Analysis of a Single-Agent Search

Aleš Tavčar
Department of Intelligent Systems
Jozef Stefan Institute, Ljubljana, Slovenia
E-mail: ales.tavcar@ijs.si

Keywords: minimin, LRTS, pathology, game tree, single-agent search

Received: February 12, 2012

Game playing is one of the first areas of artificial intelligence that was studied by AI researchers. The developed algorithms and heuristics combined with an ever-increasing computer speed efficiently searched large game trees and thus effectively competed with the best human players. The key advantage comes from the generally accepted notion that a deeper search produces better results. However, while trying to provide a mathematical explanation, researchers have discovered that under certain conditions the opposite situation occurs, i.e., a deeper search results in worse decisions. In this paper we analyse single-agent search algorithms and the influence of three properties of the search trees on the quality of the search. The analysis was performed on one-player search models and in the maze-path-finding problem.

Povzetek: Na modelu enoagentnega preiskovanja in iskanja poti v zemljevidu analiziramo vpliv različnih faktorjev na kvaliteto preiskovanja dveh algoritmov.

1 Introduction

A common way of solving search problems is to represent the problem with a directed search graph. The nodes of the graph correspond to states, e.g., the tiles of a map, and the edges are the moves leading from one state to the other. Moreover, the problem definition includes special states: one initial state and one or several goal states. The solution to the problem is a path leading from the initial to the most appropriate goal node.

Smaller problems can be solved by completely searching the state space [6]. Unfortunately, the search trees of most real-world problems are too large to be searched completely in a reasonable amount of time. Korf introduced an incomplete search method called real-time A* [8] that tackles this problem. Real-time A* expands the game tree to a certain depth, and using a heuristic function it evaluates the quality of the nodes at that depth; it then backs the computed values to the current state. The action leading to the node with the optimal value is then selected. Another algorithm is minimin [8]. Figure 1 demonstrates how the minimin algorithm determines the next action.

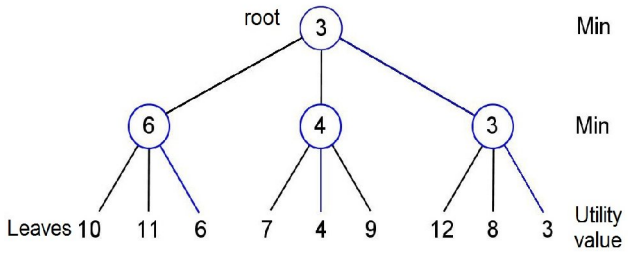

Figure 1: Minimin search.

The nodes of the tree contain utility values, i.e., the quality of a node. For each node the minimum value of its successors is selected and backed-up. Finally, the action leading from the root to the node with the value 3 is selected.

Common sense dictates that a deeper search should provide better decisions, and this can indeed be observed in practice. Evaluating many moves ahead leads to better decisions since obtaining better estimates of the possible strategies can help with selecting the appropriate responses in the current situation. However, while trying to provide a mathematical explanation of this principle, researchers have discovered the opposite: looking ahead does not always provide better decisions. This phenomenon was termed a lookahead pathology [10] and it occurs when the quality of a shallower heuristic search is higher than the quality of a deeper one. This property makes the pathology suitable for measuring the performance of search algorithms at certain depths.

In this paper we study the pathological situations of search algorithms where a deeper search does not guarantee better decisions. On synthetic search-tree models we explore the influence of three important factors on the occurrence of pathological behaviour: the **granularity g** of the heuristic, the **branching factor b** of the game tree and the **similarity s** of the nearby nodes. Next, we perform experimental tests to determine whether these factors have the same influence on the maze-path-finding problem.

This paper is organized as follows. First, we provide a brief literature review of related works in Section 2. In Section 3 we present the game-tree models used during the analysis and we describe the three observed factors. Section 4 explores the influence of these factors in the

real-life domain of maze-path finding. Section 5 concludes this article.

2 Related Work

The pathology measure used in this paper was first discovered independently by Nau [10] and Beal [1] in the minimax algorithm [17]. Since then, extensive research and many explanations of the minimax pathology [1,2,7,10,11,12] have broadened our understanding of the principles behind the occurrence of pathological behaviour. Three factors have been identified that greatly contribute to the pathological behaviour in the minimax algorithm: the independence or low similarity of the sibling nodes, the low granularity of the selected heuristic function and the large branching factors of the search trees.

