Data-driven Sequential Goal Selection Model for Multi-agent Simulation

Wenxi Liu[1], Zhe Huang[1], Rynson W. H. Lau[1], and Dinesh Manocha[2]

[1]Department of Computer Science, City University of Hong Kong
[2]Department of Computer Science, University of North Carolina at Chapel Hill

Abstract

With recent advances in distributed virtual worlds, online users have access to larger and more immersive virtual environments. Sometimes the number of users in virtual worlds is not large enough to make the virtual world realistic. In our paper, we present a crowd simulation algorithm that allows a large number of virtual agents to navigate around the virtual world autonomously by sequentially selecting the goals. Our approach is based on our sequential goal selection model (SGS) which can learn goal-selection patterns from synthetic sequences. We demonstrate our algorithm's simulation results in complex scenarios containing more than 20 goals.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: crowd simulation, data-driven animation, goal selection

1 Introduction

With the popularity of massive multiplayer online role-playing games (MMORPGs) and distributed virtual worlds, users now have access to larger virtual environments: for example, virtual campuses, virtual shopping malls, and even virtual cities. Recent developments in multi-modal interfaces such as realtime and high-fidelity rendering mean that users can experience a high level of immersion and presence in these virtual worlds. Unlike in the real world, however, virtual worlds with only a few human-like agents may not be adequately realistic. For example, in online games, the presence of nonplayer characters (NPCs) plays a vital role in enhancing the virtual environment's realism. Hence, it is useful to develop techniques to automatically simulate a large number of pedestrians or virtual agents in a large virtual environment.

Various approaches have been proposed to simulate large, heterogeneous crowds [Reynolds 1999; Treuille et al. 2006; Ju et al. 2010]. In crowd simulation, each virtual agent's path is computed in a distributed manner, and its movement is guided by goals. Therefore, given a goal position, the agent updates its position based on global navigation and local collision avoidance techniques to move towards the desired destination [Curtis et al. 2014]. In current crowd simulation systems, the goal positions are manually determined by the users or designers, or are generated using behavior models. But these techniques do not scale to large crowds. As the virtual environment becomes larger and more complex, it becomes increasingly more difficult and tedious to manually assign goals to each of the virtual agents. In addition, in large virtual worlds, virtual agents should behave like real people: guided not by a single goal, but by a sequence of goals. Hence, a method is needed to efficiently, realistically, and automatically simulate agent goal selection. In recent years, researchers have been paying attention to the crowd behavior simulation [Yu and Terzopoulos 2007] and, while some interesting works have studied the traces of online players [Pittman and GauthierDickey 2010], modeling goal selection for virtual agents based on the goal-selection behaviors of real agents or of real online players remains a challenge.

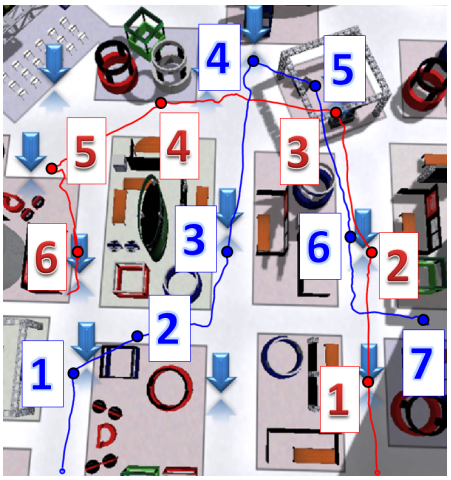


Figure 1: *To show our goal selection model, we simulate the visiting tour in a virtual tradeshow scene. The blue arrows indicate the goals in the scene, and the numbers indicate the order of the goal selection for the virtual agents. There are two simulated traces.*

In this paper, we address the problem of plausibly simulating the behavior of large crowds in which each agent has automatically been assigned a specific, different goal. We present a novel algorithm to automatically generate sequential goals for each virtual agent. In this context, the goals usually refer to locations that geographically correspond to different regions in the virtual world. We use a sequential goal selection (SGS) model to automatically generate goals one after another in a distributed manner. At a higher level of the crowd simulation framework, the overall crowd simulation algorithm uses an efficient goal selection model. We demonstrate that our model can be learned from synthetic sequences.

Due to the complexity of a large virtual environment, the goals can usually be modeled as a directed graph. We therefore propose a probabilistic model to simulate the process of automatically generating sequential goals. Since the edges of the graph embed the transition probabilities between adjacent goals, sequentially sampling from the transition probability graph allows us to efficiently generate sequential goals. To learn the transition probability, we first estimate it using the Hidden Markov Model (HMM). However, due to the bias in the training data, the transition probability estimated by HMM may not be accurate; therefore, in order to refine the estimated probability, we adopt In order to refine the transition probability, we adopt the Discrete Choice Model (DCM) [McFadden 1974; Train 1978] from Econometrics to model goal selection.

