Belief-Driven Pathfinding Through Personalized Map Abstraction

Davide Aversa and **Stavros Vassos**
Department of Computer, Control, and Management Engineering
Sapienza University of Rome
Rome, Italy
{aversa,stavros}@dis.uniroma1.it

Abstract

We investigate the case of belief-driven pathfinding (BDP) according to which characters hold a personalized account of a dynamic changing game-world. BDP is concerned with maintaining and revising a set of beliefs that persists over time as a character navigates to subsequent target destinations. This allows for a differentiation among characters with different observations in the game and can provide better believability. We present BGCA*, a practical BDP approach that is based on (*i*) decomposing the map into regions, (*ii*) using personalized beliefs per character about the connectivity of regions, and (*iii*) employing a regular pathfinding component as a service. We evaluate BGCA* in terms of computational effort and precision wrt a regular solver over several benchmark maps. Our results motivate a simple belief revision strategy that induces small overhead and amortizes effort spent toward precision.

Introduction

Traditionally, AI for non-player characters (NPCs) in games has to do a lot with pathfinding, i.e., finding an appropriate path so as an NPC can navigate from their current location to a desired target destination. This is an essential mechanism for every game with characters and is crucial for the quality of the interaction between the human player and the NPCs. As the demand for a higher level of believability for NPCs increases, many techniques have been developed that handle different aspects of pathfinding, e.g., smoothing the trajectories so that paths look more realistic or taking into account the different types of terrain and NPC capabilities.

Nonetheless, there is one aspect that is typically neglected, namely, the *personalized* view of the game-world regarding what each NPC *knows* or *believes* about the connectivity of the areas in a dynamic world. In other words, it is not possible two different NPCs to follow a different path based on what they *observed* in the world or what they *missed to observe* in areas that were not visible to them.

Typically a game engine includes a pathfinding module that is used by all NPCs in order to navigate in the game map (perhaps also conditioned on their size and capabilities). But what happens when one NPC has seen, hence should know, that the shortest path to the target destination is blocked while another NPC does not possess this information? Even though these special cases would probably arise less frequently in existing games, these are exactly the cases where the believability of NPCs is challenged. When game AI is more or less "cheating" by allowing the characters to do things that they are not supposed to, an immersion-breaking moment happens that significantly lowers the quality of the gameplay. This is similar to the frustration of players in the past when opponent NPCs would always follow them in the most optimal way, simply because they could always see the location of the player (even if this should not be possible by means of their location and observations). What we identify here is a similar case but more refined as it refers to all observations about the connectivity of areas in the game-world, and not just a single piece of information such as the last known location of the human player.

There is a good reason for not including personalized beliefs per NPC, namely real-time efficiency. As requests need to be resolved almost instantly for many characters in parallel, it is important that a pathfinding service in the game is optimized in terms of CPU time. Similarly, as the game-worlds become larger it is equally important that their representations used for pathfinding are compact in terms of memory usage. Nonetheless, as available computing power of consoles and computers increases, there is more space for experimenting with such aspects.

In this work we investigate BGCA*, a practical approach for *belief-driven pathfinding (BDP)* that combines intuitions from work on map abstraction, e.g., (Sturtevant and Buro 2005), and transit routing (Abraham et al. 2010). We employ a simple map decomposition into disjoint *regions* and identify *gates* that connect them, and use a personalized *gate connectivity graph* per NPC as a means to capture the beliefs of eah NPC. The proposed approach is tested in a number of maps of different size and over a large number of synthetic pathfinding requests. Our results motivate a simple belief revision strategy that induces small overhead and amortizes effort spent toward precision.

Related Work

There is a lot of work on pathfinding in dynamic (or unknown) maps based on interleaving planning and execution in a similar manner as in the one adopted by BDP. According

to *agent-centered* search (Koenig 2001) the agent searches over a local part of the map that is typically centered around the agent, computes a minimal path from its current location and the most promising location on the local boundaries, and retries after moving to the found location. In the general setting of *real-time* search (Korf 1990) each deliberation step is limited by time restrictions typically expressed by means of a maximum number of expanded nodes. The aim is to optimize the nodes expanded while leading (through replanning) to paths with high quality, e.g., some recent work is LRTA* with subgoals (Hernández and Baier 2011) and TBA* (Björnsson, Bulitko, and Sturtevant 2009).

In these approaches the path quality is measured wrt the shortest path found by a regular pathfinder that searches for a complete solution. In BDP though, the aim is not to find the shortest path (by possibly also "physically" moving back to previous locations from time to time) but to generate a path that is consistent with the acquired (and possibly outdated) information about the map that was accessible to the character. Note also that in BDP the acquired information is intended to *persist* for more than the duration of a planning and execution cycle for a single target destination.

