Skeleton-based Wayfinding for Computer Games

Hans Guesgen and Paul Shotbolt

Dept. of Computer Science, University of Auckland
Auckland, New Zealand
hans@cs.auckland.ac.nz, paul.shotbolt@ihug.co.nz

Abstract
This is an extended version of a paper published in the ECAI conference (Guesgen and Shotbolt 2004). In this paper, a system is proposed in which case-based reasoning principles are applied to the wayfinding problem for computer games. As case-based reasoning is thought to mimic aspects of human thinking, applying this methodology may provide more convincing non-player character behaviour. Additionally, CBR allows for re-use of knowledge, and performance gains could be achieved through a reduction in redundant computation. This approach incorporates aspects of island search, case-based reasoning and traditional wayfinding techniques. Agents store a skeleton set of known paths and adapt these skeleton paths to solve new wayfinding problems. New solutions, along with paths obtained through communication with other agents, are reincorporated to create a dynamic, evolving case-base of skeleton paths.

Introduction
Most humans perform wayfinding tasks heuristically without considering the incredible complexity of the task they are performing (Kuipers 2001); however, computer wayfinding is usually approached from a series of mathematical calculations over a pre-computed graph or grid. This disparity means that certain problems are quickly solved by one approach but difficult to solve with another, and that for a given problem, solutions obtained by one approach may be noticeably different from the other.

In the context of a computer game, a goal of the game designer is the suspension of disbelief of the player. Many computer games feature 'human' non-player characters (NPCs), so increasing importance is placed on bestowing human-like behaviour to these characters. Differences between human and simulated behaviour result in the onset of disbelief, which subsequently detracts from enjoyment of the game. Recent investigations into wayfinding and game playing, such as (Laird 2001) have shown that there is still tremendous scope for improvement in the simulation of convincing human behaviour.

In this paper, the case-based reasoning (CBR) methodology is applied to the wayfinding problem in computer games. As this method more closely approximates human thinking, this provides more convincing agent behaviour and the added benefits of knowledge re-use allow for more efficient wayfinding. Another research goal is to allow this deeper simulation of NPC wayfinding to contribute to emergent gameplay. Emergent gameplay is discussed and defined in (Smith 2001), and involves deeper simulations giving rise to complex scenarios in-game that are not considered by the game designers during production. Successful, believable emergent gameplay is thought to result in a more rewarding play experience.

The aim of this experiment was to give agents in a computer game environment the ability to obtain, store and reuse spatial information about paths and locations in their wayfinding processes. Agents should be able to obtain new paths through both derivation from level geometry, and also through communication with other agents. Once obtained, paths should be retained in an agent’s knowledge base (KB), and future wayfinding decisions should incorporate this knowledge. This skeleton set of known paths should grow and change dynamically as the agent moves about the level, affecting future decisions to an increasing extent. Humans appear to build up and utilise a base of these skeleton paths in a similar manner, and the similarity between the approaches provides more believable simulated human wayfinding.

Many special factors must be considered when designing a system for computer games, particularly the significant restrictions on memory and processing cycles (Fairclough et al. 2001). An interesting aspect of wayfinding in computer games is that the path found does not have to be optimal or even near optimal. In some cases it is acceptable that an agent attempts a route and backtracks if that route fails, as the importance lies with suspending the user's disbelief. In these cases the agent may make mistakes, as long as they are mistakes that a human might make under similar circumstances, and are not blindly repeated. Further, in computer games, the environment is usually completely accessible, yet we often wish to simulate a computer-controlled opponent agent with only limited knowledge of the domain.
Approach

The dynamic skeleton-based wayfinding system is composed of three major areas: a representation system, a wayfinding system, and a communication system. Combined, these provide each agent with the ability to solve wayfinding problems through re-using path knowledge in a manner that approximates social humans exploring and interacting with each other.

Knowledge Representation

The representation system governs how each agent stores and uses its KB of known paths. Agents are required to possess enough data about a certain area of terrain in order to navigate it. This approach involved placing a grid of waypoints over the terrain and setting up neighbourhood information for each waypoint \( w \). This information includes an associated \( x,y,z \) coordinate and also references to other waypoints that are immediately near \( w \) and traversable directly from \( w \).

For reasons of simplicity, all agents are given awareness of all waypoints in this current implementation, although this is not a requirement for the dynamic skeleton wayfinding system. Agents also possess a set of known skeleton paths that can be used to reason with. While in this implementation this set is initially empty, it is reasonable to suggest that agents could be given an initial skeleton set to reason with.