Given a set of alternative choices, DCM computes the utility (or benefit) of each alternative based on attributes, and the alternative with the highest utility value will be selected. In our method, DCM is built upon the graphical structure of the probabilistic goal transitions. We use an optimization algorithm to iteratively refine the estimated transition probability.

Main Results: Some of the novel components of our work include: (1) We simulate sequential goal selection process for a virtual crowd in a virtual world with many goals; we do this by sampling from a transition probability map, which can be built using any heterogeneous crowd simulation algorithm. We present a sequential goal selection (SGS) framework that integrates the discrete choice model (DCM) from Econometrics and the hidden Markov model (HMM) to learn the transition probability from the sequence data. (2) We present an optimization algorithm using DCM to refine the transition probability map, which improves the estimation of HMM. (3) We show that, by defining several crowd motion features for filtering training sequences, our method can generate crowd simulation with varied patterns. By measuring the length of the sequence and the efficiency of the path planning, we evaluate the quality of the simulated traces. As a result, we generate sequential goal selection for 200 agents' simulation in two virtual scenes consisting of more than 20 goals.

The paper is organized as follows. In Section 2, we introduce related work on goal selection and crowd simulation. In Section 3, we present the DCM algorithm and the SGS model. In Section 4, we propose the optimization algorithm to learn the SGS model from sequence data. We discuss the implementation details of the simulation in Section 5. In Section 6, we verify our method and the simulation results. Finally, we summarize our work in Section 7.

2 Related Works

2.1 Crowd Simulation

There is extensive literature on crowd simulation; we refer the reader to nice recent surveys [Pelechano et al. 2008; Thalmann et al. 2006]. In this section, we briefly describe some techniques for local collision avoidance, global navigation, and agent-based methods.

Local collision avoidance could be considered as an agent's very short-term decision-making process since its goal is to avoid collisions with the obstacles and other agents. There are several approaches in modeling agents' collision avoidance: e.g., local rule-based models [Reynolds 1987; Reynolds 1999], social force model [Helbing and Molnar 1995] and geometry-based algorithms [van den Berg et al. 2008; Ondrej et al. 2010; Pettre et al. 2009].

Global navigation techniques are used to compute a global collision-free path for each agent to reach a relative long-term goal. These include potential field methods [Koren and Borenstein 1991] [Patil et al. 2011], probabilistic roadmaps [Barraquand et al. 1997; Kavraki et al. 1996], corridor map [Karamouzas et al. 2009], navigation graph [Pettre et al. 2005], and the continuum models[Narain et al. 2009; Treuille et al. 2006].

Both local collision avoidance and global navigation focus, however, on low-level behavior, or the spatial movements of the crowd. They are concerned more with "how they move" rather than "where they move". We are more interested in where the agents move, and in what kind of sequential goal selection strategy is reasonable.

Agent-based methods are developed to model heterogeneous crowd behavior. As happens in the real world, in agent-based crowd modeling, individuals usually make decisions based on their own preferences and their individual choices then influence the crowds behavior. Braun et al. [2003] study the impact of individuals' characteristics in an emergent group by considering each agent's dependence level and altruism level. Pelechano et al. [2005] focus on simulating evacuation behaviors, including panic propagation and crowd impatience. Guy et al. [2011] simulate heterogeneous crowd behaviors using personality trait theory.

2.2 Goal Selection in Crowd Simulation

We classify existing works relevant to goal selection into two types: geographic and behavioral. The geographical type uses locations in the virtual world as their goals; the behavioral type defines goals in terms of actions (such as "standing still" or "meeting with other individuals") rather than specific locations.

Geographical locations as goals. Treuille et al. [2006] consider goal selection as an external parameter set by the animator; a large number of agents share a few common goals. Many other works use a similar strategy [Gayle and Manocha 2008] [Pelechano et al. 2005] [Sud et al. 2007] [Patil et al. 2011]. In particular, Gayle et al. [2008] perform goal selection in their model using two methods: one is to randomly select a goal within the range of a specified radius around the agent, and the other is to script goals for agents as an offline process. Ikeda et al. [2013] propose a prediction model based on transition probability as well. Wolinski et al. [2014] present a method to estimate the optimal goal positions from real data.

Actions as goals. Most methods with action goals integrate individual personalities into the procedure of action selection, constructing a complete decision scheme for agents. These types use a high-level decision scheme to direct agents' behaviors and integrate the impact of mental states. Funge et al. [1999] propose a sophisticated cognitive framework to model agents' decision-making by including knowledge of environment and logic. Yu et al. [2007] apply a decision network integrated with agents' different characteristics to infer agents' desired behaviors. Jaklin et al. [2013] present a framework containing high-level action planning and lower-level path planning. Curtis et al. [2014] present a crowd simulation framework that combines agents goal selection and local behavior. In the game community, Orkin [2005] proposes a Goal-Oriented Action Planning framework, based on a simplified STRIPS-like architecture, to model characters' action planning.

