DEEP FORKS IN STRATEGIC MAPS
Playing to Win*

Susan L. Epstein
Hunter College of The City University of New York
695 Park Avenue, New York NY 10021

HOYLE is a system intended to learn to play a broad class of two-person, perfect information game well. During play, HOYLE bases its decisions on the recommendations of its Advisors, one of which is dedicated to the detection of deep forks. Informally, a deep fork is a set of overlapping plans to win at a game, and the depth of a fork is a measure of its maximum time to success. HOYLE generates certain kinds of deep forks exhaustively, and uses them both to plan a skillful offense and to mount a prescient defense. The result is a powerful competitor that can play many different games adaptively, with clearly specified levels of speed and anticipation.

Because HOYLE's domain is a class of games, it is important to construct a uniform terminology for them. The first section of this paper provides HOYLE's definitions for games, strategy, and learning to play well. Section 2 places HOYLE in the context of other work in computer game playing. The third section defines strategic maps and deep forks, and demonstrates their efficacy. Section 4 explores the selection of a "best" deep fork and how to defend against one. Subsequent sections describe Pitchfork, the HOYLE Advisor responsible for deep forks, analyze the results of the generation of deep forks, report on some early results of their application, and describe future work.

1. GAMES, SEARCH, AND MEMORY

Informally, a game consists of some material (e.g., a board, playing pieces), a roster of one or more participants, and a list of rules. The rules describe how the participants

* This work was supported in part by NSF 9001936 and PSC-CUNY 668287.

are to take turns affecting the position and/or status of the material (e.g., moving pieces from the board or changing their location) until some rule terminates the process. When the process terminates, the rules declare either which participant has won or that the process was a draw. If all information about the game is disclosed and equally available to all the players (e.g., no closed hands, no uncertain outcomes as with dice), it is said to be a *perfect information* game. For the games considered here, there are exactly two participants: *Player* (the one who moves first) and *Opponent*. Player and Opponent are the *roles* in a two-person game.

Any game can be represented as a finite, directed graph (the *game graph*) in which a node (*state*) both identifies the participant whose turn it is (the *mover*) and describes a possible arrangement of the material. The participant who is not the mover is sometimes referred to here as the *non-mover*. An edge in the game graph from one (*originating*) state to another (*resultant*) state represents a *move*, the changes that occur when the mover in the originating state behaves (*takes a turn*) in any one manner permitted by the rules. Together, the material, participants, and game graph are called the *elements* of a game.

In the game graph there are three kinds of nodes: starts, finishes, and *intermediate* states. A *start* is an initial state of the material before any participant has taken a turn, with Player as mover. A *finish* is a node with no out edges, and is labeled exactly one of *win* (Player wins and Opponent loses), *loss* (Player loses and Opponent wins), or *draw* (no winner or loser). Each game has one or more starts and one or more finishes. A node that is neither a start nor a finish is called an *intermediate state*.

A *path* is a sequence $n_1e_1n_2e_2\ldots n_ke_kn_{k+1}$ of nodes $n_1,n_2,\ldots,n_k,n_{k+1}$ and edges $e_1,e_2,\ldots,e_k$, from the game graph such that e_i is an edge from n_i to n_{i+1}, for $i = 1, \ldots, k$. If there is a path $n_1e_1n_2e_2\ldots n_ke_kn_{k+1}$ in a game graph, node n_{k+1} is said to be at *depth k* or *k-ply* from node n_1. If there is any path in the game graph from a node back to itself, the game graph is said to contain a *cycle*. A move from a given node is said to be *invertible* if and only if it is the first edge in some path to a (not necessarily distant) node where the mover's position is identical to the current one; otherwise it is *uninvertible*. A path in a game graph from a start to a finish represents one complete experience of the game, and is called a *contest*.

During a contest, Player tries to arrive at a win and Opponent tries to arrive at a loss. Since from any state there are usually many legal moves, the challenge in play is for the mover to select the best move, i.e., one that will maximize the mover's opportunity to arrive at a desired finish, while minimizing the non-mover's chances to arrive at his own desired finish. A *strategy* is a set of heuristics for a specific game with a control method for choosing among them. A strategy is thus a procedure that, given any state, returns a move for the mover. To *learn to play a game well* is to produce and refine a strategy for it, by constructing, testing, and revising theories.

True expertise is perfectly backed up knowledge from the entire game graph. Such information is error free, but exhaustive forward search for it is feasible only in games (such as tic-tac-toe) with a manageable number of states, or from nodes all of whose immediate neighbors are relatively close to finishes. Most interesting games have extremely large game graphs; for them, true expertise through exhaustive search of the game graph is computationally intractable. In such games, people, and programs, must attempt a reasonable approximation of true expertise; the goodness of a strategy for such games can be measured only relative to other strategies for them.

2. COMPUTER GAME PLAYING AND HOYLE'S MEMORY

Computer game playing programs usually approximate true expertise by heuristic search of the game graph. From a given node in the game graph, the program estimates the unseen finishes of deeper intermediate nodes and backs up this approximate knowledge through the game graph (using minimax with alpha-beta pruning (Nilsson), the B^* algorithm (Berliner, 1979; Palay, 1982), or Rivest's min/max approximation (Rivest, 1987) to the given node to select the best possible legal move. Both Deep Thought (Anantharaman, et al., 1990) and Hitech (Berliner and Ebeling, 1989; Ebeling, 1986), two outstanding chess programs, rely heavily on this technique. Human experts can foil such programs, however, by executing plans that extend deeper than the programs' search.