To facilitate agent communication, knowledge representation was implemented using semantic networks (Russell and Norvig 1995); however, conceptual graphs would be equally suitable. As emergent, believable gameplay is one of the primary concerns of this research, it was decided that each agent should maintain its own separate KB instead of storing path knowledge in a central repository. While the centralized approach may increase the efficiency and optimality of wayfinding, agents will be able to access information that they should not be privy to, and their resulting near-prescient behaviour may detract from the user’s suspension of disbelief.

Each agent stores its individual knowledge in a semantic network made up of concepts and relations between these concepts. Concepts store information about some game object, eg another agent, or an abstract idea such as the notion of a ‘universal set’. Concepts have a name, and a list of properties describing their attributes; for example, an ‘Agent’ concept might have a ‘health’ property. Each property has a name, a typeless value, and a timestamp describing the creation time or last update of the property.

Wayfinding System

The wayfinding system incorporates elements from CBR, A* search, and skeleton-based wayfinding.

Agents employ CBR techniques to solve wayfinding problems. A query is specified in this system by a start position and an intended goal position for some agent. When given a query, an appropriate skeleton path from the case-base is first retrieved, and then adapted into a solution. If no path suitable for adaptation is found in the case-base, then a solution must be generated entirely. This can occur either because there are no paths in the case-base, or because there was no path that was examined that scored above some minimum rating threshold.

Retrieval of an appropriate solution, given a query, is a difficult process (Bartsch-Spörl, Lenz, and Hübner 1999), and there are a multitude of approaches to it; for example, defining attributes for the query and the cases and examining and combining weighted elements from ‘similarity matrices’. In general, the approach involves retrieval function \( f \) that maps candidate path and a ‘start point, destination query’ to some rating, in most cases a ‘similarity metric’, usually over the real numbers.

There are a large number of possible retrieval functions, \( f \), which could be used for evaluating how useful a candidate path is for solving a given query. For simplicity, a weighted combination of the following factors was chosen.

- A measure of how well the orientation of the candidate path matches the orientation of the general direction of travel for the query
- An approximation of the distance from the start point of the query to the candidate path
- An approximation of the distance from the destination point of the query to the candidate path
- A measure of how the curvature of the path influences the approximations in the two points above
- The length of the candidate path
- The agent’s perception of how useful a path has been to them (or other agents) in the past: the ‘agent rating’

If a path is found that is similar enough to the query problem, it is adapted into a solution. The obvious approach is the same as that described in (Kuipers 2001), finding connections between the start point and the chosen skeleton path \( s \), and then from \( s \) to the end point. This path adaptation is handled by a more classical wayfinding technique; in this implementation A* search was used. Alternative techniques are possible, providing that they have the ability to find a path between a pair of locations, the ability to find a path between a start location and a given skeleton path, and the ability to find a path between a given skeleton path and a given destination location.

The A* algorithm is a well-known form of guided search that is both complete and optimal\(^1\) (Russell and Norvig 1995), and has been widely used in computer games (Sergent 2003). It operates by minimising the expected cost from the current node to the goal, using some measure \( g^*(x) \) to evaluate the cost so far, and some heuristic \( h(x) \) to estimate the remaining cost to the goal.

---

\(^1\) Provided that the heuristic function \( h \) is admissible.
For many wayfinding problems, \( A^* \) becomes intractable, and alternatives such as introducing sub-goals, e.g. Island search or skeleton-based wayfinding become preferable. With these approaches, path optimality is sacrificed for search efficiency.

Skeleton-based wayfinding is an approach that selects a path as a subgoal rather than a specific node. A route from start node \( N \) to the skeleton path \( S \) is found, and then a route from \( S \) to goal node \( G \) is found, and finally the complete path is composed of routes NS and SG, along with an intermediate part of skeleton path \( S \). The template \( A^* \) algorithm in (Heyes-Jones 2003) was modified to allow finding a new path between a node and a given path. This involves defining a goal state to be any one of the waypoints along the given path (or paths). The \( h \) heuristic had to be altered also; for most wayfinding tasks, the preferred \( h \) heuristic is simply the Euclidean distance between the current node and the target node. This heuristic is admissible and therefore never underestimates the actual cost of reaching the goal node from the current position.