So what is the main challenge with BDP? One can adopt the very simple solution that each NPC maintains a personalized copy of the whole game map. This representation offers some advantages, in particular the ability for NPCs to consider beliefs about every tile of the map and that orthogonally to this any additional tricks can be applied to the pathfinding over the copy of the map, including real-time search methods. However this solution is not efficient in many practical cases because it needs to allocate a large amount of map tiles in memory when there are many NPCs in the game, either for the copy of the map alone or additionally for the cached search nodes that are explored. A more refined solution can be developed by exploiting map abstraction techniques so that only some abstraction level of the map is personalized and the rest is common to all NPCs.

Moreover, there are several aspects that need to be explored in this setting that arise from the use of beliefs: (*i*) how long should a belief about the free/blocked state of a particular map tile be kept? (*ii*) what is a practical representation for the belief about a map tile that is "forgotten" or discarded from the belief set? (e.g., it is different to assume that it is now free following an open-door assumption than marking it as unknown) (*iii*) if a path cannot be found using the personalized information of an NPC, should the pathfinding procedure discard beliefs in order to relax its assumptions? (*iv*) and if so, which beliefs should be discarded?

Our analysis aims at giving answers to some of these questions over a practical BDP approach that keeps a "time-window" of the recently observed information about a particular set of tiles called gates. In our approach a simple abstraction is employed that is based on decomposing the map into regions, while gates connect these regions and their free/blocked state is personalized per NPC. In this way a high-level path over regions is personalized for each NPC while a low-level path is generated by a regular pathfinding service that operates over the true, updated, map.

Observe that when the time-window of our BDP approach is zero, meaning that no beliefs are maintained between searching for different destinations, then BDP reduces to a simple hierarchical pathfinding method (with no real-time search tricks) that replans whenever new information is acquired or when the perceived path fails in execution. When the window allows for some beliefs to persist, then different paths are found based on the beliefs.

Our work is then closely related to hierarchical abstractions, e.g., HPA* (Botea, Müller, and Schaeffer 2004) in which the high-level structures are fixed in regular regions (e.g., squares) as well as abstractions based on map topology such as the one in PRA* (Sturtevant and Buro 2005) and those examined in (Sturtevant and Jansen 2007). Note that since we are interested in ascribing beliefs to the elements of the abstraction, the approach we adopt based on gates can be more intuitive in practice, mapping to points of interest that may be open, closed, or blocked, in contrast to the other representations that are, in a sense, more mathematical. PRA*(k) also employs an approximation so that only a path number up to length k is searched per abstraction level. As mentioned before, this type of approximating, time-slicing, or keeping search information such as D* (Stentz 1995) and D*-Lite (Koenig and Likhachev 2005), are orthogonal to the issues that BDP is concerned.

Finally, BDP is also relevant to the work in heuristics based on identifying and exploiting the high-level structure of the game map. In particular, the case of pre-computed heuristics for optimal pathfinding such as those described in (Felner, Sturtevant, and Schaeffer 2009), and the gateway-based (Björnsson and Halldórsson 2006) and portal-based heuristic (Goldenberg et al. 2010).

Belief-Driven Pathfinding (BDP)

A centralized pathfinding service is a typical core component of many games that deals with finding an optimal (e.g., shortest) path between two points on the game map. Assuming a *dynamic* environment where passages may be blocked by obstacles or locked doors, the map may be changing constantly under the effects of the player and NPCs in the game. Typically, a pathfinding service operates on the current (updated) game map for all requests. We call this *omniscient pathfinding* because it assumes that every NPC has an accurate view of the whole game map at any given time.

On the contrary, *belief-driven pathfinding (BDP)* assumes that each NPC has a *personalized* version of the game map which is appropriately updated only for the areas that the NPC visits as it moves. This personalized version of the map is represented in terms of *beliefs* that the NPC has about the connectivity of the game map. When a pathfinding request is made by an NPC, a BDP service returns a path that is optimal wrt the NPC's beliefs, but not necessarily also optimal with respect to the actual game map. The motivation for such a personalized BDP service is that the resulting path provides a more believable behavior for the NPC navigation. This is illustrated in the following example scenario.