Although many AI programs excel at games, such as checkers (Samuel, 1963, 1967), Othello (Lee and Mahajan, 1988; Rosenbloom, 1982), backgammon (Berliner, 1980; Tesauro, 1989), and chess, each of them plays only a single game, and derives its power from an arsenal of domain-dependent heuristics tailored to that specific task. HOYLE takes a more general approach. (For a more detailed description of HOYLE, see (Epstein, 1989.) HOYLE's thesis is that, in learning to play a new game, people apply a weak theory for the domain, one that encapsulates their experience playing other games and enables them to learn to play well. The metatheory captures commonalities among two-person, perfect information games. HOYLE's metatheory identifies a set of valid viewpoints and weighs them carefully against each other, according to the game, the stage the current state lies in, and the nature of its opponent.

Compared to these one-game programs, HOYLE does relatively little heuristic search and has a limited memory. Instead, HOYLE makes decisions based on a variety of advice. Although good human game players certainly look ahead a ply or two, and do try to recall analogous previous experiences during play, they are not limited to these techniques alone. At any given state in a game, there is usually a variety of possible moves, each of which has something to recommend it (a *positive comment*) or advise against it (a *negative comment*). A comment is attributable to some particular view of the game and the current state, usually a very narrow one. Each comment cites supporting evidence and is advanced with some measure of strength. During a contest, a human participant generates many such comments on the current state and then uses them to select the next move.

A Fork

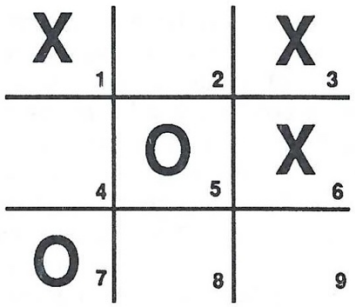


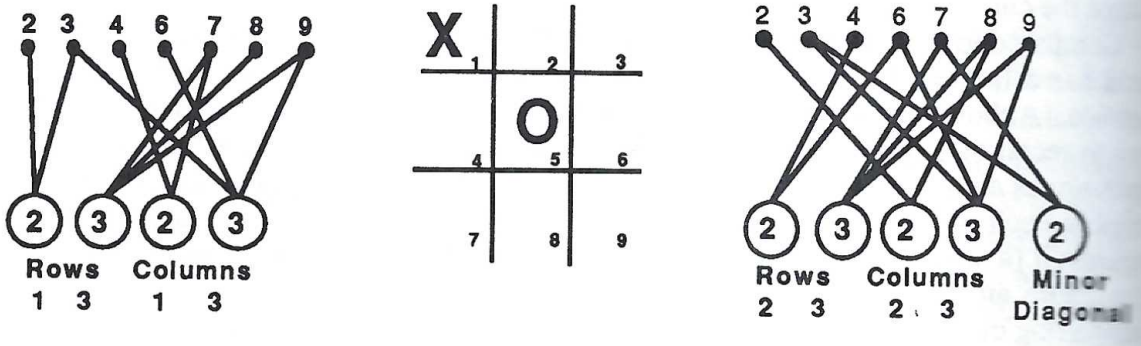
Figure 1

HOYLE simulates this comment-and-evaluate process with a panoply of Advisors and a flexible control structure to determine move selection. At the moment there are 20 Advisors, each of which generates one or more comments about the current state, and each of which has a deliberately narrow perspective on game playing. Among HOYLE's Advisors is one to exploit forks, called Pitchfork.

3. DEFINING FORKS IN STRATEGIC MAPS

Informally, a fork is two or more overlapping plans to win or to achieve an advantage. Figure 1 shows a fork in tic-tac-toe where Player's most recent move, into position 3, threatens a win in both the first row and the third column. Opponent is defenseless before this onslaught. Banerji (1980] devised a graph representation for the mover in which he identified certain patterns as "kernels," renamed "forks" here. An early implementation of Banerji's ideas (King] used a few simple forks to suggest moves in several positional games, like tic-tac-toe, where the placement of a piece in a location is irrevocable. HOYLE continues this research: it extends Banerji's original terminology, defines and explains the significance of a fork's depth, explores the generation and application of forks, and integrates forks with other techniques to construct both *offensive* (i.e., directing a path toward a win) and *defensive* (i.e., diverting a path away from a loss) plans.

A Game State and its Strategic Maps



Strategic Map for Player X **Strategic Map for Opponent O**

Figure 2

In games with only uninvertible moves, for any placement of pieces on the board, a strategic map can be constructed for each participant. A *strategic map* is a labeled bipartite graph in which one set of nodes (the *top vertices*) represents the available positions on the board, and the second set (the *bottom vertices*) represents the possible winning configurations with the number of moves each requires to be achieved. A strategic map may be denoted as <T,B,E>, where T is a set of top vertices, B a set of bottom vertices, and E a set of undirected edges from a vertex in T to a vertex in B. In Figure 2, for example, positions 2, 3, 4, 6, 7, 8, and 9 are available to both participants, rows 1 and 3 and columns 1 and 3 are possible winning configurations for Player, and rows 2 and 3, columns 2 and 3, and the minor (from upper right to lower left) diagonal are possible winning configurations for Opponent. In a strategic map, an edge is drawn from every still-possible winning configuration to every still-available position it

requires, and each bottom vertex is labeled with its degree. Figure 2 shows the strategic maps for Player and Opponent for the given board position.

