Spatially-Oriented Agents Improve a Multi-Agent Decision-Making Program

Susan L. Epstein
Department of Computer Science
Hunter College and
The Graduate School of
The City University of New York
New York, NY 10021
epstein@roz.hunter.cuny.edu

Jack Gelfand
Department of Psychology
Princeton University
Princeton, NJ 08544
jjg@princeton.edu

Esther Twersky-Lock
Department of Computer Science
The Graduate School of
The City University of New York
New York, NY 10021
elock@csp.gc.cuny.edu

Abstract

A multi-agent game-learning program that already epitomizes a variety of high-level reasoning is augmented with both an associative visual memory and spatially-oriented reasoning agents. From a pattern-oriented language based on small collections of markers, patterns persistently associated with wins, losses, and draws are identified and stored. A new, game-independent agent relies upon increasingly improved knowledge about associations between these patterns and the outcomes of the contests in which they are observed. Meanwhile, additional new, game-specific agents are spawned as generalizations over the learned patterns. The agents are automatically created and incorporated into the program. As they demonstrate their reliability, they acquire increasing influence over decision making. A dynamic filtering process continually refines knowledge about the patterns to ensure that, as the game-learning program becomes more expert, concept formation becomes increasingly accurate. An important contribution of this work is the successful self-expansion of a multi-agent architecture during skill acquisition. The resultant spatially-oriented system is shown to enhance performance.

Introduction

There is increasing evidence that people integrate multiple, parallel, possibly conflicting strategies to make decisions (Biswas, Goldman, Fisher, Bhuva, & Glewwe 1995; Crowley & Siegler 1993; Ratterman & Epstein 1995). Different parts of the brain, for example, are activated when decisions are being made about different strategic aspects of chess (Nichelli, et al. 1994). In humans, perception is an integral part of problem solving. While learning to play games, for example, people report noticing and applying pattern knowledge (Ratterman, et al. 1995). Even experts describe their rationale for move selection in terms of control of the center of the board or the edges (Fine 1989; Gelfer 1991; Lee & Mahajan 1990; Samuel 1963; Samuel 1967).

It is the thesis of this work that those interested in the simulation of human intelligence cannot afford to overlook perception, and, in particular, those interested in the design of adaptive game players cannot afford to overlook pattern learning. The work reported here simulates the way people learn pattern-oriented play. People integrate both pattern knowledge and spatially-oriented concepts into their decision making. Game players, for example, apply pattern knowledge that associates particular patterns with successful or unsuccessful play. In Figure 1(a), the empty locations are blanks, # denotes "don't care," and a skilled tic-tac-toe player associates the pattern with a win for X. The pattern may appear in any symmetric orientation, the # squares may be empty or contain any marker; the human expert still will associate the pattern with a win for X. Game players also use more general but equally salient spatially-oriented heuristics. An example is the generalization "reflect O's move through the center," produced from the three patterns of Figure 1(b) and proved to be optimal play for X in the game lose tic-tac-toe (Cohen 1972). In general, spatially-oriented heuristics are much faster than high-level reasoning and search because they function as compiled knowledge; they are applied without reasoning about them.

Like people, the *FORR* architecture for learning and problem solving simulates the integration of a variety of strategies and employs multiple concurrent processing streams (Epstein 1994a). Each strategy is represented as an agent that contributes advice on how to make a decision. *Hoyle* is a FORR-based program that learns to play two-person, perfect information, finite-board games. To integrate spatially-oriented reasoning with other kinds of strategies, the program has recently been enhanced with pattern learning. One new agent produces advice based upon these patterns, essentially applying pattern associations like those of Figure 1(a). Individual additional agents are learned as generalizations over patterns, like the generalization over Figure 1(b).

At an earlier workshop we reported a preliminary design for this system based on small samples of play experience (Epstein & Gelfand 1995). The contribution of this paper is a revised and much improved version that now processes all playing experience, plus evidence that this *spatially-oriented reasoning is readily integrated into the pre-*

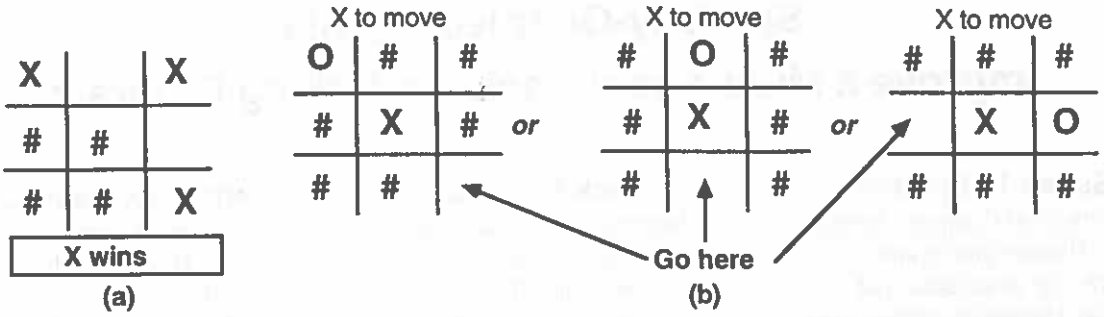

Figure 1: (a) In tic-tàc-toe, X associates this pattern with a win. # denotes "don't care." (b) "Reflect through the center," a spatially-oriented heuristic for lose tic-tac-toe.