All these works require a suitable decision-making scheme (such as a finite state machine or a decision network), and to ensure that agents choose appropriate actions, each must be individually designed for its scenario. It can therefore be difficult to extend these methods to complex simulations composed of hundreds or thousands of agents. The decision-making scheme developed for one scenario may not be easily adapted to others. In addition, these methods treat path planning and action performing as one integral process. However, some action clips may only need to perform local behaviors, like "look around", while other tasks, like go to the store, may need agents to do path planning only. [Ikeda et al. 2013] is most similar to ours. One major difference is that their method requires real trajectories for training, and their goals are auto-extracted from the intersections of trajectories without semantic meaning. Further, their transition probability is naively computed based only on the frequency of the training data, which may be inaccurate because of data noise and goal extraction. Our goal selection model is set upon the semantic goal graph and we optimize the transition probability using prior.

We present a approach that improves on previous work in several ways. Our method is concerned only with the autonomous process of sequential goal selection. It thus removes the need to manually assign goal sequences for all virtual agents. It also obviates the

need for individual designing of complex intelligence systems and action systems for each agent.

3 Sequential Goal Selection model

In this section, we explain our sequential goal selection model (SGS).First we describe the modeling of the virtual environment. Next we explain the DCM, which is used to model the goal selection of a single agent facing multiple choices. Finally, we present the principle of the SGS model built on DCM.

3.1 Modeling the Virtual Environment

In large virtual worlds, as in the real world, most player avatar behaviors are guided by a series of goals, e.g., the player needs to complete task A before moving on to task B. These goals usually take place in specific geographical portions of the virtual world in which players can interact with others, complete tasks, etc. A large virtual world usually contains a large number of goals. We are interested in simulating how agents consecutively select goals in these complex, goal-filled virtual worlds. For our model, a directed graph of the goals is usually adequate. In this graph, each node depicts a goal; each edge implies that agents can travel from one goal directly to the adjacent one. By incorporating the transition probabilities for all edges between adjacent goals, we can construct a transition probability map.

3.2 Goal Selection Algorithm: DCM

Before digging into the SGS model, we first introduce the fundamental goal selection algorithm that simplifies the real person's decision making. The discrete choice model (DCM) [Train 2003] is used to model a decision-maker's choices from an alternative set. These alternatives should be discrete, finite and mutually exclusive: for instance, a customer choosing a product or a student answering a multiple-choice question. DCMs have been used to predict consumer demands, and are used for business decision-making in areas such as pricing and product development [Train 2003]. They have also been used to predict demand for a transportation system being planned [Train 1978] and so forth.

The idea of DCM is to maximize the utility (or benefit) of a decision that a decision maker is trying to make. It can simply describe the relation of explanatory variables to the outcome of a choice without referencing to exactly how the choice is made. In our formulation, the DCM serves to simulate virtual agents' goal-selection behaviors. Of the various DCM models available, we choose the Logit model [McFadden 1974], which compares the differences among the alternatives, rather than the exact utility values of them. In particular, when an agent (or a decision maker), a, faces a choice among N *potential goals* (i.e., N potential targets that a may choose from), a net benefit or utility, U_i, of each potential goal is computed. Agent a would choose the potential goal that provides the largest utility, i.e., potential goal i given $U_i > U_j, \forall j \neq i$, as his/her next goal. As it may be too costly for us to model or to provide the full information for agent a to compute the exact utility value of each potential goal, we model the utility of each potential goal by two components: an *observable component* and an *unobservable component*. Agent a could only observe some attributes of potential goal i, x_i, which are generally denoted as $\mathbf{x}$. We denote the utility of all these observable attributes as a *representative utility*. For agent a, the representative utility of potential goal i is, $\mathbf{O}_{i,a} = \mathbf{O}(x_i, s_a) = wx$, where w is the weight to the set of observable attributes $\mathbf{x}$. Hence, the utility of potential goal i to agent a can be expressed as:

$$\mathbf{U}_i = \mathbf{O}_i + \overline{\mathbf{O}}_i \tag{1}$$

where $\overline{\mathbf{O}}_{i,a}$ refers to the unobservable component of the utility of potential goal i. The Logit model assumes that each $\overline{\mathbf{O}}$ is independently and identically subject to Gumbel distribution. The probability of any specific agent choosing potential goal i is:

$$P_i = \mathbf{P}(\mathbf{U}_i > \mathbf{U}_j, \forall j \neq i) \tag{2}$$

$$= \mathbf{P}(\overline{\mathbf{O}}_j < \overline{\mathbf{O}}_i + \mathbf{O}_i - \mathbf{O}_j, \forall j \neq i) \tag{3}$$