Certain connected subgraphs of a strategic map, induced by bottom vertices, can be used to execute simultaneously more than one plan to win. (These subgraphs are denoted here as F-dn, where F stands for "fork," d is its depth, to be defined shortly, and n provides a distinct label.) For example, in a subgraph of the form F-31 in Figure 3(a), a move in either degree-two position at the top furthers the two plans represented by the bottom vertices adjacent to it. Such a move, in the left degree-two position, for example, transforms F-31 into Figure 3(b), two connected graphs, of the forms F-1 on the left and F-21 on the right. (Immediately before her most recent move into position 3 in Figure 1, the only edges in Player's strategic map form exactly F-21, where the top vertices represented 2, 3 and 9 and the bottom vertices represented Row 1 and Column 3.) A move in the second degree-two position transforms Figure 3(b) into Figure 3(c), three copies of F-1. Thus there are two moves in F-31 that, played in any order, construct three potential winning situations.

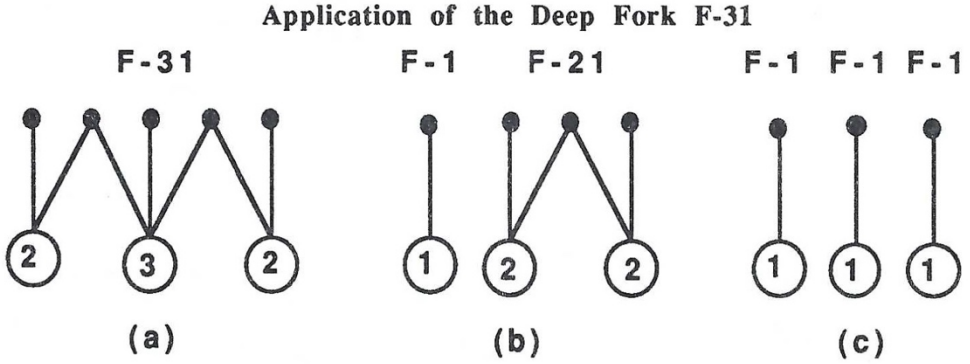


Figure 3

More formally now, a *fork* is defined to be either F-1 = <{t},{b},{(t,b)}> or a strategic map containing at least one top vertex whose removal, along with its edges, disconnects the graph into two or more connected components, each of which is a strategic map. The following theorem is self-evident:

Theorem

The set of all forks is defined, and may be completely and correctly generated, by the following:

- F-1 is a fork.
- The following construction produces a fork: Copy any n forks. (They are the *parents* of the new fork under construction.) Add a new top vertex v. (This is the *join* of the new fork under construction.) Add an edge from v to at least one bottom vertex in each parent.
- Nothing else is a fork.

Observe that a move in the join transforms the graph into n connected graphs, its parents, each of which is itself a fork. F-21, for example, is the result of joining two copies of F-1.

F-31 is called a fork of depth three because a win in three moves is certain if the fork belongs to the mover. (A win in fewer moves is possible, of course, but against

optimal defense, the win will require three moves.) F-1 is the *trivial fork*, a win in one move. In general, a *fork of depth d* is a fork where play in at least one specified key top vertex produces at least one connected component that is a fork of depth d-1. (For example, the keys in F-31 are the two top vertices of degree two.) Any fork with depth greater than 2 is called a *deep fork*. (Thus F-31 is a deep fork of depth 3.) A move that key of a fork is called an *execution* of the fork. Clearly, a deep fork of depth d is a plan to win in d moves.