existing system and can enhance its performance. The next section describes Hoyle's decision making process. Subsequent sections sketch how Hoyle learns patterns and applies them in decision making. We offer results on both successful integration and improved performance.

Decision Making in Hoyle

Hoyle learns to play in competition against a hand-crafted, external expert program for each specific new game. Hoyle has a set of 23 resource-limited, heuristic procedures called *Advisors*. The Advisors are advice-giving agents organized in two tiers, as shown in Figure 2. A tier-1 Advisor relies upon correct knowledge, shallow search, and simple inference. For example, the Advisor Victory advocates any move that wins the contest immediately. A tier-2 Advisor, however, has a narrow and heuristic viewpoint, such as Material's "maximize the number of your markers and minimize the number of your opponent's."

The input to every Advisor is the same: the current game state, the legal moves, and any useful knowledge (described below) already acquired about the game. The output of every Advisor is advice represented as comments. A *comment* is an ordered triple <Advisor, action, strength> where *strength* is an integer in [0, 10] measuring the intensity (distance from 5) and nature

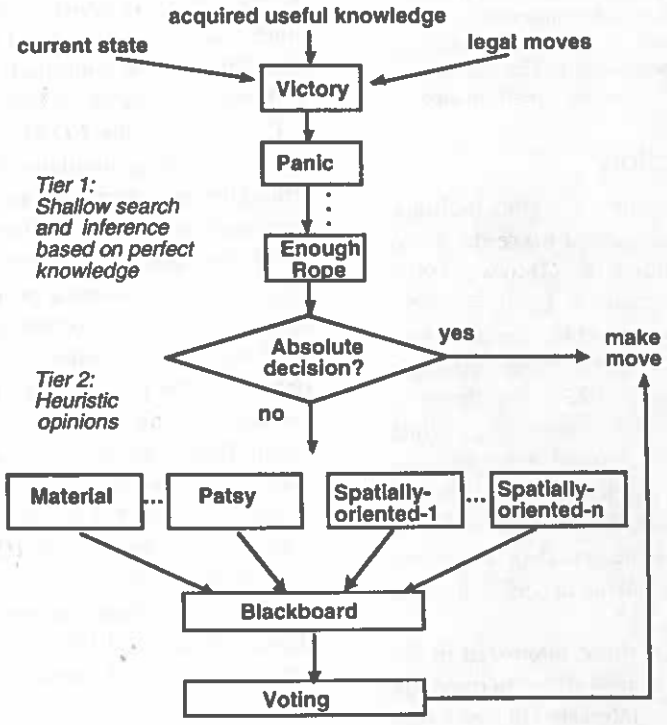

Figure 2: Hoyle's multi-agent decision-making process augmented with the new spatial Advisors described in the text.

(above 5 for support and below 5 for opposition) of the Advisor's opinion on the action. Whenever it is Hoyle's turn to move, the hierarchy is provided with the input, and tier 1 sequentially attempts to compute a decision. If any tier-1 Advisor makes a decision, it is executed as Hoyle's move. If no tier-1 Advisor makes a decision, then the second tier collectively makes its many, less reliable recommendations. After all the tier-2 Advisors have commented, the move that is forwarded to the game-playing algorithm for execution is the one that has the most support and least opposition, i.e., the one that produces

$$\max_i \left(\sum_A s(A, i) \right) \quad [1]$$

where s(A, i) denotes the strength of the comment from Advisor A on move i.

Hoyle's performance improves with experience because many of the Advisor's formulate their comments based upon the useful knowledge the program learns from play. *Useful knowledge* is expected to be relevant to future play and is probably correct in the full context of the game tree. Examples of useful knowledge include recommended openings and states from which a win is always achievable with perfect play on both sides. Each item of useful knowledge is associated with a learning algorithm that is highly selective about what it retains. The learning methods vary; they may generalize and they may discard previously acquired knowledge. Further details on Hoyle are available in (Epstein 1992).

Learning Patterns as Advice

Our system limits the cost of creating strategically meaningful, higher-order concepts by formulating them from a limited group of patterns associated with wins, losses, and draws. A *pattern*, for our purposes, is a small geometric arrangement of marker types (e.g., black or X) and *blanks* (unoccupied positions) with a particular orientation to each other, such as Figure 1(a). Patterns are normalized for symmetry, i.e., each represents all the ways it could be placed on the board under the eight classic symmetries of the plane (0-, 90-, 180- and 270-degree rotation, reflection across the horizontal and vertical axes, and reflection across both diagonals). Patterns are initially identified by templates (Epstein, et al. 1995).

