2010

Perceptually Realistic Behavior through Alibi Generation

Ben Sunshine-Hill
University of Pennsylvania, bsunshin@seas.upenn.edu

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

Perceptually Realistic Behavior through Alibi Generation

Abstract
Real-time pedestrian simulation for open-world games involves aggressive behavior simplification and culling to keep computational cost under control, but it is diffficult to predict whether these techniques will become unrealistic in certain situations. We propose a method of perceptually simulating highly realistic pedestrian behavior in virtual cities in real- time. Designers build a highly realistic simulation, from which a perceptually identical "perceptual simulation" is generated. Although the perceptual simulation simulates only a small portion of the world at a time, and does so with inexpensive approximations, it can be statistically guaranteed that the results are perceptually indistinguishable from those of the original simulation.

Disciplines
Artificial Intelligence and Robotics | Graphics and Human Computer Interfaces | Numerical Analysis and Scientific Computing

Comments
Suggested Citation:
Sunshine-Hill, B. and N.I. Badler. (2010). "Perceptually Realistic Behavior through Alibi Generation" *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. October 11-13, 2010. Stanford, California, USA.

Perceptually Realistic Behavior through Alibi Generation

Ben Sunshine-Hill and **Norman I. Badler**
Center for Human Modeling and Simulation
Computer and Information Science
University of Pennsylvania
bsunshin@seas.upenn.edu, badler@seas.upenn.edu

Abstract

Real-time pedestrian simulation for open-world games involves aggressive behavior simplification and culling to keep computational cost under control, but it is diffficult to predict whether these techniques will become unrealistic in certain situations. We propose a method of perceptually simulating highly realistic pedestrian behavior in virtual cities in real-time. Designers build a highly realistic simulation, from which a perceptually identical "perceptual simulation" is generated. Although the perceptual simulation simulates only a small portion of the world at a time, and does so with inexpensive approximations, it can be statistically guaranteed that the results are perceptually indistinguishable from those of the original simulation.

1 Introduction

In recent years, the "open world" style of video game, in which players wander freely through a large populated area, has become very popular. Most agents in open world games are not enemies, but residents, going about their life as the backdrop of the main gameplay. At times they are drawn into the gameplay, and must react to the player's behavior and to the indirect results of that behavior. In order to maintain suspension of disbelief in these games, it is important that agents behave realistically in all situations, giving the impression of a living city. Patterns of unrealistic or unlikely behavior negatively impact the realism of the game.

The increased demands for agent realism and large, open worlds present practical difficulties for game developers. Memory and processing power is limited on video game hardware, and much of it is devoted to graphics, sound, and other resource-intensive tasks. As the size and complexity of the virtual population grows, it quickly becomes intractable to simulate the virtual population in real-time.

Luckily, real-time simulation of the entire virtual population is not required for these games. Because the player can only see a small portion of the world at a time, he cannot evaluate the overall realism of the world, only that portion which is visible to him. As far as the player is concerned the world is realistic as long as the visible portions of it at any given time are realistic.

These twin demands — thrifty use of computational resources, and the appearance of realistic simulation — have driven the development of a variety of clever methods (tricks) which maintain the *perception* of realism while reducing the actual realism. For instance, out-of-view portions of the world will have their agents deleted entirely, with a set number randomly created when the player returns to the vicinity. Even agents which are invisible because they are behind the character may be frozen in place in order to free up simulation resources. Agents will often have only a veneer of realistic behavior, appearing at first glance to be moving towards some desired destination while actually turning randomly at each intersection.

Such tricks are difficult to get just right. User testing is required to determine whether they compromise realism in unanticipated ways. Parameter tweaking is often necessary to arrive at a realistic configuration, and the parameters may not map cleanly to quantities which designers wish to directly control. For example, if the rate at which agents enter an area does not match the initial population, the agents' speed of motion, and the geometry of the area, agents may exit the area at a greater rate, and the player will perceive that an area is always well-populated when he arrives, only to mysteriously empty out soon afterwards. In contrast, if the rates of exit and entry are directly tied to each other, the player may perceive that a small area always happens to have three people in it, with one entering every time another leaves. Agents with only random turning will quickly reveal their unrealistic behavior if followed by the player. These tricks may produce realistic results early in development, only to become unrealistic later due to designers' varying demands on the world.

