This paper appeared in the *Proceedings of the AAAI Fall Symposium Games: planning and Learning*, 1993. Menlo Park , CA: AAAI.

Toward a Theory for Well-Guided Search

Susan L. Epstein
Hunter College and The Graduate School of the City University of New York
695 Park Avenue
New York, NY 10021 USA
sehhc@cunyvm.cuny.edu

Abstract

A game learner's experience is no more than the nodes it encounters in the search space. For a challenging game, only a small fraction of that space can ever be visited. People's ability to learn to play difficult games well is a clear indication that not all nodes are equally relevant to learning. There are, it is argued here, *key nodes* particularly important to the development of expertise, and those key nodes appear in clusters in the game tree. A game learning program might arrive at a key node by chance, be drawn there by a choice it makes, or be driven there by the moves of its opposition. Trainer guidance has some similarity to varying the sequence of training examples in ordinary induction, but here is delegated to the program itself. This paper offers empirical evidence of substantial improvement in the quality of play when a program is steered to clusters of key nodes, and considers several ways to do so.

The thesis of this work is that people are able to learn to play difficult games[1] expertly because the knowledge they require is not evenly distributed across the game tree, but located at *key nodes*. The quality of learned performance depends in part on the learner's familiarity with these key nodes. *Well-guided search* takes the learner to the key nodes, the places where the most valuable learning can occur.

This paper identifies and offers empirical evidence for some basic principles of well-guided search. The first section explains the role of key nodes in experience. The second provides background on a game-learning program whose empirical data instigated the theory described here. The next two sections document how the learner arrives at key nodes through both external direction (training) and internal direction (choice). Subsequent sections discuss the results, their impact on discovery learning, and future work.

1. Key Nodes and Experience

Even a simple game offers evidence of how limited an expert player's experience is. Consider tic-tac-toe, for example, a game that many nine-year-old children play perfectly, but one whose statistics are initially daunting. It has a branch factor of 4.5 and a contest averages nine moves. Theoretically, there could be as many as $9^{4.5}$ = 19,683 states in the game graph, but the search space is far smaller than it first appears. A *reachable state* is a game board, along with a designated mover, that can actually occur during competition. There are only 5478 distinct, reachable states in tic-tac-toe. These states can be normalized for symmetry, so that, for example, all four of the corner openings are considered as a single state. After normalization there remain only 170 distinct paths through the game graph, beginning with the initial empty board, that must be explored before one has enough information to play perfectly.[2] The average person would have difficulty memorizing either of these reduced spaces, and few have exhaustively visited all their nodes. Yet people, with a few memorized openings and a reliable winning endgame detector, often play expert, even perfect, tic-tac-toe.

Data indicate striking similarities between the way ordinary folk cope with the somewhat oversized tic-tac-toe search space and the way grandmasters at chess cope with an enormous one. The literature on expert chess players confirms that they neither search nor remember very much of such a space. With a branch factor of 20 and contests that average 80 moves, the chess game graph could have as many as $20^{80} \approx 10^{120}$ nodes. Although there may be as many as 40,000 distinct chess openings, most tournament players encounter no more than 400 of them in the course of a year of play, and prepare only three or four for any given tournament (Charness, 1991). If one restricts the graph to these 40,000 openings, assumes a memorized endgame, and restricts moves to those that masters might select, there would still be 4×10^{20} "sensible" contests (paths through the space) with exhaustive search (Charness, 1983). An expert chess

[1] For clarity, we distinguish carefully here between a *game* (a board, playing pieces, and a set of rules determining play) and a *contest* (a complete experience at a game, from some initial state specified by the rules to some final state where the rules designate a winner or declare a draw).

[2] This computation is based upon data from (Berlekamp, Conway, & Guy, 1982).

player is estimated to see 1,200,000 nodes per year (Charness, 1991). During 20 years of learning, that amounts to only about 6×10^{-14} of the "sensible space" and 2.4×10^{-113} of the entire space.