Figure 3 provides an overview of the five-stage process Hoyle uses to move from visual perception to new spatial Advisors. The *associative pattern store* is a heuristically-

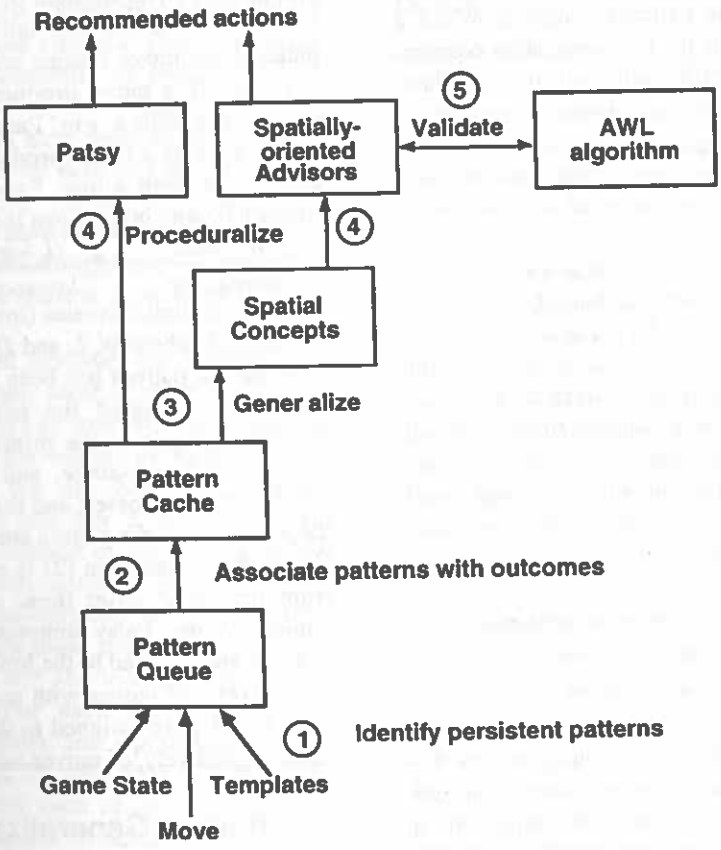

Figure 3: A schematic of Hoyle's associative pattern learning and spatially-oriented agent formation.

-organized database that includes the templates that screen for patterns, a frequency table, a pattern queue, a pattern cache, and spatial concepts for each game. After each contest is played, Hoyle examines each move, applying a temporal difference method that identifies only those patterns created or destroyed by that move. (In a game with sliding moves, patterns may be destroyed as well as created.) This ensures that patterns created early in the contest do not overly influence the decision process, and also helps constrain the potential combinatoric explosion of patterns. Each time it is encountered, a pattern is recorded in the frequency table. Once it has appeared several times (stage 1 of Figure 3), the pattern joins the *pattern queue*, along with *association counters* that tabulate the number of times the pattern has been seen in a contest where the first competitor won, lost, or drew.

When a pattern that is already in the pattern queue reappears, the appropriate association counter is incremented by 1. Frequently, however, particularly during early play when Hoyle is still a novice, there is likely to be conflicting evidence about patterns. A valuable pattern may be associated with a loss because Hoyle does not know how to take advantage of it, or an irrelevant one will appear to be associated with a win. We have therefore devised a way to temper experience with some measure of the program's ability to use patterns properly. When a pattern in the queue reappears, the two association counters that are not incremented by 1 are adjusted to reflect how certain the system is that they report relevant experience.

An *outcome sequence* $a_1, a_2, \ldots, a_n$ is a list of W's, L's, and D's (denoting wins, losses, and draws from Hoyle's perspective). Each outcome sequence of n a_i's can be represented as an integer:

$$\sum_{i=1}^{n} \frac{\alpha_i}{n - i + 1} \text{ where } \alpha_i = \begin{cases} +2 \text{ if } a_i = W \\ -2 \text{ if } a_i = L \\ +1 \text{ if } a_i = D \end{cases}$$

Raw-confidence is the integer associated with the program's actual confidence sequence. The best case (*max-confidence*) would be to have all the losses, followed by all the draws, followed by all the wins. The worst case (*min-confidence*) would be to have all the wins, followed by all the draws, followed by all the losses. After n contests, Hoyle measures its *confidence* within [0, 1] as:

$$confidence = \begin{cases} 0, \text{ if } n \leq 3 \\ 0, \text{ if } max\text{-}confidence = min\text{-}confidence \\ \dfrac{raw\text{-}confidence - min\text{-}confidence}{max\text{-}confidence - min\text{-}confidence} \end{cases}$$

When a pattern in the queue reappears, the two association counters that are not incremented are multiplied by 1 – *confidence*. For example, perhaps a pattern that has thus far been associated with a loss is suddenly associated with a draw. If the program's confidence in its ability is .99, the draw counter will be incremented by .01, but if the program's confidence is only .3, the counter will be incremented by .7.