Given $\overline{\mathbf{O}}_i$, based on (2), the cumulative distribution of the probability of selecting $\overline{\mathbf{O}}_i$ over all potential goals $j \neq i$ could be denoted as:

$$P_i|\overline{\mathbf{O}}_i = \prod_{i \neq j} \mathbf{P}(\overline{\mathbf{O}}_j < \overline{\mathbf{O}}_i + \mathbf{O}_i - \mathbf{O}_j) = \prod_{i \neq j} e^{-e^{-(\overline{\mathbf{O}}_i + \mathbf{O}_i - \mathbf{O}_j)}} \tag{4}$$

Integrating $P_i|\overline{\mathbf{O}}_i$ with the Gumbel distribution function, we have:

$$P_i = \int (P_i|\overline{\mathbf{O}}_i) f(\overline{\mathbf{O}}_i)\, d\overline{\mathbf{O}}_i \tag{5}$$

Then, we substitute Eq. (4) into Eq. (5). We can derive a closed-form expression of the probability as follows. It represents the final formulation used in our framework:

$$P_i = \frac{e^{\mathbf{O}_i}}{\sum_j e^{\mathbf{O}_j}} = \frac{e^{wx_i}}{\sum_j e^{wx_j}} \tag{6}$$

With this formulation, we could compute the probability of an agent selecting potential goal i as his/her next goal.

3.3 The SGS model

DCM can handle only local goal selection. Our method models sequential goal selection by adjoining DCM to the directed goal graph. The edges of the graph represent the transition probabilities of agent moving from one goal to a nearby goal (node). We can use the selected synthetic sequences (automatically generated but manually selected realistic visiting behaviors) to train the graphical model. For the details of the symbols and terms (e.g. sequences) in the following section, please refer to Tab. 1. Our purpose is to maximize the conditional probability and estimate the transition probability:

$$\hat{A} = \arg\max_A \prod_{p=1}^{P} \mathbf{P}(M_p|A) \tag{7}$$

$$= \arg\max_A \prod_{p=1}^{P} \mathbf{P}(I_1^p, \ldots I_k^p, \ldots |A) \tag{8}$$

In fact, the previous description is a typical setting for Hidden Markov Model (HMM) [Rabiner 1989], i.e., maximizing the probability of the observation sequence given the model. Previous works have also proposed using iterative approaches, such as EM algorithms or gradient techniques, to estimate the transition probabilities between the joint goals [Rabiner 1989].

However, rather than simply estimating the transition matrix A, we adopt DCM to model the probability of the transition between goals. Taking the example in Fig. 2, following the expression of DCM (Eq. 6), the transition probability between any consecutive goals L_i and L_j, $\mathbf{P}(L_j|L_i)$, is close to the value of the DCM computation, $\hat{\mathbf{P}}(L_j|L_i)$:

$$\mathbf{P}(L_j|L_i) \approx \hat{\mathbf{P}}(L_j|L_i) = \frac{1}{Z_i} \exp(wx_{ij}) \tag{9}$$

where x_{ij} is the attribute (e.g. the likelihood or the tendency of the goal j moving from i) that explicitly affects players' choice and w refers to the corresponding DCM weight. Z_i is the sum of all utilities of moving to the goal i.

There are several reasons why we should embed DCM into the probability transition of goals: 1) DCM has been extensively used for modeling choice selection, and has been demonstrated as suitable for this situation. 2) Applying DCM to model the goal selec-

Table 1: *The implication of symbols and terms*

	Implication
i, j, k	General index
p	Index of a player
P	Number of players
N	Number of goals
L_j	Global index of jth goal
I_k^p	kth goal of the player p; $I_k^p \in \{L_1, \ldots, L_N\}$
M_p	pth sequence, i.e. $M_p = \{I_1^p, \ldots, I_k^p, \ldots\}$
x_{ij}	Attribute of transition from L_i to L_j
$\mathbf{x}$	The set of all attributes; $\mathbf{x} = \{x_{ij}\}, \forall i, j$
w	DCM weight of the attributes
$\mathbf{P}(L_i\|L_j)$	Probability of moving from from the goal L_i to L_j
A, $\hat{A}$	Transition probability matrix
$A_{i,j}$	Entry at (i, j) of the matrix A
Sequence	A ordered set of goal indices
Trace	Trajectory (i.e. spatio-temporal coordinates) of an agent

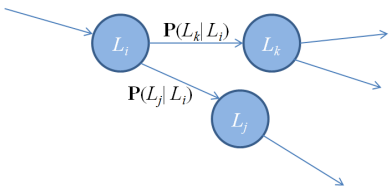


Figure 2: *An example of our goal graph, where the node i connects to the node j and k.*

tion simplifies the process of interactively modifying goal selection for crowds in specific virtual scenes, which is tedious for simulation designers. if the scenes are large and some certain desired global crowd motion patterns are required. 3) As we will show later, our DCM-based method improves the estimation from HMM.