That the tiny, experienced fraction of the search space so well supports learning to play, suggests that some of these nodes are particularly significant, and so we call them *key nodes*. Since nodes encountered during contests delineate a path in the game tree, and since human study often focuses on variations that explore the neighborhood of those paths, we hypothesize that key nodes are not scattered randomly about the space, but appear in isolated clusters.

If we presume that expertise develops when key nodes are visited, then a game-learning program must somehow be exposed to them. Key nodes could be encountered by chance, they could be presented by a teacher as a sequence of training examples with explanation or solution, they could be encountered during competition in a carefully structured environment, or the learner itself could seek them out. The first of these, random encounters, is unreasonable given the apparent sparsity of key nodes in a game tree. The second, a series of "puzzle problems" for instruction, would require that the teacher have a clear test for key nodes and a road map to reach them. Unfortunately, neither is available for difficult games; the only evidence that one has visited key nodes is in the improved quality of one's play. After some background, we explore the other two possibilities, competition and internal direction .

2. Background

Let a *perfect player* for a game be one that always makes the best possible move from every node in the game graph. A *draw game* is one in which every contest between two perfect players must, by the nature of the game graph, end in a draw. (Tic-tac-toe is an example of a draw game.) A contestant who loses a contest at a draw game has made an error; one who wins or draws a contest at a draw game is said to have *achieved an expert outcome*. A *tournament* is a sequence of contests between two participants in which they alternate moving first.

The experiments in the next two sections were run with Hoyle, a program that learns to play two-person, perfect information, finite board games (Epstein, 1992). Hoyle relies upon its *Advisors*, procedures that recommend and advise against the current choices available to the mover. Each Advisor takes a particular, game-independent perspective and searches no more than two-ply forward from the current node. When it is Hoyle's turn to move, the program consults its Advisors. The input to every Advisor is the same: the current game state, the current legal moves, and the current useful knowledge (see below) accrued for the game from experience. An Advisor's output is one or more *comments*, ordered triples naming the Advisor, a legal move, and an integer that

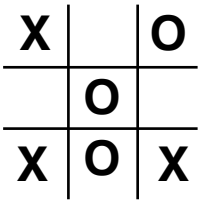

Figure 1: A significant win state from tic-tac-toe, with X to move. The contest is not over, but with perfect play X will win.

indicates an opinion somewhere between strong aversion (0) and enthusiastic support (10). Hoyle's 7 primary Advisors have a priority ranking; if they do not designate a move, the remaining 16 all comment. The move with the most support is chosen, and ties are broken by random selection. Hoyle begins with only the rules of a game and is expected to learn to play it expertly during competition against an external program, its *trainer*. For Hoyle, internal direction to key nodes comes from its Advisors, and external direction from its trainer.

Useful knowledge in Hoyle is game-dependent data computed and stored in game-independent slots. It is possibly relevant and probably correct. There are, for example, useful knowledge slots to record average contest length, applicable two-dimensional symmetries, good openings, relevant forks, important contest histories, whether going first or second is an advantage, and significant states. A *significant win state* is one that must result in a win for the mover when she plays perfectly, no matter how the other contestant plays. An example of such a state appears in Figure 1. A *significant loss state* is one that must result in a loss for the mover when the other contestant plays perfectly, no matter how the mover plays. An example of a significant loss state appears in Figure 2. By definition, all the children of a significant loss state are themselves significant win states for the other contestant, and at least one of the children of a significant win state is a significant loss state for other contestant.

Hoyle has one or more heuristic, game-independent learning procedures associated with each useful knowledge slot. These learning strategies vary from one procedure to the next, and include explanation-based

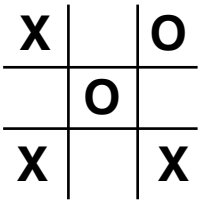

Figure 2: A significant loss state from tic-tac-toe, with O

to move. O cannot prevent a perfect-playing X from a win.

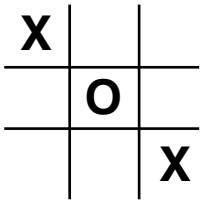

Figure 3: A key node for Hoyle from tic-tac-toe, with X to move. This is not a significant state.

learning, induction, and deduction. Learning occurs after contests and after tournaments.