In stage 2 of Figure 3, patterns that persist over time and are identified with a single consistent outcome are eventually included in the *pattern cache* as well, along with their association counters. We use thresholding, aging, and consistency mechanisms to filter the patterns (Epstein, et al. 1995). The *threshold* is the number of times a pattern must appear in the frequency table before it enters the pattern queue. Patterns are *aged* by multiplying all their counters by an aging parameter in (0, 1). The threshold and aging screen for persistent patterns. If a pattern is eventually consistent in its association with a single outcome, aging will reduce two of its counters to very small numbers. *Effective zero* is the value treated as equivalent to zero for entry into the pattern cache. The remaining stages of Figure 3 are discussed in the next section.

Patsy is a pre-specified tier-2 Advisor that relies on useful knowledge in the form of entries in the pattern cache. Patsy comments in favor of moves that produce new patterns associated with wins and (to a lesser extent) draws, and against those that produce new patterns associated with losses. (Subpatterns of detected patterns are ignored.) The strength of Patsy's comment on a move is a function of the associations recorded for all the new patterns the move creates and the old patterns the move destroys. If a move produces only patterns associated exclusively with a win, Patsy votes for that move with strength 10. If a move produces only patterns associated exclusively with a loss, Patsy votes for that move with strength 0. Any other move is evaluated as the sum of

$$2 * \frac{(W - min)}{wins\text{-}since} - 2 * \frac{(L - min)}{losses\text{-}since} + \frac{(D - min)}{draws\text{-}since} \qquad [2]$$

for every pattern it creates (minus this expression for those it destroys), where W, L, and D are counters for the number of times the pattern has been seen in a contest where the mover who created the pattern won, lost, or drew, respectively; *min* is the minimum of W, L, and D; and *wins-since*, *losses-since*, and *draws-since* denote the number of wins, losses, and draws for the mover who used the pattern since the pattern entered the cache, respectively. (If any denominator in [2] is zero, that term is eliminated from the sum.) After these moves are sorted by their numeric values, Patsy comments on them. Strengths of 6, 7, and 8 are assigned to the lower, middle, and upper third, respectively, of moves with positive values, and strengths of 2, 3, and 4 are assigned to the lower, middle, and upper third, respectively, of moves with negative values.

Pattern Generalizations as Agents

As Hoyle acquires pattern knowledge, sets of individually learned patterns are generalized to form spatial concepts. In

stage 3 of Figure 3, periodic sweeps through the pattern cache attempt to generalize sets of patterns into concepts like that of Figure 1(b). Each new concept is proceduralized in stage 4 of Figure 3 into an individual, game-specific, tier-2 Advisor called a *spatially-oriented Advisor*. A spatially-oriented Advisor for concept C in game G supports a move to a state where an instance of C is newly introduced and opposes a move to a state where an instance of C is eliminated.

Credit/blame assignment in a domain such as this is difficult; even human experts may disagree on the move that won or lost a contest. The significant decision may have been early in play, or may have been a set of moves rather than an individual one. Rather than credit or blame an individual move, an algorithm called *AWL* credits or blames each tier-2 Advisor when it supports or opposes expert-like behavior (Epstein 1994b). This approach holds the rationale behind actions accountable, rather than the actions themselves.

AWL is a modification of WINNOW (Littlestone 1988). Instead of tallying the tier 2 comments with [1], AWL uses

$$\max_i \left(\sum_A w_i \, s(A, i) \right) \qquad [3]$$

For each tier-2 Advisor A_i and each game, AWL computes a weight w_i to reflect the importance and reliability of A_i in that game. Previously, if two Advisors commented on the same move with the same strength, their comments would have been treated identically. If one of the two Advisors were more trustworthy in a particular game, however, its comment should have more influence there. Irrelevant and self-contradictory Advisors in a particular game should eventually have weights of 0, and more trustworthy Advisors should have higher weights than less trustworthy ones. Prior empirical experience with Hoyle's game-independent Advisors showed such weights to be game-specific.

