

Lookahead Pathology in Real-Time Path-Finding

Mitja Luštrek

Jožef Stefan Institute
Department of Intelligent Systems
Jamova 39, 1000 Ljubljana, Slovenia
mitja.lustrek@ijs.si

Vadim Bulitko

University of Alberta
Department of Computing Science
Edmonton, Alberta, Canada T6G 2E8
bulitko@ualberta.ca

Abstract

Large real-time search problems such as path-finding in computer games and robotics limit the applicability of complete search methods such as A*. As a result, real-time heuristic methods are becoming more wide-spread in practice. These algorithms typically conduct a limited-depth lookahead search and evaluate the states at the frontier using a heuristic. Actions selected by such methods can be suboptimal due to the incompleteness of their search and inaccuracies in the heuristic. Lookahead pathologies occur when a deeper search decreases the chances of selecting a better action. Over the last two decades research on lookahead pathologies has focused on minimax search and small synthetic examples in single-agent search. As real-time search methods gain ground in applications, the importance of understanding and remedying lookahead pathologies increases. This paper, for the first time, conducts a large scale investigation of lookahead pathologies in the domain of real-time path-finding. We use maps from commercial computer games to show that deeper search often not only consumes additional in-game CPU cycles but also decreases path quality. As a second contribution, we suggest three explanations for such pathologies and support them empirically. Finally, we propose a remedy to lookahead pathologies via a method for dynamic lookahead depth selection. This method substantially improves on-line performance and, as an added benefit, spares the user from having to tune a control parameter.

Introduction

Path-finding tasks commonly require real-time response, which on large problems precludes the use of complete search methods such as A*. Imagine a real-time strategy computer game. The user commands dozens or even hundreds of units to move simultaneously towards a distant goal. If each of the units were to compute the whole path to the goal before moving, this could easily incur a noticeable (and annoying) delay. Incomplete single-agent search methods (Korf 1990; Shue *et al.* 2001; Bulitko *et al.* 2005; Hernández and Meseguer 2005; Bulitko and Lee 2006) work similarly to minimax-based algorithms used in two-player games. They conduct a limited-depth lookahead search, i.e., expand a part of the space centered on the agent, and heuristically evaluate the distances from the frontier of the ex-

panded space to the goal. By interleaving search and movement, agents using these algorithms can guarantee a constant upper-bound on the amount of search per move and compare favorably to complete methods (Koenig 2004).

Actions selected based on heuristic lookahead search are not necessarily optimal, but it is generally believed that deeper lookahead increases the quality of decisions. It has long been known that this is not always the case in two-player games (Nau 1979; Beal 1980). This phenomenon has been termed the minimax pathology. More recently pathological behavior was discovered in single-agent search as well (Bulitko *et al.* 2003; Bulitko 2003).

In this paper we investigate lookahead pathologies in real-time pathfinding on maps from commercial computer games. Computer games have a limited CPU budget for AI and current path-finding algorithms can take as much as 60-70% of it (Pottinger 2000). Thus, it is crucial that the CPU cycles be used efficiently and more expensive, deeper lookahead not be conducted when it is not beneficial or, worse yet, when it is harmful.

This paper makes three contributions. First, it presents a large-scale empirical study of lookahead pathologies in path-finding on commercial game maps. In this study, pathologies were found in *more than half* of the problems considered. Second, it shows that both learning and search are responsible for such wide-spread pathological behavior. Our explanations for the pathology are presented formally, motivated intuitively, and supported by empirical evidence. Third, it proposes an automated approach to dynamic lookahead selection. This novel technique takes advantage of automatically derived state abstraction and works with any lookahead-based single-agent search method. Its initial implementation in real-time path-finding demonstrates a promise in remedying pathologies.

Related Work

Most game-playing programs successfully use minimax to back-up the heuristic values from the leaves of the game tree to the root. Early attempts to explain why backed-up values are more reliable than static values, however, led to a surprising discovery: minimaxing actually amplifies the error in the heuristic evaluations (Nau 1979; Beal 1980). Several explanations for such pathological behavior were proposed, the most common being that de-

pendence between nearby positions is what eliminates the pathology in real games (Bratko and Gams 1982; Nau 1982; Pearl 1983; Scheucher and Kaindl 1998). Recently Luštrek *et al.* (2005) showed that modeling the error realistically might be enough to eliminate it. Common to all the work on the minimax pathology is that it tries to explain why the pathology appears in theoretical analyses but not in practice, whereas in path-finding, the pathology is a practical problem.