With the introduction of a ‘goal path’, which can be decomposed into a set of goal nodes, the \( h \) heuristic needs to remain admissible yet express the cost of reaching a path, or set of nodes, rather than a single node. This problem is easily resolved by defining \( h \) to return the Euclidean distance to the closest node in the path to the current node. This altered heuristic is still admissible but is computationally more expensive by a linear factor of \( n \), where \( n \) is the number of nodes in the given skeleton path. Bear in mind that in most cases, this increase in computational complexity of computing the heuristic function \( h \) is offset by the benefits of introducing an explicit sub-goal to the path, which reduces the complexity of the \( A^* \) algorithm from \( O( b^d ) \) to around \( O( b^{d/2} ) \), where \( b \) is the effective branching factor of the search tree, and \( d \) is the depth of the search. Further benefits occur when the skeleton path makes up a significant portion of the solution, as the search is reduced to two smaller trees, the first from the start to the skeleton path with depth \( d_{sp} \), and the second from the skeleton path to the destination, with depth \( d_{sp,d} \). The combined distance \( d_{sp,d} = d_{sp} + d_{sp,d} \) is less than the depth \( d \) of the unaided search, because the component of the combined path that is contributed by the skeleton path does not have to be searched.

Additionally, due to the difficulty of searching from a skeleton path to a destination location, the system instead searches from the destination to the skeleton path, and reverses the solution when found.

**Communication System**

Inter-agent communication is also an integral part of simulating human wayfinding. In computer games, agents usually inhabit a closed, multi-agent environment, and are free to traverse diverse terrain carrying out their goal activities for example, attacking player characters or constructing a building. Agents may share a common goal such as trying to destroy any players inhabiting the environment. With a shared objective, there is scope for cooperation and coordination.

The communication system extends the semantic-network knowledge representation by allowing agents (within some radius) to communicate by asking about a particular topic \( t \). The agent being asked about \( t \) examines its own semantic net for the concept describing \( t \), and returns a small sub-graph of that semantic net, including the concept describing \( t \), a number of neighbouring concepts and all of the relations connecting these. The returned sub-graph is calculated from a randomized depth-\( d \) breadth-first search of the semantic net, starting at the concept describing \( t \). This returned sub-graph is then reintegrated into the querying agent’s own semantic net. All unknown concepts and relations are added, and all known concepts are updated.

An agent’s skeleton set may be stored as a set of ‘path’ concepts. An agent creates a new path concept every time it discovers a new path, and stores this in their semantic net. The agent rating for this path, which measures how often it has been used, is also stored in the semantic net. The communication system then permits skeleton paths (and associated data) to propagate between agents in a semi-realistic fashion. Rather than being static, an agent’s case-base of skeleton paths will grow and change dynamically throughout the course of the game session through experience and interaction with other agents. Each agent will use their knowledge of paths in the solving of subsequent tasks, and as their case-base begins to store an increasing number of useful paths, their wayfinding abilities should improve.

**Additional Considerations**

The application of CBR techniques to computer games introduces a number of problems that must be considered. The CBR approach suffers when either the case-base is too small or too large. The case-base is prevented from being too small by continuous acquisition of new paths. If the case-base is too large, then it becomes inefficient to process, and some method for refining the case-base to an acceptable size (while only discarding the least useful cases) is necessary. In this system, a 'forgetting' algorithm removes oldest and least-used paths from the case-base, whenever the case-base exceeds a threshold size. This also provides memory savings, as ever-growing agent case-bases can begin to encroach on the limited RAM resources available in the context of a computer game.

Although attaining maximal performance was not a primary design goal, optimisations were performed to ensure the system is viable for use within a computer game, where savings in processing time and memory use are highly prized.

---

2 Assuming a good explicit sub-goal is chosen, ideally halfway between the start node and the destination node. If a less optimal sub-goal is chosen, the benefits to computational complexity of the algorithm will not be as evident.
Results

Overall, as expected, the dynamic skeleton based wayfinding system appears to reduce the time needed for solving certain classes of wayfinding problems, compared with the standard A* approach. The system was built using the Torque Game Engine (Garage Games 2003), and a number of different experiments were carried out over several different test environments. A preliminary test environment with a low waypoint resolution was initially used. Another more complicated test environment was constructed for testing the wayfinding system, and involves more complex geometry and higher waypoint resolution, as well as including non-axis-aligned buildings and a higher obstacle count. Other experiments included heavily modified versions of these environments under a range of different waypoint resolutions. Results from the experiments indicate that the general approach can provide some benefits in terms of simulating human behaviour within the context of a computer game.