As new, spatially-oriented Advisors are spawned and Hoyle's skill develops further, some of them may prove irrelevant, self-contradictory, or untrustworthy, despite prior empirical evidence of the validity of the patterns on which they were based. We use AWL to integrate spatially-oriented Advisors into tier 2, assigning them credit or blame when they support or oppose expert-like behavior. AWL runs after each contest Hoyle plays against an external (human or computer) expert. The algorithm considers, one at a time, only those states in which it was the expert's turn to move and Hoyle's tier 1 would not have made a decision. For each such state, AWL distinguishes among support and opposition for the expert's recorded move and for other moves. Essentially, Hoyle learns to what extent each of its Advisors, both ordinary and spatially-oriented, simulates the expertise exemplified by the expert's moves. AWL cumulatively adjusts the w_i's after each contest and uses those weights to make decisions throughout the subsequent contest. In particular, AWL gradually validates spatially-oriented Advisors (stage 5 of Figure 3) for relevance and correctness during subsequent play.

Although prespecified, game-independent Advisors and learned, game-specific Advisors work together in tier 2 to select a move, a new spatially-oriented Advisor is *probationary*. Once the new Advisor demonstrates its *relevance* (ability to comment), its comments are gradually incorporated into the decision making process. When a probationary Advisor outputs comments, its comment strength is multiplied not only by w_i as in [3], but also by a discount factor d_A that reflects its novelty.

$$\max_i \left(\sum_A d_A \, w_i \, s(A, i) \right) \qquad [4]$$

Initially, d_A is 0.1. With each decision the probationary Advisor participates in, that number increases by 0.1 until, after 10 such decisions, the Advisor is relieved of its probationary status and is treated like all the others ($d_A = 1$). As a result, the collective wisdom of tier 2 not only remains intact but actually improves as the spatially-oriented agents are smoothly integrated into the pre-existing architecture. Further details on the formation of spatial concepts and spatially-oriented decision-making strategies in Hoyle are reported in (Epstein, Gelfand, & Lesniak 1996).

Results

In the experiments described here, Hoyle alternately moved first in one contest and second in the next. Such a run continued until Hoyle was said to have learned to play a game because it could draw *n* consecutive contests. Once the program met this *behavioral standard,* the game was considered *learned*, learning was turned off, and Hoyle was tested against four challengers that simulated perfect, expert (10% random move selection, 90% perfect), novice (70% random move selection, 30% perfect), and random contestants. During testing, *reliability* measures the consistency with which the program can continue to win or draw against contestants of varying strengths, and *power* measures the ability of the program to defeat those contestants (Epstein 1994c). Because Hoyle is non-deterministic, all results are averaged over five runs.

We have successfully implemented spatially-oriented decision making with the integration of spatially-oriented Advisors into Hoyle's tier 2. We tested the process on the games tic-tac-toe and lose tic-tac-toe, played exactly like tic-tac-toe except that the first person to get three of the same playing piece along any row, column, or diagonal loses. Tic-tac-toe is an easy game for Hoyle to learn; there is only one kind of contest it must lose before it plays perfectly, so that the speed with which it learns merely reflects how soon the non-deterministic external expert

Table 1: Learning time in number of tasks and performance during testing for three versions of Hoyle, with a behavioral standard of $n = 20$.

		Performance							
	Learning	Perfect challenger		Expert challenger		Novice challenger		Random challenger	
Version	time	Reliability	Power	Reliability	Power	Reliability	Power	Reliability	Power
Without patterns	170.0	84.0	0.0	82.0	14.0	74.0	52.0	66.0	40.0
With Patsy	62.0	96.0	0.0	90.0	10.0	80.0	52.0	74.0	50.0
With Patsy and learned Advisors	126.0	100.0	0.0	96.0	20.0	78.0	52.0	90.0	58.0

program presents it (Epstein 1995). Thus our results on tic-tac-toe merely reflect that different patterns are learned for the two games, and that pattern learning and the acquisition of new Advisors does not degrade what was already perfect performance.

Lose tic-tac-toe, however, is much more difficult for Hoyle. For lose tic-tac-toe we ran three experiments with a behavioral standard of $n = 20$. The results are summarized in Table 1. Without any pattern learning, Hoyle drew 20 contests in a row after playing 170 times. When it learned patterns and applied them in Patsy but did not learn spatially-oriented Advisors, Hoyle learned in 62 contests. With pattern learning, Patsy, and spatially-oriented Advisors, Hoyle learned in 126 contests. Clearly, learning individual patterns accelerates Hoyle's acquisition of expertise. Although acquiring new spatial concepts as Advisors slows down learning, it is still faster than Hoyle without pattern learning, and the remainder of Table 1 indicates that the extra learning time is well worth while. Hoyle's reliability and power against the various challengers generally improve with Patsy, and improve even more with spatially-oriented Advisors. The substantial increase in reliability against a random challenger is particularly noteworthy. The random challenger is likely to expose Hoyle to portions of the game tree it has never encountered before, and therefore provides a rigorous test of overall expertise.