The relationship between key nodes and significant states is non-trivial. Consider, for example, a state S with a dozen children. After playing one or more contests "through" each of S's children, Hoyle might deduce first that each of S's children is a significant loss state for one contestant, and then deduce that S is therefore a significant win state for the other contestant. This does not mean that S and all its children are key nodes; perhaps visiting S alone would suffice for a particular learning method. Thus a significant state is not necessarily a key node.

It may also happen that important learning occurs at a state that is not itself a significant state. An example from tic-tac-toe appears in Figure 3, where O is the mover. Although prospects look grim, O can always manage a draw from Figure 3 by avoiding the corners. (The significant states in Figures 1 and 2 make clear what happens after Figure 3 with corner play.) For Hoyle, Figure 3 is a key node; the program does not "know" the correct move there and will always make the wrong choice the first time it encounters the state. If Hoyle trains without encountering Figure 3, it will be vulnerable there. Once Hoyle experiences Figure 3 (or its symmetric equivalent) and loses, however, it will move correctly there in subsequent contests. Clearly significant states and key nodes are in some sense intertwined, and lie in each others vicinity, but they are by no means equivalent.

3. Visiting Key Nodes via Competition

Without an explicit road map to key nodes, the most obvious way to expose a program to them is to have the opposition play so that the program must confront them. The data reported here is taken from extensive experiments with a variety of trainers in which Hoyle was required to learn to play three quite different draw games with small search spaces. Full details on that work are available in (Epstein, To appear); here we cite data to demonstrate how difficult key nodes are to reach. Each experiment pitted Hoyle in a *learning tournament* against a trainer until the program had learned to play well enough to draw 10 consecutive contests. Then learning was turned off and Hoyle played a twenty-contest *testing tournament* against each of four *challengers*: a perfect player and play that simulated a (slightly imperfect) expert challenger, a novice, and a random player. Although the results were consistent for all three games, space limitations restrict discussion here to lose tic-tac-toe, the most difficult of the three for the program to learn. (Like tic-tac-toe, lose tic-tac-toe is played on a 3×3 grid between X and O. Whoever achieves three of the same marker in a row, however, vertically, horizontally, or diagonally, however, *loses*.)

Table 1 highlights representative data; each value represents an average over 5 runs. *Power* is the ability to exploit the other contestant's errors; it is measured here for a draw game by the percentage of contests won, and is reported against the expert, novice, and random challengers, in that order. (In a draw game, power against a perfect player must be zero.) For lose tic-tac-toe, maximal power was computed in a 10,000-contest tournament between the perfect player and each of the other challengers. It is 16% against the expert, 66% against the novice, and 74% against the random challenger. *Reliability* is the consistent achievement of an expert outcome; it is measured here for a draw game by the percentage of contests won or drawn, and is reported first against a perfect player, and then against the other challengers in the same order. In a draw game, ideal reliability is 100%. *Space* is long-term memory allocation for useful knowledge.

A perfect player as trainer provided no experience with key nodes beyond those that two perfect players would proffer to each other. The first line of Table 1 demonstrates that there are key nodes beyond those regularly visited during competition against a perfect player. Even though Hoyle learned to play perfectly reliably (100 as the first entry in column 3) against this trainer, when confronted during testing with the imperfect moves of the other challengers it has difficulty with their errors. Although an expert should play better against a weaker contestant, Hoyle is *less* reliable(88, 82, and 81% in column 3) against the other challengers than it is against the perfect player. Instruction with a perfect player shows the learner ideal play, but only in an extremely narrow context, as evidenced by the relatively small long-term memory (in column 4) allocated for useful knowledge. For example, a key node like Figure 3 that is a once-only (and deliberately avoided thereafter) experience may never arise during competition against a perfect player.