The first wayfinding environment was chosen primarily to evaluate whether or not the implementation was functioning correctly, and this test environment served its purpose. The waypoint grid was of resolution 50x50 units. Due to the smooth curvature of the landscape in test environment A, agents did not have significant trouble navigating the terrain. After the integration of the dynamic skeleton-based wayfinding system, agents visibly re-use past paths in their solutions to subsequent queries. Additionally, a preliminary analysis shows that the wayfinding process appears to complete quicker when using the dynamic skeleton based system, rather than the original A* system, provided there are a significant number of paths in the agent’s KB. At higher waypoint resolutions, these time savings are more significant; however, a thorough empirical testing of this is outside the scope of this paper. Agents do not often make poor choices in their selection of which skeleton path to adapt to solve a query. This is a very difficult aspect of the system to test, and a more thorough investigation of the choices made would be appropriate for future work.

Observing the behaviour of the agents as they acquire knowledge of new skeleton paths, it is evident that the agents learn. The path-finding ability of the agents improves with experience, not only in terms of moderately shorter times taken to find paths, but also in terms of agents using paths that have been created and taught by other agents. Certain paths become much more frequently used than other paths, and Kuipers’ (Kuipers 2001) positive feedback loop appears to be emulated to some extent by the agent ratings system.

This preference for common routes, while not completely convincing, appears to be more reminiscent of the human school of problem solving than the non-skeleton based wayfinding system.

The more complicated test environment involved a larger number of buildings acting as obstacles. In addition, some of the buildings were aligned at unusual angles from each other. The second environment provides a much more interesting test-bed for investigating agent wayfinding behaviour, but is by no means a complete evaluation of the approach for the task. As expected, the more complex environment also highlighted a few more problems with the design. Of the more interesting problems noted, certain cases arise where an agent will take an extremely poor choice of route, as illustrated in Figure 1.

![Figure 1](image-url)

Figure 1. An illustration of the poor choice problem. Here, an agent decides to use a known skeleton path (dashed). It takes the most direct route to that path from S, and follows this path before departing towards destination D; however, this results in a route that is significantly worse than optimal.

While optimality was not an essential design criterion for the skeleton based wayfinding system, the choice that the agent makes here is far removed from optimal. There are two ways of looking at this issue – is the problem at the path selection phase, or at the adaptation phase? A better adaptation phase might result in the agent choosing a better point to rendezvous with the skeleton path. However, another way of looking at the issue is that the problem is in the selection phase. The problem has already occurred; observe that skeleton path selected is already significantly worse than optimal. In fact, even if travelling between two points directly on each end of the skeleton path itself, following the path already defined might be a poor choice as it involves a long detour. This skeleton path, then, is not a good path to have in the case-base, and certainly not a useful path for adapting to solve many queries.

Another interesting phenomenon was observed in this environment: agents performed slowly at first, but after some initial trials, they became very quick at making good skeleton path adaptations. The agents performed well for a moderate time period, but in the longer sessions of ten to thirty minutes, their performance began to degrade, until their behaviour dropped to an infantile level, and became too frustrating to continue working with. The reasons behind this behaviour appear to be as follows.
Initially, agents have no paths in their case-base, and do all their wayfinding through A*, guaranteeing that each new path is an optimal route. After building up a small number of cases, agents begin to adapt these cases to solve queries. These solutions are new paths, which are added to the case-base, however, these paths are based on skeleton-based wayfinding and are non-optimal. The case-base now contains some optimal paths, and some non-optimal, derivative paths. As these derivative paths are adapted from optimal paths, we will call these first-level derivative paths.

After another period of path building, the case-base contains some optimal paths, some first-level derivative paths, and second-level derivative paths that are adapted from first-level derivative paths. This process continues until there are many levels of derivative path contained in the case-base.

This process is compounded by two factors. Second level derivative paths are by nature less optimal than first level derivative paths. Observation confirmed that for a given start and end point, in general, a second-level derivative path will be longer than a first-level derivative path. Recall that the path selection process, wherein agents choose which path to adapt to solve a new query, has a positive weight for path length. This means that a longer path, such as a less optimal second-level derivative path, is more like likely to be chosen for adaptation than a more optimal first-level derivative path. There are two results of this – poor path choice (for example, choosing a second level derivative path instead of a first) and the creation of an even poorer choice path (eg the third level derivative path created by adapting the poor choice for the above example). The new path is added to the case-base, in process that initiates a positive feedback loop for poor path choices. Poor path choices cause poorer path choices, and this action and reaction sequence continues on ad infinitum via a cycle of cumulative causation.