AWL was used in all the experiments reported here, and works well with the new spatially-oriented Advisors. In a fourth experiment, we let Hoyle learn in 250 contests (without a behavioral standard) to measure the impact of extended playing experience. By inspection, those spatially-oriented Advisors considered relevant and significant by human game playing experts are precisely those whose weights rise over the course of Hoyle's learning. By way of example, we show in Figure 4 three spatial concepts typically learned for lose tic-tac-toe, and graph the evolution of their weights under AWL during a single, 250-contest run in Figure 5. When an Advisor is described with α and β, the symbols are interpreted as "either α = X and β = O or α = O and β = X." Advisor 1 advocates playing in an *outside* (not through the center) row where each contestant already has one marker. These are safe, not particularly aggressive moves which AWL accepts in moderation, stabilizing the Advisor's weight around 1.3 about 60 contests after Advisor 1 is identified. Advisor 2 recommends that X play a corner in an outside row where O already holds a corner. Such moves do not always constitute good advice; Advisor 2's weight plummets in Figure 5 as its advice fails time and again to correlate with the expert's move choice. Advisor 3 represents the horizontal and (through symmetry) vertical portion of the strategy "reflect the other contestant's move through the center." This has been proved optimal for X in lose tic-tac-toe (Cohen 1972), is often a good move for O, and is weighted highly by AWL. The growth in the weights of the significant spatially-oriented Advisors, along with the corresponding decline in the weights of the

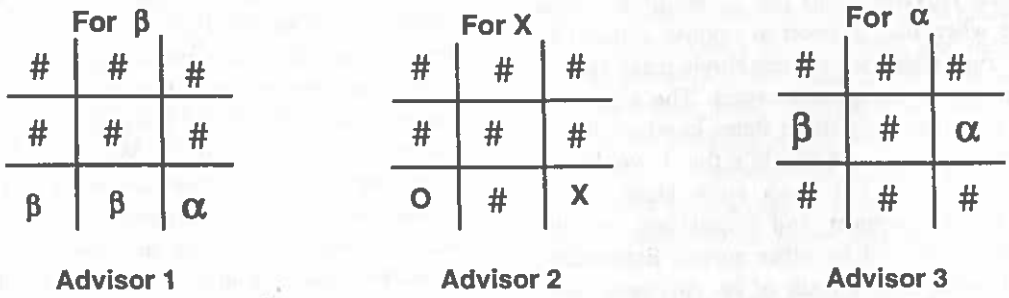

Figure 4: Some patterned-based Advisors learned for lose tic-tac-toe. The mover for each Advisor is in the current state; the pattern is matched for in the subsequent state.

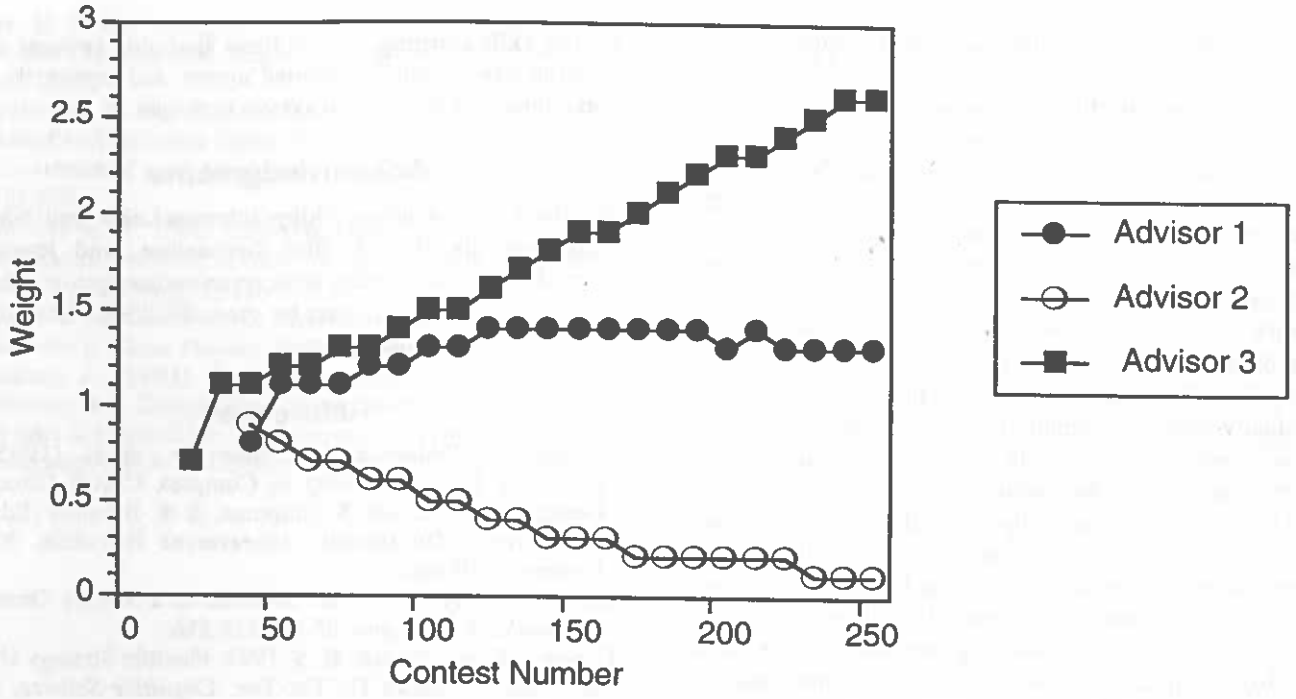

Figure 5: The weights attributed to three learned, spatially-oriented Advisors during 250 consecutive learning contests.

