Evolving Players that Use Selective Game-Tree Search with Genetic Programming

Amit Benbassat
Dept. of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
amitbenb@cs.bgu.ac.il

Moshe Sipper
Dept. of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
sipper@cs.bgu.ac.il

ABSTRACT

We present the application of genetic programming (GP) to evolving game-tree search in board games. Our work expands previous results in evolving board-state evaluation functions for multiple board games, now evolving a search-guiding evaluation function alongside it. Our system implements strongly typed GP trees, explicitly defined introns, and a selective directional crossover method.

Categories and Subject Descriptors

I.2.6 [**Parameter learning**]: Knowledge acquisition; I.2.1 [**Applications and Expert Systems**]: Games; I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods

General Terms

Design

Keywords

Genetic programming, games, board games, search

1. INTRODUCTION

Developing players for board games has been part of AI research for decades. Board games have precise, easily formalized rules that render them easy to model in a programming environment. In this work we will focus on full knowledge, deterministic, two-sum board games, expanding on our previous results [1, 2]. Our previous work applied GP to evolving an evaluation function to work in conjunction with the existing alpha-beta search algorithm. We now begin to explore how search may be directed using evolved functions.

We chose to use Nine Men's Morris as our prototype game for this purpose. Our method of choice to guide search is *Forward Pruning*, meaning that while search is going on some nodes are discarded and not expanded further due to an assessment that they do not represent good moves. Unlike standard alpha-beta pruning this approach does not ensure that play strategy will not change due to pruning, as the pruning is implemented using a heuristic function that attempts to assess the relative value of sibling nodes in the game tree.

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

Table 1: Nine Men's Morris: Results of top runs with unselective $\alpha\beta$ search. *Player* uses $\alpha\beta$ search of depth 3 coupled with evolved evaluation function, while *Benchmark Opponent* uses $\alpha\beta$ search of depth 4 coupled with a material evaluation function. Benchmark score refers to the number of games against *Benchmark Opponent* won by *Player* in a 1000-game tournament. A drawn game is considered half a victory for the player in this calculation.

Run identifier	Fitness Evaluation	Benchmark Score
r00144	20Co	953.0
r00145	20Co	978.5
r00146	20Co	947.0
r00147	20Co	923.0
r00148	20Co	919.0
r00149	20Co	932.0

2. NINE MEN'S MORRIS

Nine Men's Morris is a board game that was popular in the middle ages and dates back to ancient times. It is played on a relatively small, 24-square board (which our implementation translates into a 3x8 board, though that is not how it appears with actual, physical boards). The objective of the game is to eliminate all opponent pieces by way of making special patterns called "Mills" using the friendly pieces on the board.

Though solved by Gasser [3] it is still considered challenging for human players. This game has two distinct stages, the first being a piece-placing stage and the second being a piece-moving stage. We have used our system (described in [1, 2]) to evolve alpha-beta players using a search depth of 3 that vastly outperform handcrafted alpha-beta players using a search depth of 4. The results of these evolutionary runs can be seen in Table 1. All runs in this work use a population size of 150, 100 generations, crossover probability of 0.8, mutation probability of 0.2. All runs utilize tournament selection with a tournament size of 2, as well as our selective crossover method. Fitness in all runs is calculated by having the individuals play against each other (i.e., coevolution). Individuals receive 1 fitness point per victory, and half a point for each drawn game.

In order to apply our system to Nine Men's Morris, we used some domain specific nodes. These are shown in Table 2.

3. RESULTS USING FORWARD PRUNING

Our system currently supports two approaches to selective search via forward pruning. One relies on a parameter called *SelectiveSearchRatio*. This parameter is a floating-point number in the range $(0, 1]$. It sets the ratio of sibling states that get further expanded as long as the final search depth has not been reached (the number of sib-

Table 2: Nine Men's Morris specific terminal nodes.