This explains the behaviour observed whereby agents’ wayfinding abilities increase initially, but under longer testing sessions their performance begins to degrade. Compounding the problem of this case-base degeneration is another problem involving the ‘forgetting function’. The most optimal paths created near the beginning of the session are shorter, and are therefore less likely to be used. They then do not achieve a relatively high agent rating due to being under-used, and they have an earlier timestamp than all other paths. This combination of under-use and early timestamps mean that these paths are early candidates for the ‘forgetting system’. The most optimal paths in the case-base are then removed soonest. This means that the forgetting system as it was originally designed is actually adverse to the case-based reasoning process.

As described above, even without the added complication of the forgetting system, the entire ‘dynamic skeleton-based wayfinding system’ would need an overhaul to make it viable, due to a cumulative causation situation that arises that causes the case-base to degenerate.

### Possible Extensions

There are a number of ways in which this system could be extended, and indeed, the system urgently requires some of the more major changes described here before it could even be considered for further investigation.

A solution to the degenerative case-base situation is to only store restricted classes of solutions; for example, optimal and first level derived solutions. At creation time, each path and path concept should be given a derivation level property, so that it can be dealt with accordingly. The ‘forgetting’ process would also have to be adjusted so that it was no longer biased against more optimal solutions. One approach might be to give a large negative weight to the ‘derivation level’ of the paths, and this would make the agent more likely to forget 2nd and 3rd level derived paths than 1st level or optimal paths. This is somewhat similar to human reasoning; humans tend to place more emphasis on their first impressions (Gleitman, Fridlund, and Reisberg 1999), and less on their repeated subsequent experiences.

The system could be improved by viewing it as a ‘hardest problem’ solution cache. This takes the approach that there is no point in caching problems that can be easily recalculated; speed benefits may be made primarily on the more difficult problems, even if they are encountered less frequently. The dynamic skeleton-based wayfinding system could be developed in such a way that only those difficult paths that take A* a long time to solve would need to be stored. Routes such as those that involve immediately moving directly away from the goal destination are notoriously slow to solve with A* but could be improved by this modified approach.

An adept agent could use its knowledge of skeleton paths to help predict the movement of enemies. Assuming that the agent and its adversary traverse the same environment, it is likely that the enemy would also utilise the skeleton paths that the agent finds. An agent could use its knowledge of skeleton paths to either set up an ambush in a room that is likely to be traversed (because it is on a skeleton path), or to help predict which direction an opponent may head in future, given the opponent’s current position and direction. These are only two of a number of different applications to spatial reasoning that could be made once an agent has a working knowledge of paths and routes in an environment.

The current system only allows for incorporating a maximum of one skeleton path into any given wayfinding request. Conversely, a human is able to recursively string together skeleton paths to solve problems involving a significant number of skeleton paths. Incorporating this recursive process into the wayfinding system could produce significant computational benefits and more convincing simulated human behaviour.
Currently, the use of linear weighted attributes for path selection is arbitrarily chosen for ease of computation and implementation. A more sophisticated function, perhaps with weights selected by utilizing techniques from the field of Machine Learning such as genetic algorithms, might provide a more accurate and useful measure of the utility-level of a particular skeleton path regarding a specific query.

Conclusions

Overall, the project was a successful investigation of an unusual approach to the wayfinding problem. A large number of contributing fields provided inspiration and stimulus for areas of this research.

The dynamic skeleton-based wayfinding approach still has some problems. These problems range from moderate flaws, such as imperfections in the ‘path selection’ system, to fundamental flaws such as the conflux of factors that cause positive feedback to more derivative path choices, resulting in a continuously degenerating case-base. The most important of these problems have been outlined, and overall the impression is given that the whole idea of dynamic skeleton-based wayfinding may need a major revision before continuing to any further stage of research.

Despite these problems, there have been a number of successes with this investigation, including the occurrence of several emergent phenomenon. An example might be the positive feedback loop present when agents interact with each other. Information about more useful paths is propagated between agents, and as agents are then more likely to share common routes, future interaction with other agents is encouraged.

A number of optimisations were made to keep the system efficient for use in the context of a computer game; however, others were simply suggested, as efficiency is not critical to this research. A wide range of potential extensions have been proposed, in addition to possible solutions for the most significant remaining problems including both the degenerative case-base issue and the adverse ‘forgetting system’ issue.

Another significant issue with this research is the small amount of empirical testing of the system, as efficient algorithms are critical for computer games. However, the results of this research imply that there is significant work to be done resolving the critical problems in the approach, and thus concerns about efficiency should be raised only after the approach has been re-examined.

References


