. Karpov and R. Miikkulainen and came 6th (out of 12 contestants) in 2006 GGP Competition (5th in the preliminary rounds).

nnrg.hazel makes use of minimax game tree search method based on alpha-beta pruning algorithm with a depth limit of one ply. Game state evaluation function is represented as an artificial neural network (ANN). It is generated by a co-evolutionary method, evolving network topology and connection weights simultaneously. This is done using a method called NeuroEvolution of Augmenting Topologies (NEAT) [10]. It starts with the simplest possible fully-connected network with input and output layers only, which is then incrementally complexified to incorporate gradually more complex concepts without forgetting the knowledge already acquired.

The most obvious weakness of nnrg.hazel is its lack of any 'intelligent' ANN input data generation algorithm. Instead, the agent simply relies on random projection of game state features onto a 40-node ANN input layer. Improvements in this area, described more closely in the following section, seem to be the most obvious path of further development of NEAT-based General Game Players.

4.2 Game state features identification in existing solutions

While nnrg.hazel is able to achieve surprisingly promising results with only random projection of state description facts onto the evaluation function input vector, even its authors admit that it can be expected to fare much better with better input features generation routines. The input vector of a state evaluator should ideally include intelligently selected both static (e.g. pieces counts) and dynamic (e.g. mobility) features of the game state.

The first widely cited and influential paper on game features identification and generation was published by G. Kuhlmann, K. Dresner and P. Stone [5]. Their approach consists in syntactical analysis of game description (supported by simple simulations) in search for a set of general patterns, such as *successor relations* (inducing ordering), *counters* (incrementing in each time step according to successor relation), two-dimensional *boards*, *markers* (occupying cells on a board) and *pieces* (markers that can exist at at most one cell on a board at a time). Having identified these structures, it is possible to easily create a set of game features such as Manhattan distances between pieces or number of occurrences of markers.

Cluneplayer's game state features identification process is to some extent similar. A set of candidate features contains initially all expressions found in the game description. Additional features are then generated by automatic discovery of possible constants that can be substituted for all the variables in these expressions. Dedicated analysis algorithm can furthermore identify constant values that are, in some way, 'special' (i.e. differ significantly from the rest). Afterwards, Cluneplayer attempts to impose interpretations on the resulting features. The three available interpretations are *solution cardinality*, *symbol distance* and *partial solution*.

Fluxplayer takes the concept even further by replacing syntactic analysis of game description with exploration of semantical properties of rules definitions. That way higher level concepts can be detected more easily and with greater confidence. Fluxplayer evaluation function may make use of structures such as successor and order relations.

4.3 Constructing game state features in CI-based solutions

All the above-mentioned approaches have proved relatively successful as part of AI programs and we think that they may also be utilized in CI-based solutions for creating input vectors for evaluation functions in any form. What is worth noting, is that the patterns identified by the described applications generally fall into two categories:

1. universal logical predicates (e.g. successor and order relations) and expressions explicitly provided in game rules;
2. 'real-world' mind game specific predicates (e.g. boards, pieces).

We believe that truly universal and purely CI-based GGP agent should not include any of the latter, as there is no inherent reason stemming from the problem specification to assume that, e.g., the concept of board should be generally applicable.

In many cases the number of identified game features will be too big to directly include them all in the evaluation function (whatever its form). In such cases, some selection or dimensionality-reduction mechanism will be required. This task can be accomplished by deterministic methods; they may,

The simplest approach might be parallel construction of several state evaluators with varied input features sets and occasional comparison to decide which of them should be maintained in the pool of candidate solutions and which should be discarded. New candidate features sets can be generated randomly or via modification of existing sets depending on training results, e.g. utilizing sensitivity analysis.

4.4 Soft learning in GGP research direction proposals

Although the specifics of multigame playing application must differ significantly from the single-game programs, we believe that much of the knowledge gathered during development of single-game learning agents is transferable to the multi-game case. One of the distinguishing aspects of GGP agents is that game tree traversal and identification of legal moves may prove much more time-consuming than in the case of single-game programs, since in GGP both these operations require costly theorem proving. Depending on the detailed setup of the learning approaches, this fact may make some methods less useful than others. Particularly, coevolutionary training schemes may turn out to be slower than expected.

We believe that one of the approaches that may be effective in terms of game tree traversal cost might be the layered learning scheme described, in the context of evolutionary algorithm, in [7]. The method requires dividing the game into several disjoint stages and gradually creates evaluation function applicable to all of them. First, a number of end-game positions are generated via random play. These positions are analyzed with minimax algorithm without evaluation function with the expectation that the search will reach terminal states in most of the cases; the rest will have neutral value assigned. This way a training set of game states with their evaluations is generated and any supervised learning method can be used to construct the heuristic evaluation function. A number of new positions from an earlier game stage is then generated and evaluated using minimax search with depth limit of at least game stage length and the heuristic evaluation function from previous step. The new training set is used for further improvement of evaluation function. This process is repeated until the game tree root is reached.

GGP covers a very wide spectrum of possible games. Many machine learning algorithms can prove very sensitive to the choice of their parameters and game-specific features. Ideal General Game Player should, therefore, be able to tune its learning patterns to the problem at hand. We propose two basic approaches allowing to achieve this goal.

Firstly, the agent's training scheme could involve tuning the learning algorithm's steering parameters. This task could be performed by employing some kind of an evolutionary approach analogical to the one proposed for input features selection. A population of candidate solutions would be trained with separate sets of steering parameters. Their training results would occasionally be compared in order to identify the most promising individuals. The weakest would then be disposed of and the strongest reproduced into the next population. At some point the parameters would usually have to be frozen, and training of single candidate would continue to further optimize the evaluation function until some stop condition (e.g. running out of time) was met.