Building New Forks

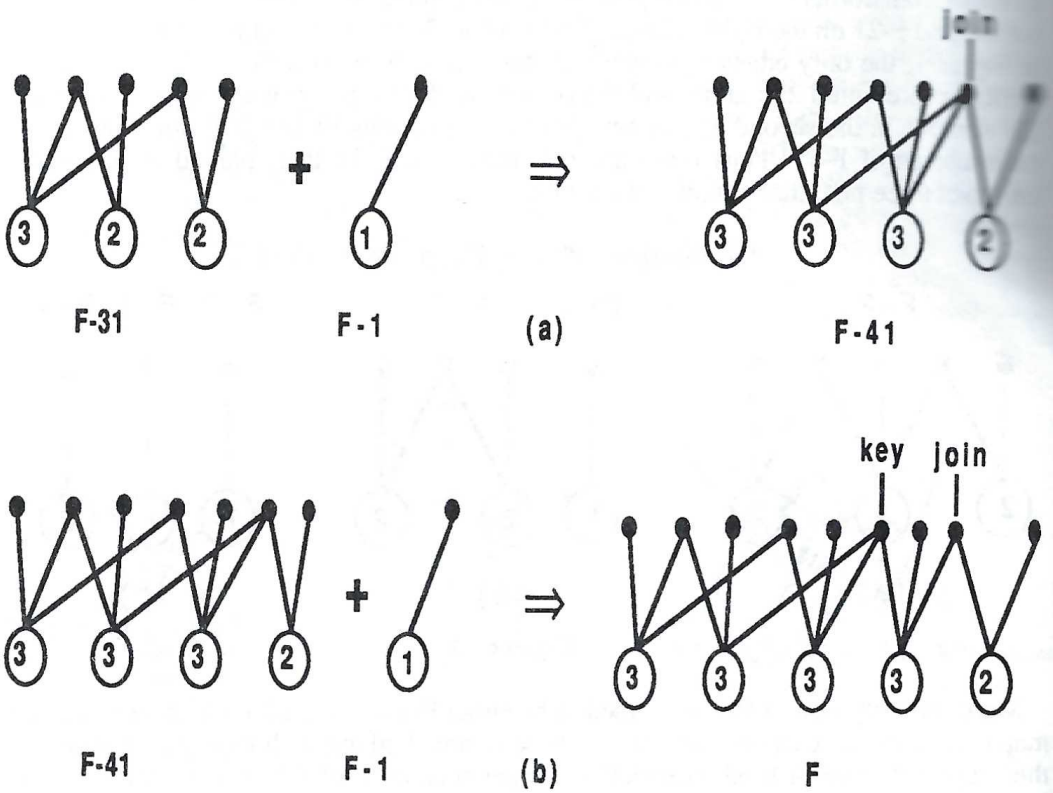


Figure 4

The depth of a fork is a non-trivial computation, however. Banerji expected that, since each of his forks was built by combining two earlier ones, every new fork would be, in some way, more difficult to defeat than its components. Interestingly, this is not the case. In his original work, Banerji sketched several forks of increasing size, building each one from a pair of earlier, intuitively simpler, forks and a join adjacent to at least one bottom vertex. The fork F-41 in Figure 4(a) is constructed from a copy of F-31 and a copy of F-1; the new join vertex is made adjacent to all the bottom vertices of degree one and two in both parent forks. F-41 has its join as its key and a depth of four. The fork F in Figure 4(b), although it is built from F-41 and F-1, does not have the join as its key. A move in the indicated key fragments F into F-31 and F-21, so F has depth four, rather than its expected five. When a two-parent fork like F is built from parents of depth m and n but has depth d less than the maximum of m+1 and n+1, it is because the fork's key is not its join. In this case, it can be shown that the key of the new fork

is a key of one of its parents, and that the new fork is constructible from parents of depths d-1 and m+n-d, respectively. (F, for example, can be built from F-31 and F-21.) How quickly a fork can achieve its objective, quantified here as its depth, must be computed as the minimum of all fork depths arising in the map when first the mover plays any top vertex and then the other participant plays another.

4. THE STRATEGY OF DEEP FORKS

Deep forks are significant both offensively and defensively in the play of a game. To exploit forks properly, one must consider both strategic maps for the current state. When the mover's strategic map contains a fork of depth d, a win in d turns is guaranteed against optimal defense only if the other participant has no possible winning configurations. Several examples illustrate these ideas.

Attacking a Fork

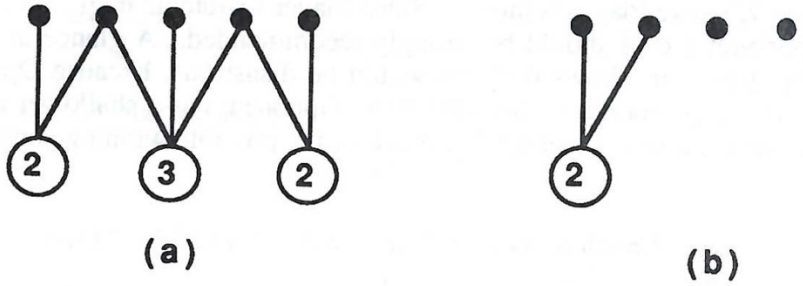


Figure 5

An opposing offensive fork may require the mover's consideration. The best defense against any opposing fork is to move in one of its keys yourself, permanently eliminating many of your opponent's possible winning configurations. For example, if the non-mover has a strategic map containing F-31, redrawn in Figure 5(a), the mover can destroy the fork by moving in either of the degree-2 positions (say, the right one), immediately reducing that fork to Figure 5(b). In what remains of the mover's strategic map, the only possible winning configuration is readily prevented on the next turn.

Another Game State and Two Strategic Maps

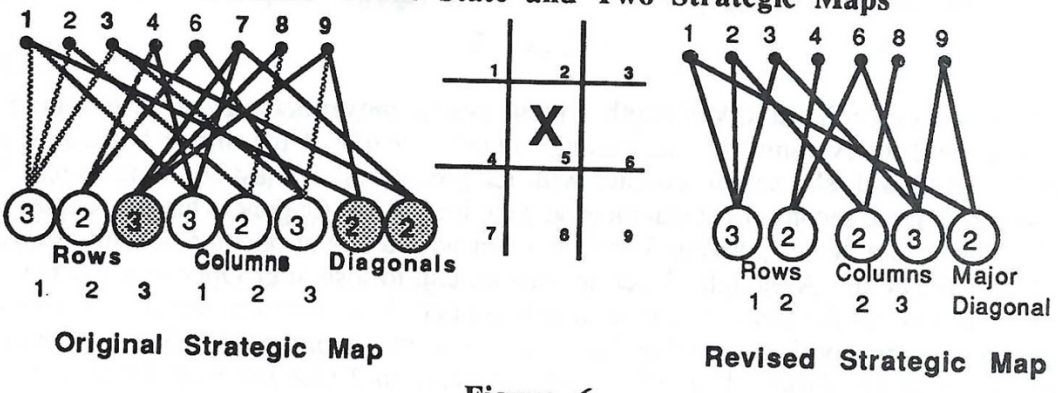
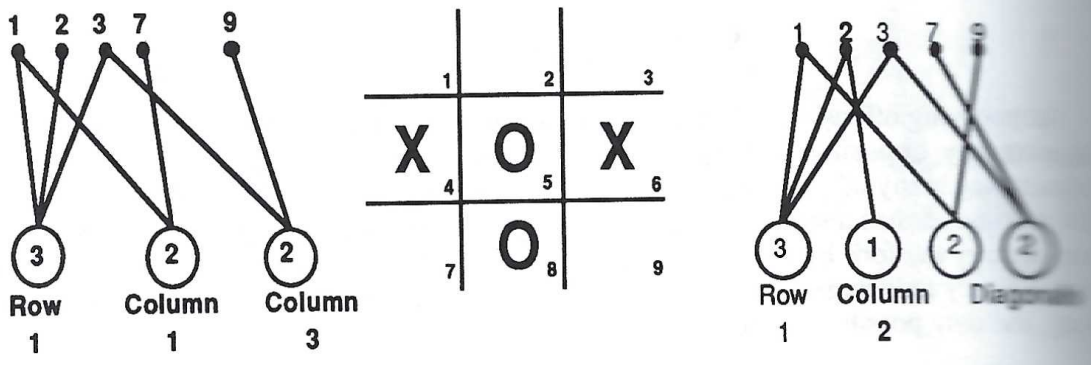