The pathology in single-agent search was discovered by Bulitko *et al.* (2003) and was demonstrated on a small synthetic search tree. Later it was observed in solving the eight puzzle (Bulitko 2003). The same paper also showed the promise of dynamic lookahead depth selection, but did not propose a mechanism for doing so. An attempt to explain what is causing the pathology on synthetic search trees was made by Luštrek (2005), but it does not translate well to path-finding and practically used heuristics. To the best of our knowledge, this paper presents the first study of lookahead pathology in path-finding.

Problem Formulation

In this paper we are dealing with the problem of an agent trying to find a path from a start state to a goal state in a two-dimensional grid world. The agent’s state is defined by its location in one of the grid’s squares. Some squares are blocked by walls and cannot be entered. The traversable squares form the set of states S the agent can enter; $s_g \in S$ is the goal state. From each square the agent can move to the eight immediate neighbor squares. The *travel cost* of each of the four straight moves (north, west, south and east) is 1 and the travel cost of the diagonal moves is $\sqrt{2}$. A search problem is defined by a map, a start state, and a goal state.

The agent plans its path using the Learning Real-Time Search (LRTS) algorithm (Bulitko and Lee 2006) configured so that it works similarly to the classic LRTA* (Korf 1990). LRTS conducts a lookahead search centered on the current state $s_c \in S$ and generates all the states within its lookahead area (i.e., up to d moves away from s_c). The term generate refers to ‘looking at’ a state, as opposed to physically visiting it. Next, LRTS evaluates the states at the frontier of the lookahead area using the heuristic function h , which estimates the length of shortest path from the frontier states to s_g . The agent then moves along the shortest path to the most promising frontier state $s_{f_{opt}} \in S$. State $s_{f_{opt}}$ is the state on the path to s_g with the lowest projected travel cost $f(s_{f_{opt}}) = g(s_{f_{opt}}) + h(s_{f_{opt}})$, where $g(s_{f_{opt}})$ is the travel cost from s_c to $s_{f_{opt}}$ and $h(s_{f_{opt}})$ the estimated travel cost from $s_{f_{opt}}$ to s_g . When the agent reaches $s_{f_{opt}}$, another lookahead search is performed. This process continues until s_g is reached. The initial heuristic is the *octile distance* to s_g , i.e., the actual shortest distance assuming a map without obstacles (the term octile refers to the eight squares or tiles the agent can move to). After each lookahead search, $h(s_c)$ is updated to $f(s_{f_{opt}})$ if the latter is higher, which raises the heuristic in the areas where it was initially too optimistic, making it possible for the agent to eventually find the way around the obstacles even when the lookahead radius is too

small to see it directly. This constitutes the process of learning. The updated heuristic is still guaranteed to be admissible (i.e., non-overestimating), although unlike the initial one, it may not be consistent (i.e., the difference in heuristic values of two states may exceed the shortest distance between them).

Definition 1 For any state $s \in S$, the *length of the solution* (i.e., the travel cost of the path from s to s_g) produced by LRTS with the lookahead depth d is denoted by $l(s, d)$.

Definition 2 For any state $s \in S$, the *solution-length vector* $\vec{l}(s)$ is $(l(s, 1), l(s, 2), \dots, l(s, d_{\max}))$, where d_{\max} is the maximum lookahead depth.

Definition 3 The degree of *solution-length pathology* of a problem characterized by the start state s_0 is k iff $\vec{l}(s_0)$ has exactly k increases in it, i.e., $l(s_0, i + 1) > l(s_0, i)$ for k different i .

The length of the solution an agent finds seems to be the most natural way to represent the performance of a search algorithm. However, we will require a metric that can also be used on a set of states that do not form a path.