We propose an alternative approach, whereby designers build a highly realistic whole-world pedestrian simulation, from which a perceptually identical "perceptual simulation" is generated. Although the perceptual simulation simulates only a small portion of the world at a time, and does so with inexpensive approximations, it can be statistically guaranteed that the results are perceptually indistinguishable from those of the original simulation.

The basis of the simulation is the probabilistic modeling of observable information about an agent. Agents are initially created with only a minimal set of information, and any information which is later required for a higher level of detail is created on demand in a manner which ensures consistency with previous observations of the agent. We refer to such an incremental addition of information about an agent

as an **alibi**, which fills in details of the agent's goals and back-story to give the impression that his behavior is driven by concrete needs and goals. The act of giving an agent an alibi causes no immediately visible changes, but allows the realism of the agent's actions to hold up under closer scrutiny from the player.

Related work

Simulation level-of-detail The application of level-of-detail (LOD) techniques to simulation, rather than rendering, was pioneered by Carlson and Hodgins (1997). Sung, Gleicher, and Chenney (2004) applied simulation level-of-detail (SLOD) by freezing out-of-view agents, advancing them by large steps at infrequent intervals to maintain the appearance of continued action. These approaches minimized but did not eliminate the computational burden of simulating an individual out-of-view agent, and were targeted to situations where the long-term plausibility of all individuals' actions was important.

Approaches to agent level of detail O'Sullivan et al. (2002) developed the ALOHA framework to apply LOD techniques to both simulation and rendering of humans. MacNamee et al. (2002) extended this framework to "role passing": the temporary assumption of specific roles by previously generic agents; their work is philosophically similar to ours, in that it leverages role passing as necessary to mitigate the perception of generic behavior. Likewise, Brom, Šerỳ, and Poch (2007) used hierarchical representation of goals to determine current behavior from preexisting goals as necessary. These approaches are limited in their ability to simulate large areas, because the full population exists at all times.

Niederberger and Gross (2005) dynamically prioritized agents within a group to maximize the perception of realism. Osborne and Dickinson (2010) used hierarchical groups to optimize crowd simulation with the Reynolds (1987) flocking model. These approaches improve the perceptual quality of pathing, but do not address realism in long-term behavior.

Population distribution modeling Stylianou, Fyrillas, and Chrysanthou (2004) synthesized population distributions from a roadmap by performing random walks and sampling agent positions in the steady state. Plausible population densities could thus be created without any semantic information about land use, and the system allowed manual modification of population densities to correct unrealistic areas. They briefly discuss the use of as-needed agent creations to minimize simulation time.

Haciomeroglu, Laycock, and Day (2008), likewise, found shortest paths between all road segment pairs in an unannotated road graph (using a distance metric which minimized turning angles) and based population density on the resultant road utilizations. They then dynamically inserted pedestrians into in-view and nearby areas to maintain these densities, giving agents destinations which took them into and then back out of the simulation area.

Population simulation in industry practice Brockington (2002) discussed the use of LOD to manage agent simulation time in the video game *Neverwinter Nights*; agents still existed when out of view, but could be frozen or simulated using large time-steps. Adzima (2001) used a fixed radius around the player for ambient car simulation in *Midtown Madness 2*, with cars turning randomly and being replaced as necessary to keep them close to the player. More recent open world games, such as *Grand Theft Auto IV*, tend to use weighted random walks for visible characters, and proactively remove occluded characters and objects from memory. This leads to populations which appear realistic in the aggregate, but individual agents act unrealistically over time.

Organization

The remainder of the paper is organized as follows. First we give a motivating example of an agent behavior simulation whose details will serve to illuminate various aspects of our approach. Secondly, we describe the form of our perceptual simulation strategy, which can perceptually approximate these simulations. We then derive the data which is needed to perceptually simulate our motivating example, and describe how this data is used. Finally, we present the results of our simulation strategy, and describe how the method can be adapted for various complications not given in our motivating example.