Example 1. Imagine a scenario according to which the player decides to block a passage by putting some obstacle in the way after he passes through. This means that a hypo-

thetical chaser NPC would have to go to the blocked passage, find the path obstructed and compute a different route in order to get close to the human player. This is an important piece of information that actually should affect greatly how the NPCs will decide to navigate in the game-world. Using omniscient pathfinding it is as if the NPC is *instantaneously* informed about the obstacle, defeating, in practice, the attempt by the player to slow down the NPC. On the contrary, with BDP the NPC can act in a more believable way: since it does not know that the passage is blocked it first tries to go through the normal (previously shortest) route, sees that it is blocked, and then goes around using the alternative route.

First, we describe a simple hierarchical pathfinding approach over gates in the game map and then we discuss how this can be used to form a practical BDP approach.

GCA*: Gate connectivity A*

We assume that the game-world is represented as an *octile map* M such that each tile $m \in M$ may be blocked or free, and appropriate functions exist for retrieving neighbouring tiles and the state of the tile. We choose the grid representation for simplicity but note that the methods we present can be adapted to work for other representations.

Gate connectivity A (GCA*)* relies on a simple map decomposition of M into a set $\mathcal{R}$ of n regions $R_i \subseteq M$, such that $\bigcup_{i=1}^{n} R_i = M$, and $R_i \cap R_j = \emptyset$ for every $i \neq j$. Given $\mathcal{R}$, we define M_P as the set of tiles that are in the edge of one region in the sense that they have a neighbor that belongs to a different region. Then, we define a *portal* p as a pair (m_i, m_j) such that $m_i \in M_P$, $m_j \in M_P$, $m_i \in R_i$, $m_j \in R_j$ and $i \neq j$. We say that p *belongs* to R_i iff $m_i \in R_i$ or $m_j \in R_i$. As a consequence, any portal always belongs to exactly two regions. Finally we define a *gate* G as a set of portals, such that (*i*) every portal $p \in G$ belongs to the same pair of regions R_i, R_j and (*ii*) the portals in G form a consequtive horizontal or vertical line of one or more portals. We also say then that gate G connects regions R_i and R_j and use $conn(G)$ to denote the set $\{R_i, R_j\}$.[1]

Given a map M and an appropriate set of regions $\mathcal{R}$ (e.g., one that decomposes the map into coherent areas) and an appropriate set of gates $\mathcal{G}$ (e.g., one that segments the borders of regions into gates of bounded size), the *gate connectivity graph* $\Gamma_{\mathcal{G}} = (V, E)$ is an undirected graph such that: (*i*) the set of vertices V is the set of gates $\mathcal{G}$; (*ii*) an edge (G_i, G_j) is in E iff $conn(G_i) \cap conn(G_j) \neq \emptyset$; (*iii*) each edge (G_i, G_j) is labeled with the true distance between G_i and G_j (considering the center of each gate). Note that, $\mathcal{R}$ can be fixed and $\Gamma_{\mathcal{G}}$ can be precomputed offline.

Similar to more sophisticated hierarchical planning approaches, e.g., (Botea, Müller, and Schaeffer 2004; Sturtevant and Buro 2005; Sturtevant 2007), GCA* first searches for a path in the higher-level portal connectivity graph, and then searches for low-level paths that realize each of the high-level moves between regions. GCA* handles a request between tiles m_a and m_b as follows:

1. If m_a and m_b belong to the same region, the algorithm returns the path $m_a \rightarrow m_b$.
2. Otherwise, graph $\Gamma_{\mathcal{G}}$ is expanded with two vertices v_a and v_b into graph $\Gamma^{\star}_{\mathcal{G}}$. New edges are added between v_a and each gate that connects the region where m_a is in, and similarly for the new vertex v_b and m_b.
3. An A^* search is launched on $\Gamma^{\star}_{\mathcal{G}}$ in order to find a high-level path of the form $v_a \rightarrow p_i \rightarrow \cdots \rightarrow p_j \rightarrow v_b$. If no path is found GCA* returns with failure.
4. Otherwise, the high-level path found is expanded by a regular omniscient pathfinding algorithm such that for each step $p_i \rightarrow p_j$ a low-level path is found that connects p_i and p_j through tiles of the map M while searching only inside the area that p_i, p_j reside.

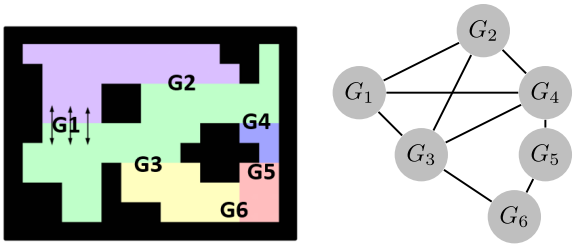


Figure 1: A small map decomposed in five regions, and a corresponding gate connectivity graph. Gate G_1 is a set of three portals, each of which is denoted by a double arrrow.