Based on Eq. 9, the sequential goal selection can be modeled in the form of the transition probability graph. Therefore, the process of sequential goal selection is to sample from such a probability transition graph. In the following section, we will explain how to learn the parameters (i.e. the DCM weight) and the attributes.

4 Learning the SGS model

In the previous section, we make use of HMM to estimate the transition probabilities of the entire goal graph. Here we need to learn the parameters w and the attributes $\mathbf{x}$ (which is a matrix consisting of individual attributes, e.g. x_{ij}) of the SGS model built on this probabilistic goal graph. On one hand, the estimated probability from HMM cannot be very accurate even when trained on a large data set. On the other hand, it is difficult to estimate goal attributes that both fit DCM and are consistent with the data. In order to accomplish the task, we introduce the priors of the attributes and formulate it as an optimization problem:

$$[w, \mathbf{x}] = \arg\min_{w,\mathbf{x}} \mathbf{E} \tag{10}$$

$$= \arg\min_{w,\mathbf{x}} \mathbf{E}_{prior}(\mathbf{x}, \mathbf{x}_0) + \gamma \mathbf{E}_{DCM}(w, \mathbf{x}), \tag{11}$$

where $\mathbf{E}_{prior}$ refers to the prior term. Minimizing the prior term indicates that the attributes $\mathbf{x}$ should be similar to the user-defined prior $\mathbf{x}_0$. Similarly, minimizing $\mathbf{E}_{DCM}$ tries to fit the values of transition probability and w and $\mathbf{x}$ in the DCM formulation in the meantime. γ balances the importances of these two terms.

4.1 The Prior Term $\mathbf{E}_{prior}$

The prior term $\mathbf{E}_{prior}$ actually measures the dissimilarity between $\mathbf{x}$ and $\mathbf{x}_0$, which can be simply defined as:

$$\mathbf{E}_{prior} = \frac{1}{N}||\mathbf{x} - \mathbf{x}_0||_2^2, \tag{12}$$

where N is the number of goals in the graph.

4.2 The DCM Term $\mathbf{E}_{DCM}$

The second term, $\mathbf{E}_{DCM}(w, \mathbf{x})$, concerns the transition probability and the DCM algorithm. In other words, any estimated transition probability (e.g. $\mathbf{P}(L_j|L_i)$) should fit to the DCM model. Therefore, this term measures the difference between the probability estimated by HMM and the probability computed by DCM. Based on Eq. 9, it is expressed as below:

$$\mathbf{E}_{DCM} = \frac{1}{Q}\sum_{i,j} ||\mathbf{P}(L_j|L_i) - \hat{\mathbf{P}}(L_j|L_i)||_2^2 \tag{13}$$

$$= \frac{1}{Q}\sum_{i,j} ||\mathbf{P}(L_j|L_i) - \frac{1}{Z_i}\exp(wx_{ij})||_2^2, \tag{14}$$

where $\mathbf{P}(L_j|L_i)$ is estimated from HMM and $\hat{\mathbf{P}}(L_j|L_i)$ is computed using DCM. Q refers to the number of edges in the graph. Note that x_{ij} is one of the elements in the attribute matrix $\mathbf{x}$.

a difficult optimization problem, since Z_i are the sum of the utilities of all possible goals starting from the goal i (which could be any goal). To simplify the expression, let us start from a simple case in Fig. 2. We assume that node L_i only directs to nodes L_j and L_k. Intuitively speaking, at the branch with nodes $\{i, j, k\}$, an agent standing at i has an opportunity to select goals L_j or L_k. Based on the definition of DCM (Eq. 6), we can cancel out the term Z_i:

$$\frac{\hat{\mathbf{P}}(L_j|L_i)}{\hat{\mathbf{P}}(L_k|L_i)} = \frac{\exp(wx_{ij})}{\exp(wx_{ik})}. \tag{15}$$

To measure the difference between the probability estimated by HMM and the probability computed by DCM is a process similar to the minimization in the following expression (with $\mathbf{E}_{prior}$ as constraints):

$$\mathbf{E}_{DCM} = \left|\left|\frac{\mathbf{P}(L_j|L_i)}{\mathbf{P}(L_k|L_i)} - \frac{\hat{\mathbf{P}}(L_j|L_i)}{\hat{\mathbf{P}}(L_k|L_i)}\right|\right|_2^2 \tag{16}$$

$$= \left|\left|\frac{c_1}{c_2} - \frac{\exp(wx_{ij})}{\exp(wx_{ik})}\right|\right|_2^2, \tag{17}$$

where, to simplify the notation, we let $\mathbf{P}(L_j|L_i) = c_1$ and $\mathbf{P}(L_k|L_i) = c_2$. Let