Noise in the trainer does not lead the learner to more key nodes, i.e., they are not randomly distributed. A series of *fallible trainers* was tested, each a perfect player with a stochastic opportunity to err. The tested random move selection rates were multiples of 10% from 10% to 100%, inclusive. The next ten lines of Table 1 show

representative data for the fallible trainers. A fallible trainer increased the demand on long-term memory and

Instruction with	Power	Reliability	Space
Perfect player	27 - 65 - **73**	**100 - 88 - 82 - 81**	89.2
10% fallible	16 - 70 - 61	57 - 91 - 84 - 78	135.6
20% fallible	15 - 54 - 68	60 - 96 - 86 - 79	224.2
30% fallible	31 - 59 - 68	57 - 78 - 78 - 82	137.6
40% fallible	24 - 65 - 78	58 - 89 - 81 - 86	325.6
50% fallible	28 - 61 - 74	49 - 71 - 84 - 82	306.8
60% fallible	35 - 61 - 78	52 - 72 - 81 - 84	273.8
70% fallible	29 - 69 - 77	41 - 63 - 83 - 82	301.6
80% fallible	41 - 72 - 71	45 - 68 - 80 - 85	154.8
90% fallible	32 - 60 - 76	42 - 72 - 78 - 88	209.2
100% fallible	41 - 64 - 75	57 - 66 - 80 - 87	185.4
Self-training	14 - 66 - 69	39 - 54 - 77 - 78	186.2
Lesson and practice	18 - 63 - **85**	**100 - 98 - 97 - 100**	299.0

Table 1: A comparison of the power and reliability achieved after different instruction in lose tic-tac-toe.

directed the learner to more nodes than the perfect player did. If those additional nodes were equally likely to be key nodes, then the program should have learned to play better with more fallible instruction. In fact, it generally played worse, further evidence that key nodes are organized some way within the search space, a way that a perfect player somehow reflects.

Left on its own, a program is unlikely to encounter key nodes. When the trainer is eliminated, its lack of guidance to key nodes proves costly. In *self-training* Hoyle is expected to learn while playing against itself. Although self-training is often presumed to be a natural environment in which to improve expertise gradually, the data in the penultimate line of Table 1 indicate otherwise. With self-training, Hoyle was the least reliable against every challenger and the least powerful against the perfect challenger. Although Hoyle had achieved 10 consecutive draws during self-training, the nodes it chose to visit alone did not include enough of the knowledge learnable at key nodes that it would later require in its encounter against the challengers. Clearly, one of the functions of a trainer is to guide the learner to key nodes.

A new training paradigm, called *lesson and practice training,* provides good guidance to key nodes. Lesson and practice training advocates interleaving relatively few learning contests against the best available expert (the *lesson*) with relatively many self-training contests (the *practice*). The application cited here gave Hoyle experience in cycles of 2 lessons followed by 7 practice contests. The last line in Table 1 is from learning tournaments where termination was after 20 consecutive wins or draws, instead of the 10 used in all the other instruction. This gave Hoyle almost as much experience against the perfect player (an average of 28 contests rather than 35.6) as the program had when the perfect player was its trainer. (Extension to 20 for self-training and for instruction with a perfect player showed no appreciable improvement. Extension to 20 for fallible trainers proved more helpful, but only against a more fallible opponent.) Lesson and practice training proved more reliable and powerful against all the challengers than most other kinds of instruction; it was never less reliable or less powerful at the 95% confidence level.

Our explanation for this improved performance is that Hoyle was able to derive useful knowledge because it was forced by lesson and practice training to visit more key nodes. During each practice contest against itself, Hoyle makes non-random choices, driven both by the general game-playing knowledge encoded in its Advisors and by the useful knowledge it has thus far acquired in competition against its trainer. These knowledge-based decisions effectively search in the immediate vicinity of nodes Hoyle has already seen during lessons with the trainer. The program's improved performance argues for the clustering of key nodes.

4. Visiting Key Nodes via Internal Direction

Hoyle is currently learning nine men's morris, an ancient African game played with 18 pieces on a 24-position board

.

Most people find the search space of approximately 143 billion nodes quite challenging. Hoyle's contests average 60 moves in two stages: a placing stage with an average of 15.5 legal moves for every turn, and a sliding stage

with an average of 7.5 legal moves. Nine men's morris is *unsolved* (the backed up value of the root in the game graph is unknown), but generally believed to be a draw game (Gasser, 1991).

