When Is It Better Not To Look Ahead?

Dana S. Nau[a], Mitja Luštrek[b], Austin Parker[*,a], Ivan Bratko[c], Matjaž Gams[b]

[a] *University of Maryland, College Park, MD, USA*
[b] *Jožef Stefan Institute, Ljubljana, Slovenia*
[c] *University of Ljubljana, Ljubljana, Slovenia*

Abstract

In situations where one needs to make a sequence of decisions, it is often believed that looking ahead will help produce better decisions. However, it was shown 30 years ago that there are "pathological" situations in which looking ahead is counterproductive. Two long-standing open questions are (a) what combinations of factors have the biggest influence on whether lookahead pathology occurs, and (b) whether it occurs in real-world decision-making.

This paper includes simulation results for several synthetic game-tree models, and experimental results for three well-known board games: two chess endgames, kalah (with some modifications to facilitate experimentation), and the 8-puzzle. The simulations show the interplay between lookahead pathology and several factors that affect it; and the experiments confirm the trends predicted by the simulation models. The experiments also show that lookahead pathology is more common than has been thought: all three games contain situations where it occurs.

Key words: lookahead pathology, minimax, game-tree search

1. Introduction

In situations where one needs to make a sequence of decisions, it is often believed that looking ahead (to predict the possible results of one's actions) leads to better decisions. There have been some dramatic demonstrations of

*Current address: Center for Computing Sciences, 17100 Science Drive, Bowie, MD 20715.

this principle in board games such as chess and checkers, where the performance of computer programs has greatly improved as improvements in computers and algorithms have made it possible to look farther ahead in the same amount of time [1, 2].

On the other hand, there are theoretical results saying that looking ahead does not always lead to better decisions. Thirty years ago a class of games was discovered [3] having a counterintuitive property called *lookahead pathology*, in which searching farther ahead consistently led to worse decisions rather than better ones. Furthermore, a game-tree model that was considered realistic at the time turned out to be pathological [4].

Although many explanations of lookahead pathology have been proposed and several factors affecting it have been found [3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19], these studies have several limitations:

- They have typically focused on a single factor at a time, without giving a clear understanding of the interplay among the factors. Consequently, several researchers have claimed—erroneously, in our view—that one or another of these factors was the reason for the absence of lookahead pathology in most real-world situations.

- Although lookahead pathology occurs in mathematical models and "artificial" games such as Pearl's game [5], it has remained unclear as to whether lookahead pathology occurs in real-world decision-making or is just a mathematical curiosity.

The purpose of this paper is to resolve the above problems. Our main contributions are as follows:

1. We use simulations of lookahead search for two- and one-player games to explore the interplay between lookahead pathology and three major factors affecting it: the heuristic evaluation function's *granularity* (the number of possible returned values), the game tree's *branching factor* (the number of successors of each node), and the tree's *local similarity* (the similarity

among the values of closely related nodes). The simulations show that the benefit provided by a deeper search increases with granularity, decreases with branching factor, and increases with local similarity. Consequently, lookahead pathology is most likely to occur with a low granularity, a high branching factor, and a low local similarity.

2. Experimental tests on substantially different games (two chess endgames, modified kalah,[1] and the 8-puzzle) show that the benefit provided by a deeper search follows the trends predicted by the simulations.
3. The experimental tests also show that even though all three games are mostly nonpathological, they all contain situations in which lookahead pathology occurs. This suggests that similar situations may occur in many decision-making environments; and our models may be useful in helping to predict what these situations are and how to deal with them.

The paper is organized as follows. Section 2 defines the minimax and minimin algorithms, lookahead pathology, and several other terms and quantities used throughout the paper. Section 3 describes the factors influencing pathology that we use as parameters in our game-tree models. Section 4 describes several other influences on pathology, and explains why we do not include them as explicit parameters in our models. Section 5 describes simulations showing how the factors described in Section 3 affect lookahead pathology in three game-tree models (two for two-player games and one for one-player games). Section 6 explores the influence of these factors in three games: chess, kalah and the 8-puzzle.

2. Definitions

2.1. Two-Player Games

A finite, two-player, perfect-information, zero-sum game can be represented by a game tree in which the nodes are the states and the edges are the moves.

[1] We made some simple modifications (see Section 6.2) to ensure a constant branching factor, strict alternation of play, and a uniform-depth game tree.

Each terminal node x in the tree is assigned a utility value $u(x)$. This utility value may be the game's final score or some other way of expressing which outcomes are preferable. We follow the usual convention of calling the two players Max and Min, where Max is trying to maximize the utility value and Min is trying to minimize it.

From the Minimax Theorem [20] it follows that each node x has a unique value $m(x)$, which is the utility value that will be obtained if both players play optimally:

$$m(x) = \begin{cases} u(x), & \text{if } \operatorname{suc}(x) = \emptyset, \\ \max_{y \in \operatorname{suc}(x)} u(y), & \text{if it is Max's move at } x, \\ \min_{y \in \operatorname{suc}(x)} u(y), & \text{if it is Min's move at } x, \end{cases} \tag{1}$$

where $\operatorname{suc}(x)$ is the set of x's successors.[2] Most game trees are so large that computing $m(x)$ is infeasible, but an approximation $m_e(x, d)$ can be computed using the minimax algorithm [21]:

$$m_e(x, d) = \begin{cases} u(x), & \text{if } \operatorname{suc}(x) = \emptyset, \\ e(x), & \text{if } d = 0, \\ \max_{y \in \operatorname{suc}(x)} m_e(y, d-1), & \text{if it is Max's move at } x, \\ \min_{y \in \operatorname{suc}(x)} m_e(y, d-1), & \text{if it is Min's move at } x, \end{cases} \tag{2}$$

where d, the *lookahead depth* or *search depth*, is a nonnegative integer saying how far to look ahead from x; and $e(x)$ is a *heuristic evaluation function* that computes an approximation of $m(x)$ from various features of the current game position. In order to clearly differentiate between $m(x)$ and $m_e(x, d)$, we will call the former a *utility value* and the latter a *heuristic value*.

We let $\operatorname{opt}(x, d)$ be the set of successors of x that look optimal according to the minimax algorithm, i.e.,

$$\operatorname{opt}(x, d) = \{y \in \operatorname{suc}(x) : m_e(y, d-1) = m_e(x, d)\}. \tag{3}$$

[2] We use "children" and "successors" synonymously throughout.

If a player moves to one of these nodes at random, then the probability of making an optimal move is

$$P_{\text{opt}}(x,d) = \frac{|\operatorname{opt}(x,d) \cap \operatorname{opt}(x,\infty)|}{|\operatorname{opt}(x,d)|}. \tag{4}$$

The minimax algorithm produces optimal play (i.e., $P_{\text{opt}}(x,d) = 1$) if e is completely accurate (i.e., $e = u$) or if d exceeds the height of the game tree. Neither is usually the case in practice, but variants of the minimax algorithm such as alpha-beta [22] and its derivatives are still widely and successfully used.

The minimax algorithm's *decision error* at a node x of a game tree is the probability of moving to a successor y of x such that $m(y) \neq m(x)$. If the player to move at x chooses at random from among the nodes in $\operatorname{opt}(x,d)$, then the decision error is

$$P_{\text{err}}(x,d) = 1 - P_{\text{opt}}(x,d) = 1 - \frac{|\operatorname{opt}(x,d) \cap \operatorname{opt}(x)|}{|\operatorname{opt}(x,d)|}. \tag{5}$$

The *degree of pathology* at a node x is the ratio between the decision errors when searching to two different lookahead depths:

$$p(x,i,j) = \frac{P_{\text{err}}(x,i)}{P_{\text{err}}(x,j)}, \tag{6}$$

where x is a node in the game tree, i and j are the search depths, and $i > j$. Values of $p(x,i,j) > 1$ indicate lookahead pathology; $p(x,i,j) = 1$ means that the quality of the decisions will be the same when searching to the depths i or j; and $p(x,i,j) < 1$ means that searching deeper is worthwhile. Note that this definition of the degree of pathology refers to a particular node x and depths i and j.