BGCA*: Belief-based gate connectivity A*

We now build on GCA* to provide a BDP approach such that each NPC maintains its own gate connectivity map capturing the beliefs of the NPC about the map. This is a practical way to capture beliefs about connectivity as it essentially filters out the local obstacle avoidance inside each region, which may be also fine-tuned at execution. Also, each NPC need only maintain beliefs about a small subset of the map, i.e., the gates, which are points of interest that are expected to map well to meaningful positions, e.g., doors or places where an obstacle may be put to obstruct the passage, or can be specified at design time to actually correspond to such.

Note that a gate connecting two regions may actually be just a couple of tiles wide, while two automatically decomposed regions may be connected through a large number of tiles, e.g., as illustrated the map of Figure 1. At the one extreme one can consider every portal as a gate (that the NPC believes to be free or blocked), and at the other extreme one can consider just one gate per two neighboring regions. In Figure 1 the gate connectivity graph is composed by six gates that abstract the connections of the regions according to the latter extreme. An edge between two gates indicates that they belong to the same region (and thus are assumed to be always reachable one to the other).

Depending on the resolution of the map, many portals can be grouped together to form a meaningful gate that corresponds to a part of the map that can be blocked during play

[1] Our definition of portals is inspired by (Goldenberg et al. 2010) but we use a pair of tiles connecting exactly two regions, instead of a single tile that may connect multiple regions. A gate then is a (part of a) gateway in the sense of (Björnsson and Halldórsson 2006).

in a way that actually obstructs navigation from one region to another. In this sense the extreme case of putting together all portals in the border between two regions in a single gate works well, as the personalized portal connectivity map would hold information about the high-level connectivity of regions and not about the actual detailed position that the NPC should enter or leave the region. Nonetheless, since in very wide borders this detail may provide a nice refinement to the path followed by a belief-driven NPC, we expect that a better approach is to group portals together into gates according to an average size of gateways in the game. For example, in the map of Figure 1 a reasonable choice for the size of a gate would be three portals.

We assume that gates are formed based on a maximum size in tiles. This then gives some bounds on the memory that each NPC needs to allocate for the personalized beliefs as well as the total memory overhead for this BDP approach. In the worst case the memory required for storing the beliefs for a single NPC is proportional to the size of the portal set $|M_P|$, which with an average decomposition algorithm is expected to be much smaller than the total number of tiles in the map. The total memory needed for n NPCs is then just the amount needed for a single one multiplied by n.

Maintaining and discarding beliefs. A generic procedure for an NPC navigating with BGCA* is one that interleaves planning and execution in a straightforward way as studied in AI in various contexts. The NPC starts with an empty set of beliefs and updates the beliefs during execution accordingly for the portals that are in his field of view. As time goes by the NPC may need to *"forget"* or discard some of his beliefs because they may have become obsolete.

At one extreme the NPC may focus solely on his perception, keeping only the beliefs that come from his field of view and discard anything else, and at the other extreme he may give strong trust to his memory, keeping all beliefs until they are revised by his perception. A middle ground is desired where beliefs are kept in memory only up to a certain time-limit so that they guide his navigation but without hurting too much the precision of his results. Note that it is actually desired that the NPC follows paths that may be obsolete at the time of execution or even fails to find a path when in fact there is one, but this should happen within an overall precision that makes his navigation reasonable.

It is important to specify how beliefs are stored and what does it mean to discard a belief about the state of a gate. In knowledge representation terms, a precise way to formalize this is by allowing the belief about a gate to range over three states: free, blocked and unknown. The information that is received by the perception of the NPC would then update beliefs about gates to free or blocked accordingly, while the discarded beliefs (due to a time-window that passed) would be updated to unknown. This allows for different search strategies, e.g., one that promotes paths that go through unknown areas and are more exploratory, and one that promotes paths with gates whose state is known.

BGCA* adopts a practical solution that simplifies the handling of unknown gates by an *open-door assumption*, i.e., the discarded beliefs are updated to beliefs that the corresponding gates are free. This is a common assumption followed by several approaches for pathfinding in unknown maps. We believe that a correct account of unknown gate states would be prove useful to generating paths that follow different dispositions, but it is out of the scope of this analysis and is left for future work. BGCA* also adopts a practical one-step strategy for discarding beliefs: essentially all beliefs are kept until the NPC gets to a point that a pathfinding request fails to find a solution. Then beliefs that are older than a specified time window are discarded.