Originally, Hoyle was not a very good nine men's morris player; it lost every contest against its trainer, a hand-crafted, very strong, expert program. Recently, however, two new Advisors were added that capitalize on the visual cues provided by predrawn lines on the game board (Epstein, Gelfand, Lesniak, & Abadie, 1993). With these Advisors, the program initially had some ability to draw, and then began to learn to win. Table 2 shows the outcome of a 50-contest tournament with both the new Advisors in place. During the 50 contests in Table 2, Hoyle lost 24 times, drew 17 times, and won nine times. (If nine men's morris is indeed a draw game, then the wins mean that the hand-crafted program made errors. We have located and corrected a few, but have begun to suspect, based on more recent experiments, that nine men's morris may not be a draw game after all.) The first win was not until the 27th contest, and five of the wins were in the last eight contests, suggesting that Hoyle was learning to play better as it acquired more experience. (The likelihood that when 10 wins are distributed at random among 50 contests, none would appear before the 27th contest is .02%, and that five would appear in the last eight is 0.4%.) Comparison against what we believe to be a perfect player for the placing stage indicates that the program made no identifiable errors in the placing stage of any of the last 10 contests.

#	Outcome	#	Outcome	#	Outcome	#	Outcome	#	Outcome
1	draw	11	loss	21	loss	31	loss	41	draw
2	loss	12	loss	22	draw	32	loss	42	draw
3	draw	13	loss	23	loss	33	**win**	43	**win**
4	loss	14	draw	24	draw	34	draw	44	draw
5	loss	15	loss	25	loss	35	**win**	45	**win**
6	draw	16	draw	26	loss	36	loss	46	**win**
7	loss	17	draw	27	**win**	37	loss	47	**win**
8	draw	18	draw	28	loss	38	loss	48	**win**
9	loss	19	loss	29	loss	39	**win**	49	loss
10	draw	20	loss	30	draw	40	loss	50	draw

Table 2: The outcome of a 50-contest tournament at nine men's morris between Hoyle and a hand-crafted expert program. "**win**" indicates that Hoyle defeated the trainer; "loss" indicates that Hoyle lost to the trainer.

There are several possible explanations for this playing improvement:

• **New application of useful knowledge:** One possibility is that the new Advisors were associated with a new category of useful knowledge. These two Advisors, however, do not reference any useful knowledge at all, and so could not benefit from learning.

• **Unusually accurate new Advisors:** Another possibility is that the new Advisors were so clever that they quickly dominated the decision making process. This is not the case either; Hoyle gathers statistics on the performance of its Advisors and, although the new ones were active, they by no means overwhelmed the others. Although there were now some early draws, the ability to win was not immediate, as it should have been if these Advisors "knew" the right moves from the start; the program's improvement was clearly gradual.

• **Direction without learning:** Yet another possibility is that learning was already drawing the program to places where all Hoyle needed was some good advice from these new Advisors to play well. If that were the case, then the useful knowledge store should be roughly the same size without the new Advisors as with them. This is not the case either. The useful knowledge store increased with the new Advisors.

• **Visiting the key nodes:** The final possible explanation, and the one advocated here, is that *the program performed better because of the new nodes it chose to visit during learning*. The new Advisors drew the program to places in the search space where the learning strategies presumably extracted different, more powerful, useful knowledge upon which the other Advisors were able to capitalize. Hoyle definitely plays smarter with these new Advisors, and smart play directs it to the key nodes.

5. Discussion

There are clear lessons here for the development of game-playing expert programs. The typical, competitively successful game-playing program has incorporated only some of the work from cognitive science: an extensive opening book and an error-free closing book. It is in the middlegame where such a program makes most of its mistakes, when it selects a move with a heuristic approximation of exhaustive search. These decisions are dependent on the accuracy of a feature-based evaluation

function and, to some extent, on the search depth. This paper shows that, unless those features direct the program's attention to key nodes, and unless the program can learn there, middlegame play will be both inefficient and inaccurate.