Definition 4 In any state $s \in S'$ where $S' \subseteq S$ is the set of states visited by an agent while pursuing a given problem, the agent can take an optimal or a suboptimal action. An optimal action moves the agent into a state lying on a lowest-cost path from s to s_g . The *error* $e(S')$ is the fraction of suboptimal actions taken in the set of states S' .

Definition 5 In any set of states $S' \subseteq S$, the *error vector* $\vec{e}(S')$ is $(e(S', 1), e(S', 2), \dots, e(S', d_{\max}))$, where d_{\max} is the maximum lookahead depth.

Definition 6 The degree of *error pathology* of a problem characterized by the set of states S' is k iff $\vec{e}(S')$ has exactly k increases in it, i.e., $e(S', i + 1) > e(S', i)$ for k different i . We can also speak of the ‘‘pathologicalness’’ of a single state ($|S'| = 1$), but in such a case the notion of the degree of pathology makes little sense. A state $s \in S$ is pathological iff there are lookahead depths i and $i + 1$ such that the action selected in s using lookahead depth i is optimal and the action selected using depth $i + 1$ is not.

Finally, we will need ways to measure the amount of work done by the LRTS algorithm.

Definition 7 $\alpha(d)$ is the average number of states generated per single move during lookahead search with depth d .

Definition 8 $\beta(d)$ is the average volume of updates to the heuristic encountered per state generated during lookahead search with depth d . The volume of updates is the difference between the updated and the initial heuristic.

Pathology Experimentally Observed

We chose in-game path-finding as a practically important task with severe constraints on CPU time. Five maps from a role-playing and a real-time strategy game were loaded into Hierarchical Open Graph, an open-source research testbed provided by Bulitko *et al.* (2005). The map sizes ranged from 214×192 (2,765 reachable states) to 235×204 (16,142

reachable states). On these maps, we randomly generated 1,000 problems with the distance between the start and goal state between 1 and 100.

We conducted two types of experiments: on-policy and off-policy. In an on-policy experiment, the agent follows the behavioral policy dictated by the LRTS algorithm with lookahead depth d from a given start state to the goal state, updating the heuristic along the way. We vary d from 1 to 10. The length of the path and the error over the entire path are measured for each d , giving the solution-length and the error vectors from which the pathology is computed. In an off-policy experiment, the agent spontaneously appears in individual states, in each of which it selects the first move towards the goal state using the LRTS algorithm with lookahead depth d . The heuristic is not updated. Again, d ranges from 1 to 10. The same 1,000 problems were used in both types of experiments, but in the off-policy experiments, the start states were ignored. The error over all states is measured for each d and the error pathology is computed from the resulting error vector. Since it is not possible to measure solution-length pathology in off-policy experiments, only error pathology is considered in this paper (with the exception of the first experiment).

First we conducted the basic on-policy experiment. The results in Table 1 show a that *over 60%* of problems are pathological to some degree. ‘Degree’ in the table means the degree of pathology (0 indicates no pathology), ‘Length %’ means the percentage of problems with a given degree of length pathology and ‘Error %’ means the same for error pathology. The notation in the rest of the tables in this and the following section is similar.

Table 1: Percentages of pathological problems in the basic on-policy experiment.

Degree	0	1	2	3	4	≥ 5
Length %	38.1	12.8	18.2	16.1	9.5	5.3
Error %	38.5	15.1	20.3	17.0	7.6	1.5

The first possible explanation of the results in Table 1 is that the maps contain a lot of states where deeper lookaheads lead to suboptimal decisions, whereas shallower ones do not. This turned out *not* to be the case: for each of the problems we measured the “pathologicalness” of every state on that map in the off-policy mode. Surprisingly, there were only 3.9% of pathological states. This is disconcerting: if the nature of the path-finding problems is not pathological, yet there is a lot of pathology in our solutions, then perhaps the search algorithm is to blame.

Comparing the percentage of pathological states to the results in Table 1 is not entirely meaningful, because in the first case the error is considered per state and in the second case it is averaged over all the states visited per problem. The average number of states visited per problem during the basic on-policy experiment is 188. To approximate the setting of the on-policy experiment, we randomly selected the same number of states per problem in the off-policy mode and computed the error over all of them. Pathology measurements from the resulting error vector are shown in Table

2.