More sophisticated applications might take the same idea further and try to not only choose the best steering parameters but the training algorithm itself, comparing, for instance, various representations of evaluation function and/or learning schemes. The principles of the selection process would remain similar to those described in the previous paragraph, with parallel application of all the schemes and occasional comparison of their quality to abandon the weakest solutions. Alternatively, a whole ensemble of evaluation functions could be constructed along with rules describing how their results should be combined depending, for instance, on game phase.

5 GGP framework extension proposals

5.1 Opponent modeling

GGP, in its current form, does not offer enough information to seriously attempt long-term modeling of individual opponents. Contestants typically resort to one of popular simplifications: assuming that other players will implement decision process analogical to the playing agent's one, treating other players as opponents attempting to minimize the agent's score or simply considering random reactions of other players.

GGP framework can, however, easily be extended to include unique identifiers of players, thus forming a useful opponent modeling test bed. Game playing agents would then be able to gather knowledge about individual players' behavior and transfer it from match to match. Full opponent modeling scheme should also attempt to identify cross-game elements of player style, such as its aggressiveness (tendency to play risky moves, that may yield both high rewards and losses, depending on other players' responses), ability of long-term, strategic planning, probability of employing mobility strategies and so on. Identified traits could then be incorporated into game simulation and minimax tree analysis algorithms to better predict opponents' behavior.

5.2 Knowledge transfer

Analogically to the case of opponent modeling, cross-game knowledge transfer is a goal very difficult to pursue in the current form of GGP framework. Game playing agents are not provided with game names or categorization and, during the tournament, even the GDL constants and predicate identifiers are garbled to devoid the programs of any lexical clues to their meaning. In this context, knowledge transfer would require very sophisticated game description analysis able to identify and extract patterns common to the games.

GGP framework communication protocols could, however, yet again be slightly modified in order to create a cross-game knowledge transfer testing environment, in which agents are provided with game names and categorization and GDL atoms with same names represent analogical data in varied games. In the simplest approach, game agent could attempt to store metaparameters that proved most successful in evaluation function learning for each game and attempt to select one of those sets whenever a new game is encountered (relying on game similarities). In case of games classified into the same category and sharing a significant number of concepts, the learning phase could gain a head-start by starting from solutions based on evaluators successful in similar games, instead of learning from scratch.

6 Summary

Although not new, the concept of multigame playing remains a very interesting and still little explored research area. We believe that it is well suited for application and comparison of various Computational Intelligence methods. One of the important factors that could help this research area flourish, would be a common framework defining class of games along with their description language that should be understood by the game playing agents and allowing direct comparison of various approaches. General Game Playing competition offers exactly that.

Although so far the tournaments seem better suited for symbolic approaches, we argue that it is possible to create a reasonable CI-based GGP player. In this paper, we have discussed the key aspects of developing CI-based General Game Players, identified the elements of the existing applications that can be transferred to machine learning approaches and proposed several promising research directions for the CI community. Implementation and validation of some of these proposals is, at the moment, a work in progress.

At the same time, we have pointed out that some aspects of multigame playing, such as opponent modeling and cross-game knowledge transfer, are prohibitively hard to tackle in the current form of the GGP framework. We therefore propose slight modifications to its definition so that it can become a standard multigame playing platform for testing and comparison of various approaches, even outside the scope of the annual championship.

References

1. J. Clune: Heuristic evaluation functions for General Game Playing. In Proceed- ings of the Twenty-Second AAAI Conference on Articial Intelligence (AAAI- 07), pages 1134–1139, Vancouver, BC, Canada, 2007. AAAI Press.
2. H. Finnsson, Y. Björnsson: Simulation-based approach to General Game Playing. In Proceedings of the Twenty-Third AAAI Conference on Articial Intelligence (AAAI-08), pages 259–264, Chicago, IL, 2008. AAAI Press.
3. General Game Playing website. http://games.stanford.edu/.
4. M. Genesereth, N. Love: General Game Playing: Overview of the AAAI Competition. http://games.stanford.edu/competition/misc/aaai.pdf, 2005.
5. G. Kuhlmann, K. Dresner, and P. Stone: Automatic heuristic construction in a complete General Game Player. In Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence (AAAI-06), pages 1457–1462, Boston, MA, 2006. AAAI Press.
6. N. Love, T. Hinrichs, D. Haley, E. Schkufza, M. Genesereth: General Game Playing: Game Description Language Specification. http://games.stanford.edu/language/spec/gdl_spec_2008_03.pdf, 2008.
7. J. Mańdziuk, M. Kusiak, K. Walędzik: Evolutionary-based heuristic generators for checkers and give-away checkers. Expert Systems, 24(4): 189–211, Blackwell-Publishing, 2007.
8. J. Reisinger, E. Bahçeci, I. Karpov and R. Miikkulainen: Coevolving strategies for general game playing. In Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG'07), pages 320–327, Honolulu, Hawaii, 2007. IEEE Press.
9. S. Schiffel, M.Thielscher: Automatic Construction of a Heuristic Search Function for General Game Playing. In Seventh IJCAI International Workshop on Non-monotonic Reasoning, Action and Change (NRAC07).
10. K. O. Stanley and R. Miikkulainen: Evolving neural networks through augmenting topologies. Evolutionary Computation, 10(2):99–127, 2002.