The supervision of instruction is non-trivial. The role of a trainer is to provide guidance in an otherwise intractable search space, and a trainer can fail to direct the learner to key nodes for many reasons. A lack of breadth, like instruction with a perfect player, may overlook nodes that the nascent expert should visit for robustness. A lack of reliability, like instruction with fallible players, may fragment experience and distract the learner from causal associations. A lack of expertise coupled with a lack of inherent variation, like self-training, may also miss crucial nodes. High quality instruction requires variation that addresses the vicinity of the key nodes; that is what lesson-and-practice training appears to do. The lessons direct the learner to the nodes an expert would be likely to visit, while the practice permits the learner to explore variations on that guidance. One interesting option would be to have a game-learning program deliberately explore variations, i.e., sets of short paths in the vicinity of nodes encountered during training.

For most challenging games one must presume an imperfect trainer. If a program takes an imperfect trainer as a model of expert behavior, however, it will eventually learn something that is incorrect. One reason lesson and practice training succeeds is probably that Hoyle distinguishes carefully between its own moves and those of its trainer. A trainer's move is more highly regarded, and trainer error is less likely to be suspected. Thus nodes are in some sense labeled by the reason that they were visited, and treated accordingly.

A sudden, rather than a gradual, improvement in performance during learning may be explainable as the crystallization of developing internal direction, as if the learner has finalized some good decision-making technique. The described impact of the two new Advisors in Hoyle simulates this. Another example is TD-gammon, a neural net program that learns to play backgammon as well as the best human experts (Tesauro, 1992). Work on TD-gammon supports the principles of well-guided search. The program learned in about a month, playing 200,000 contests against itself. Contests averaged 60 moves, so that the program encountered no more than $2 \cdot 10^8$ nodes, a tiny fraction of its search space. During the first week, however, the program played "long, looping contests that seemed to go nowhere" (Tesauro, 1991). Then the program improved rapidly, suggesting that first the network learned to find key nodes, and *then* to make good decisions. A less-accomplished precursor, Neurogammon, had a trainer, but repeatedly experienced the same 400 contests, without any ability to encounter additional key nodes. When TD-gammon learned against Neurogammon instead of against itself, it was able to progress away from random moves much more quickly (Tesauro, 1991).

A theory of well-guided search is applicable to domains beyond game playing. Recent work in cognitive science indicates that studies of chess experts are representative of expertise in other fields as well. Regardless of the domain, experts are distinguished from novices in several ways: they rely heavily on mental models and special-purpose knowledge representations, they have larger long-term memories and expert procedural techniques, and they have extensive, readily accessible knowledge (Chase, & Simon, 1973; Ericsson, & Smith, 1991; Ericsson, & Staszewski, 1989). Their search, however, is distinguished by its limited focus, not its breadth or speed (Charness, 1991).

In particular, the work described here highlights the dangers inherent in learning with a reactive program. A reactive program is in some sense the victim of its environment; it is expected only to experience, generalize and correct, and keep reacting. It has no control over the nature of its next experience, and can take no initiative. Since the environment *is* its trainer, a reactive learning program can only succeed to the extent that its environment shapes its experience appropriately.

6. Driving Discovery

The distinctive property of *discovery learning* is that the program is expected to formulate and direct its own tasks, rather than to follow the examples or the tasks set by an instructor. Discovery learning is notoriously difficult, in part because the program must focus its own attention. Early work on mathematical discovery, for example, was found to be unintentionally biased by LISP, the language in which the concept definitions were formulated (Lenat, 1976; Lenat, 1984). Focus of attention requires meta-knowledge: a measure of self-awareness and a metric for *interestingness*. A discovery program must therefore either begins with a bias as to what is interesting or must learn such a bias.

If interestingness were domain-independent, or if the program were restricted to a set of related domains, interestingness could be defined in advance. Three fundamental elements of the definition are *surprise* (i.e., expectation failure), *curiosity* (i.e., partial but inconclusive knowledge), and *ignorance* (i.e., computation failure). Each of these is an important indication of inadequate knowledge often cited by people as the reason that they formulate a task.