irrelevant and incorrect ones, proves AWL to be an able filter for the newly-learned agents.

Our implementation successfully avoids the potential combinatoric explosion in detected patterns. Even in so small a board as the one for these games, there are 3^9 possible patterns, but to date we have seen no more than 35 patterns in the cache, and even fewer are emphasized as the conceptual grounds for a spatially-oriented Advisor. Despite the potentially large number of lose tic-tac-toe patterns, for example, on a typical run after learning there were 39 patterns in the pattern queue, 34 patterns in the cache, and 16 relevant (validated and actively commenting) spatially-oriented Advisors.

Memory of patterns is refined by the threshold for a pattern to enter the queue (set to 3 in these experiments), aging in both the pattern queue and the pattern cache (set to .9 and .99, respectively), and the management of both consistent entries (with effective zero .01) and inconsistent entries with the confidence value. Although we did not perform a quantitative study of this memory refinement process, we did find that without it performance degraded. These processes are ongoing and constantly refine the system's perspective on patterns.

Discussion

We do not advocated reliance on pattern learning alone. That would ignore the other, higher-level processes quite evident in human decision making. Indeed, Hoyle learns much useful knowledge that is not pattern-oriented and has been detailed elsewhere (Epstein 1992).

The implementation of pattern learning and its ability to spawn new tier-2 Advisors were inspired by repeated laboratory experiences with people, in the context of many different games. College students spoke about, reacted to, and relied upon familiar, sometimes symmetrically transposed, patterns while learning to play well. They spoke of "the death triangle" or "the black L," for example. Later, they relied heavily (sometimes too much so) on these patterns as a kind of compiled expertise (Ratterman, et al. 1995).

Other researchers have created programs that capitalize upon learned patterns (George & Schaeffer 1991; Levinson, Beach, Snyder, Dayan, & Sohn 1992; Painter 1993). Typically, however, a great many patterns are acquired, so that matching against them becomes non-trivial (Fawcett & Utgoff 1991; Yee, Saxena, Utgoff, & Barto 1990). In one run, for example, Fawcett and Utgoff's T2 learned 45 tic-tac-toe concepts as predicate calculus expressions with 52 clauses after 800 contests. This is a great deal of experience and a great many patterns with a heavy matching cost for so simple a game. The problem, we believe, is that predicate calculus lacks incisiveness, and that such programs are forced to represent many inconsistencies that may have no consequence in actual strategic play. The process we describe, in contrast, identifies inconsistent experience early on, and filters down

to patterns that are both reliable and occur frequently.

Each of the spatially-oriented Advisors Hoyle now learns is a generalization over a class of states that occurs with some frequency, is consistently associated with a single outcome, and contains a simple configuration of spatially-related markers. The high-level reasoning of the prespecified Advisors tempers possible premature guidance from the spatially-oriented components. When we force patterns to prove their reliability and importance before they can enter the cache, we reduce the combinatorics that would otherwise confront the generalizer. More experienced, spatially-oriented Advisors gradually emerge to emphasize broader generalities than those expressed in a simple pattern, and must advocate expert play to retain their status.

Although the new spatially-oriented Advisors comment in parallel with the other tier-2 Advisors, the former comment much more quickly than the latter. Eventually we would like to see these spatially-oriented Advisors supplant the others. This would greatly increase the speed of Hoyle's decision making, and model the shift to automaticity regularly observed in human players.

Future work includes more difficult, morris and Go-like games, as well as the incorporation of spatial biases like center, edge, perimeter, bounded region, length, and area. We intend to experiment with spatially-oriented Advisors spawned from other spatially-oriented Advisors and with new generalization methods. The process employed here could be extended to broader temporal reasoning. It should be possible to remember important patterns of activity during decision making and generalize these relations into temporal concepts that are readily integrated into a FORR-based system.