A game or a model is considered pathological if $p(x,i,j)$, averaged over x, is greater than 1.[3] When a game or a model is pathological for some values of i and j, usually it will also be pathological for other values.

[3]Depending on the game being studied and the objectives of the study, sometimes the average is over all x, and sometimes it is over a proper subset. For example, in the studies of Pearl's game in [5], $p(x,i,j) > 1$ for all x such that a depth-i search reaches neither the terminal nodes (all of which are at the same depth) nor their parents.

2.2. One-Player Games

In one-player games, the algorithm is similar to the minimax algorithm, except that there is only one player. If the objective is to minimize the cost of reaching a goal, then the player plays the role of Min at every node, and the minimum cost at each node is

$$m(x) = \begin{cases} u(x), & \text{if } \mathrm{suc}(x) = \emptyset, \\ \min_{y \in \mathrm{suc}(x)} c(x,y) + u(y), & \text{otherwise,} \end{cases} \tag{7}$$

where $c(x, y)$ is the cost of the edge from x to y.[4] An approximation of $m(x)$ can be computed using the *minimin* algorithm [23]:

$$m_e(x,d) = \begin{cases} u(x), & \text{if } \mathrm{suc}(x) = \emptyset, \\ e(x), & \text{if } d = 0, \\ \min_{y \in \mathrm{suc}(x)} c(x,y) + m_e(y, d-1), & \text{otherwise,} \end{cases} \tag{8}$$

where e is an A*-style heuristic function.

If the objective is to maximize a gain, then "min" is replaced with "max" in the above equations, and the algorithm is referred to as *maximax*. Maximax is what we used in the model in Subsection 5.3.

3. Influences on Pathology that Appear Explicitly in our Models

Researchers have investigated many different factors that influence whether or not lookahead pathology occurs in games. This section describes three factors that we will use as explicit parameters in our game-tree models in Section 5. These include the *branching factor* of the game tree (Subsection 3.1), the *granularity* of the heuristic function (Subsection 3.2), and *local similarity* among nodes of the game tree (Subsection 3.3).

Later, Section 4 describes several other factors that influence pathology, and explains why they are not explicit parameters in our game-tree models. Among

[4] It is conventional to include $c(x, y)$ in one-player games (Eqns. 7 and 8) and omit it in two-player games (Eqns. 1 and 2), but in principle it could be included in both.

other things, several of them can be viewed as special cases of the three factors mentioned above.

3.1. Branching Factor

A game tree's branching factor, b, is the number of successor nodes at each node in the tree. In game-tree models, b usually refers to a uniform branching factor, i.e., exactly b choices at each nonterminal node. In games where the number of successors may vary, b is the mean number of successors.

Nau [3] showed mathematically that for an infinitely large class of games, lookahead pathology was inevitable if b was sufficiently large. Intuitively, this happened because of the minimax algorithm's tendency to eliminate low values at Max's move and high values at Min's move. Increasing the values of b and d made it more and more likely that all values but one would be eliminated. This increased the probability that all successors of the current node would get the same heuristic value, regardless of which successors were the best moves.

In Nau's experiments with Pearl's game, a simple game designed for the analysis of search algorithms [5], lookahead pathology was indeed more likely with large branching factors [9]. In Beal's [4] game-tree models, for large branching factors, the error at the root of the tree increased by approximately $\log b$ with every additional level of the search.

3.2. Granularity of the Heuristic Function

An evaluation function's granularity, g, is the size of its range, i.e., the number of different values that it can return. Luštrek et al. [17] discovered that as the branching factor increases, the granularity (discussed in the next subsection) required to avoid lookahead pathology also increases. Intuitively, decreasing the granularity of an evaluation function makes it less able to distinguish among situations that are similar but not identical, and makes it more likely that a deeper search will return the same value for every child of the current node, making it less likely that the search will tell us which of the children are actually better.

In some of the early research on lookahead pathology (e.g., the work of Beal [4]), only granularity 2 was considered. Bratko and Gams [6] and Pearl [8]

compared granularity 2 to higher granularities and concluded that higher granularities do not prevent lookahead pathology. Scheucher and Kaindl [12], however, considered multi-valued evaluation functions (i.e., high granularities) essential to the prevention of lookahead pathology. Their work is described in more detail in Subsection 4.3.

3.3. Local Similarity

Several researchers have attempted to explain lookahead pathology by means of *local similarity*, i.e., similarity among the utility values of nearby nodes in the game tree [5, 7, 12, 17, 9]. Local similarity is probably the most widely accepted inhibitor of lookahead pathology; it generally is present in real games but absent in pathological models such as Pearl's game [5].

Intuitively, local similarity inhibits pathology in the following manner: if node a is better than node b, then the higher the amount of local similarity, the more likely it will be that most of a's descendants of are better than most of b's descendants, making it more likely that a deeper search will return a higher value for a than for b.

Researchers have introduced local similarity into game-tree models in a variety of ways. Beal [7] included in his game trees a fraction of nodes with all the successors having the same utility. Several other authors used "incremental" approaches in which the current position's utility changes gradually as moves are made in the game:

- Nau [5, 9] modified Pearl's game by randomly giving each edge in the game tree a value of $+1$ or -1. Each terminal node x was assigned utility $u(x) = 1$ if the sum of the values on the path from the root to x exceeded 1, and $u(x) = 0$ otherwise.

- Game-tree models by Luštrek et al. [17] used real-valued utilities, which were set in a similar way: instead of assigning $+1$ or -1 to game-tree edges, they assigned normally distributed real values; terminal utilities were then simply the sums of these values.

- Scheucher and Kaindl [12] also used an incremental approach, which was inspired by chess programs.

All of the above types of local similarity resulted in the elimination of lookahead pathology, although it should be noted that in the work of Scheucher and Kaindl, other factors described in Subsections 3.2 and 4.3 are also important. There is no standard way to measure local similarity, and in this paper we will use two different measures:

- In our models (see Section 5), local similarity is expressed as a parameter $0 \leq s \leq 1$, where $s = 0$ corresponds to complete independence among sibling nodes' utility values, and $s = 1$ corresponds to the maximum amount of similarity or dependence among sibling nodes that the model allows. If $s = 0$, this means that each node's utility value is in no way affected by the values of its siblings, as in Pearl's game where all leaf nodes have independently assigned random values. If $s = 1$, this means that the value of each node is as similar to the values of its siblings as the model allows.

- Although s is useful for expressing local similarity in a game-tree model, it is not a measure of local similarity in an arbitrary game. In games we therefore use the game tree's *clustering factor*, f, which is the ratio between the standard deviation of the sibling nodes' utilities and the standard deviation of the utilities throughout the tree [15]. If f is low then s is high, and vice versa, but the precise numeric correspondence between s values and f values will generally be different in different game-tree models.

4. Other Factors that Influence Pathology

This section describes several other factors that influence whether or not lookahead pathology occurs in games, and explains why we do not include them as explicit parameters in our game-tree model. One factor, graph structure (Subsection 4.1), can be mapped directly into local similarity. A second factor,