BGCA* handles a request between a source and destination tile by interleaving planning and execution as follows.

Execution procedure:

1. Loop over step 2 until a maximum number of *repaths* is reached. If the maximum is reached then return failure.
2. Use the planning procedure to compute a path from the current tile to the destination. If a path is found then execute this path and update the beliefs for all gates that lie inside the perception *radius* of the NPC, otherwise return failure. If the destination is reached then return success. Stop the execution (and break the loop) if some gate believed to be blocked is sensed to be free or if the path execution fails.

Planning procedure:

1. GCA* is run using the personalized gate connectivity graph of the NPC that made the request. If a path is found then returns the path.
2. Otherwise, the gate connectivity graph is updated by discarding all beliefs that are older than a specified *time-window*, i.e., the corresponding gates are considered to be free. Another GCA* instance is run on the same tiles using the updated gate connectivity graph. If a path is found then returns the path, otherwise returns failure.

Note that the behavior of BGCA* is configured by three parameters: (*i*) the maximum number of *repaths* (i.e., planning and execution iterations) allowed, (*ii*) the perception *radius* of the NPC, and (*iii*) the *time-window* for maintaining beliefs. Next, we proceed to discuss a testbed for BGPA* (and BDP in general) and a set of experiments that we conducted to identify the configurations for these parameters that generate the intended results.

A BDP testbed in Unity

In order to investigate the correlation between the parameters of BGCA* and the performance of the algorithm, we have developed a testbed in the Unity game engine (unity3d.com) that automatically iterates over a set of configuration files and maps[2]. For each configuration and each map the benchmarking tool acquires a map area decomposition using the flood-filling method of (Björnsson and Halldórsson 2006), generates the gate connectivity graph, and goes over a number of BGCA* pathfinding requests as specified in

[2]The testbed is publicly available at https://github.com/THeK3nger/BDP-Benchmark

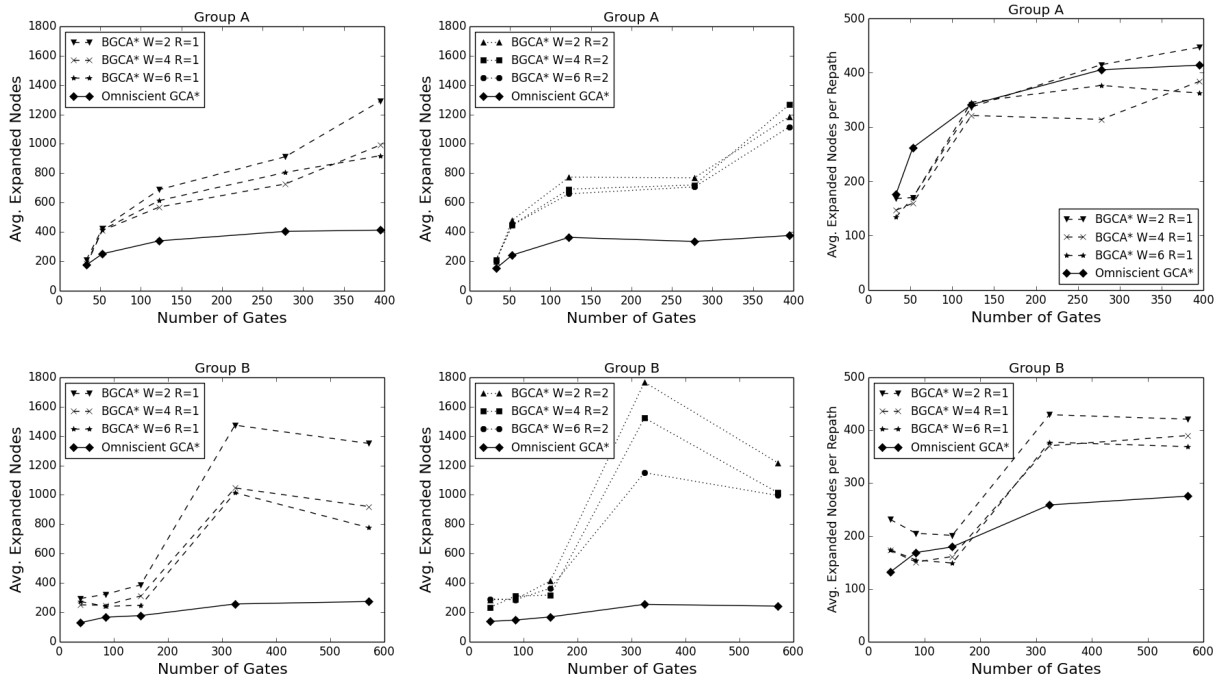


Figure 2: Total effort and effort per repath for maps in Group A (top) and Group B (bottom).

the configuration file. Unlike other benchmarks that generate paths with random start and destination tiles, here only random destinations are generated and the request uses the current position of the NPC as the start. This is needed in order to measure the use of BGCA* in a scenario where beliefs about the areas that the NPC visited recently can be used to drive the new paths requested.