2 Motivating example

Here we present a simple but detailed pedestrian behavior model, which describes where agents go, when they go there, and why they go there. This example is intended to showcase various features that are difficult to model with perceptual simulation. By itself this model is unsuitable for actual use in a large, heavily populated world, as it requires full simulation of out-of-view agents at all times, thereby motivating the need for a perceptually identical real-time simulation. We do not intend this model to be perfectly realistic: it ignores certain important dimensions, like time of day, in favor of a straightforward presentation. We describe how some of these additional dimensions can be added in section 7.

World schema

The world is partitioned into walkable region "cells", which are connected to each other by "portals". Agents walk from cell to cell through the portals. In a city model, a length of sidewalk or a crosswalk will be represented as a cell.

In addition to walking between portals, agents may exit and enter buildings. The entryways of these buildings are known as "targets". Targets are categorized into a dozen or so "types", representing what the building is used for. Examples of target types in our model are "house", "office building", "restaurant", and "grocery store". (A building with multiple uses will be represented as several co-located targets.) The number of targets in the world is much greater than the number of target types. Targets are additionally organized by which cell they are in.

All portals and targets in a given cell can be directly reached from all other portals and targets in the same cell, without passing through any other portals. In our navigation graph, portals and targets form the nodes of our world, and any two nodes which are in the same cell have an edge

between them. Each portal is present in two cells, and is connected to all other nodes in both. We refer to a directed edge in our graph as a "segment", a linear-shaped area along which pedestrians may be found, walking in a particular direction away from one node and towards another. We denote by n the number of targets in the world, by m the number of target types, and by s the number of segments in the world.

Agent behavior

An agent, at any given time, will either be inside a target, or will be travelling from his previous target to his next one. Agents always travel along the shortest path through the world to their next target. (For distance calculations, portal locations are measured as the midpoint between their edges; for pedestrian simulation along sidewalks, this is not a significant oversimplification.) Upon arriving at a target, an agent spends a randomly determined amount of time there, which we model using a normal distribution. Afterwards, he chooses a "goal" randomly, from a distribution conditioned on the type of the target he is currently at. A goal consists of a type of a target to choose as a destination, as well as instructions on how that target is determined; the agent may have a particular preferred target for that goal which is determined when the agent is created (for instance, the agent's own home will always be chosen for the "go home" goal), may choose the closest target of that type (as would be appropriate for grocery stores and other such fungible destinations), or may uniformly randomly choose a target of that type (for, say, a courier delivery to an office building). Certain goals may be round-trip; after leaving a target reached by a round-trip goal, the agent will return to the target from which he had previously departed. Other than returning from round-trips, the agent's choice of goals is independent of his previous goals and the goals of other agents; a target type therefore has an associated goal distribution, consisting of the probability of choosing each goal next.

We simulate low-level pedestrian motion control using a simple social forces model. The method is compatible with other crowd motion controllers, such as the many cited in (Pelechano, Allbeck, and Badler 2008), and in section 7 we describe the practicalities of changing it.

3 Components of perceptual simulation

Maintaining the perceptual realism of our real-time simulation requires two main tasks: Adding and removing agents as necessary to ensure the realism of aggregate behavior without overtaxing simulation resources, and generating alibis to upgrade agents to keep their behavior realistic under scrutiny.

Creating and deleting

Agents are created in two circumstances: *in media res* when a new area of the world comes into view, and entering at random intervals from targets or areas of the world which are out of view.

For the first circumstance, whenever a new cell comes partially into view, agents are generated along each of its segments. An individual agent from among the entire population of the world will have a particular steady-state probability of being on a given segment when that segment's tile comes into view, and (without knowledge of agents' bound targets) each agent in the world will have the same such probability. The number of agents present on a segment when it comes into view, therefore, can be modeled as a binomial process, with n the population of the world not currently visible and p the probability that a given agent is on the segment at a given time. Simple and efficient algorithms to generate binomial variates are well-known; for several examples refer to (Devroye 1986). Because of the high population of the world and the high number of segments in the world, this distribution can also be approximated by a Poisson distribution if desired (Devroye 1986).