Figure 6

An example of such play occurs in Figure 6, in a tic-tac-toe contest where, in the strategic map on the left, X has an F-31 subgraph, as indicated by shaded vertices and solid edges. The fork is induced by the two diagonals and the bottom row in the strategic map; its execution requires a move in position 7 or 9. It is not Player's turn, however. An intelligent Opponent will destroy the fork by moving in 7 or 9 herself. After a move in 7, for example, Player is left with the strategic map on the right in which to search for forks on her next turn. There are actually four F-31 subgraphs to be found in the left strategic map of Figure 6. The others are induced by the two diagonals and any one of Row 1, Column 1, or Column 3. Collectively, their keys are positions 1, 3, 7, and 9, the corners. Exhaustive search and minimax of the tic-tac-toe game graph confirms that, in the Figure 6 state, the best next move will indeed be any of the corners the forks have enumerated. Thus deep forks provide a clear mathematical explanation for one kind of game playing expertise.

The detection of a fork in the mover's strategic map, even a relatively shallow fork like F-31, does not guarantee a win for the mover. Consider, for example, the tic-tac-toe state in Figure 7, where Player is mover. Since Player's strategic map is precisely F-31, a move to position 1 or 3 should be strongly recommended. A glance at Opponent's strategic map, however, shows that this would be disastrous, because Opponent will have a clear win at position 2 on her next turn. Opponent has a shallower fork, a copy of F-1. The correct move is clearly 2, a block of the possible winning configuration in Column 2.

Yet Another Game State with Strategic Maps



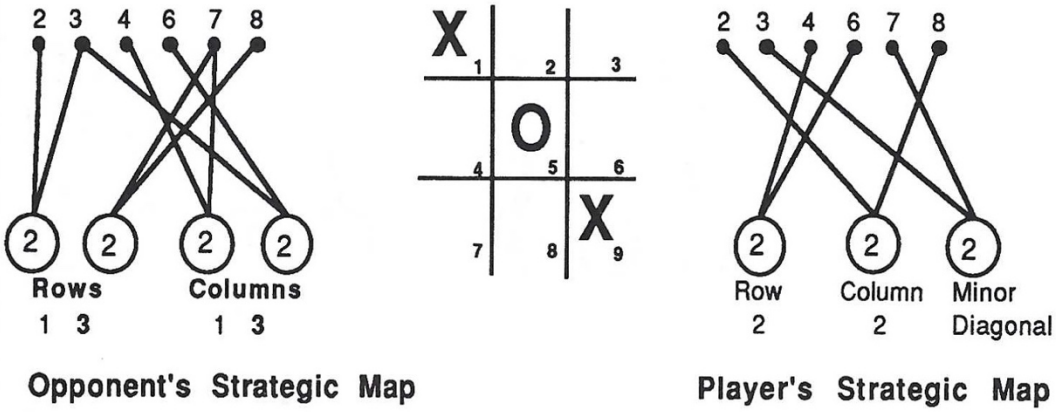
Strategic Map for Player X **Strategic Map for Opponent O**

Figure 7

If the non-mover has a fork of depth d, how does the mover defend against certain loss in only the d intervening turns remaining to her? Clearly, if the mover has a fork of depth less than d, she should counter with its execution. It may be, as in Figure 6, however, that the second participant has no such fork. Here Opponent has two copies of F-21, one with a key at position 3 and the other with a key at position 7, while Player has no forks at all. Although Player appears certain to lose after Opponent's next two moves, this is not the case. The correct defense here is to play a move in a (non-fork) possible winning configuration that forces Opponent to respond in anywhere except the key of one of her forks. Thus Player should move in 2 (forcing a response in 8), 4 (forcing a response in 6), 6 (forcing a response in 4), or 8 (forcing a response in 2), but

not in 3 or 7, where Opponent's correct response (7 or 3, respectively) would simultaneously defeat Player's possible winning configuration and execute a fork. On each remaining turn, Player can continue to distract Opponent from her forks. Despite Player's apparently strong position and Opponent's lack of forks, the contest in Figure 8 should result in a tie.