The parameters that can be specified are the following:

- `max-gate-size`: the maximum size for gates.
- `iterations`: the number of destination requests,
- `initial-closed`: the ratio of closed gates,
- `shuffle-amount`: the ratio of gates to be updated,
- `shuffle-rate`: the frequency of updating gates,
- `max-repaths`: the maximum number of repaths,
- `update-radius`: the perception radius of the agent,
- `belief-window`: the time-limit for discarding beliefs.

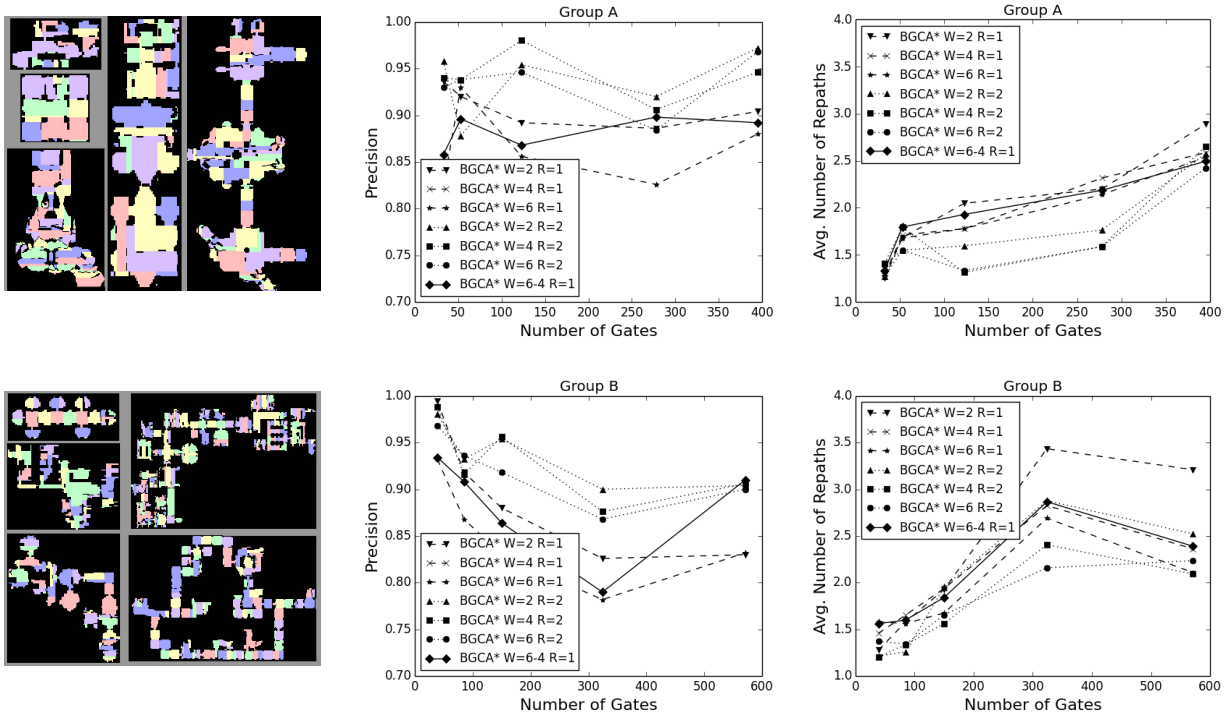
During the generation of destinations, positions that lie in the same area as the NPC are filtered out. The initial ratio of closed gates is specified in the parameters and a shuffling of them takes place every few iterations. This changes the state of the percentage of gates specified but keeps the total ratio of closed gates constant. The last three parameters correspond to the parameters of BGCA*. In particular, the update radius refers to the depth in terms of areas that the NPC can perceive, i.e., a value of 1 means that the NPC can perceive all gates in lying in the same area, while a value of 2 means that he can also perceive gates in areas connected to the one he is located. Also, as far as the time-window is concerned, since the navigation of the NPC is simulated, we chose to specify the window in terms of the number of shuffling events that have occurred. For example, if a shuffling rate of 10% is specified, then belief window of 4 would correspond to the time required on average before 40% of the map has changed in a real-time running game.