For the second circumstance, each segment beginning in a target, and each segment whose starting portal is out of view, has a given probability during each small time-step of having an agent appear, and in the steady state of the simulation this probability is independent of previous appearances (with the exception of social forces, discussed later). As a result, the random variable representing the time before the next agent appears is exponentially distributed, with λ the inverse of the expected time between arrivals on that segment (Devroye 1986), and can be sampled in constant time. This arrival process is not applicable to segments whose starting node is a portal in view of the player — that is, whose companion cell is visible — as these segments will have already-generated agents arriving from the companion cell.

At the moment when a new tile comes partially into view after being completely out of view, therefore, three tasks must be performed: The number of agents along each segment within the tile must be sampled and the resultant number of agents generated, and each segment whose starting node is a target or a portal whose other tile is still out of view must have a "next arrival time" sampled and remembered. Arrival times for all segments of this type are placed in a priority queue. Whenever a scheduled arrival occurs, a new agent is generated at the beginning of that segment, and a new exponential variate is sampled for the segment and placed back in the priority queue. Finally, any segments currently in the arrival priority queue whose starting node was a portal on that cell are removed from the queue, as their agents will henceforth arrive in already-generated form.

Cells which pass entirely out of view must continue to be simulated for a short time, or a return to the cell may produce obvious discrepancies. The cell must continue to be simulated for at least as long as it takes for the set of agents in the cell to change entirely. Following (Sung, Gleicher, and Chenney 2004), this burden may be alleviated by freezing simulation of the agents, and advancing them at once only if and when the cell comes back into view. Likewise, agents which pass from a visible cell into an invisible cell must continue to be simulated for some time, but can also be simulated in a frozen manner.

Alibi generation

While the previous section suffices to distribute momentarily plausible agents over the visible portion of the world, it does not by itself give them plausible long-term behavior. A short-term solution is to store the conditional probability table for transitioning to a given segment given the agent's last few segments; this table can be stored in a compressed form

(Gagie 2006) but the memory requirements quickly become intractable as the segments grow longer. If the player follows an agent for any significant period of time, interacts with the agent in a nontrivial manner, or inspects his behavior in any other way (for instance, by blocking the agent's segment and forcing him to replan) an actual goal destination is required.

The agent's destination, and any other information about the agent generated some time after his creation, forms the agent's *alibi*. As far as the player is concerned, this information had always existed but was not known to him. An alibi, therefore, must be consistent with all previously observed information about the agent, and the geometry of the world and the distribution and habits of agents in the world in general. Additionally, over time there must be no observable patterns to alibis, even those generated for different agents in exactly the same situation. In short, any alibi must be an independent, fair sample from the conditional distribution of possible alibis given observed behavior. In section 5, we show how this may be done without explicitly storing the conditional probability table.

4 Deriving the probabilities

We now return to the motivating example in section 2, and show how the distribution parameters mentioned above can be analytically determined. The general approach is to determine the different circumstances under which an agent can have a particular alibi (source and destination), and then to weight and sum these circumstances to determine the prior probability of having that alibi. We then filter to determine the posterior probability of an alibi given the agent's observed path.

First, some definitions. For a given pair of nodes i, j, we denote the distribution of the time required for an agent to travel from target i to target j by D_{ij}. (Here and for the remainder of this paper, we use i and j as target indices, and k and l as target type indices.) We denote the type of target i by $S(i)$, and the number of targets of type k in the world by $\|S_k\|$. We denote the conditional probability of choosing a goal with type l from target type k, assuming the previous goal was not round-trip, by p^G_{kl}; we further split this into one-way and round-trip probabilities p^O_{kl} and p^R_{kl}, which are the normalized conditional probabilities of choosing a goal with type l given that the previous one-way goal was of type k and that the next goal is constrained to be, respectively, one-way or round trip. We denote the conditional probability of picking any one-way goal from a target of type k by p^R_k, giving the identity $\sum_{l=1}^{m} p^R_k p^R_{kl} + \sum_{l=1}^{m}(1-p^R_k)p^O_{kl} = 1$. We denote the distribution of the waiting time at a target of type k by W_k. Finally, we denote by $closest(i,l)$ the closest target to i of type l.