Surprise occurs when a program expects something different from what actually occurs, e.g., a robot "believes" it has picked up a wrench and then sees that it is still on the table. Such states are a kind of key node. Indeed, the position in Figure 3 is interesting *because* it violates the expectation that X will win. The program needs to learn knowledge that will correct its expectation

for this state the next time. Rather than begin each contest from the initial state, a game-learning program could begin a practice session from such a key node. This would still not be exhaustive search, but a series of carefully organized "what if's" that addressed problematic positions.

Curiosity is spurred by partial but inconclusive data. Pell has considered incomplete experiential data about the efficacy of a move in Go on a 9 × 9 board (Pell, 1991). His program learned (gathered statistics on) the wins and losses associated with individual moves. It pursued (became curious about) moves that had a true mean of winning most likely to be better than a "fickleness parameter." This amounts to a forced exploration of situations which are not clearly categorized by the feature set and the program's playing experience. Against an external, hand-coded expert, Pell's program with the parameter set to .5 performed better than a program that simply chose a move based on the historical data. In the context of key nodes, the primary problem with this approach is that it is directed toward move preference, rather than toward knowledge acquisition for expert behavior. Thus the learning it results in is reflexive and not particularly transparent.

Moore has a program for a learning control system that attempts to concentrate experience in "regions of the control space that are relevant to the task at hand" (Moore, 1991). His program introduces some randomness into an otherwise carefully learned controller. He shows that random perturbations of a decision are better than random decisions, and that predictive analysis of random perturbation is even more effective. This last class of methods attempts to harness some regularity in the space to identify good decisions. The tacit assumption in that approach is that the value of an action is to some extent continuous in the space. Although that may be true for Moore's toy car driving domain, it is not necessarily true for many others, including game playing.

Hunter has taken ignorance as an impetus to plan to learn, primarily in the context of biological research (Hunter, 1989). His IVY was an opportunistic planner that generated knowledge goals. An IVY-inspired extension of the work described here would be to have a game-learning program deliberately construct a state (or a set of states) from which to begin a practice session. For example, if a program encounters a new opening and loses to it, why should it have to wait until it encounters that opening again to seek an appropriate defense? The program could instead undertake a deliberate investigation of the opening, alternately playing each side, preferably against the same expert who introduced it. Here the key node (or nodes) may expose important strategic strengths or weaknesses.

Another example of ignorance-driven search would be to seek a situation in which one decision-making principle (Advisor) would be inapplicable or always take precedent over another one. Rather than attempt to construct such states, the program could save their partial descriptions and alert itself for analysis when enough of them arise.

7. Future Work

There are many interesting issues associated with the theory of well-guided search as proposed here. The identification of key nodes is non-trivial; although Hoyle has some successful, domain-dependent detectors, "important" or "interesting" remains an elusive property. The distribution of key nodes, in particular their tendency to cluster, may vary with the domain. Perhaps combinations of key nodes, rather than individual ones, are what drives learning. Different learning strategies may perform better with different key nodes. Explicit labeling of key nodes may even suggest a new class of learning strategies that rely upon their significance as exemplars or counterexamples, or require clusters of key nodes as input. All of these are topics for future work.

Hoyle already saves contests that it considers significant because they violate its expectations. Key nodes arise, as Hunter suggests, in the context of "good teaching cases." Thus a natural way to turn a skilled learner into a skilled teacher would be to focus on these paradigmatic experiences. Where the program has learned, so may its students, particularly if the instructor and students have similar learning algorithms.

8. Conclusions

If the knowledge intrinsic to expert performance can only be extracted after all, or even most, of the nodes in an intractably large search space are visited, then prospects for a person to learn expertise there would be dim. *The existence of human experts in spaces intractable for them argues that most of the important knowledge is located at a limited number of key nodes.* Thus, visiting the key nodes should become a priority during learning in a very large space, regardless of the learning strategy used after arrival there.