Table 2: Percentages of pathological problems in the off-policy experiment with 188 states per problem.

Degree	0	1	2	3	≥ 4
Problems %	57.8	31.4	9.4	1.4	0.0

This correction reduces the discrepancy between the presence of pathology in on-policy and off-policy experiments from 61.5% vs. 3.9% to 61.5% vs. 42.2%.

The fact that only 3.9% of states are pathological indicates that the pathology observed in the off-policy experiment with 188 states per problem may be due to very minor differences in error. This interferes with the study of pathology since the results may be due to random noise instead of systematic degradation of lookahead search performance when lookahead depth increases. To remove such interference, we extend Definition 6 with noise tolerance t :

Definition 9 The degree of error pathology of a problem characterized by the set of states S' is k iff $\vec{e}(S')$ has exactly k t -increases in it, where a t -increase means that $e(S', i + 1) > t \cdot e(S', i)$.

The new definition with $t = 1.09$ will be used in lieu of Definition 6 henceforth. The particular value of t was chosen so that fewer than 5% of problems are pathological in the off-policy experiment with 188 states per problem. Table 3 compares the on-policy and off-policy experiments with the new definition.

Table 3: Percentage of pathological problems in the basic on-policy and off-policy experiments with $t = 1.09$.

Degree	0	1	2	3	4	≥ 5
On-policy %	42.3	19.7	21.2	12.9	3.6	0.3
Off-policy %	95.7	3.7	0.6	0.0	0.0	0.0

The data indicate that the noise tolerance hardly reduces the pathology in the on-policy experiment (compare to Table 1). This suggests that map topology by itself is a rather minor factor in on-policy pathologies. In the rest of the paper, we will investigate the real causes for the difference in the amount of on-policy and off-policy pathologies. We refer to data in Table 3 as the basic on-policy and the basic off-policy experiments henceforth.

Mechanisms of Pathology

The simplest explanation for the pathology in the basic on-policy experiment is as follows:

Hypothesis 1 The LRTS algorithm’s behavioral policy tends to focus the search on pathological states.

This hypothesis can be verified by computing off-policy pathology from the error in the states visited during the basic on-policy experiment instead of randomly chosen 188 states. This experiment differs from the basic on-policy experiment in that the error is measured in the same states at all lookahead depths (in on-policy experiments, different states

may be visited at depths depths) and there is no learning. The results in Table 4 do show a larger percentage of pathological problems than the off-policy row of Table 3 (6.3% vs. 4.3%), but a much smaller one than the on-policy row of the same table (6.3% vs. 57.7%). So while the percentage of pathological states visited on-policy is somewhat above average, this cannot account for the large frequency (57.7%) of pathological problems in the basic on-policy experiment.

Table 4: Percentages of pathological problems measured off-policy in the states visited while on-policy.

Degree	0	1	2	3	≥ 4
Problems %	93.6	5.3	0.9	0.2	0.0

Notice that the basic on-policy experiment involves learning (i.e., updating the heuristic function) while the basic off-policy experiment does not. The agent performs a search with lookahead depth d every d moves. If there were no obstacles on the map, the agent would move in a straight line and would encounter exactly one updated state during each search (the one updated during the previous search). Each search generates $(2d + 1)^2$ distinct states, so $1/(2d + 1)^2$ of them would have been updated: a fraction that is larger for smaller d . We can now formulate:

Hypothesis 2 Smaller lookahead depths benefit more from the updates to the heuristic. This can be expected to make their decisions better than the mere depth would suggest and thus reduce the difference in error between small and large lookahead depths. If the error is reduced, cases where a deeper lookahead actually performs worse than a shallower lookahead should be more common.

A first test of Hypothesis 2 is to perform an on-policy experiment where the agent is still directed by the LRTS algorithm that uses learning (as without learning it would often get caught in an infinite loop), but the measurement of the error is performed using only the initial, non-updated heuristic. To do this, two moves are selected in each state: one that uses learning, which the agent actually takes, and another that ignores learning, which is used for error measurement. The results in Table 5 strongly suggest that learning is indeed responsible for the pathology, because the pathology in the new experiment is much smaller than in the basic on-policy experiment shown in Table 3 (20.2% vs. 57.7%). This is not the only reason, because the pathology is still larger than in the basic off-policy experiment also shown in Table 3 (20.2% vs. 4.3%), but it warrants further investigation; we will return to the other reason later.