Recall that goals may be round-trip, forcing a return to the original target after leaving the destination, or one-way, with future goals sampled independently of past goals. Consider the sequence of targets which an agent reaches as a result of one-way goals only, disregarding for the moment the sequence of round-trip goals which he may perform between each one-way goal. This continuous-time process of one-way goals may be modeled as a *semi-Markov process* (Ross 1996). A semi-Markov processes is a random process which, like a standard Markov chain, moves from state to state with transition probabilities independent of all but the most recent state, but which will stay in a particular state for a real-valued period of time whose distribution depends on the current and/or the next state. The limiting behavior of a semi-Markov process can be described by the tuple $\langle p_{ij}, F_{ij}\rangle$, where p_{ij} is the probability of transitioning to state j given current state i (the "jump process"), and F_{ij} is the distribution of the time for that transition to occur. We denote an agent's current state in the semi-Markov process as the last target the agent reached by a one-way goal, irrespective of whether they are still inside that target as well as of whether they have performed other round-trip goals since reaching it. For this process, $p_{ij} = p^O_{S(i)S(j)} / \|S(j)\|$ if $S(j)$ is not fungible, $p^O_{S(i)S(j)}$ if it is fungible and $closest\,(i, S\,(j)) = j$, and 0 otherwise. (Without prior information on bound targets, there is no need to differentiate between goals to bound targets and goals to uniformly chosen targets.) Note that the jump process is independent of all but the target types, because the agent chooses from between goals directly, but the transition time is dependent on the specific targets, because different targets may take more or less time to transition due to their positioning in the world.

While p_{ij} is simple to calculate as above, the calculation of $F_{ij}$is more complex. We consider the agent's actions upon arriving at a target i, of type k. First he waits for a time given by W_i. He will then carry out zero or more round trip goals; for each goal to some target j, he walks to that goal, taking time given by D_{ij}, waits at that goal for time W_j, walks back to the original target taking time D_{ji}, and then waits again at target i for time W_i. Finally, he walks to his next one-way destination j for time D_{ij}, at which point he is in the next state and the clock stops. We denote the round-trip travel time for a round-trip destination to a target of type l by D^R_{il}, remembering that this may be dependent on whether l is a fungible target type. The number of round-trip goals he chooses before leaving on a one-way goal is given by a negative binomial random variable (Ross 1996) with $r = 1$ and success probability $p = p^R_k$, with expected value $\frac{p}{1-p}$, and the proportion p^R_{kl} of those are specifically round-trip goals to target type l; we denote the number of those round-trips taken by N_{kl}. Organizing one-way goals by their target type, we thus have

$$\mathrm{E}[F_{ij}] = \mathrm{E}[W_k] + \sum_{l=1}^{m} \mathrm{E}[N_{kl}]\left(\mathrm{E}\left[D^R_{il}\right] + \mathrm{E}[W_l] + \mathrm{E}[W_k]\right) + \mathrm{E}[D_{ij}]\,. \quad (1)$$

It remains to calculate these expected values. $\mathrm{E}[W_k]$ is simply the mean of the normal distribution, and $\mathrm{E}[N_{kl}]$=$p^R_{kl}\left(p^R_k / \left(1-p^R_k\right)\right)$, as above. $\mathrm{E}\left[D^R_{il}\right]$ can be calculated as $\min_{j,S(j)=l} \mathrm{E}[D_{ij}] + \mathrm{E}[D_{ji}]$ if l is fungible, and $\|S_l\|^{-1} \sum_{j,S(j)=l} \mathrm{E}[D_{ij}] + \mathrm{E}[D_{ji}]$ otherwise.