On the other hand, the pathology of a single-agent search has been much less investigated. It was first discovered in 2003 by Bulitko et al. [3], and it was first demonstrated on a synthetic, two-level search tree. Bulitko only demonstrated that the tree is pathological, but he did not provide an adequate explanation. Luštrek [9] continued the research on synthetic search trees and he provided two key factors influencing the pathology. The first of these are certain domain and search-tree properties, with the most important being the difference between the values of the sibling nodes. This is given by the domain and therefore, cannot be altered. The second of these are the factors related to the heuristic function: he showed that consistent and admissible heuristics prevent the pathology. On the other hand, Sadikov and Bratko [14] determined the opposite: pessimistic heuristic functions perform better in the eight-puzzle game. They analysed the influence of several heuristic functions on the number of non-optimal nodes and the quality of the final solutions at greater search depths. Additionally, Bulitko [5] observed that the type of the heuristic function used affects the number of search-tree nodes expanded at greater depths. Their research was limited to the occurrence of pathological behaviour and no insights into the underlying reasons were given.

Piltaver et al. [13] analysed the influence of various properties of the search tree and the heuristic function on the gain and pathology in the eight-puzzle game. They extended the original puzzle to allow additional diagonal moves and thus obtain a different branching factor for the search trees. Next, they analysed the similarity of the sibling nodes and the effect on the pathology. Even though the similarity is given by the domain and therefore cannot be modified at will, they were able to observe that a greater similarity decreases the degree of pathology. Experimentations with heuristic functions included the granularity of the function, the amount of heuristic error and the type of heuristic function (optimistic, pessimistic and uniform).

Nau et al. [12] presented a unifying view of the pathology in the minimax and single-agent searches. The paper shows the interplay between the lookahead pathology and three factors that affect it: the dependence of the sibling nodes in the search tree, the branching factor and the granularity of the heuristic function. The experiments were performed on synthetic trees, the 8-puzzle, two chess endgames and kalah. The content of their paper is related to our work; however, in this paper additional findings relating to the one-player model, as well as an analysis of the maze-path-finding problem, are presented.

3 One-Player Model

In the following section we investigate the influence of three factors on the quality of the search in synthetic search trees: the **granularity g** of the heuristic function (the number of possible heuristic values returned), the **branching factor b** of the search tree (the number of successors of each node), and the **similarity s** of the search tree (the similarity among values of sibling nodes). First, we start with some basic definitions of the model.

3.1 Definitions

Any one- or two-player game can be represented by a game tree in which the nodes are the states and the edges are moves leading from the current to the next state. The goal is to find a path from the initial node to one of the terminal nodes. The quality of a terminal node x is denoted by the utility function u(x). If the overall objective of the search problem is to maximize the cost of reaching a goal, then the player selects the successor with the highest utility score. To each node of the tree a unique value v(n) can be assigned, i.e., the utility value obtained if the player always selects the action leading to the node with the maximum cost:

$$v(n) = \begin{cases} \mathrm{v(n)} & \text{if } \mathrm{suc(n)} = \{\} \\ \max_{\mathrm{m \in suc(n)}} c(n,m) + v(m), & \text{otherwise,} \end{cases}$$

where suc(n) is the set of n`s successors. The player following this concept would always select the optimal move. Unfortunately, most search trees are too large to compute v(n) in a reasonable time. Therefore, an approximation of v(n) can be computed using the maximax algorithm, which is the same as the minimin algorithm where the objective is to maximize the gain:

$$v_h(n,d) = \begin{cases} v(n), & \textit{if } suc(n) = \{\} \\ h(n), & \textit{if } d = 0, \\ \max_{m \in suc(n)} c(n,m) + v_h(m, d-1), & \textit{otherwise} \end{cases}$$

where *h(n)* is a *heuristic evaluation function* used to compute an approximation of *v(n)* at a certain *search depth d.* The computed value $v_h(n,d)$is called a heuristic value, since it is an approximation obtained using a heuristic function. Since the heuristic function is not completely accurate, an error is introduced in the estimation of the utility value. In practice, maximax performs well; therefore, while backing-up values to the root of the search tree it should reduce the error introduced with the heuristic function. Pathology occurs when maximax amplifies the error of the heuristic function. In this paper we will focus on the *decision*

error, which is the probability of selecting an action that does not lead to a successor with the optimal utility value. The decision error at a certain search depth d and heuristic function h is denoted as *E(h,d)*.