Table 5: Percentages of pathological problems in an on-policy experiment with error measured without learning.

Degree	0	1	2	3	4	≥ 5
Problems %	79.8	14.2	4.5	1.2	0.3	0.0

During the basic off-policy experiment, all lookahead depths are on an equal footing with respect to learning as there are no updates to the heuristic function. Since learning is inevitable during on-policy experiments, the best one

can do to put all lookahead depths on an equal footing during an on-policy experiment is to have all states within the lookahead radius updated as equally as possible. We implement such uniform learning as follows. Let s_c be the current state, S_i the set of all the interior nodes of the lookahead area and S_f the frontier of the lookahead area. For every inner state $s \in S_i$ an LRTS search originating in s and extending to the frontier S_f is performed. The heuristic value of s is then updated to travel cost of the shortest path to the goal found during the search, just like the heuristic value of s_c is updated with the regular update method. The results for an on-policy experiment with uniform learning are found in Table 6.

Table 6: Percentages of pathological problems in an on-policy experiment with uniform learning.

Degree	0	1	2	3	4	≥ 5
Problems %	40.9	20.2	22.1	12.3	4.2	0.3

The results in Table 6 seem to contradict Hypothesis 2, since they are even more pathological than in the basic on-policy experiment shown in Table 3 (59.1% vs. 57.7%). Figure 1 explains the apparent contradiction: even though more states are updated with uniform learning than with the regular learning, the volume of updates encountered decreases more steeply, which means that the difference in the impact of learning between lookahead depths is more pronounced with uniform learning. This may happen because the uniform learning helps the agent find the goal state more quickly: the length of the solution averaged over all the problems is 2.3 to 3.3 times shorter than in the basic on-policy experiment. Shorter solutions mean that the agent returns to the areas already visited (where the heuristic is updated) less often. What is common to both on-policy experiments in Figure 1 is that unlike in the basic off-policy experiment, the volume of updates encountered does decrease with increased lookahead depth. This supports Hypothesis 2, even though the reasoning that led to it may not be entirely correct.

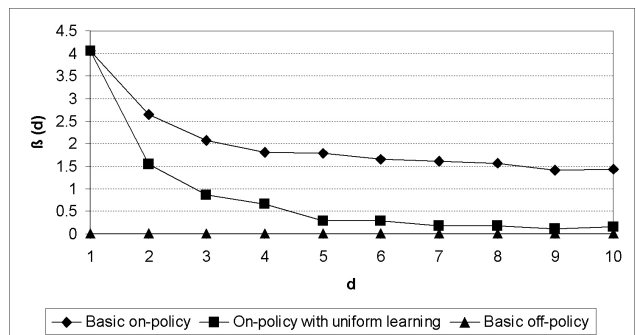


Figure 1: The volume of heuristic updates encountered with respect to the lookahead depth in different experiments.

Unlike the regular learning in LRTS, uniform learning preserves the consistency of the heuristic. Experiments on synthetic search trees suggested that inconsistency increases

the pathology (Luštrek 2005), but considering that the consistent uniform learning is more pathological than the inconsistent regular learning, this appears not to be the case in path-finding.

Hypothesis 3 Let $\alpha_{\text{off}}(d)$ and $\alpha_{\text{on}}(d)$ be the average number of states generated per move in the basic off-policy and on-policy experiments correspondingly. In off-policy experiments a search is performed every move, whereas in on-policy experiments a search is performed every d moves. Therefore $\alpha_{\text{on}}(d) = \alpha_{\text{off}}(d)/d$ (assuming the same number of non-traversable squares). This means that in the basic on-policy experiment fewer states are generated at larger lookahead depths than in the basic off-policy experiment. Consequently lookahead depths in the basic on-policy experiment are closer to each other with respect to the number of states generated. Since the number of states generated can be expected to correspond to the quality of decisions, cases where a deeper lookahead actually performs worse than a shallower lookahead should be more common.