This is sufficient to describe the steady-state probability of the semi-Markov process. First, the steady state of the jump process σ is given by the equation $\sigma_j = \sum_{i=1}^{n} \sigma_i p_{ij}$ (recall the definition of p_{ij} as above), and can be found as the eigenvector of $[p_{ij}]$ with eigenvalue 1. We can marginalize F_{ij} over the destination target as $F_i = \sum_{j=1}^{n} p_{ij} F_{ij}$, and

then entirely as $F = \sum_{i=1}^{n} \sigma_i F_i$. The steady state of the semi-Markov process itself is then $\hat{\sigma}_i = \sigma_i \frac{\mathrm{E}[F_i]}{\mathrm{E}[F]}$. The probability $\hat{\sigma}_i$ of having last visited one-way target i at any given time can be further broken up; for instance, the probability of having just reached target i and being currently waiting inside it before leaving either for the first round-trip goal or for the next one-way goal is $\sigma_i \frac{\mathrm{E}[W_i]}{\mathrm{E}[F]}$. Additionally, $\hat{\sigma}_i$ can be used to determine the probability of being on one's way to, or coming back from, a particular goal. Since any agent who is not inside a target will be on a one-way goal, going to a round-trip goal, or coming back from a round-trip goal, we can find the summed joint probability of being on one's way from target i to target j for any reason as

$$P(i,j) = \alpha_{ij} + \beta_{ij} + \gamma_{ij}, \text{ where} \tag{2}$$

$$\alpha_{ij} = \hat{\sigma}_i p_{ij} \frac{\mathrm{E}[D_{ij}]}{\mathrm{E}[F_{ij}]}, \tag{2a}$$

$$\beta_{ij} = \hat{\sigma}_i \delta_{ij} \frac{\mathrm{E}\left[N_{S(i)S(j)}\right] \mathrm{E}[D_{ij}]}{\mathrm{E}[F_i]}, \tag{2b}$$

$$\gamma_{ij} = \beta_{ji}, \text{ and} \tag{2c}$$

$$\delta_{ij} = p^R_{S(i)S(j)} \cdot \begin{cases} \|S(j)\|^{-1} & \text{if } S(j) \text{is not fungible,} \\ 1 & \text{if } closest(i, S(j)) = j, \text{ or} \\ 0 & \text{otherwise.} \end{cases} \tag{2d}$$

From here, we can easily find the joint probability of going from i to j and also being on a segment $\{a,b\}$ as

$$P(i,j,\{a,b\}) = P(i,j)\, SP_{ij}(\{a,b\})\, \frac{\mathrm{E}[D_{ab}]}{\mathrm{E}[D_{ij}]}, \tag{3}$$

with $SP_{ij}(\{a,b\})$ equal to 1 if $\{a,b\}$ is on the shortest path from i to j and 0 otherwise, and can sum over i and j as

$$P(\{a,b\}) = \sum_{i=1}^{n} \sum_{j=1}^{n} P(i,j,\{a,b\}) \tag{4}$$

which gives us the parameter for our binomial and exponential variates, to sample populations and arrival intervals. (This can be extended to observed paths consisting of multiple segments.)

5 Alibi Sampling

In this section we demonstrate how to use the above derivations for practical alibi sampling. While it is straightforward to find $\mathrm{P}(i,j|\,\{a,b\})$ for an individual situation, sampling from this distribution would be more difficult. It would be possible to precompute all values and store their cumulative sums, and sample using a binary search, but this would require $O\left(n^2 s\right)$ space for single-segment observed routes and even more for longer observed routes, far from practical for worlds of large size. (It is important to sample both i and j, in case the resultant alibi is the first half of a round trip.) Instead, we express this probability with the equivalence $\mathrm{P}(i,j|\,\{a,b\})=\mathrm{P}(i,j,\{a,b\})/\mathrm{P}(\{a,b\})$. Note from equations 1–3 that this quantity is entirely dependent on $\mathrm{E}[D_{ij}]$, $SP_{ij}(\{a,b\})$, $\mathrm{E}[D_{ab}]$, $\mathrm{E}\left[D^R_{il}\right]$, $\mathrm{E}[W_k]$, $\mathrm{E}[F_i]$, $P(\{a,b\})$, $\hat{\sigma}_i$, $closest(i,l)$, $\mathrm{E}[N_{kl}]$, p^O_{kl}, and $\|S_k\|$. Of these, the first two can be determined through shortest path searches, the third is taken directly from the world, and the remainder can be precomputed and stored with size $\Theta\left(s + m^2 + mn\right)$ in the number of segments, target types, and targets. (Recall that $m \ll n$.)