Next, we define the *pathology p* as:

$$p(h,d_1,d_2)=\frac{E(h,d_2)}{E(h,d_1)},1\le d_1<d_2\le d_{\max}$$

Pathological behaviour occurs when an error at a deeper level is greater than an error at a shallower level:

$$\exists d_1<d_2:\left[E(h,d_1)<E(h,d_2)\right]$$

When this happens the values of $p(h,d_1,d_2) > 1$ indicate that the search algorithm performs poorly, while $p(h,d_1,d_2) < 1$ indicates the absence of pathology, meaning that searching deeper is worthwhile. In order to evaluate whether a problem is pathological we have run several experiments with the Monte Carlo method and compared the averaged *p* with 1.

3.2 Independent game-tree model

This section describes the experiments with synthetic, independent game trees. Namely, we constructed game trees where the sibling nodes are selected randomly and are thus independent of each other.

We start building the game-tree models by assigning utility values to the terminal nodes from a uniform distribution over the interval [0,1]. These values are real; however, we wanted to examine the influence that the number of possible heuristic values has on the search gains. This is called the granularity of the heuristic function. In order to adapt the heuristic function to the desired granularity g, the original values are grouped into g intervals of equal size. Each interval contains values from a certain range from the interval [0,1] and each interval is represented by the mean value of the lower and upper interval bounds. In the one-player model, at small granularities there is a tendency for the values towards the root to converge to a single value: the maximum utility. Due to the nature of the backing-up procedure this phenomenon occurs as soon as every sub tree of the root node contains at least one terminal node with the maximum utility. We solve this by limiting the probability that a maximum utility is reached at the root to at most 50%. In the majority of cases, only half of the values at the root will be the maximum utility. A direct descendant of the root *x* does not have the maximum utility, except when none of the terminal nodes in this sub tree have the maximum utility. Therefore, we have introduced the following equations that describe the relation between the probability of a direct successor of the root having the maximum utility ($P_{max}(x)$) and the probability of a terminal node having the maximum utility ($P_{max}(t)$):

$$1-P_{\max}(x)=(1-P_{\max}(t))^{b^{\Delta d}},$$

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

where *b* is the branching factor of the tree and Δd is the depth from the node x to the terminal node t. The second equation provides the size of the last interval containing the largest utilities. The boundaries of all the intervals are shifted until the lower boundary of the last interval reaches $1-P_{max}(t)$.

Next, the values from the terminal nodes are propagated throughout the tree using the maximax principle.

3.3 Dependent model

The next task was to introduce a local similarity or a dependence of the sibling nodes into the one-player model.

A search tree has a high dependence or similarity of the sibling nodes when the utility values of the descendants of a node have similar values. In this case, a deeper search is beneficial when the similarity of the sibling nodes increases. The opposite also holds true: the deeper search is becoming less efficient when the similarity of the sibling nodes decreases.

In our model, the dependence is expressed as a parameter $s \in [0,1]$, where $s = 0$ denotes independence between the sibling nodes and $s = 1$ corresponds to the maximum similarity between the sibling nodes. Models with similarity $s = 1$ are constructed by generating b^h random numbers in the interval [0,1] and the values are mapped into granular classes to obtain the desired granularity. Next, the mapped numbers are sorted in increasing order and assigned to the terminal nodes from left to right. The lowest value is assigned to the left-most node, continuing through the inner nodes and finishing with the highest number assigned to the right-most node.

Intermediate similarities for the models ($0 < s < 1$) are created by randomly mixing the terminal nodes from the independent and completely dependent trees. All the sibling terminal nodes in the independent tree are replaced with terminal nodes from the dependent tree with the probability *s*.

The introduced measure of similarity is useful when we are dealing with game-tree models; however, local similarity in an arbitrary game is expressed in a different way. If we want to compare results from the models and the game trees we need a common similarity measure. For this purpose the *clustering factor f* was used. Roughly, it is the ratio between the standard deviation of the utility values of the sibling nodes averaged across all the possible parent nodes, and the standard deviation of the utilities in the entire search tree [15]. When *f* is low, then the sibling nodes are similar, and vice versa. We calculated the clustering factor for every similarity *s*.