Hypothesis 3 can be verified by an on-policy experiment where a search is performed every move instead of every d moves as it is in the regular LRTS. Table 7 supports the hypothesis. The percentage of pathological problems is considerably smaller than in the basic on-policy experiment shown in Table 3 (13.1% vs. 57.7%). It is still larger than in the basic off-policy experiment also shown in Table 3 (13.1% vs. 4.3%), but the remaining difference can be explained with Hypothesis 2 and (to a smaller extent) with Hypothesis 1.

Table 7: Percentages of pathological problems in an on-policy experiment when searching every move.

Degree	0	1	2	3	4	≥ 5
Problems %	86.9	9.0	3.3	0.6	0.2	0.0

Hypothesis 3 can be further tested as follows. Figure 2 shows that in the basic off-policy experiment and in the on-policy experiment when searching every move, the number of states generated per move increases more quickly with increased lookahead depth. This means that lookahead depths are less similar than in the basic on-policy experiment, which again confirms Hypothesis 3.

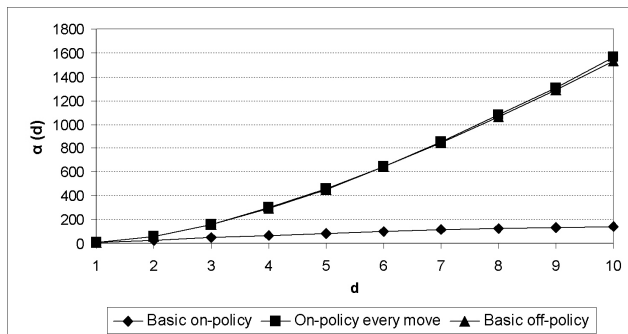


Figure 2: The number of states generated with respect to the lookahead depth in different experiments.

Summary. Different lookahead depths in the basic on-policy experiment are closer to each other in terms of the quality of decisions they produce than the differences in depth would suggest. This makes pathological cases where a deeper lookahead actually performs worse than a shallower lookahead relatively common. The reasons stem from the volume of updates encountered per move (Hypothesis 2) and the number of states generated per move (Hypothesis 3). A minor factor is also that the LRTS algorithm’s behavioral policy tends to visit states that are themselves pathological more frequently than is the average for the map (Hypothesis 1).

Remedying Lookahead Pathologies

We have shown that pathological behavior is common in real-time path-finding. We now demonstrate how much an LRTS agent would benefit were it able to select lookahead depth dynamically. Table 8 shows the solution length and the average number of states generated per move in the basic on-policy experiment averaged over our set of 1,000 problems. If the optimal lookahead depth is selected for each problem (i.e., for each start state), the average solution length is 107.9 and the average number of states generated per move is 73.6. This is a 38.5% improvement over the best fixed depth (which, curiously, is 1). The number of states generated falls between lookahead depths 4 and 5. The improvement in solution length is quite significant and motivates research into automated methods for lookahead depth selection.

Table 8: Average solution length and the number of states generated per move in the basic on-policy experiment.

Depth	Length	States	Depth	Length	States
1	175.4	7.8	6	221.0	102.2
2	226.4	29.0	7	209.3	115.0
3	226.6	50.4	8	199.6	126.4
4	225.3	69.7	9	200.4	137.2
5	227.4	87.0	10	187.0	146.3

The most straightforward way to select the optimal lookahead depth is to pre-compute the solution lengths for all lookahead depths for all states on the map. We did this for the map and the goal state shown in Figure 3. The right side of the map represents an office floor and the left side is an outdoor court area. The shades of gray represent the optimal lookahead depth for a given start state. Black areas are blocked.

Table 9 shows the solution length and the average number of states generated per move averaged over all the start states on the map in Figure 3. If the optimal lookahead depth is selected for each start state, the average solution length is 132.4 and the average number of states generated per move is 59.3. This solution length is a 47.7% improvement over the best fixed depth (which is again 1) and the number of states generated falls between lookahead depths 3 and 4 – results not unlike those in Table 8, which suggests that this map is quite representative.