When a Fork Should be Ignored



Opponent's Strategic Map **Player's Strategic Map**

Figure 8

These examples are summarized as follows. Let d_m be the minimum of the degree of the mover's shortest plan and the depth of the mover's shallowest fork, and let d_n be the minimum of the non-mover's shortest plan and the depth of the non-mover's shallowest fork. If $d_m \leq d_n$, play at any top vertex in the mover's shallowest fork that fragments off a fork of degree less than d_n guarantees the mover a win in at least d_m moves. If $d_n < d_m$, however, the mover must defend the non-mover's fork or plan that presents the most immediate threat. There are several important observations to be made here:

• *Play should not necessarily be in the key*, i.e., the key of a fork merely determines its degree, but defensive requirements may demand that play be in a non-key that fragments off a small threat to force the other participant into a defensive stance. Such would be the case, for example, if mover had the fork F_{33} (see Figure 9) in her strategic map but the non-mover had a plan of degree of two. The key of F_{33} only would mount two offensive forks of degree two, which the other participant could ignore while she forwarded her own degree two plan. Play in either degree two vertex of F_{33}, however, fragments the fork into F_1 and F_{31}. The other participant must immediately reply to F_1, and then the mover can continue with F_{31}. Thus play in the key is not always the best choice.

• *Even with a fork of degree d_m, the win may require more than d_m moves*, i.e.,d_m is a lower bound for perfect play. Defensive considerations may delay the win, as in Figure 8 above.

• *Even with a fork, the contest may end in a draw before the win is achieved.* As in the contest of Figure 8, there may not be enough moves remaining in the contest for either participant to execute a winning fork or plan.

Thus, although a fork of depth d is a plan to win in d moves, its execution can either be prevented by a defense that plays in the fork or be delayed by the execution of possible winning configurations of degree d or less.

HOYLE's Two-Parent Forks of Depth 3 and Their Keys

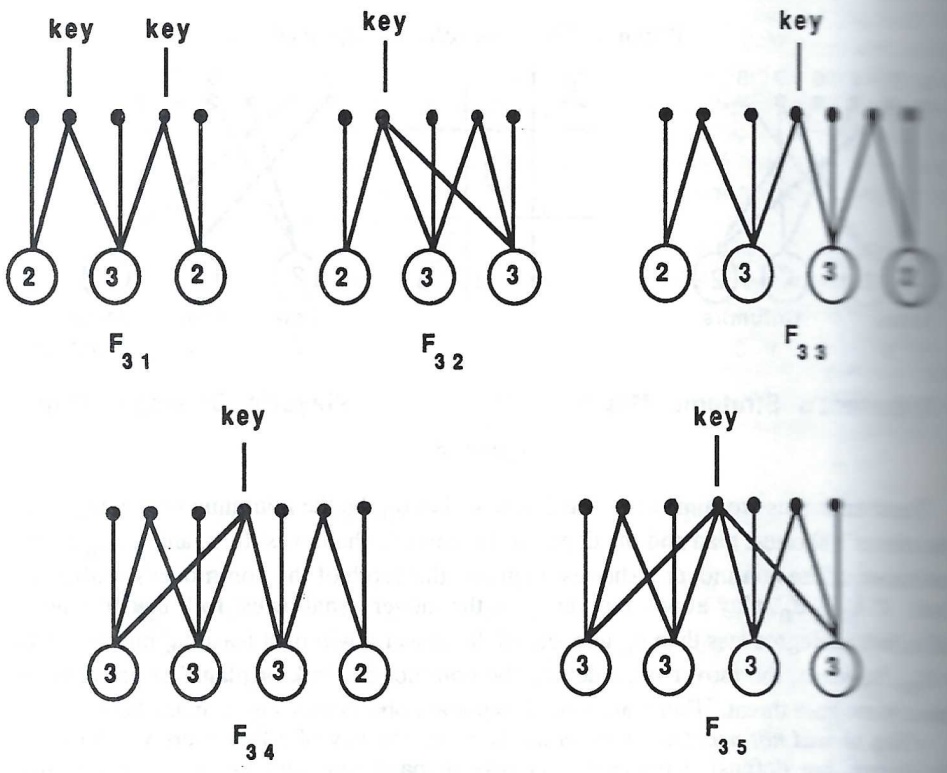


Figure 9

5. HOYLE APPLIES DEEP FORKS

Clearly, true expertise in positional games with uninvertible moves requires only the construction of strategic maps and the detection and execution of forks and possible winning configurations in them, as described in the previous section. For a machine to play perfectly, then, it would have to recognize all deep forks and their depths correctly. One way to recognize deep forks is to maintain a library of them and search for subsets of bottom vertices in a strategic map that induce an element of the library. As described in Section 3, all forks may be generated correctly and completely from the combination of any two or more known parent forks with a single new top vertex adjacent to at least one bottom vertex in each of the parents. Implementation of such construction is problematic, however. The algorithm generates infinitely many forks at any given depth (for example, one could combine k copies of F-1 to produce F-1,k), and it can also generate isomorphs of the same fork, as in the construction of F in Figure 4(b). As discussed earlier, the construction provides only an upper bound (one more than the

maximum depth of any of its parents) on the depth of the new fork, and so the fork must be explored several moves down to determine its true depth.

HOYLE's initial foray into two-parent fork construction through depth four warns of a combinatoric explosion. From the single fork F-1 at depth one, HOYLE builds one fork at depth two, and then 5 distinct forks at depth three, shown with their keys in Figure 9. There were, however, 520 distinct forks constructed at depth four. In two instances, HOYLE discarded a fork constructed from a depth three parent and a depth one parent because it proved to be isomorphic to some depth three fork already constructed from two parents of depth two. These discarded forks are analogous to F in Figure 4(b).