Conclusions

Our long-range objective in this work is to create a heuristic decision maker that learns rapidly enough to simulate intelligent behavior while it is learning. Although restricting potential patterns by applying templates could overlook some information, we found that the concepts formed were those deemed important in published analyses of the games that we tested. The pre-existing, more general Advisors prevent egregious errors, particularly during early experience. Our results show that pattern learning can enhance the level of play of a high-level reasoner and support its transition towards expertise. Simple pattern learning within Hoyle now improves the quality of the decisions made during learning, and increases the speed of the program's transition toward expertise. The validation process for newly created agents performs properly, and the system works smoothly as knowledge is refined during learning. The transitions in the way Hoyle treats patterns model some of the automaticity shifts detected in humans during skill learning. We believe that this process of creating new, spatially-oriented agents and testing their correctness in a multi-agent system is unique.

Acknowledgments

We thank Ron Kinchla, Philip Johnson-Laird, and Nick Littlestone for their helpful discussions, and Joanna Lesniak for her pioneering work on an earlier version. This work was supported in part by grant #9423085 from the National Science Foundation.

References

Biswas, G., Goldman, S., Fisher, D., et al. (1995). Assessing Design Activity in Complex CMOS Circuit Design. In P. Nichols, S. Chipman, & R. Brennan (Ed.), *Cognitively Diagnostic Assessment* Hillsdale, NJ: Lawrence Erlbaum.

Cohen, D. I. A. 1972. The Solution of a Simple Game. *Mathematics Magazine*, 45 (4): 213-216.

Crowley, K. and Siegler, R. S. 1993. Flexible Strategy Use in Young Children's Tic-Tac-Toe. *Cognitive Science*, 17 (4): 531-561.

Epstein, S. L. 1992. Prior Knowledge Strengthens Learning to Control Search in Weak Theory Domains. *International Journal of Intelligent Systems*, 7: 547-586.

Epstein, S. L. 1994a. For the Right Reasons: The FORR Architecture for Learning in a Skill Domain. *Cognitive Science*, 18 (3): 479-511.

Epstein, S. L. 1994b. Identifying the Right Reasons: Learning to Filter Decision Makers. In *Proceedings of the AAAI 1994 Fall Symposium on Relevance*, 68-71. New Orleans: AAAI.

Epstein, S. L. 1994c. Toward an Ideal Trainer. *Machine Learning*, 15 (3): 251-277.

Epstein, S. L. 1995. Learning in the Right Places. *Journal of the Learning Sciences*, 4 (3): 281-319.

Epstein, S. L. and Gelfand, J. 1995. Learning Spatial Concepts through Experience. In *Proceedings of the IJCAI Workshop on Spatial and Temporal Reasoning*, 47-56. Montreal:

Epstein, S. L., Gelfand, J. and Lesniak, J. 1996. The Integration of Pattern-Based Learning and Spatially-Oriented Reasoning with a Multi-Agent, Decision-Making Expert. *Computational Intelligence*, 12 (1): 199-221.

Fawcett, T. E. and Utgoff, P. E. 1991. A Hybrid Method for Feature Generation. In *Proceedings of the Eighth International Workshop on Machine Learning*, 137-141. Evanston: Morgan Kaufmann, San Mateo.

Fine, R. 1989. *The Ideas behind the Chess Openings*. New York: Random House.

Gelfer, I. 1991. *Positional Chess Handbook*. New York: Macmillan.

George and Schaeffer. (1991). Chunking for Experience. In D. F. Beal (Ed.), *Advances in Computer Chess VI* (pp. 133-147). London: Ellis Horwood.

Lee, K. F. and Mahajan, S. 1990. The Development of a World Class Othello Program. *Artificial Intelligence*, 43 (1): 21-36.

Levinson, R. A., Beach, B., Snyder, R., et al. 1992. Adaptive-Predictive Game-Playing Programs. *Journal of Experimental and Theoretical Artificial Intelligence*, 4: 315-337.

Littlestone, N. 1988. Learning Quickly when Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning*, 2: 285-318.

Nichelli, P., Grafman, J., Pietrini, P., et al. 1994. Brain Activity in Chess Playing. *Nature*, 369: 191.

Painter, J. (1993). Pattern Recognition for Decision Making in a Competitive Environment. Master's thesis, Hunter College of the City University of New York.

Ratterman, M. J. and Epstein, S. L. 1995. Skilled like a Person: A Comparison of Human and Computer Game Playing. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, 709-714. Pittsburgh: Lawrence Erlbaum Associates.

Samuel, A. L. (1963). Some Studies in Machine Learning Using the Game of Checkers. In E. A. Feigenbaum, & J. Feldman (Ed.), *Computers and Thought* (pp. 71-105). New York: McGraw-Hill.

Samuel, A. L. 1967. Some Studies in Machine Learning Using the Game of Checkers. II - Recent Progress. *IBM Journal of Research and Development*, 11 (6): 601-617.

Yee, R. C., Saxena, S., Utgoff, P. E., et al. 1990. Explaining Temporal Differences to Create Useful Concepts for Evaluating States. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 882-888. Boston, MA: AAAI Press, Palo Alto, CA.