3.4 Evaluation

This section describes experiments with the one-player models. The performance of the maximax algorithm was estimated by observing how the degree of pathology is influenced by three factors: the branching factor *b*, the granularity *g*, and the similarity *s*. For each combination of values for the three observed factors 10,000 synthetic trees were generated using the one-player model. The following values were systematically used for each of the

three factors: b = 2,3,…,10; g = 2,3,…,300; and s = 0.0,0.1,…, 1.0.

In the first experiment we observed how the quality of the search is influenced by the branching factor and the granularity in independent trees. Figure 2 shows the pathology p in relation to the granularity g and the similarity s in the results of the experiment. On one hand the search is inefficient at lower granularities and as the branching factor increases. On the other hand, a deeper search becomes worthwhile with an increased granularity and at lower branching factors.

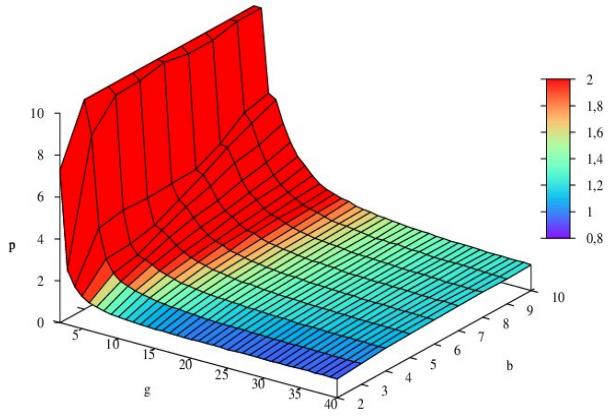

Figure 2: The degree of pathology for independent search trees.

In the next experiment we computed the required granularity to avoid situations where a deeper search is inefficient, as a function of the branching factor b and the local similarity s. The surface in Figure 3 represents the border between pathological and non-pathological behaviour where p = 1. The area below the surface is pathological, while the area above the surface is non-pathological. The results are as expected: higher similarity decreases the granularity needed to avoid search inefficiency, while a higher branching factor increases the needed granularity.

The unified influence of the branching factor, the local similarity and the granularity on the quality of the search is presented in Figure 4. Again, the pathology was used to estimate the performance of the maximax algorithm. Darker dots denote a slightly weak performance ($1<=p<1.5$) and an extremely weak performance ($p>1.5$), respectively. The dots in light grey denote the parameter settings where a deeper search is worthwhile.

The observed relations among the factors are the same as presented in the previous figures. The degree of pathology decreases with the increased granularity g and the local similarity s and increases with the branching factor b. Moreover, one can see that the similarity of the sibling nodes affects the degree of pathology the most. In a high-similarity search tree the influence of the granularity and the branching factor on the pathology is diminished. This can be clearly observed when comparing independent and completely dependent trees. The pathological behaviour is present for most granularities and branching factors in the independent trees, while the pathology is mostly eliminated in the completely dependent trees. In addition, the degree of pathology is not noticeably increasing with higher branching factors. Another effect that can be observed in Figure 4 is that large values of b decrease the difference in the degree of pathology between values near the pathological and non-pathological border. For example, at lower branching factors, values transit from a high pathology degree (p >= 1.5) to a low pathology degree (p < 0.5) relatively quickly.

The next experiment investigated how well the synthetic trees model the 8-puzzle, a real-world game.