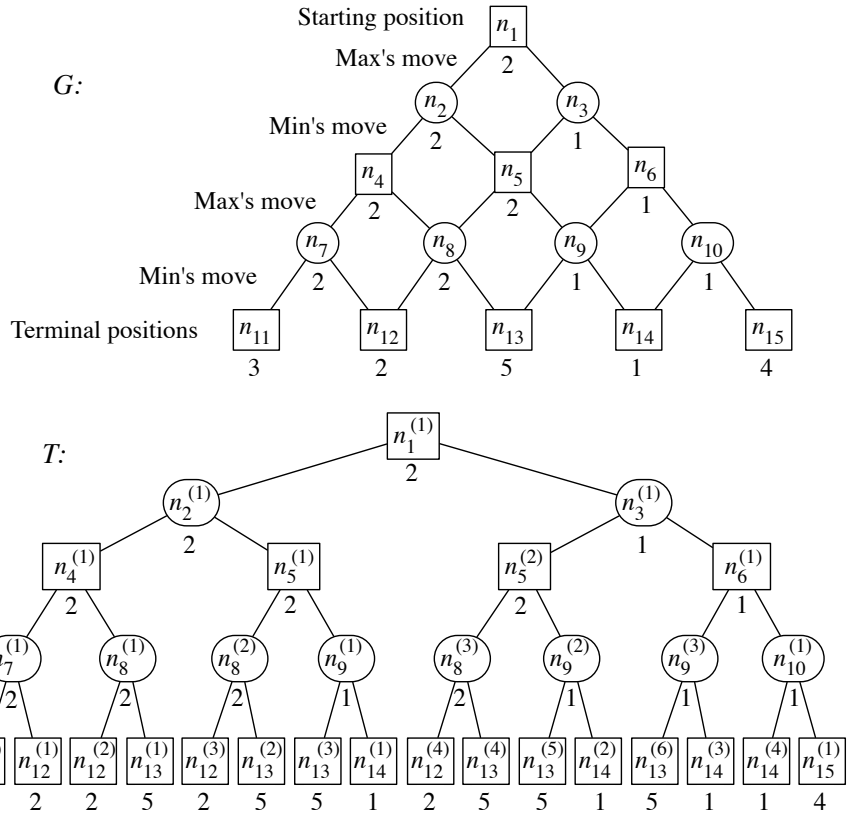

Figure 1: A game G whose state space is a graph, and the corresponding game tree T. The number below each node is its utility value.

reliably evaluated nodes (Subsection 4.3), maps into local similarity but only in an approximate fashion (the mapping involves applying the evaluation function in cases where one already knows the exact value). A third factor, improved evaluations deeper in the game tree (Section 4.3), is related (though there is not a direct mapping) to evaluation-function granularity. The remaining factors (Subsection 4.4) are specific to one-player games.

4.1. Graph Structure

Pathology has been shown to vanish if there are sufficiently many different paths to the same position [10]. As we will now explain, this can be viewed as a special case of local similarity.

If G is a game whose state space is an acyclic graph, then we can map G into a game tree T that is an "unfolded" version of G. If n_i is a node of G and there are p paths from G's root to n_i, then there are nodes $n_i^{(1)}, \ldots, n_i^{(p)}$ in T that are "duplicates" of n_i, and all of these duplicates have the same utility value as n_i. For example, in Fig. 1, the nodes $n_8^{(1)}$, $n_8^{(2)}$, and $n_8^{(3)}$ are duplicates of n_8, and $u(n_8) = u(n_8^{(1)}) = u(n_8^{(2)}) = u(n_8^{(3)}) = 2$.

Let n_i and n_j be sibling nodes in G, and suppose they have duplicates $n_i^{(k)}$ and $n_j^{(l)}$ in T. The more children n_i and n_j have in common, the more duplication there will be among the children of $n_i^{(k)}$ and $n_j^{(l)}$, hence the higher T's local similarity will be.

For example, consider Fig. 1 again. No two terminal nodes of G have the same utility value; but there are multiple paths to nearly every terminal node, hence T contains many duplicates of G's terminal nodes. Consequently, in 3 of the 7 pairs of sibling nodes in T, the siblings have the same the utility value.

4.2. Reliably Evaluated Nodes

If sufficiently many nodes in a game tree are evaluated reliably (i.e., without error or with a very small error) compared to other nodes on their level, minimaxing reduces the heuristic error, which means that a larger search depth is beneficial. One place where reliable evaluations are likely to occur is in subtrees with homogenous utilities, in which most of the terminal nodes are either all lost or all won [6]. It is easy to see that this is a special case of local similarity, by noticing that for nearly every node in the subtree, most or all of the nearby nodes will have the same utility (e.g., see the two rightmost subtrees in Fig. 2).

Another place where reliable evaluations occur is at terminal nodes encountered by lookahead search, such as early checkmates in chess [8, 24]. Early terminal nodes may also be considered as a case of local similarity, since they can be interpreted as the roots of subtrees in which all the nodes have the same value. This also is illustrated in Fig. 2.

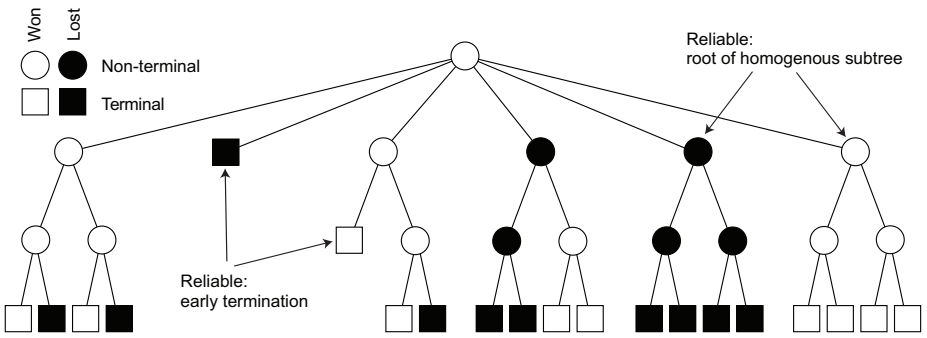

Figure 2: A two-player game tree with two types of reliably evaluated nodes: early terminations and roots of homogenous subtrees.

4.3. Improved Evaluations Deeper in the Game Tree

Scheucher and Kaindl [12] proposed a game-tree model in which heuristic values have a local similarity inspired by chess: after each player's move, only a limited change of value in favor of the moving player is possible, reflecting the change in material caused by that move. The heuristic value of the root of the tree is set to 0. As the search depth increases, the heuristic values tend to get ever further apart.

Scheucher and Kaindl observed that a lost position is less likely to be mistaken for a won one or vice versa in positions with extreme (strongly positive or negative) heuristic values, since those positions are more clearly decided in favor of one of the players. Therefore they introduced a function that determined the error in each game-tree node such that the probability of mistaking a loss for a win or vice versa was inversely correlated with the absolute heuristic value of that node. Since heuristic values lower in the tree tend to be further apart, the frequency of extreme values increases with the search depth and the probability of a misevaluation decreases. This depth-related decrease—probably combined with the effect of local similarity itself—was shown to be sufficient to eliminate lookahead pathology.

It was later shown that if the heuristic error is modeled appropriately, lookahead pathology disappears for a similar reason as in the work of Scheucher and Kaindl even without local similarity [14]. If real-valued utilities drawn ran-

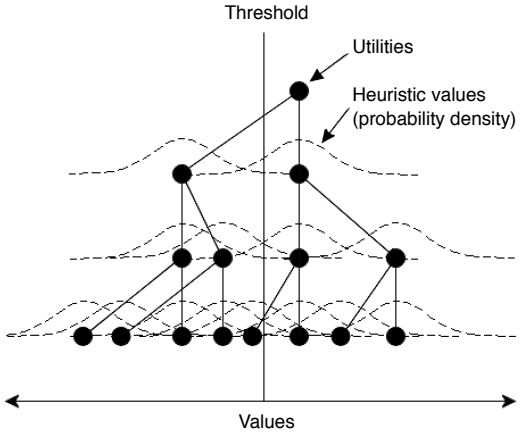

Figure 3: Utilities (circles) and heuristic values (dashed curves showing probability densities) are more likely on the same side of the threshold lower in the game tree.

domly from a uniform distribution are assigned to terminal nodes of the game tree, they are also further apart lower in the tree. The mechanism that achieves this is minimaxing itself: the player who is maximizing the utility eliminates low values, and the opponent, who is minimizing the utility, eliminates high values. Moving from terminal nodes towards the root, this results in an ever narrower range of utilities. The heuristic values are obtained by adding Gaussian noise to the utilities. If the heuristic values and utilities are mapped to losses and wins using a threshold, it is more likely that both the heuristic value of a node and its utility are on the same side of the threshold lower in the game tree. This happens because utilities lower in the tree are further apart and thus also further from the threshold, as is illustrated in Fig. 3. Since a loss is mistaken for a win or vice versa when a heuristic value is on the opposite side of the threshold from the corresponding utility, the probability of such a mistake decreases with the search depth. This decrease is sufficient to eliminate lookahead pathology.