In the play of a positional game with uninvertible moves, HOYLE capitalizes upon the information about deep forks discussed here. For any given state, HOYLE's Advisor Pitchfork constructs the strategic maps for both Player and Opponent, and then considers forks according to the following heuristics and rationale:

Heuristic 1: Search only for two-parent forks. A fork that has more than two parents always has some two-parent fork as a subgraph. Although a fork that fragments into more than two smaller forks may seem to offer a greater challenge to the opposition, in reality it is totally unnecessary, since a player in these games need only attain one winning configuration. (In an n-person game, however, each fork would have to be built from n smaller forks at a time.) HOYLE maintains a library of all the two-parent forks it has constructed, with depth and key.

Heuristic 2: Search only in connected components of a strategic map for a fork. By definition, all forks are connected.

Heuristic 3: Search first in the mover's strategic map, then in the non-mover's, for a bottom vertex that induces a subgraph isomorphic to F-1, and terminate the search as soon as any one fork is recognized. If an offensive fork is discovered, move in its key. If a single defensive fork is discovered, move in its key. If more than one defensive fork of depth 1 is discovered, resign.

Move Recommendation with Respect to Fork Depth

```
Procedure fork-play(fork-list)
Offensive ← The first subgraph in the mover's strategic map isomorphic to a fork
            in fork-list
If Offensive
   then recommend moves in Offensive that fragment off a fork of degree less than
        any possible winning configuration in the non-mover's strategic map
   else Defensive ← The set of all subgraphs in the non-mover's strategic map
                    isomorphic to a fork in fork-list
        If |Defensive| = 1
          then recommend a move in each key of Defensive
          else recommend a move in the first applicable one of the following:
            all vertices that are keys in all elements of Defensive
            all vertices that are most often a key in Defensive and top vertices in
              each element where they are not keys
            all vertices that are top vertices in all elements of Defensive
            all vertices that are most often top vertices in Defensive
            all vertices that are most often keys in Defensive
```

Figure 10

Heuristic 4: If no fork has been detected at depth one, execute the procedure [illegible] in Figure 10 for the fork-list {F-21}. Fork-play makes one or more recommendations for moves, based upon the shallowest offensive and defensive forks it detects in the strategic maps for the current state. (Another HOYLE Advisor, Candide, recommends moves that advance possible winning configurations, such as those in Figure [illegible]. Ultimately, move choice is determined by HOYLE's architecture for selection among disagreeing recommendations (Epstein, 1990b).) If the procedure detects no offensive or defensive forks at depth 2, execute the procedure for a fork-list of all forks at depth [illegible] those in Figure 9. Fork-list forks must all be of the same depth. Fork-play terminates its search as soon as any one offensive fork in fork-list is recognized, and recommends moves in top vertices that produce forks of depth smaller than that of the non-mover's shortest possible winning configuration. There is no reason to discover all offensive forks; any one will suffice. If no offensive fork is detected but a single defensive fork is discovered, fork-play recommends moves in all the keys of that fork, i.e., blocks. At the same depth, offense takes priority over defense. If no offensive fork and more than one defensive fork in fork-list is discovered, fork-play recommends moves intended to block as many defensive forks as possible.

Heuristic 5: Restrict fork-list in Figure 10 to known forks whose number of top vertices is no larger than the current number of legal moves, and whose bottom vertices are of degree no larger than the number of positions required for a win. At least for simple games, not all deep forks are relevant. For example, in tic-tac-toe only deep forks whose bottom degrees are at most 3, and that have no more than 9 top vertices, can be relevant in any state in the game. Only 47 of HOYLE's generated forks at depth four meet these restrictions at the onset of the game; as play progresses, even fewer are relevant.

Heuristic 6: If no deep fork has been discovered thus far, call fork-play in Figure [illegible] on some small subset of the known forks of depth four and five. The supposition is that a human opponent is unlikely to construct or recognize such deep forks. For its own strategic map, HOYLE varies these selected forks from one contest to another to discourage an adaptive opponent. For the non-mover's strategic map, HOYLE remembers and searches for any that have arisen, particularly against the same opponent.

Heuristic 7: Inspection of HOYLE's constructed forks indicates that the key to the deep fork of a strategic map is usually its maximally-connected (highest degree) cutpoint (a vertex whose elimination, along with its edges, disconnects the graph). When no deep fork has been identified, recommend the maximally-connected cutpoint in the strategic map; it may well be a key, and at the very least, it forces the other participant to fragment her attention to multiple subgraphs. If there is no cutpoint, recommend the maximally-connected top vertex, on the theory that it forwards the greatest number of possible winning configurations. Among such vertices, the one whose neighbors have minimal degree sum is preferable because it advances the shortest paths to a win. This process is performed both offensively (in the mover's strategic map) and defensively (in the other participant's strategic map). In this case, HOYLE may be said to be playing the strategic maps rather than executing a fork.

Heuristic 8: If, early in play or in a complex game, either strategic map is particularly large and/or dense, examine only the subgraph of the map induced by bottom vertices of degree less than n, where n is the number of positions required for a winning configuration. This effectively focuses the attention of the search on the pieces already

in play and their most promising lines of attack. If n does not sufficiently curtail the size of the map, use n-1 or n-2.

6. RESULTS AND FUTURE WORK

Previously, a few known deep forks were used as naive offensive plans in limited computer play, and succeeded only when they went unobserved by the opposition. The work done for HOYLE and summarized here achieves the following:

- It rigorously analyzes the origin, nature, and applicability of forks.
- It defines and explains the significance of the depth and key of a fork.
- It exhaustively generates game-independent, two-parent deep forks through depth four.
- It strategically applies deep forks to defensive as well as offensive play.
- It constructs successful heuristics to exploit both deep forks and strategic maps in an efficient, adaptive manner.