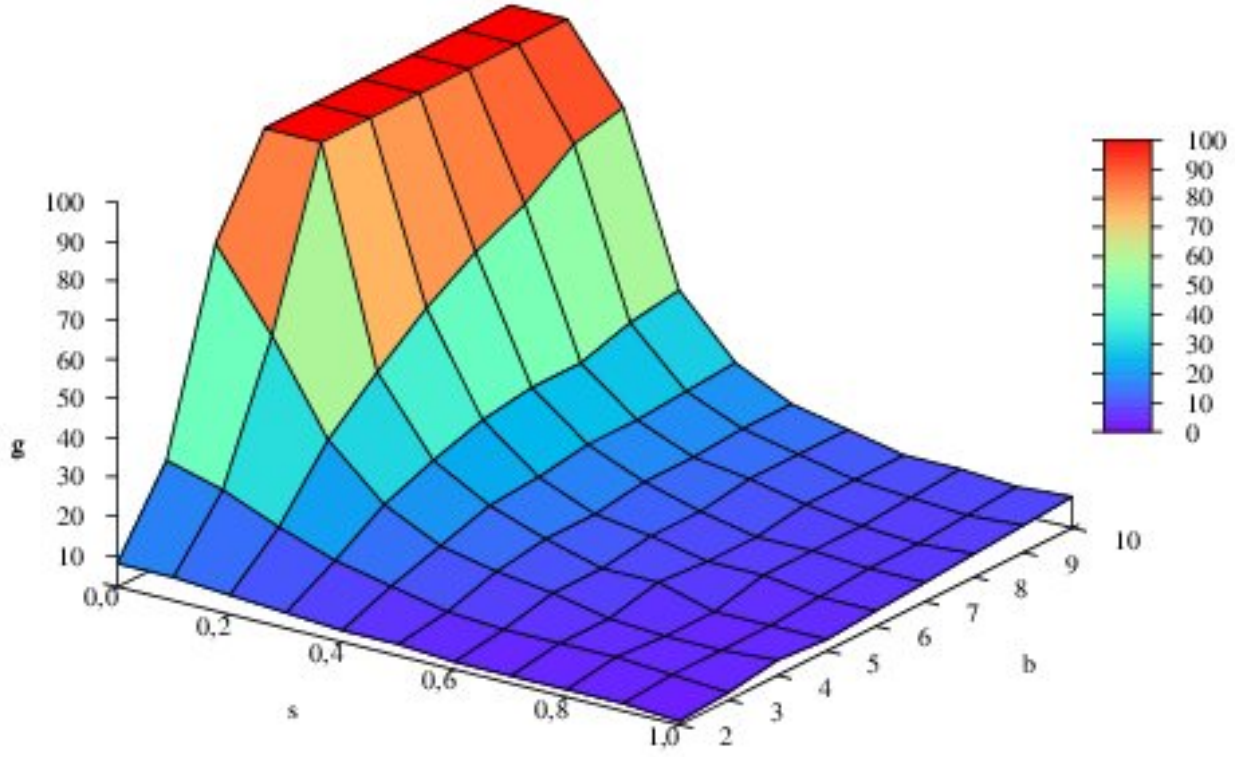

Figure 3: Granularity values needed to avoid pathology in a deeper search.

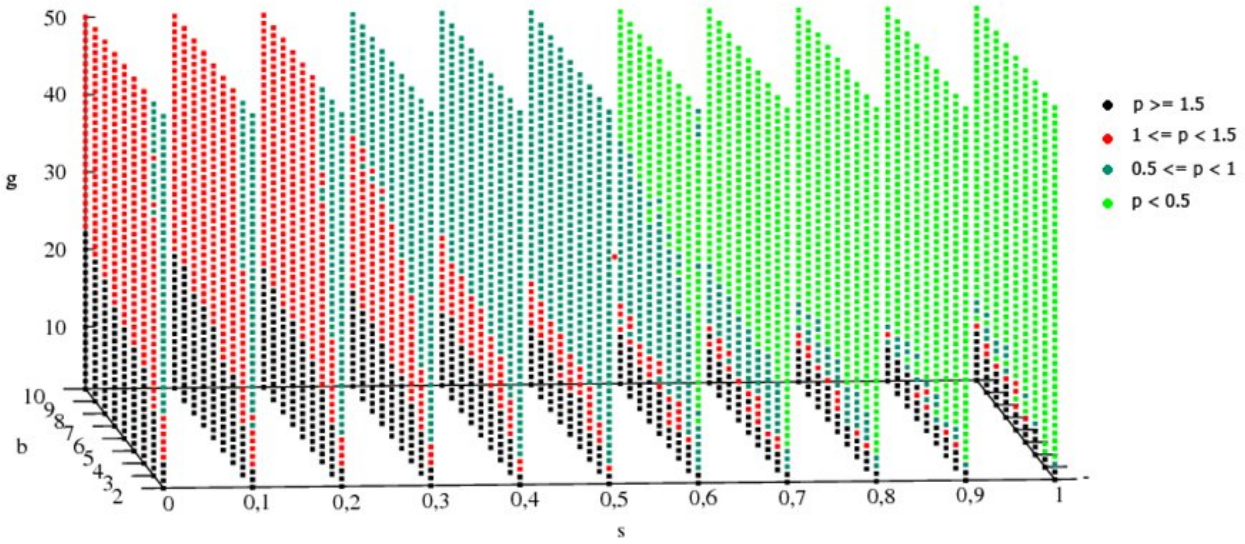

Figure 4: Degree of pathology in the minimin search model.