The map decomposition can be specified manually or automatically by means of a map partitioning algorithm, e.g., the *counter-based graph-partitioning* algorithm (Goldenberg et al. 2010) based on the *betweenness centrality* of the portal edges, or the flood-filling algorithm used in (Björnsson and Halldórsson 2006) for their pathfinding *gateway heuristic*. While the first algorithm provides a more robust map decomposition, the second one, which is the one used in our testbed, always provides horizontal or vertical portals that are easier to use and store. On the other hand, this partitioning method does not perform well on every map and may generate non-intuitive regions.

The testbed reports a detailed log for each configuration file and map. As we intend to compare the performance of BGCA* with an omniscient pathfinder, the testbed also performs the same requests with an omniscient PCA*. This is so that the comparison is fair, as PCA* is also hierarchical and works in the same way but in one-shot using the accurate version of the current map.

Experimental Results

For evaluating BGCA* we set up a set of experiments using the testbed of the previous section and benchmark maps from the video game "Dragon Age: Origins" provided by (Sturtevant 2012). We formed two groups of representative



maps of increasing size, each of which consisting of five maps. *Group A* contains "open" maps that allow multiple solutions to pathfinding requests, while *Group B* contains "maze-like" maps where one blocked passage may cut-off connectivity between big parts of the map.[3]

We measure the outcome of the different configurations for BGCA* wrt *effort* and *precision*. As far as the effort is concerned, we look into the number of *total expanded nodes* for the whole set of planning execution steps until the destination is reached or BGCA* quits. We also look into the number of expanded nodes for each execution step. We refer to this as the *expanded nodes per repath* and it corresponds to the effort needed before the NPC can start moving in the game map. As far as precision is concerned we focus measure the ratio of destination instances that BGCA* and the omniscient PCA* agree about success or failure. Also, we take into account the *number of repaths* per destination.

Our intention is to identify the parameters that achieve a balance between the NPC finding paths with a precision above 75% while inducing the least effort as possible and keeping the number of repaths to a small constant, e.g., 2 or 3.[4] For example, if we would allow the NPC to discard all beliefs and retry as many times as needed then he would eventually find a path if one exists. At the other extreme if we never allow the NPC to discard beliefs then at some point he may not be able to find paths outside of his area of vision when the last information he had was that the surrounding areas next to his location were blocked.

The following results show how this balance can be achieved for the two groups of maps that we identified. We performed several experiments and we report here the ones that are more suitable for drawing conclusions. For each one we set the number of iterations to 500, and for all reported sets we set the ratio of closed gates to 20% and a shuffling ratio of 10% after every destination iteration. Also, we used a fixed number for the maximum size of gates to be at most 5 portals. As far as the parameters of BGCA* are concerned we report on combinations of belief-window values of 2,4,6 and update-radius values of 1,2. Figure 2 shows the effort for these experiment sets, where "W" stands for the belief-window and "R" for the update-radius. Figure **??** shows the precision and repaths for all combinations for Group B, along with screenshots of the maps. From these graphs we can conclude the following:

- The average effort per repaths of BGCA* in the "open" maps of Group A is comparable with the one of omniscient PCA*. This means that as far as the NPC making the first move is concerned, BGCA* is not inducing any significant overhead.
- The average effort per repaths of BGCA* for the "maze-like" maps of Group B is greater than that of omniscient PCA*. However, it seems to be bounded by a factor of 2-3 times the effort of PCA*. This shows that some constant overhead is to be expected to maps of this type.
- The total effort of BGCA* is larger than that of omniscient PCA* which is expected as the NPC makes a num-

[3]Group A consists of maps: `orz500d`, `den504d`, `rmtst`, `arena`, `den101d`. Group B consists of maps: `lak302d`, `den005d`, `lak202d`, `den020d`, `orz601d`.

[4]This is also because unlike other contexts where the shortest path is desired regardless of repath steps, here we are interested in behaviors that generate only a few "re-thinking" events for the NPC that are easier to be acknowledged by the player.

ber of repaths before reaching the destination.

- There is a trade-off between total effort and precision. A lower value for the belief-window can help the agent to reach a precision above 90%, however leads to a larger total effort. On the other hand, an increase of the perception radius can help the agent achieve a higher precision with the same effort. For instance in Figure **??**, every experiment with radius 2 leads to a precision score above 90% but the effort is comparable with the effort of the corresponding experiments with a perception radius of 1. The use of a radius larger than 1 is a way to achieve better performance by giving the NPC more perception power.