$$\mathcal{K} = \frac{\exp(wx_{ij})}{\exp(wx_{ik})} - \frac{c_1}{c_2}, \tag{18}$$

so our goal is to minimize $\mathcal{K}^2$. Hence we have:

$$\frac{\exp(wx_{ij})}{\exp(wx_{ik})} = \mathcal{K} + \frac{c_1}{c_2} \tag{19}$$

$$w(x_{ij} - x_{ik}) = \ln(\frac{c_1}{c_2} + \mathcal{K}). \tag{20}$$

Based on Taylor's theorem, we can derive that $\ln(\frac{c_1}{c_2} + \mathcal{K}) \approx \ln(\frac{c_1}{c_2}) + \frac{c_2}{c_1}\mathcal{K}$, when $\mathcal{K}$ is small enough. Eq. 19 can be rewritten as:

$$w(x_{ij} - x_{ik}) = \ln(\frac{c_1}{c_2} + \mathcal{K}) \approx \ln(\frac{c_1}{c_2}) + \frac{c_2}{c_1}\mathcal{K} \tag{21}$$

$$\mathcal{K} \approx \frac{c_1}{c_2}(w(x_{ij} - x_{ik}) - \ln(\frac{c_1}{c_2})). \tag{22}$$

Therefore, the term needed to be minimized is equal to:

$$\mathbf{E}_{DCM} = \mathcal{K}^2 = ||w(x_{ij} - x_{ik}) - (\ln c_1 - \ln c_2)||_2^2 \tag{23}$$

$$= ||w(x_{ij} - x_{ik}) - C_{i,j,k}||_2^2. \tag{24}$$

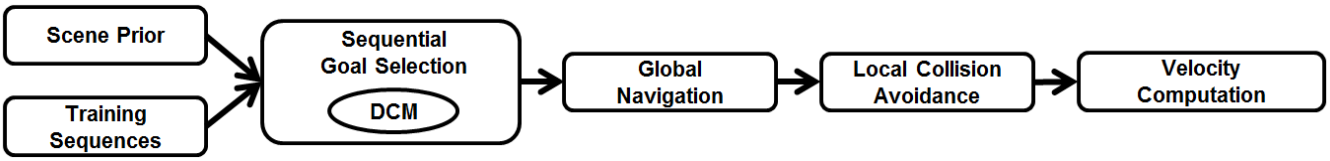


Figure 3: ***Crowd Simulation using SGS:*** *The DCM algorithm is used as a part of the sequential goal-selection scheme. In the offline stage, it estimates the transition probability of goals in the virtual environment. In the online stage, it probabilistically generates the goal position, using each agents status, to direct its global navigation and local collision avoidance schemes are used to compute the velocity of each agent during each simulation time step. The combination of goal selection, global navigation and local collision avoidance results in heterogeneous crowd behaviors.*

where we denote the constant $(\ln c_1 - \ln c_2)$ as $C_{i,j,k}$.

Extending this theory to more general cases, for any branch (at least two out-going edges) in the graph, we can find out any triplet $\{i, j, k\}$ and measure the differences between its transition probability ratio and its DCM utility ratio, given as follows:

$$\mathbf{E}_{DCM} = \frac{1}{Q'} \sum_{i,j,k} ||w(x_{ij} - x_{ik}) - C_{i,j,k}||_2^2, \tag{25}$$

where Q' is the number of the triplets in the branches of the graph. Therefore, our objective is formulated as follows:

$$\begin{aligned}[w, \mathbf{x}] &= \arg\min_{w,\mathbf{x}} \mathbf{E} \\ &= \arg\min_{w,\mathbf{x}} \mathbf{E}_{prior}(\mathbf{x}, \mathbf{x}_0) + \gamma \mathbf{E}_{DCM}(w, \mathbf{x}) \\ &= \arg\min_{w,\mathbf{x}} \frac{1}{N}||\mathbf{x} - \mathbf{x}_0||_2^2 + \gamma \frac{1}{Q'} \sum_{i,j,k} ||w(x_{ij} - x_{ik}) - C_{i,j,k}||_2^2.\end{aligned}$$

We optimize the formulation above in two main steps: (1) Fix $\mathbf{x}$ and optimize w (denoted as f_1):

$$w = \arg\min_{w} \sum_{i,j,k} ||w(x_{ij} - x_{ik}) - C_{i,j,k}||_2^2; \tag{26}$$

(2) Fix w and then optimize $\mathbf{x}$ (denoted as f_2):

$$\mathbf{x} = \arg\min_{\mathbf{x}} \frac{1}{N}||\mathbf{x} - \mathbf{x}_0||_2^2 + \frac{\gamma}{Q'} \sum_{i,j,k} ||w(x_{ij} - x_{ik}) - C_{i,j,k}||_2^2. \tag{27}$$