From [13] we obtained the degree of pathology for two heuristic functions, with a branching factor of 2 and a similarity of 0.4. The similarity in the 8-puzzle was measured with the clustering factor; therefore, we had to map the value of the clustering factor to the similarity used in the models as described in Subsection 3.3. The comparison between the model and the 8-puzzle is shown in Figure 5. Qualitatively, the performance of the model is similar to the 8-puzzle. Quantitatively, the minimax algorithm from the model performs worse at lower granularities ($g < 20$), but performs very similarly at higher granularities. The granularity needed to avoid inefficiency in a deeper search is higher in the model than in the 8-puzzle. The reason that the synthetic model performs worse than the Manhattan distance is probably related to the specific properties of the function, e.g., the Manhattan function is exact for distances less than eight moves to the finish.

4 Pathology in the Maze Search

This section presents experimental tests on a real-world problem, i.e., maze path finding, and an analysis of the Learning Real-Time Search (LRTS) algorithm [4] The algorithm conducts a lookahead search centred on the current state and generates all the states up to d moves away from the current state.

The analysis of LRTS consisted of determining if the algorithm selects the optimal first move from the initial node. First, we computed the shortest path from the initial to the goal node of the maze to determine the correct move from the initial node. Next, we run one basic step of the LRTS algorithm with different lookahead depths (d=1 and d=5). Finally, we compared the move computed with the two LRTS runs with the computed optimal move. If the move selected with LRTS differs from the optimal then this is a decision error for a certain lookahead depth.

As in the single-player model, the measure of the search quality is the pathology p. In addition we compare the performance of the LRTS algorithm and the minimin algorithm used in the model.

Using the Randomized Kruskal's algorithm we created mazes of different sizes and structures. In the generated mazes it is possible to vary only the granularity g of the utility and the heuristic function since in real-world problems the branching factor b and the local similarity s are part of the problem. However, we were still able to generate mazes with different local similarities and calculate the corresponding degree of pathology. In Figure 6 it is clear that the degree of pathology decreases with the similarity of the sibling nodes. The similarity is measured with the clustering factor; therefore, a smaller clustering factor means a larger similarity. The similarities in the figure range from 0.03 to 0.67, i.e., game trees that are almost completely dependent to games trees with a smaller dependence. The same cannot be achieved for the branching factor since the average branching factor in an arbitrary maze is 2.

The experiments thus analyse the influence of the amount of granularity and type of heuristic function on the performance of the LRTS algorithm. The first step of the experiments is to compute the utilities for all the positions of the maze. The utility of a given position is equal to the minimum number of moves needed to reach

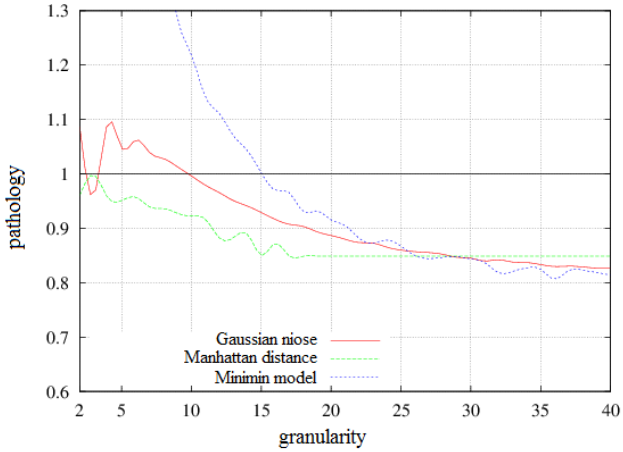

Figure 5: Search efficiency of three single-agent search problems.

the exit of the maze from that position. Using the retrograde analysis [16] we start at the goal position and while moving towards the start we assign to each position the number of moves currently traversed. Next, the heuristic values are computed from the utility values using the *generic heuristic evaluation function h*. The error of the heuristic function is the difference between the utility and the heuristic values and is modelled using Gaussian noise. The advantage of such an approach is the ability to adjust the noise distribution and thus approximate any heuristic function used in practice. In our experiments the heuristic values were computed by adding the Gaussian noise with a standard deviation σ = 2.5 to the utility values. The selected standard deviation is the same as the standard deviation of the Manhattan-distance heuristic function.

The desired granularity g of the utility and the heuristic values is obtained by creating g intervals of equal size. We start by limiting the heuristic values to the [0,N] interval, where $N = \max_n\{h^*(n)\}+\lfloor\sigma\rfloor+1$. If any values are lower than 0, it is set to 0, and if it is greater than N, it is set to N. Next, all the heuristic values are multiplied by g/N to further reduce the interval of possible heuristic values to [0,g].