Hoyle is predicated on the idea that general expertise in a broad domain can be efficiently instantiated for a subdomain to develop specific expertise there, as when general game-playing knowledge is applied to a particular game. For Hoyle, this is discovery learning, triggered by its experience during play. Although Hoyle exploits non-experiential knowledge (like the rules, the Advisors, and various knowledge representations), the program will learn nothing unless it plays. For human experts, too, experience is the catalyst for learning. Thus the focus on key nodes is appropriate.

Deliberate direction to key nodes is well-guided search. Given the goal of learning to perform expertly in a large space, a theory for well-guided search begins with the

following principles, supported by results described here:

- There are key nodes where important knowledge may be extracted by the learner.
- Key nodes appear in clusters.
- Guidance to those key nodes is both appropriate and necessary.
- Internal direction to key nodes is available from high-quality decisions made by the learner.
- External direction to key nodes can be managed by the trainer.
- A mixture of external direction to key nodes and exploration in their vicinity is a productive way to exploit the clusters of key nodes in a large space, and to compensate for trainer error.

Acknowledgments

This work has benefited from conversations with Tom Mitchell and the suggestions of several anonymous referees. The author was supported in part by NSF 9001936. Pascal Abadie and Joanna Lesniak provided programming support.

References

Berlekamp, E. R., Conway, J. H., & Guy, R. K. (1982). *Winning Ways for Your Mathematical Plays* . London: Academic Press.

Charness, N. (1983). Human Chess Skill. In P. W. Frey (Ed.), *Chess Skill in Man and Machine, second edition* (pp. 34-53). New York: Springer-Verlag.

Charness, N. (1991). Expertise in Chess: The Balance between Knowledge and Search. In K. A. Ericsson, & J. Smith (Ed.), *Toward a General Theory of Expertise - Prospects and Limits* (pp. 39-63). Cambridge: Cambridge University Press.

Chase, W. G., & Simon, H. A. (1973). The Mind's Eye in Chess. In W. G. Chase (Ed.), *Visual Information Processing* (pp. 215-281). New York: Academic Press.

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. (To appear). Toward an Ideal Trainer. *Machine Learning*.

Epstein, S. L., Gelfand, J., Lesniak, J., & Abadie, P. (1993). *The Integration of Visual Cues into a Multiple-Advisor Game-Learning Program.* Raleigh.

Ericsson, K. A., & Smith, J. (1991). Prospects and Limits of the Empirical Study of Expertise: An Introduction. In K. A. Ericsson, & J. Smith (Ed.), *Toward a General Theory of Expertise - Prospects and Limits* (pp. 1-38). Cambridge: Cambridge University Press.

Ericsson, K. A., & Staszewski, J. (1989). Skilled Memory and Expertise: Mechanisms of Exceptional Performance. In D. Klahr, & Kotovsky (Ed.), *Complex Information Processing: The Impact of Herbert A. Simon* (pp. 235-267). Hillsdale, NJ: Erlbaum.

Gasser, R. (1991). Applying Retrograde Analysis to Nine Men's Morris. In D. N. L. Levy, & D. F. Beal (Ed.), *Heuristic Programming in Artificial Intelligence 2 - The Second Computer Olympiad* (pp. 161-173). Chichester: Ellis Horwood.

Hunter, L. (1989). *Knowledge Acquisition Planning: Results and Prospects.* Morgan Kaufmann, 61-65.

Lenat, D. B. (1976). *AM: An Artificial Intelligence Approach to Discovery in Mathematics.* Ph.D., Department of Computer Science, Stanford University,

Lenat, D. B. (1984). Why AM and EURISKO Appear to Work. *Artificial Intelligence*, *23*(3), 249-268.

Moore, A. (1991). *Knowledge of Knowledge and Intelligent Experimentation for Learning Control.* 683-688.

Pell, B. (1991). Exploratory Learning in the Game of GO: Initial Results. In D. N. L. Levy, & D. F. Beal (Ed.), *Heuristic Programming in Artificial Intelligence 2 - The Second Computer Olympiad* (pp. 137-152). Chichester, England: Ellis Horwood.

Tesauro, G. (1991). *Personal communication.*

Tesauro, G. (1992). Practical Issues in Temporal Difference Learning. *Machine Learning*, *8*(3/4), 257-277.