By iteratively running (1) and (2), the objective function is minimized and we should get the optimal values of w and $\mathbf{x}$ until it converges. Since both (1) and (2) aim to optimize the same objective function, it is guaranteed that the iterative optimization process will reach the local optimal solution. In our tests, the algorithm usually converges in only a few loops. The process is especially efficient in step (1), which is exactly the formulation of least squares method and therefore has a closed-form solution that can be found using linear regression techniques. For step (2), we can apply any gradient-descent based method (e.g. Quasi-Newton algorithm) to find out its local optimal solution. Our method is illustrated in Algorithm 1. In Algorithm 1, the function DCM refers to Eq. 9 and $HMM\_estimate$ refers to the process of estimating transition probability based on HMM.

Algorithm 1 Learning SGS model

Require: The prior of attributes $\mathbf{x}_0$; the training data $\mathbf{M}$.
return The estimated $\mathbf{x}$, w.
Initial guess of the transition probability based on Eq. 6:
$\hat{\mathbf{P}}_0(\cdot) = DCM(\mathbf{x}_0, w_0)$
Estimate the transition probability $\mathbf{P}(\cdot)$ using HMM given the initial transition probability:
$\mathbf{P}(\cdot) = HMM\_estimate(\mathbf{M}, \hat{\mathbf{P}}_0(\cdot))$
Initialization: $w = w_0$, $\mathbf{x} = \mathbf{x}_0$
while $\mathbf{E}$ NOT converges **do**
    Fix $\mathbf{x}$ and optimize w (Eq. 26):
    $\hat{w} = f_1(\mathbf{x}, \mathbf{P}(\cdot))$
    Fix w and optimize $\mathbf{x}$ (Eq. 27):
    $\hat{\mathbf{x}} = f_2(\mathbf{x}_0, w, \mathbf{P}(\cdot))$
end while

5 Implementation Issues

5.1 Simulation

In crowd simulation in large virtual environments, our framework serves as a high-level layer on top of local navigation and collision avoidance algorithms. Based on the virtual environment attributes, our framework sequentially generates the goal positions that direct the global navigation of the virtual agents. After goal-position generation, the local collision avoidance schemes are used to compute collision-free velocities of each agent in any crowded scenario. Fig. 3 shows the framework structure.

Crowd simulators. We claim our framework can be easily integrated into existing crowd simulators, including the well-known local collision-avoidance algorithms (the social force model [Helbing and Molnar 1995] and Reciprocal Velocity Obstacle (RVO) [van den Berg et al. 2008]). In our experiments, we use RVO as our underlying motion model. RVO handles the collision avoidance among virtual agents. Its input is the configuration of all virtual agents: their positions, their actual velocities, and their preferred velocities (desired direction of motion). Given these configurations, RVO can infer the collision-free velocity for each agent in a distributed manner. We use our DCM algorithm to compute the goal positions and combine them with these local collision avoidance schemes. The multi-agent simulation algorithm is illustrated in Algorithm 2, where $\mathbf{P}_{\text{new agent}}$ is the sampling process of the new agent generation. In Algorithm 2, we also profile the agents by keeping track of their positions, velocities, preferred velocities, and their traveling histories. Here the preferred velocity is computed as the normalized velocity along the direction of the difference between the agent's position and its destination, i.e., $\mathbf{v}_{pref} = v_0 \cdot (\mathbf{p}_0 - \mathbf{p}_{goal})/||\mathbf{p}_0 - \mathbf{p}_{goal}||_2$, where v_0 is the speed of agents' average movement.

Emergence of virtual agents. We model the frequency with which new virtual agents enter into the scene over time, or the timing of virtual agents' emergence ($\mathbf{P}_{\text{new agent}}$ in Algorithm 2). This can be modeled by sampling from a Poisson distribution. From the Poisson distribution, the number of entries, or new virtual agents, at each time interval is controlled. The probabilities of choosing one of the six portals as entry are computed based on the proportion of the number of observations in a specific portal from the data. This is a weighted sampling taken with replacements.

Environment setting. In order to mimic real persons' or players' behavior, we need to set an affordance area for each goal (e.g. the range of the booth in the tradeshow). As the virtual agents reach the affordance area of the goal, it will pause, signaling that it has reached the goal. Since our framework is extensible to other applications, agents can be assigned more rich and varied actions, such as interacting with other agents, to perform upon reaching the affordance area. In multi-agent simulation, as the virtual agents make goal selection, the virtual agents choose some position in the affordance area as their desired destination when making their goal selection. We model it as a Gaussian distribution, i.e., sampling the goal position $\mathbf{p}_{goal} \sim \mathcal{N}(\hat{\mathbf{p}}, \sigma^2)$, where $\hat{\mathbf{p}}$ is the center of the affordance area.