This approach does not, however, allow for a binary search-based sampling. Instead we use the Metropolis-Hastings algorithm (Hastings 1970) to sample from this distribution based only on the ability to calculate the ratio of individual probabilities, and a function to iteratively perturb i and j randomly in space. This algorithm uses a Markov chain whose steady state distribution is the desired distribution, sampling from it after a burn-in period which allows the distribution to converge. For the perturbation function, we store at each target a table of the nearest q targets, with one-way transitions pruned to avoid a division by zero in the transition probability ratio. The probability ratio is calculated from the two i,j pairs as in equation 3, noting that $P(\{a,b\})$ and $\mathrm{E}[F]$ cancel out. $\mathrm{E}[D_{ij}]$ and $SP_{ij}(\{a,b\})$ are iteratively maintained using A* searches from a to i, from b to j, and from i to j; the first two change only their goal nodes, and thus computations can be reused by keeping the closed list and lazily recomputing the heuristic for all nodes on the open list. For the third search, MT-Adaptive A* search (Koenig, Likhachev, and Sun 2007) shows promise for speeding up computation, but we have not yet implemented this method.

6 Results

We have tested the perceptual simulation described using a world with approximately 400 targets, 1200 segments, and 20,000 agents (approximately 80% of whom were inside at any given time), on a 3.6 GHz dual-core Intel processor, with computational power roughly equivalent to current-generation video game consoles. The persistent memory overhead of the precomputed data, not including data about the geometry of the world, was approximately 52 kB, scaling roughly linearly with the number of targets. Sampling the population of all segments in a cell took less than 0.1 ms. Using the reverse Kullback-Liebler divergence to monitor the convergence of the Metropolis-Hastings chain to the equilibrium, we found satisfactory convergence for alibi sampling with 500 iterations and transition tables of size $q = 50$. The optimum transition table size was dependent on the number of iterations (Figure 1). Alibi sampling took approximately 9 ms; this computational load can easily be spread over hundreds of frames if desired, making the additional computational burden affordable for even the stingiest of AI time-slices, and can be used as an anytime algorithm in especially processing-intensive situations.

Alibi sampling, therefore, produces agents whose observed behavior is virtually indistinguishable from those in the original simulation model. Players would have to analyze the behavior of hundreds or thousands of agents in order to determine whether the simulation they were viewing was the original simulation or the perceptual simulation, an activity which is not usual for video game players.

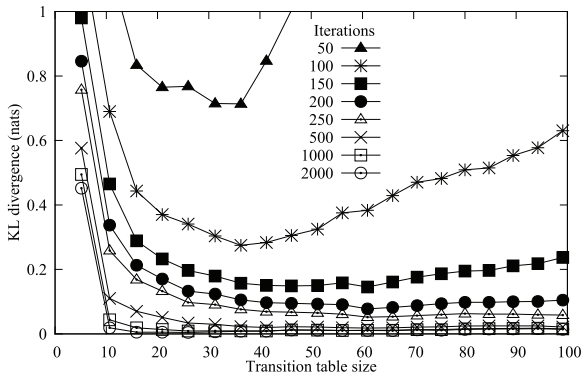

Figure 1: Kullback-Liebler divergence of alibi sampling versus ground-truth probabilities, under varying Metropolis-Hastings transition table size and number of iterations.

7 Extensions

As mentioned earlier, our motivating example is deliberately simplified for clarity of exposition. Here we discuss elaborations to the agent behavior model that may be desired, and how they can be reflected in the perceptual simulation.

Heterogeneous agents In the motivating example, all agents have the same goal probabilities and uniformly bound targets. In the real world, however, agents are obviously heterogeneous: From a person's clothes, one can often reasonably infer his social status, his career, or his current goal. This would be represented in the original simulation description by a separate population of each type of agent, differing goal probabilities, and non-uniformly bound targets. In the perceptual simulation, this would manifest itself as a distribution of agent types over each segment (an agent's type being sampled immediately after his creation) and per-agent-type multipliers in equations 2a, 2b, and 2d. Non-uniformly bound targets can also be used to simulate agents' bound targets tending to be nearby each other, as for agents who live and work in the same neighborhood.