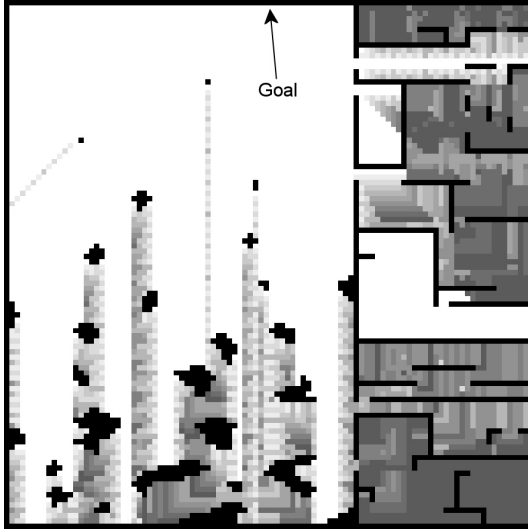


Figure 3: Starting states shaded according to the optimal lookahead depth (white: $d = 1$, darkest gray: $d = 10$).

Table 9: Average solution length and the number of states generated per move for the map in Figure 3.

Depth	Length	States	Depth	Length	States
1	253.2	7.8	6	318.8	101.2
2	346.3	29.4	7	283.6	116.2
3	329.1	50.4	8	261.5	126.7
4	337.0	69.3	9	282.6	133.2
5	358.9	85.7	10	261.1	142.7

Once we know the optimal lookahead depth for all the states on the map, we can adapt the depth *per move*. Selecting the lookahead depth in state s_c that would be optimal if the agent started in s_c should yield an even shorter solution than keeping the same depth all the way to the goal state. However, this is not guaranteed, because the heuristic will probably have been modified prior to reaching s_c and thus the search from s_c may not behave in exactly the same way as if the agent started in s_c with the initial heuristic. On the map in Figure 3, this approach gives the average solution length of 113.3, which is an additional improvement of 14.4% over the optimal depth per start state. The average number of nodes generated per move decreases by 42.6% to 34.0.

Determining the optimal lookahead depth for every pair of states on the map is computationally very expensive: the 8,743-state map in Figure 3 contains over 7.6×10^7 directed pairs of states. We can make the task more tractable with state abstraction such as the clique abstraction of (Bulitko *et al.* 2005). In this approach cliques of states are merged into single abstract states, reducing the number of states by a factor of 2 to 4 each time the merge operation is applied. The cost of building the abstract states is linear in the number of states, so the process is reasonably fast. For each abstract state s_a , we find a ground-level state s_{gl} closest to the average coordinates of all states abstracting into s_a . We then compute the optimal lookahead depth for s_{gl} and associate

it with s_a . During the on-line search, the lookahead depth for the agent’s state is retrieved from the state’s abstract parent. Table 10 shows how the abstraction level at which the optimal lookahead depths are stored affects the search performance.

Table 10: Number of abstract states, the resulting solution length, and number of states generated per move for different levels of abstraction.

Level	Abs. states	Length	States/move
0	8,743	113.3	34.0
1	2,463	124.6	38.3
2	783	129.2	39.2
3	296	133.4	40.9
4	129	154.0	51.2
5	58	169.3	50.5
6	26	189.2	45.1
7	12	235.7	55.5
8	4	253.2	7.8
9	1	253.2	7.8

Optimal lookahead depths at abstraction level 5 are shown in Figure 4. The average solution length in this case is 33.1% better than with the best fixed depth and the optimal depths needs to be pre-computed for only 0.004% of state pairs.