It is clear that sufficiently improved evaluations deeper in the game tree can reduce or eliminate lookahead pathology. The papers discussed in this subsection are not trying to show that, but rather how such improved evaluations can be achieved realistically through increased granularity. For that reason, this paper does not address improved evaluations, but does address granularity.

4.4. Influences on Pathology in One-player Games

Lookahead pathology in one-player games has not been studied in as much detail as that in two-player games, probably because it was discovered more than 20 years later [13]. The factors described in the previous subsections have not been thoroughly investigated in one-player games (which this paper attempts to remedy), but some other factors have been investigated.

Search algorithms for one-player games typically use an A*-style heuristic function $h(x)$ that attempts to estimate the optimal cost of reaching a goal from the state x. The first factor that influences lookahead pathology in one-player games is $h(x)$ is optimistic or pessimistic. Pessimistic functions were found to be less prone to lookahead pathology in synthetic game trees [16], in the 8-puzzle [25, 18] and in path-finding [19]. Monotonically non-decreasing functions (the heuristic value of a node is not smaller than the heuristic value of its parent) were also found to be less prone to lookahead pathology [16].

Two factors were shown to influence lookahead pathology in synthetic trees [16]: how the utilities of the children of the root node's optimal successor compared to the utilities of the children of the root node's other successors, and how the utilities of the root node's successors compared to each other. However, the influence of these two factors seems to be tied to the heuristic function in [16], which was an artificial one that is only suitable for one-player games.

Finally, there are some results regarding lookahead pathology in path-finding using the LRTS algorithm [19]. This algorithm learns updates to the heuristic values during the search. A shallower search benefits more from learning than a deeper search. This makes decisions based on a shallower search better than mere depth would suggest and thus closer to a deeper search. Learning was therefore found to increase the degree of pathology and if turned off, the degree of pathology decreased (although the quality of decisions also decreased). The second reason was that after each lookahead search, the player makes a number of moves equal to the search depth. Thus if a deeper search sends the player in a wrong direction, the mistake is larger than if a shallower search does so, which again increases the degree of pathology. These results appear to be specific to

the LRTS algorithm, which is only one of the algorithms used for the real-time search in one-player games.

This paper is concerned with factors that influence lookahead pathology in both one- and two-player games, attempting to present a unified picture of pathology in both types of games. The factors discussed in this subsection, while certainly having an influence on lookahead pathology in one-player games, are either not present in two-player games or are of little interest in that domain. Therefore we do not address them in this paper.

5. Game-tree Models

This section describes simulations with synthetic game trees showing how the degree of pathology varies as a function of three factors: granularity g, branching factor b, and local similarity s [26]. Three probabilistic game-tree models are described in the following three subsections. These models were used to build 10,000 trees for each combination of the settings of the three factors of interest: $g = 2, 3, \ldots, 60$ for the two-player models and $g = 2, 3, \ldots, 300$ for the one-player model; $b = 2, 3, \ldots, 10$; and $s = 0.0, 0.1, \ldots, 1.0$. The values of g went as high as was needed to obtain nonpathological trees for most of the settings of the remaining two factors. We made a simplifying assumption that the branching factor is uniform within each tree. It was limited to at most 10 because the size of the tree is exponential in b. Building trees with large branching factors is thus computationally expensive and the few that we did build exhibited the behavior one might expect based on the smaller trees.

Each game tree was searched once to depth 1 and once to depth 5 to measure the degree of pathology $p(\text{root}, 5, 1)$. Depth 5 (and not more) was chosen to ensure that enough trees could be generated in a reasonable time (i.e., a few weeks), to yield statistically reliable analyses. Depth 1 was chosen as the minimal depth needed for an informed move selection. Some experiments were also performed with other pairs of depths, such as (6, 2) and (7, 1), giving similar results. Minimax results with odd and even depths, such as (6, 1), were somewhat different due to one player having one move more. Lookahead pathology

was measured in the root of the game tree so that we did not need to build larger trees than required for searching to the chosen depth. All three models build subtrees in the same manner as the whole tree, so choosing a non-root node as the starting point would not give qualitatively different results.

The heuristic evaluation function was the utility of a node corrupted by Gaussian noise with $\sigma = 0.1$. We assumed that no early terminations (such as checkmates), which could be evaluated perfectly, are encountered within the searched space. Furthermore, we made the classic assumption that the error of the evaluation function is the same at all search depths. We also did not attempt to model any specific phenomena that may appear in a real game tree, such as a position appearing unattractive in the short run, but eventually leading to victory. All these phenomena certainly occur in real games, but such detailed modeling is probably game-dependent and is beyond the scope of this paper. Despite these limitations, we believe that the chosen evaluation function models a typical game situation reasonably well. The magnitude of the static heuristic error was chosen so that the error was not large compared to the utilities. At the same time, we did not want it to be so small as to produce few wrong moves, since in that case too many game trees would have to be generated for statistically reliable analyses.

All three game-tree models model situations that are not strongly in favor of one of the players. Such situations are of greatest interest and previous work on lookahead pathology in two-player games was often focused on them [4, 6, 7, 12]. We use two mechanisms to ensure that the probabilities of a victory for both players are comparable. Each mechanism is explained in the section describing the model it belongs to.

5.1. Two-Player Top-Down Model

The first step in generating a game tree according to this model is assigning the utilities to the terminal nodes. There are three cases, depending on what value we want for the tree's local similarity s:

- If $s = 0$ (the "independent" case), the terminal nodes' utilities are inde-

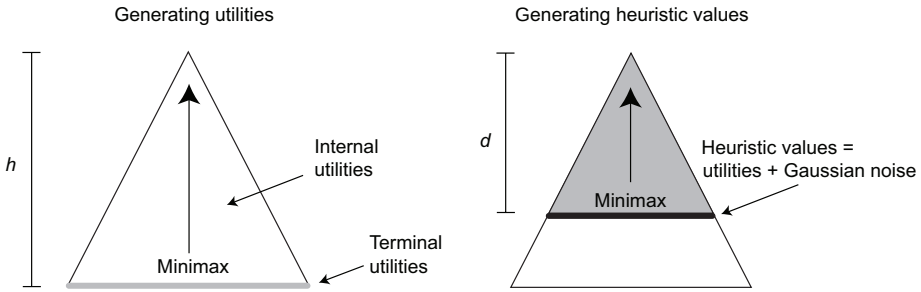

Figure 4: Generation of utilities and heuristic values.

pendently chosen from a uniform distribution over the interval [0,1].

- If $s = 1$ (the "maximally similar" case), the root of the game tree is first assigned a so-called auxiliary value, which is then propagated to the terminal nodes. The root node's auxiliary value is 0. For each non-root node, the auxiliary value is the sum of its parent's auxiliary value and a random value drawn from a Gaussian distribution (if the random value exceeds 3σ of the distribution, a new value is drawn). The utilities of the terminal nodes are their auxiliary values normalized to the interval [0,1].

- If $0 < s < 1$, the utility of each terminal node is taken with probability s from a game tree with $s = 0$, and with probability $1 - s$ from a tree with $s = 1$.

The utilities of the internal nodes are computed from the terminal utilities using the minimax algorithm. When searching to depth d, heuristic utility estimates are assigned to the nodes at level d (the levels are numbered downwards, starting with 0 for the root). They are generated by corrupting the utilities at level d with Gaussian noise representing the error of the heuristic evaluation function. The heuristic values of the nodes above depth d are computed from the heuristic values at depth d using the minimax algorithm. This is illustrated in Fig. 4.