The second property of the heuristic function that we investigated is how the heuristic error is applied to the utility values. We considered the influence of three heuristic functions: balanced, pessimistic and optimistic heuristic functions. Optimistic heuristic functions always underestimate the utility value, i.e., the heuristic estimate is smaller than the utility value. On the other hand, pessimistic heuristic functions always overestimate the utility value, and balanced heuristic functions are a combination of both. Optimistic and pessimistic heuristic functions were obtained by subtracting or adding the absolute value of the Gaussian noise to the utility values.

Pessimistic functions were found to be less prone to a lookahead pathology [14]; therefore, the use of this heuristic function should improve the performance of the LRTS algorithm.

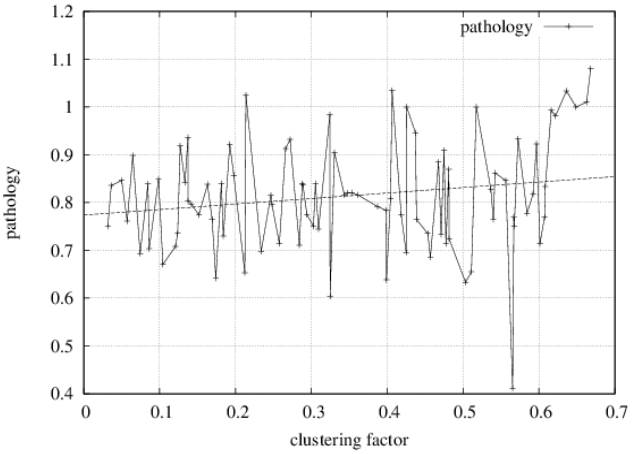

Figure 6: The influence of the similarity on the degree of pathology.

The last step is to analyse the influence of the selected factors on the performance of the algorithm. Using Monte Carlo simulations we generated 10,000 mazes for different granularities and heuristic functions. The results of the analysis are presented in Figure 7. The figure shows how the three heuristic functions and the granularity contribute to the LRTS algorithm`s quality of search. For all three functions it is clear that an increased granularity increases the quality of the search, i.e., decreases the degree of pathology. Moreover, the obtained results confirm that pessimistic heuristic functions perform better than optimistic ones. At granularities higher than 5, a deeper search is beneficial and the degree of pathology stabilizes around 0.75 at g>13. The performance of the balanced, generic, heuristic function is worse than that of the pessimistic

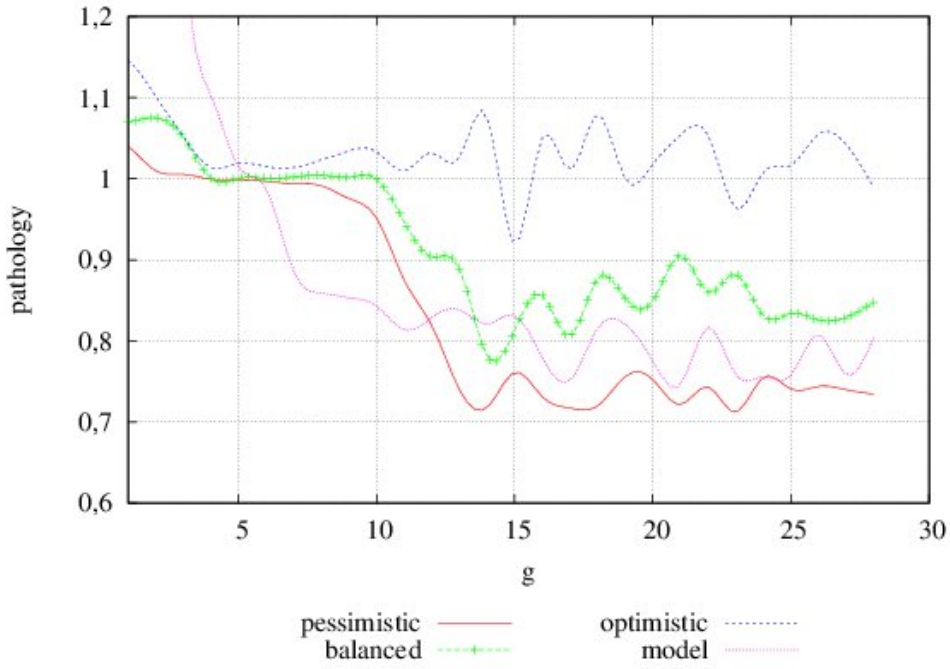

Figure 7: Performance of the three heuristic functions and the reference model.

heuristic function, but better than that of the optimistic heuristic function. For comparison, we added the performance of the minimin algorithm from the one-player model with the appropriate local similarity. The model behaves in a similar way to the LRTS algorithm with the pessimistic heuristic function.