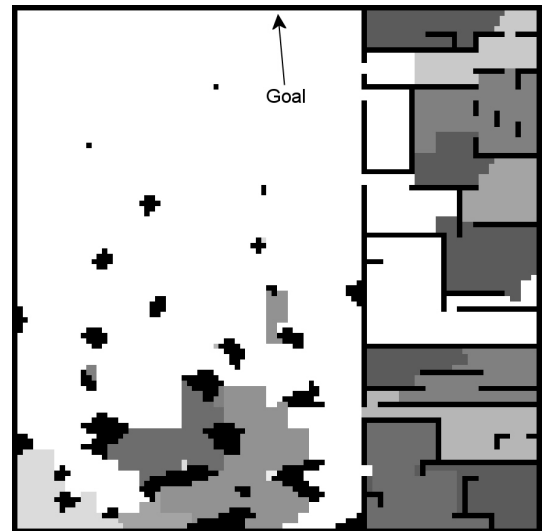


Figure 4: Optimal lookahead depth with abstract states (white: $d = 1$, darkest gray: $d = 10$).

Conclusion and Future Work

This paper made three contributions. First, we demonstrated that lookahead pathology is a common phenomenon in real-time path-finding. Second, we proposed and empirically supported three complementary causes of lookahead pathology. Finally, we proposed a method for selecting lookahead depth dynamically per current state. Doing so not only eliminated another user-tuned parameter, but also substantially increased search performance by eliminating pathologies.

Further research will be directed towards a practical method for dynamic selection of lookahead depth. We showed that this can reap a significant benefit, but our initial implementation is computationally still somewhat expensive and needs to be tested on more maps. In order to speed it up, we will look into computing the optimal lookahead depth at abstract levels, where the state space is smaller. We will also consider lookahead search with more flexible lookahead spaces such as (Koenig 2004). Finally, we would like to extend this research to dynamic and stochastic environments.

References

- Donald F. Beal. An analysis of minimax. In M. R. B. Clarke, editor, *Advances in Computer Chess*, volume 2, pages 103–109, Edinburgh, UK, 1980. Edinburgh University Press.
- Ivan Bratko and Matjaž Gams. Error analysis of the minimax principle. In M. R. B. Clarke, editor, *Adv. in Computer Chess*, volume 3, pages 1–15, Oxford, UK, 1982. Pergamon Press.
- Vadim Bulitko and Greg Lee. Learning in real time search: A unifying framework. *JAIR*, 25:119–157, 2006.
- Vadim Bulitko, Lihong Li, Russell Greiner, and Ilya Levenner. Lookahead pathologies for single agent search. In *Proceedings of IJCAI, poster section*, pages 1531–1533, Acapulco, Mexico, 2003.
- Vadim Bulitko, Nathan Sturtevant, and Maryia Kazakevich. Speeding up learning in real-time search via automatic state abstraction. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1349–1354, Pittsburgh, Pennsylvania, 2005.
- Vadim Bulitko. Lookahead pathologies and meta-level control in real-time heuristic search. In *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, pages 13–16, Porto, Portugal, 2003.
- Carlos Hernández and Pedro Meseguer. LRTA*(k). In *Proceedings of IJCAI*, pages 1238–1243, Edinburgh, UK, 2005.
- Sven Koenig. A comparison of fast search methods for real-time situated agents. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, pages 864–871, 2004.
- Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2, 3):189–211, 1990.
- Mitja Luštrek, Ivan Bratko, and Matjaž Gams. Why minimax works: An alternative explanation. In *Proceedings of IJCAI*, pages 212–217, Edinburgh, UK, 2005.
- Mitja Luštrek. Pathology in single-agent search. In *Proceedings of Information Society Conference*, pages 345–348, Ljubljana, Slovenia, 2005.
- Dana S. Nau. *Quality of Decision versus Depth of Search on Game Trees*. PhD thesis, Duke U., 1979.
- Dana S. Nau. An investigation of the causes of pathology in games. *Artificial Intelligence*, 19(3):257–278, 1982.
- Judea Pearl. On the nature of pathology in game searching. *Artificial Intelligence*, 20(4):427–453, 1983.
- Dave C. Pottinger. Terrain analysis in realtime strategy games. In *Proceedings of Computer Game Developers Conference*, 2000.
- Anton Scheucher and Hermann Kaindl. Benefits of using multivalued functions for minimaxing. *AIJ*, 99(2):187–208, 1998.
- L.-Y. Shue, S.-T. Li, and R. Zamani. An intelligent heuristic algorithm for project scheduling problems. In *Proceedings of the 32nd Annual Meeting of the Decision Sciences Institute*, San Francisco, 2001.