The number of possible utility values or heuristic values in a game tree is called granularity. The initial values during the generation of the tree are real

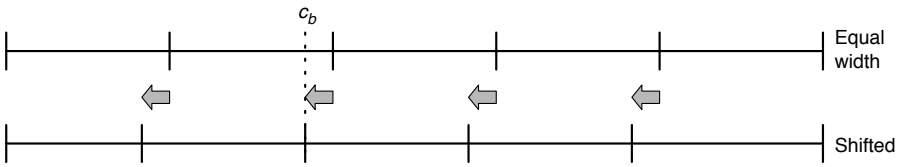

Figure 5: Shifting of buckets that ensures that not only a single value appears at the root of a two-player game tree.

numbers, but we afterwards convert them to g discrete values. The simplest way to do this is by partitioning the interval [0,1] into g buckets of equal width. However, in trees with local similarity $s = 0$, at small granularities there is a tendency for the values towards the root of the tree to converge to a single bucket containing the value c_b (called w_b in [9] and also discussed in [4], [6], and other studies on lookahead pathology in two-player games). This value is the solution of the equation $c_b = (1 - c_b)^b$; $c_2 \approx 0.38$ and then c_b slowly decreases with increasing b. The convergence to c_b happens because the player who is maximizing the utility eliminates all the low values, and the opponent, who is minimizing the utility, eliminates all the high values. At the root of such a tree no meaningful choice of a move can be made. We avoid this phenomenon by shifting the boundary between the two buckets most likely to appear at the top of the tree to c_b, as shown in Fig. 5. The other bucket boundaries are also shifted so that all the buckets except for the outermost two retain their original widths. By doing so, no single bucket contains the value c_b and thus at least the two buckets on each side of c_b have a reasonable chance of being represented at the root. Consequently, the player to move at the root may decide between at least two different values. We also generalize c_b to game trees with the local similarity larger than 0 by defining it as the value for which the ratio between the probability of a utility being smaller than c_b at one level and the probability of a utility being larger than c_b at the next level is closest to 1. This also achieves different utility values for the moves available at the root of the game tree.

5.2. Two-Player Bottom-Up Model

This model differs from the two-player top-down model in two aspects: the generation of the utilities of terminal nodes, and the mixing of independent and maximally similar game trees. The generation of nonterminal utilities and all the heuristic values and the granularization are the same as in the two-player top-down model.

There are again three cases for the generation of terminal utilities:

- Is $s = 0$, the terminal utilities are chosen in the same way as in the top-down model: they are independently chosen from a uniform distribution over the interval $[0, 1]$.

- If $s = 1$, b^h uniformly distributed random numbers in the interval [0,1] are generated. They are sorted and assigned to the terminal nodes starting with the lowest number at the leftmost node and finishing with the highest number at the rightmost node.

- If $0 < s < 1$, in principle the bottom-up model could use the same procedure as in the top-down model, i.e., randomly mixing terminal nodes from a game tree with $s = 0$ and a tree with $s = 1$. We tried this method and the relations among lookahead pathology, granularity, local similarity and branching factor were qualitatively similar to the other models presented in the paper. However, the maximally similar tree often seemed to overwhelm the independent one at relatively low values of s, because inserting even a small number of utilities into a tree – if those are the ones that are actually propagated to the root – may completely change the situation at the root. Because of that we modified the mixing procedure by increasing the probability of taking a low or high utility from the maximally similar tree. This softened the impact of the maximally similar tree because most of the low and high utilities are not propagated to the root. As mentioned before, this happens because the player maximizing the utility eliminates most low values, and the opponent, who is minimizing the utility, eliminates most high values.

5.3. One-Player Model

We designed the model for one-player games similarly to the bottom-up model for two-player games, but it differs from the two-player bottom-up model in two aspects: the procedure for backing up (i) the utilities and heuristic values and (ii) the granularization. The generation of the utilities of terminal nodes and the mixing of independent and maximally similar game trees are the same as in the two-player bottom-up model.

The utilities of the nonterminal nodes are computed from the utilities of the terminal nodes using the maximax algorithm. The same backing-up procedure is used for computing the heuristic values above depth d from the depth-d heuristic values.

The problem of the convergence of the utilities toward the root of the game tree to a single value is even more pronounced in one-player games than in two-player games. The phenomenon occurs as soon as every decision at the root of the tree leads to at least one terminal node with the maximal utility, which is quite likely if the granularity is not large. We solve this by limiting the probability of the maximal utility being reached by each decision at the root to at most 50%, which ensures some variation in the values from among which the player is choosing. Let x be a direct successor of the root, h be x's height (i.e., the path length from x to a terminal node), $P_{\max}(x)$ be the probability of x having the maximal utility, and $P_{\max}(t)$ be the probability of a terminal node having the maximal utility. Since x does not have the maximal utility only if none of the terminal nodes of the subtree rooted in x have the maximal utility, the following equations describe the relation between $P_{\max}(x)$ and $P_{\max}(t)$:

$$1 - P_{\max}(x) = (1 - P_{\max}(t))^{b^h}$$

$$P_{\max}(t) = 1 - \sqrt[b^h]{1 - P_{\max}(x)}$$

In order to limit $P_{\max}(x)$ to 50%, we allow $P_{\max}(t)$ to be at most $1 - \sqrt[b^h]{0.5}$ This is accomplished by limiting the size of the bucket containing the largest utilities. If the bucket is too large, its lower boundary is increased as shown in

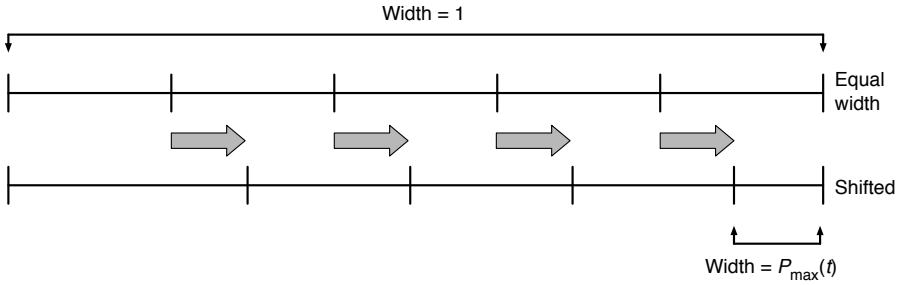

Figure 6: Shifting of buckets that ensures that the probability of the maximal utility being reached by each decision at the root of a one-player game tree is at most 50%.

Fig. 6. The other bucket boundaries are again shifted so that all the buckets except for the outermost two retain their original widths.

5.4. Results from the Models

We used all three models to compute the granularity needed to avoid lookahead pathology, as a function of the branching factor b and the local similarity s. Figs. 7, 8, and 9 show the surfaces corresponding to $p(\text{root}, 5, 1) = 1$ for all three models. In each case the results are qualitatively similar: the granularity needed to avoid lookahead pathology decreases with local similarity and increases with the branching factor. We also tried varying the granularity, the branching factor, the lookahead depth, the way that local similarity was introduced, and other parameters, such as the magnitude and distribution of the heuristic error, but the results were still qualitatively similar to those in the three figures [14, 17].

One can study not only whether lookahead pathology occurs in a given situation, but more generally how beneficial or harmful it is to search deeper. Fig. 10 shows the degree of pathology (the smaller it is, the more beneficial a deeper search is) with respect to the granularity, local similarity, and branching factor. We can observe the same relations among the factors affecting lookahead pathology as in Figs. 7, 8, and 9. In addition, Fig. 10 reveals that the branching factor has two different effects on the degree of pathology. One effect is that as b increases, the size of the pathological region also increases. This is consistent

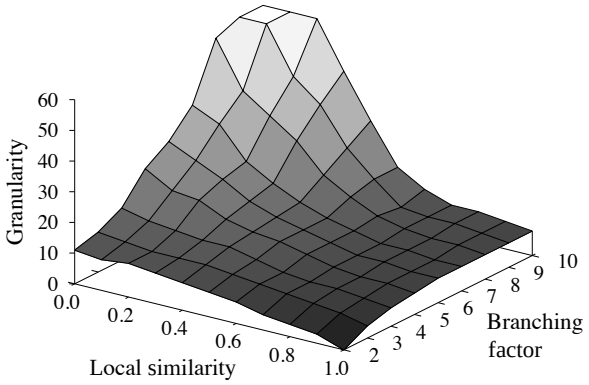

Figure 7: Amount of granularity needed to avoid pathology in the two-player top-down model. The space below the surface is pathological, the space above it is nonpathological.