5 Discussion

In this paper we analysed the performance of two one-player search algorithms, i.e., maximax and LRTS, under particularly demanding conditions. The analysis was performed on a model for one-player games and in the maze-path-finding problem. Three factors were considered during the analysis of the quality of the search for both algorithms: the granularity of the heuristic function, the local similarity of the sibling nodes and the branching factor of the game tree. We were unable to study the influence of the last factor in the maze domain since the branching factor is usually fixed in real-world domains.

Based on the experimental results several observations can be made. First, an increased granularity of the heuristic function increases the search gain of both algorithms. Second, in the one-player model an increased branching factor of the game tree reduces the search gain. Third, pessimistic heuristic functions have higher search gains than optimistic ones in the maze domain. Finally, the similarity of the sibling nodes is the single most important factor to improve the search gain: a higher similarity improves the search results.

We confirmed that under the specific, demanding conditions, both search algorithms perform poorly: a deeper search in these conditions produces worse results than a shallow search. The results are consistent with recent publications, but a certain amount of novelty is demonstrated by the analysis of the search-path finding and analyses of the LRTS algorithm.

References

[1] Beal, D. F. (1980). *An Analysis of Minimax*. In Advances in Computer Chess 2, M.R.B. Clarke (Ed.), pp. 103-109.

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

[3] Bulitko, V. (2003). *Lookahead Pathologies and Meta-level Control in Real-time Heuristic Search*. Proceedings of 15th Euromicro Conference on Real-Time Systems, Porto, Portugal, pp. 13-16.

[4] Bulitko, V., Lee, G. (2006). *Learning in real time search:A unifying framework*. *JAIR* 25:119–157.

[5] Bulitko, V., Li, L., Greiner, R. and Levner, I. (2003). *Lookahead Pathologies for Single Agent Search*. Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico, pp. 1531-1533.

[6] Hart, P.E., Nilsson, N.J. and Raphael, B. (1968). *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*. IEEE Transactions on Systems Science and Cybernetics 4(2), pp. 100-107.

[7] Kaluža, B., Luštrek, M., Gams, M., Tavčar, A. (2007). *Pathology in Minimax Searching.* Proceedings of the 16th International Electrotechnical and Computer Science Conference, pp. 111-114.

[8] Korf, R. E. (1990). *Real-Time Heuristic Search*. Artificial Intelligence 42(2, 3), pp. 189-211.

[9] Luštrek, M. (2005). *Pathology in Single-agent Search.* Proceedings of the 8th International Multiconference Information Society, Slovenia, Ljubljana, pp. 345-348.

[10] Luštrek, M., Gams, M., Bratko, I. (2006). *Is real-valued minimax pathological*. Artifcial Intelligence 170 (6-7), pp. 620-642.

[11] Nau, D. S. (1979). *Quality of Decision versus Depth of Search on Game Trees*. Ph. D. thesis, Duke University.

[12] Nau, D. S., Luštrek, M., Parker, A., Bratko, I., and Gams, M. (2010). *When is it Better not to Look Ahead?*, Artificial Intelligence, 174, pp. 1323-1338.

[13] Piltaver, R., Luštrek, M., Gams, M. *The pathology of heuristic search in the 8-puzzle.* Journal of Experimental & Theoretical Artificial Intelligence, DOI:10.1080/0952813X.2010.545997.

[14] Sadikov, A., Bratko, I. (2006). *Pessimistic Heuristics Beat Optimistic Ones in Real-time Search.* Proceedings of the European Conference on Artificial Intelligence, Italy, Riva del Garda, pp. 148-152.

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

[16] Thompson, K. (1986). *Retrograde Analysis of Certain Endgames*. ICCA Journal,3, pp. 131-139.

[17] Von Neumann, J. (1928). *Zur Theorie der Gesellschaftsspiele*. Mathematische Annalen, 100, pp. 295-320.