The last point motivated us to explore further a slightly more adaptive version of BGCA* such that in the planning procedure instead of performing a one-shot revision (point 2), relies on an incremental revision up to a smaller belief-window. Additional experiments showed that a variant of BGCA* that uses a variable belief-window from 6 to 4 (depicted as W=6-4) has a more consistent performance wrt precision in the maps of Group B, and in fact achieves superior performance wrt precision in the maps of Group A.

As far as memory usage is concerned, the total number of gates shows the memory usage per NPC. Assuming that one bit is stored per tile and gate, a map with n tiles requires n bits for a single copy, nm bits for m agents keeping a full copy each, and km bits for our approach where k is the number of gates. E.g., for map `den005d` in group B, n is 17559 and k is 571. Note that portals and gates are identified by means of an automated procedure for decomposing the map into regions and considering all tiles in the border between regions. In practice the number of gates could be much lower if game designers identified directly those points of interest that are appropriate to be considered as gates, e.g., doors, passages, breakable walls, etc.

Conclusion and Future Work

We investigated an approach for realizing a belief-driven pathfinding (BDP) service that allows for a differentiation between the paths that non-player characters (NPCs) follow, based on their beliefs and former observations. Our approach, BGCA*, appeals to a practical map abstraction in order to provide a memory-friendly way of storing a personalized copy of the map per NPC. We provide a testbed for experimenting with the parameters of BGCA* and through our analysis we identify appropriate configurations for employing BGCA* in practice. As BGCA* can be built on top of a pre-existent regular pathfinding service, it is possible to use this approach only for some particular NPCs.

We believe that this type of belief-driven deliberation and decision making is necessary for increasing the believability of characters in games, as this is perceived by means of the navigation and actions of the characters in the game-world. Moreover, we believe that exploring such approaches opens up many opportunities for developing more advanced behaviors that rely on each NPC holding a personalized view of the properties of the game-world that is separated from the real (updated and completely specified) state of affairs.

For our future work we intend to explore a more refined notion of beliefs that distinguishes between gates that are known to be free and those that are unknown (unlike the existing account that follows the open-door assumption). We also want to investigate the middle ground between pathfinding and goal-oriented action planning toward hybrid approaches such that (beliefs about) properties of the game-world may affect navigation, for instance taking into account the ability to go to a particular region and get a key in order to open a gate that blocks the path to the destination.

Finally, in addition to evaluating the BDP agent from an effort and performance point of view, it is important to define appropriate metrics for measuring the believability of the NPC behavior wrt to a belief-consistent point of view.

References

Abraham, I.; Fiat, A.; Goldberg, A. V.; and Werneck, R. F. 2010. Highway dimension, shortest paths, and provably efficient algorithms. In *Proc. of SODA'10*, 782–793.

Björnsson, Y., and Halldórsson, K. 2006. Improved Heuristics for Optimal Path-finding on Game Maps. In *In Proc. of AIIDE'06*.

Björnsson, Y.; Bulitko, V.; and Sturtevant, N. R. 2009. TBA*: Time-Bounded A*. In *Proc. of IJCAI'09*, 431–436.

Botea, A.; Müller, M.; and Schaeffer, J. 2004. Near optimal hierarchical path-finding. *Journal of game development*.

Felner, A.; Sturtevant, N.; and Schaeffer, J. 2009. Abstraction-Based Heuristics with True Distance Computations. In *Proc. of SARA'09*, 1–8.

Goldenberg, M.; Felner, A.; Sturtevant, N.; and Schaeffer, J. 2010. Portal-based true-distance heuristics for path finding. In *Proc. of SoCS'10*.

Hernández, C., and Baier, J. A. 2011. Fast subgoaling for pathfinding via Real-Time search. In *Proc. of ICAPS'11*.

Koenig, S., and Likhachev, M. 2005. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics* 21(3).

Koenig, S. 2001. Agent-centered search. *AI Magazine* 22(4):109.

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

Stentz, A. 1995. The focussed D algorithm for real-time replanning. In *In Proc. of IJCAI'95*, 1–8.

Sturtevant, N., and Buro, M. 2005. Partial pathfinding using map abstraction and refinement. In *Proc. of AAAI'05*.

Sturtevant, N., and Jansen, R. 2007. An analysis of map-based abstraction and refinement. In *Proc. of SARA'07*.

Sturtevant, N. R. 2007. Memory-Efficient abstractions for pathfinding. In *Proc. of AIIDE'07*, 31–36.

Sturtevant, N. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2):144 – 148.