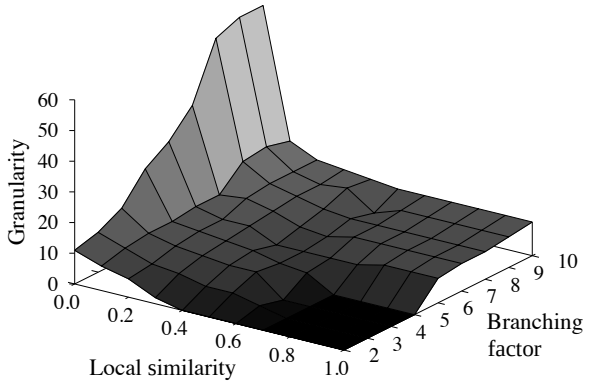

Figure 8: Amount of granularity needed to avoid pathology in the two-player bottom-up model. The space below the surface is pathological, the space above it is nonpathological.

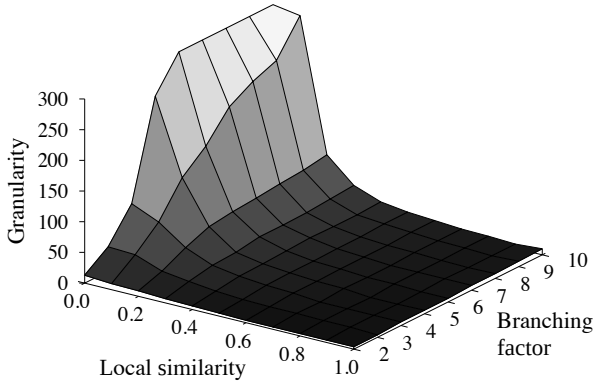

Figure 9: Amount of granularity needed to avoid pathology in the one-player top-down model. The space below the surface is pathological, the space above it is nonpathological.

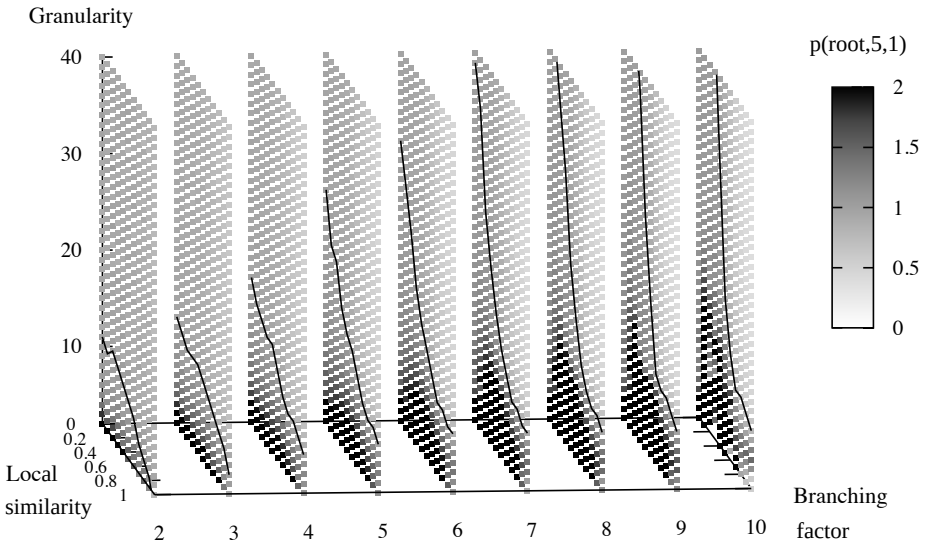

Figure 10: The degree of pathology $p(\text{root}, 5, 1)$ measured using the two-player top-down model, as a function of branching factor b, granularity g, and local similarity s. The color of each point in the graph shows the value of $p(\text{root}, 5, 1)$. The regions below the curved black lines are pathological, and the regions above those lines are nonpathological.

with the results of related studies described in Subsection 3.1. The other effect is that large values of b amplify the difference in the degree of pathology between the pathological and nonpathological regions. For example, consider any two points x and y that are close to the boundary between the pathological and nonpathological regions, and on opposite sides of the boundary. As b increases, the difference between x's degree of pathology and y's degree of pathology also increases.

The findings from the experiments with these models are consistent with the early pathological models of two-player games [3, 4, 5, 6, 7, 8, 9]: lookahead pathology occurs when $s = 0$ and $g = 2$. As shown in the following section, the findings are also consistent with real games and game-playing programs.

6. Games

The simulations described in the previous section predict how the degree of pathology depends on the granularity, local similarity, and branching factor.

This section presents experimental tests of those predictions in three real-world games: chess, kalah, and the 8-puzzle.

6.1. Chess

The relation between lookahead pathology and granularity was studied in the KBBK (king + bishop + bishop vs. king) and KQKR (king + queen vs. king + rook) chess endgames [15]. Following the example of [15], the heuristic estimate of a move's utility was defined as the number of moves on the shortest path to the end of the game from the resulting position, corrupted by Gaussian noise with $\sigma = 2$. To reflect the fact that the state space is a graph rather than a tree, the evaluation function cached each node's corrupted value, so that whenever a node is reached along more than one path in the search space, it would return the same corrupted value for that node every time.

It may be argued that distance to win is a rather artificial evaluation function because in chess the task is just to win, and not necessarily to win in the quickest way. As Sadikov et al. (2005) argued, an appropriate interpretation of a heuristic evaluation function is that such a function roughly indicates, for a given position, the chances of a (fallible) player winning the position against another (fallible) player. Distance to win can be interpreted as such an indicator. Although virtually all the positions in, say, KBBK are won for the stronger side, a fallible player may have difficulties in actually mating in 50 moves (the number of moves to mate allowed by the rules of chess). Such an imperfect player will have much better chances to win in a position where mate is possible in two moves, than in a position that requires 20 moves. Natural heuristic functions for playing a specific endgame also typically exhibit preference for shorter wins and should thus also have some correlation with our distance to mate. For controlled simulation experiments, distance to win has the advantage that it is a perfect evaluation which can be corrupted in a controlled way by our usual introduction of Gaussian noise.

The granularity was varied by partitioning the interval in which the heuristic estimates lie into g buckets of equal width. For each of the two endgames, Fig. 11

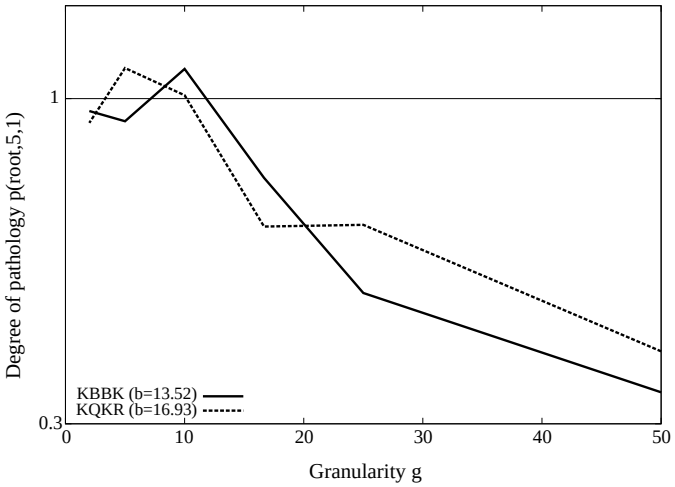

Figure 11: Degree of pathology as a function of granularity in KBBK chess endgames (average $b = 13.52$ and $f = 0.58$) and KQKR chess endgames (average $b = 16.93$ and $f = 0.37$).

shows the degree of pathology $p(x, 5, 1)$ as a function of granularity, averaged over every position x in the endgame database that is more than 5 moves away from a checkmate. Since different positions had differing branching factors and clustering factors, the figure's caption gives the average values of each. For $g \geq 10$, an increasing granularity increases the benefit of a deeper search, which is consistent with the predictions of the simulations described in the previous section. Some pathology can be observed at granularities up to 10, but this depends on the depths of search i and j used to measure the degree of pathology $p(\text{root}, i, j)$. The boundary between the pathological and nonpathological parts of the space in Fig. 10 is also around $g = 10$.