Time of day Agent behavior in the motivating example is independent of time of day, but this is unrealistic given habitual daily routines. Moving to a time-of-day-dependent model entails replacing the semi-Markov model with a non-homogeneous semi-Markov model, which significantly complicates the calculation of $P(\{a,b\}\,|t)$ (now conditioned on the time of day) but does not make it intractable. This table and certain others mentioned in section 5 can be precompiled for many times of day and linearly interpolated, with only the bracketing two copies required to be loaded in memory at any given time.

Crowded agent simulation In our calculation of $P(i,j,\{a,b\})$ we have assumed that $\mathrm{E}[D_{ab}]$ is independent of the population of segment ab and proportional to the length of the segment. For congested areas, however, this may not be the case. For pedestrian motion models which allow realistic simulation of congested areas, a more exact heuristic based on cell populations must be found and used. Since this introduces a cyclic dependency between $\mathrm{E}[D_{ab}]$ and $P(i,j,\{a,b\})$, iterative methods will be required, and in certain contrived situations a solution may not even exist.

8 Future work

The largest portion of computation was taken by the A* path searches, suggesting that optimizations in that area could greatly improve performance. Additionally, it may be possible to generate *partial alibis*, which specify the neighborhood to which the agent is headed but not the particular target. In games already using hierarchical path-planning algorithms, this could reduce pathfinding time while increasing the Metropolis-Hastings acceptance rate.

Acknowledgments

This work was supported by a Lockheed Martin Strategic Technology Thread Grant.

References

Adzima, J. 2001. AI madness: Using AI to bring open-city racing to life. *Game Developer Magazine* January 2001.

Brockington, M. 2002. Level-of-detail AI for a large role-playing game. In *AI Game Programming Wisdom*. Charles River Media. 419–425.

Brom, C.; Šerỳ, O.; and Poch, T. 2007. Simulation level of detail for virtual humans. In *Proc. IVA 2007*. Springer.

Carlson, D., and Hodgins, J. 1997. Simulation levels of detail for real-time animation. In *Proc. Graphics Interface 1997*.

Devroye, L. 1986. *Non-Uniform Random Variate Generation*. Springer-Verlag.

Gagie, T. 2006. Compressing probability distributions. *Information Processing Letters* 97(4):133–137.

Haciomeroglu, M.; Laycock, R.; and Day, A. 2008. Dynamically populating large urban environments with ambient virtual humans. *Computer Animation and Virtual Worlds* 19(3-4):307–317.

Hastings, W. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57(1):97–109.

Koenig, S.; Likhachev, M.; and Sun, X. 2007. Speeding up moving-target search. In *Proc. AAMAS 2007*.

MacNamee, B.; Dobbyn, S.; Cunningham, P.; and O'Sullivan, C. 2002. Men behaving appropriately: Integrating the role passing technique into the ALOHA system. In *Proc. AISB 2002*, 59–62.

Niederberger, C., and Gross, M. 2005. Level-of-detail for cognitive real-time characters. *The Visual Computer* 21(3):188–202.

Osborne, D., and Dickinson, P. 2010. Improving games AI performance using grouped hierarchical level of detail. In *Proc. AISB 2010*.

O'Sullivan, C.; Cassell, J.; Vilhjalmsson, H.; Dingliana, J.; Dobbyn, S.; Mcnamee, B.; Peters, C.; and Giang, T. 2002. Levels of detail for crowds and groups. In *Computer Graphics Forum*, volume 21, 733–742.

Pelechano, N.; Allbeck, J.; and Badler, N. 2008. *Virtual Crowds: Methods, Simulation, and Control*. Morgan & Claypool Publishers.

Reynolds, C. 1987. Flocks, herds and schools: A distributed behavioral model. In *Proc. SIGGRAPH 1987*, 25–34.

Ross, S. 1996. *Stochastic Processes*. Wiley and Sons, second edition. 213–218.

Stylianou, S.; Fyrillas, M.; and Chrysanthou, Y. 2004. Scalable pedestrian simulation for virtual cities. In *Proc. VRST 2004*, 72.

Sung, M.; Gleicher, M.; and Chenney, S. 2004. Scalable behaviors for crowd simulation. In *Computer Graphics Forum*, volume 23, 519–528.