HOYLE parameters quantify the balance between offensive and defensive play. For example, HOYLE may attribute a fork level of depth 3 (fairly sophisticated) to the other participant while requiring a depth of 5 for its own consideration. Thus HOYLE flexibly models the perspicacity of the other participant, as well as placing limitations on its own search time. HOYLE also remembers which deep forks have ever been identified in any state of a particular game and can search only for those. (The alternative, computing all possible deep forks that may ever arise, appears computationally infeasible for games of any reasonable challenge.)

Pitchfork recognizes that, among the deep forks of any strategic map, shallower is always better. Thus, in defense against the other participant, HOYLE can ignore any forks whose depth indicates that they will come to fruition only after HOYLE's best offensive fork. If, however, the other participant is preparing a shallower deep fork of her own, Pitchfork recognizes it and mounts a defense. Thus deep forks become plans, both offensively and defensively.

In the programmed implementation of deep forks and strategic maps, HOYLE has learned to play tic-tac-toe, lose tic-tac-toe, three dimensional tic-tac-toe on three parallel three-by-three grids and on four parallel four-by-four grids (Qubic©), tsoro yematatu, pong hau k'i, and achi. Knowledge of deep forks provided substantial initial strength. For games in which the size and number of potential forks create a combinatoric explosion, recent research (Epstein, 1990a) has shown how to learn topologically applicable forks and how to use the trace of a defeat in a contest of a game to construct an explanation that is a relevant, domain-independent, contest-independent fork. In three dimensional tic-tac-toe on three parallel three-by-three grids HOYLE now learns the depth four fork embedded in the initial game state from a single contest in which an expert is Player. After that experience, HOYLE claims victory whenever it is Player, and offers to resign after another Player makes the correct opening move. In Qubic, HOYLE now learns a depth four fork that a human expert had consistently used to defeat it. With knowledge of this fork HOYLE plays Qubic more strongly than any human opposition currently available to it.

Current research regards the extension of deep forks to games in which moves are invertible and games in which the goal is not the achievement of a specific piece configuration. The notion of a fork, for example, is well-known to chess players as an intermediate move either of whose results is desired (for example, a knight may threaten to capture two pieces, only one of which can be protected).

Because Pitchfork uses heuristics for deep forks and strategic maps, its play may be imperfect, so HOYLE must rely on other Advisors as well. For its current knowledge base, however, the exploitation of deep forks in strategic maps makes HOYLE a machine that plays to win, and usually does.

ACKNOWLEDGEMENTS

The author would like to thank Ranan Banerji for his original suggestion that HOYLE look at kernels, and Virginia Teller for her thoughtful critiques.

REFERENCES

Anantharaman, T., Campbell, M. S. and Hsu, F.-H. 1990. Singular Extensions: Adding Selectivity to Brute-Force Searching. *Artificial Intelligence* 43 (1): 99-109.

Banerji, R. B. 1980. *Artificial Intelligence: A Theoretical Approach*. New York: North-Holland.

Berliner, H. J. 1979. The B*Tree Search Algorithm: A Best-First Proof Procedure. *Artificial Intelligence* 12 (1): 23-40.

Berliner, H. J. 1980. Backgammon Computer Program Beats World Champion. *Artificial Intelligence* 14 (2): 205-220.

Berliner, H. and Ebeling, C. 1989. Pattern Knowledge and Search: The SUPREM Architecture. *Artificial Intelligence* 38 (2): 161-198.

Ebeling, C. 1986. All the Right Moves: A VLSI Architecture for Chess. Ph.D. diss., Department of Computer Science, Carnegie Mellon University.

Epstein, S. L. 1989. The Intelligent Novice - Learning to Play Better. In Heuristic Programming in Artificial Intelligence - The First Computer Olympiad, ed. D. Levy and D. Beal. New York: Ellis Horwood.

Epstein, S. L. 1990a. Learning Plans for Competitive Domains. In *Proceedings of the Seventh International Conference on Machine Learning*, 190-197. Ed. B. W. Porter and R. J. Mooney. Morgan Kaufmann.

Epstein, S. L. 1990b. Learning to Control a Blackboard System for Game Playing. In *AAAI Workshop on Blackboard Systems.*

King, P. F. 1970. A Computer Program for Playing Positional Games. M. S. diss., Case Western Reserve University.

Lee, K. F. and Mahajan, S. 1988. A Pattern Classification Approach to Evaluation Function Learning. *Artificial Intelligence* 36 (1): 1-26.

Nilsson, N. J. 1980. *Principles of Artificial Intelligence*. Palo Alto: Tioga Publishing.

Palay, A. J. 1982. The B* Tree Search Algorithm - New Results. *Artificial Intelligence* 19 (2): 145-164.

Rivest, R. L. 1987. Game Tree Searching by Min/Max Approximation. *Artificial Intelligence* 34 (1): 77-96.

Rosenbloom, P. S. 1982. A World-Championship Level Othello Program. *Artificial Intelligence* 19 (3): 279-320.

Samuel, A. L. 1963. Some Studies in Machine Learning Using the Game of Checkers. In *Computers and Thought*, ed. E. A. Feigenbaum and J. Feldman. 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.

Tesauro, G. and Sejnowski, T. J. 1989. A Parallel Network that Learns to Play Backgammon. *Artificial Intelligence* 39 (3): 357 - 390.