Fig. 10 also illustrates that in cases where b and f have opposite effects, it can be hard to say much about either of them individually. The KBBK's average branching factor ($b = 13.52$) is lower than the KQKR's ($b = 16.93$), which might lead one to expect KBBK to be less pathological than KQKR. But the KBBK's average clustering factor ($f = 0.58$) is higher than the KQKR's ($f = 0.37$), which might lead one to expect the opposite. In Fig. 11, sometimes KBBK is less pathological than KQKR, and sometimes the reverse.

Apparently it is quite rare for an entire game to be pathological, but looka-

head pathology in some game positions is more common. We analyzed 1092 chess positions from world championship matches [27]. Several chess programs searching to different depths were compared to Rybka (the strongest program available for our experiments) at depth 12. In 5.5% to 9.2% of the positions (depending on the program), we observed that the other programs chose Rybka's move at their smallest search depths but chose other moves at depth 5. Assuming that these other moves were worse than Rybka's (which seems likely since Rybka is a better player and is searching considerably deeper), this suggests that lookahead pathology occurred for these programs in 5.5% to 9.2% of positions. It should be noted, though, that Rybka's move is not guaranteed to be correct, and that occasionally several moves are all roughly comparable and can thus be considered equally best; so these results are not entirely conclusive.

Having pathology in some positions is not nearly as alarming as it would be to have pathology most positions, or to have larger average errors at deeper search depths than at shallower search depths. However, pathological positions are still of interest—firstly, because they are undesirable if unrecognized, and secondly, because we could select the best move with less effort than required by a default-depth search if we were able to recognize them.[5]

6.2. Kalah

Kalah is an ancient African game [29] played on a board with a number of pits, each containing a number of seeds, in which the objective is to acquire more seeds than the opponent, either by moving them to a special pit (called a "kalah") or by capturing them from the opponent's pits. For our experiments, we made the following modifications to produce a "regularized" version of the game:

- The game is normally played until no seeds are left on the board, but computability requires limiting the game to a small number of moves (in our case, 8 moves).

[5]Some work has been done on heuristic techniques for recognizing such positions, but the work is still at an early stage [28].

- To ensure a uniform branching factor, we allowed players to "move" from an empty pit; such a move has no effect on the board. We obtained different branching factors by varying the number of pits.
- We eliminated the "move-again" rule, where a player can move a second time when the last seed placed lands in their own "kalah".

In our experiments, we generated random initial boards by distributing seeds across the available pits, and averaged our results over these initial boards. The utility of a terminal node was the difference in the numbers of seeds captured by each player, and the utilities of the internal nodes were computed using the minimax algorithm from the terminal utilities. The heuristic evaluation function was the node's utility corrupted by Gaussian noise with $\sigma = 0.9$. The same caching and granularity techniques were used as in the chess experiments.

As shown in Figs. 12 and 13, our experimental results with modified kalah are qualitatively consistent with the predictions produced by the simulations in the previous section. Fig. 12 shows that lookahead pathology (i.e., $p(x, 5, 1) > 1$) occurs at most granularities when the branching factor $b = 6$, at small granularities when $b = 5$, and at granularity 3 (the smallest we tried) when $b = 4$. In general, when the granularity g is small, increases in g cause sharp decreases in $p(x, 5, 1)$, but when g is large, increases in g have little effect. Fig. 13 shows that $p(x, 5, 1)$ decreases with increasing local similarity, which is again consistent with the simulations. This figure also shows that lookahead pathology is stronger when b is large, as expected.

6.3. The 8-puzzle

A similar relation between lookahead pathology and granularity to that in chess and modified kalah was observed in the 8-puzzle [25]. This is a one-player game that is played on a grid of numbered tiles with one tile missing. The goal of the puzzle is to arrange the tiles sequentially by sliding them into the empty space, in turn revealing another empty space in the position of the tile moved.

For our experiments we used two heuristic evaluation functions: the number of moves to the solution, corrupted by Gaussian noise as described earlier, and

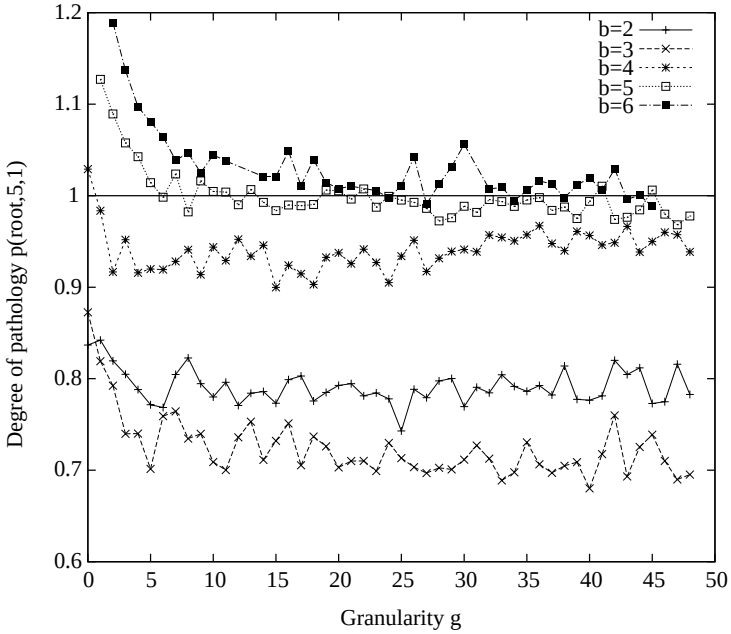

Figure 12: The degree of pathology in modified kalah as a function of granularity, at several different branching factors.

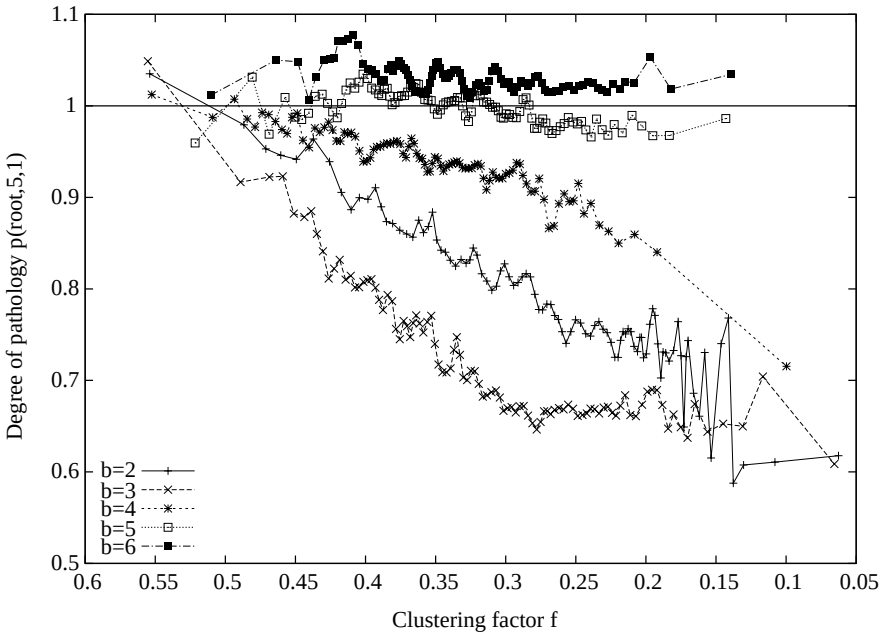

Figure 13: The degree of pathology in modified kalah as a function of clustering factor, at several different branching factors. The higher clustering factor, the lower the local similarity.

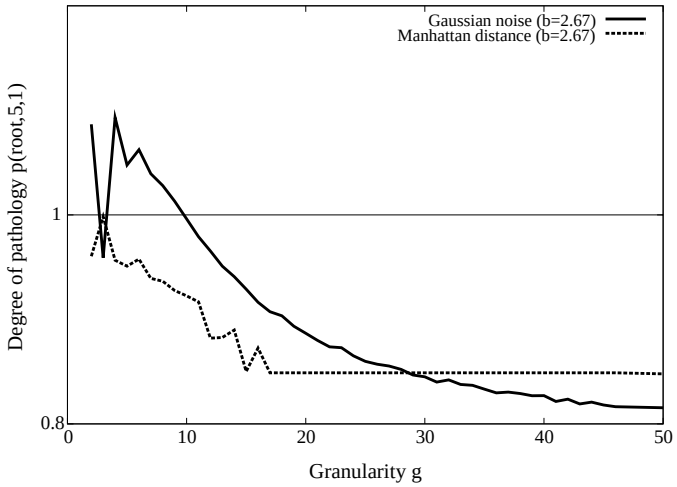

Figure 14: Degree of pathology as a function of granularity in the 8-puzzle, using two different heuristic functions. On average, $b = 2.67$ and $f = 0.73$.

the Manhattan distance of the tiles from their final positions (commonly used in search analyses).