Node name	Type	Return value
`FriendlyMillCount()`	F	Number of mills friendly player has
`EnemyMillCount()`	F	Number of mills enemy player has
`TotalMillCount()`	F	`FriendlyMillCount()` – `EnemyMillCount()`
`FriendlyAlmostMillCount()`	F	Number of almost certain mills friendly player has
`EnemyAlmostMillCount()`	F	Number of almost certain mills enemy player has
`FriendlyPossibleMillCount()`	F	Number of possible mills friendly player has
`EnemyPossibleMillCount()`	F	Number of possible mills enemy player has
`FriendlyIntersectionCount()`	F	Number of intersections friendly player holds
`EnemyIntersectionCount()`	F	Number of intersections enemy player holds
`FirstPhaseCheck()`	B	True iff game is in first (piece-placing) phase

lings actually expanded is rounded up to the nearest integer). If, for example, *SelectiveSearchRatio*=0.25, this means that out of every four children nodes a board state has in the game-tree, one will be expanded further and the others will be pruned. The other approach relies on a parameter called *MaxBranchingFactor*. This parameter is a positive integer and sets a hard limit for the effective branching factor of the searched game tree. If, for example, *MaxBranching-Factor*=5, this means that no more than five sibling nodes in the game tree will be expanded for further search. Our system can use either or both parameters to limit the breadth of its search. We also implemented a method to temper the ill effects of too much forward pruning, using a third parameter *FullSearchDepth*. This parameter is a non-negative integer and dictates that the search algorithm will behave normally up to a certain given depth. If, for example, *FullSearchDepth*=2, this means that up to depth 2 in the search tree all nodes that the base search algorithm (alpha-beta in our case) would normally expand will also be expanded by the selective search algorithm. Control of the maximal search depth is a feature that exists in our system anyway and is managed by a forth parameter.

Forward pruning in our system is achieved by using a second state evaluation function that allows us to sort sibling nodes according to their heuristic value and select those evaluated as better to be expanded and searched further. We can have this done either by using the same evolved heuristic evaluation function used for evaluating board states at the bottom of the search tree, or we can use a different evolved GP tree for this task. In this work we use a different evolved evaluation function to guide search.

We performed several evolutionary runs with selective search, all using the same basic evolutionary parameters of the runs with undirected search (population size, number of generations, method of fitness evaluation etc.). Nine Men's Morris proved challenging, as branching factor varies wildly between different parts of the game. Our best result to date is from two runs, designated 000150 and 000151.

- Run 000150 used a search depth of 4 with *MaxBranching-Factor* of 5. It yielded a best player that scored 744.0 points in a 1000-game tournament against the benchmark player.
- Run 000151 used a search depth of 3 with *SelectiveSearchRatio* of 0.75. It yielded a best player that scored 813.5 points in a 1000-game tournament against the benchmark player.

Both runs evolved players that expand less moves per turn than the evolved players that run a full alpha-beta search of depth 3, and in both runs players emerged that outperform the benchmark player. We ran an analysis of these best players by having each one play 100 games against a stochastic opponent and checked how it compared with run 000146, which uses full search, in average number of states expanded per turn. The results are presented in Table 3.

Table 3: Comparison of game-state expansion between different players. Player r00146 conducts full $\alpha\beta$ search of depth 3. The other players use forward pruning to limit search.

Run identifier	Average # of States expanded per Turn	Standard Deviation
r00146	163.83	21.12
r00150	130.02	10.08
r00151	118.62	14.16

4. CONCLUSIONS

Expanding on previous work [1, 2] we presented the genetic programming approach as a tool for discovering effective strategies for playing zero-sum, deterministic, full-knowledge board games. Using our extant GP games system, we introduced several tools that allow us to apply GP to evolving game players that use selective search with forward pruning and a heuristic evaluation function, with the search method itself being an evolvable feature. We have established that, at least for the game of Nine Men's Morris, our approach yields players that achieve similar results to full-search players, with less actual search work.

5. ACKNOWLEDGMENTS

Amit Benbassat is partially supported by the Lynn and William Frankel Center for Computer Sciences. This research was supported by the Israel Science Foundation (grant no. 123/11).

6. REFERENCES

[1] A. Benbassat and M. Sipper. Evolving lose-checkers players using genetic programming. In *IEEE Conference on Computational Intelligence and Games (CIG'10)*, pages 30–37, August 2010.

[2] A. Benbassat and M. Sipper. Evolving board-game players with genetic programming. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, GECCO '11, pages 739–742, New York, NY, USA, 2011. ACM.

[3] R. Gasser. Solving Nine Men's Morris. *Computational Intelligence*, 12(1):24–41, 1996.