Fig. 14 shows the degree of pathology $p(x, 5, 1)$ with respect to the granularity g. Both functions exhibit an increased benefit of a deeper search with increasing granularity, which agrees with the simulations in the previous section. According to Fig. 14, on average the 8-puzzle is pathological only at small granularities and only if the Gaussian-noise evaluation function is used.

We also investigated how many positions in the 8-puzzle are pathological. With the Manhattan-distance heuristic evaluation function, in 31.0% of positions a search to depth 5 gives better decisions than a search to depth 1, in 19.7% depth 1 is preferable, and in 49.3% it does not matter. Other pairs of depths give different percentages, but the observation that the advantage of a deeper search is not overwhelming is true quite generally in the 8-puzzle.

7. Conclusions

Two models of game trees for two-player games and one model for one-player games were constructed. Simulations with them showed similar behavior for both minimax and maximax: lookahead pathology was more likely when the

granularity was low, the branching factor was high, and the local similarity was low. Furthermore, large branching factors amplified the difference in the degree of pathology between pathological and nonpathological regions.

In several substantially different games—two chess endgames, modified kalah, and the 8-puzzle—our experiments confirmed that increasing the granularity generally increases the benefit of a deeper search. This effect of granularity was observed for many different combinations of the branching factor b and clustering factor f.

In modified kalah, we performed additional experiments that confirmed two more of the simulation results: the degree of pathology (measured as $p(x, 5, 1)$) decreased when the local similarity was high, and increased when the branching factor was large.

In modified kalah when the branching factor was sufficiently high, lookahead pathology occurred (i.e., $p(x, 5, 1) > 1$ on average) at almost every granularity. To the best of our knowledge, this is the first known game (other purely artificial games such as P-games [5, 9]) where lookahead pathology occurs throughout most of the game. Apparently such games are quite rare.

On the other hand, our experiments suggest that *local* pathologies, i.e., game positions where pathology occurs even though the game may be nonpathological on average, may be much more common. For example, in the chess championship matches, 6.9% to 8.5% of the positions were very likely pathological, and in the 8-puzzle, pathology occurred in 19.7% of the positions (i.e., $p(x, 5, 1) > 1$ at those positions). Thus in many games it might be useful to look for ways to detect and overcome local pathologies [28].

Generalizing beyond board games, we suspect that in many decision-making problems there may be particular kinds of situations where lookahead pathology is likely to occur. If so, then our experiments may have a practical utility in suggesting what those situations are, so that decision-makers can recognize such situations and take appropriate measures to deal with them. For example, one should be cautious about lookahead pathology when the branching factor is high and the local similarity is low, and in such situations one might consider

modifying the heuristic evaluation function to do finer-grained evaluations that capture more properties of the domain.

Acknowledgments

This work has been supported in part by AFOSR grant FA95500610405, NAVAIR contract N6133906C0149, NSF grant IIS0412812, the Slovenian Ministry of Higher Education, Science and Technology and the Slovenian Research Agency under the Research Programme P2-0209 Artificial Intelligence and Intelligent Systems. The opinions in this paper are those of the authors and do not necessarily reflect the opinions of the funders. In addition, we would like to acknowledge valuable intellectual contributions from Matej Guid, Boštjan Kaluža, Rok Piltaver, Aleksander Sadikov, and Aleš Tavčar. Finally, we appreciated the useful comments of the anonymous reviewers.

References

[1] M. Campbell, J. A. Joseph Hoane, F.-H. Hsu, Deep blue, Artificial Intelligence 134 (1–2) (2002) 57–83.

[2] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, S. Sutphen, Checkers is solved, Science 317 (5844) (2007) 1518–1522.

[3] D. S. Nau, Quality of decision versus depth of search on game trees, Ph.D. thesis (1979).

[4] D. F. Beal, An analysis of minimax, in: M. Clarke (Ed.), Advances in Computer Chess 2, Edinburgh University Press, 1980, pp. 103–109.

[5] D. S. Nau, An investigation of the causes of pathology in games, Artificial Intelligence 19 (3) (1982) 257–278.

[6] I. Bratko, M. Gams, Error analysis of the minimax principle, in: M. Clarke (Ed.), Advances in Computer Chess 3, Pergamon Press, 1982, pp. 1–15.

[7] D. F. Beal, Benefits of minimax search, in: M. Clarke (Ed.), Advances in Computer Chess 3, Pergamon Press, 1982, pp. 17–24.

[8] J. Pearl, On the nature of pathology in game searching, Artificial Intelligence 20 (4) (1983) 472–453.

[9] D. S. Nau, Pathology on game trees revisited, and an alternative to minimaxing, Artificial Intelligence 21 (1–2) (1983) 257–278.

[10] D. S. Nau, On game graph structure and its influence on pathology, International Journal of Computer and Information Sciences 12 (6) (1983) 367–383.

[11] B. Abramson, A cure for pathological behavior in games that use minimax, in: Proc First Workshop on Uncertainty and Probability in Artificial Intelligence, 1985.

[12] A. Scheucher, H. Kaindl, Benefits of using multivalued functions for minimaxing, Artificial Intelligence 99 (2) (1998) 187–208.

[13] V. Bulitko, L. Li, R. Greiner, I. Levner, Lookahead pathologies for single agent search, in: Proc IJCAI, posters section, 2003, pp. 1531–1533.

[14] M. Luštrek, I. Bratko, M. Gams, Why minimax works: An alternative explanation, in: Proc IJCAI, 2005, pp. 212–217.

[15] A. Sadikov, I. Bratko, I. Kononenko, Bias and pathology in minimax search, Theoretical Computer Science 349 (2) (2005) 261–281.

[16] M. Luštrek, Pathology in single-agent search, in: Proc Information Society, 2005, pp. 345–348.

[17] M. Luštrek, M. Gams, I. Bratko, Is real-valued minimax pathological, Artificial Intelligence 170 (6–7) (2006) 620–642.

[18] A. Sadikov, I. Bratko, Pessimistic heuristics beat optimistic ones in real time search, in: Proceedings of the 17th European Conference on Artificial Intelligence, 2006, pp. 148–152.

[19] M. Luštrek, V. Bulitko, Thinking too much: Pathology in pathfinding, in: Proceedings of the European 18th Conference on Artificial Intelligence, 2008, pp. 899–900.

[20] J. von Neumann, O. Morgenstern, Theory of Games and Economic Behavior, Princeton University Press, 1944.

[21] C. Shannon, Programming a computer for playing chess, Philosophical Magazine 41 (1950) 256–275.

[22] D. E. Knuth, R. W. Moore, An analysis of alpha-beta pruning, Artificial Intelligence 6 (4) (1975) 293–326.

[23] R. Korf, Real-time heuristic search, Artificial Intelligence 42 (2–3) (1990) 189–211.

[24] B. Abramson, Control strategies for two-player games, ACM Comput. Surv. 21 (2) (1989) 137–161.

[25] R. Piltaver, Search pathology in eight-puzzle (in Slovene) (2008).

[26] B. Kaluža, Analysis of pathological minimax models and Pearl's game (in Slovene) (2008).

[27] M. Guid, I. Bratko, Computer analysis of world chess champions, ICGA Journal 29 (2) (2006) 64–65.

[28] B. Wilson, A. Parker, D. S. Nau, Error minimizing minimax: Avoiding search pathology in game trees, in: International Symposium on Combinatorial Search (SoCS-09), 2009.

[29] H. Murray, A History of Board-Games Other than Chess, Oxford University Press, 2002.