5.2 Graph Modeling in HMM

Based on our observation in real players' movements, we realize the goal selection does not follow Markovian manner, i.e. $\mathbf{P}(L_j|L_i) \neq \mathbf{P}(L_{j-1}, L_j|L_i)$, where L_{j-1} and L_j are the agent's consecutive goals. In other words, an agent's previous experience does impact its current decision-making. First-order HMM is not adequate to model such behaviors, and keeping track of complete traveling histories for all agents in crowd simulation is impossible. Therefore, we believe implementing a 2nd-order HMM offers a workable trade-off between accuracy and efficiency. Therefore, a 2^{nd}-order graph model is adopted, whose nodes represent the edges of the original graph.

Algorithm 2 Multi-agent simulation

$T = 1$
for all time steps **do**
  if $\mathbf{P}_{\text{new agent}} \geq P_{thres}$ **then**
    Initialize new agent k:
    Initialize the position $\mathbf{p}_t^k$ and the velocity $\mathbf{v}_t^k$
    Profile agent k in agt_k
    Sample a goal position:
    $\mathbf{p}_{goal}^k = goal\_select(\text{agt}_k)$
    Compute the preferred velocity $\mathbf{v}_{pref}^k$
  end if
  Infer the configurations of all agents:
  $[\mathbf{p}_{t+1}, \mathbf{v}_{t+1}] = RVO(\mathbf{p}_t, \mathbf{v}_t, \mathbf{v}_{pref})$
  if any agent q's goal is reached **then**
    Sample another goal position:
    $\mathbf{p}_{goal}^q = goal\_select(\text{agt}_q)$
    Update the preferred velocity $\mathbf{v}_{pref}^q$
  end if
  Update the profiles all of the valid agents
  $T = T + 1$
end for
function $goal\_select(\text{agt})$
Query the agent's current goal from the profile agt
Compute the sampling probability based on Eq. 9
Random draw the next goal $\mathbf{p}_{goal}$
Return $\mathbf{p}_{goal}$

6 Results and Analysis

In this section, we first verify the optimization algorithm in the SGS model. To accomplish this, multiple sets of training sequences are randomly generated using varied manually-configured SGS models. These training sequences are used to train and estimate the DCM weights and the transition probability matrices. By measuring the output and comparing it with ground-truth values, we evaluate how well the algorithm performs. Next, we generate the training sequences using specific motion features from the four sections of the tradeshow scene. In these trials, we illustrate that our use of different local motion features does indeed generate various global motion patterns. Lastly, we run sequential goal selection for two scenes: the tradeshow and the campus. We present a metric based on traces to measure the quality of the simulation.

6.1 Verification

We verify the proposed algorithm for training the SGS model from the synthetic sequence data. Synthetic data is used because we can verify the algorithm comparing against ground truth.

6.1.1 Data and Metrics

To create the desired sequence data, we create different goal graphs, each consisting of 5 to 10 goals and 10 to 90 edges. And the graphs are all strongly connected (see Tab. 2).

We assume that the graphs' transition probabilities between goals are subject to, and can be computed by the DCM algorithm (i.e., $\mathbf{P}(L_j|L_i) = \hat{\mathbf{P}}(L_j|L_i)$). The attributes are randomly settled in advance, $\mathbf{x} \in [-1, 1]$, and the DCM weight is also determined, $w \in (0, 5)$. From the configured graph structures and their transition probabilities, we can easily randomly sample a large number of sequences. For each graph, around 3000 sequences with different lengths (from 5 to 20) are randomly sampled.

It is worth noting that all experiments indicate that our algorithm cannot directly access the ground-truth transition and the true value of the DCM weight.

For evaluation, we compute the absolute tolerances (AT), $|w - \hat{w}|$, and the relative tolerances (RT), $|\frac{w-\hat{w}}{w}|$, of the DCM weight. We also measure the estimation error by comparing the true transition probability matrix with the estimated one (DT), i.e. $DT = (\sum_{i,j} |A_{i,j} - \hat{A}_{i,j}|^2)^{1/2}$. We also compare our algorithm with HMM. HMM can also be used to estimate the transition probability matrix from sequences, denoted as $\hat{A}^H$. The estimation error is computed similarly, i.e., $DT^H = (\sum_{i,j} |A_{i,j} - \hat{A}_{i,j}^H|^2)^{1/2}$.

6.1.2 Learning the SGS Model

To verify how robustly our algorithm learns the weight of our SGS model, we first allow the algorithm to start from different initial weights. In our experiments (Tab. 2), the algorithm always converges in less than 5 iterations, with only small deviations from the ground-truth weight value. DT indicates that the estimated transition probability matrix $\hat{A}$ is very close to the ground-truth. The average value and the standard deviation of the estimated weights indicate that for different initial weights and graph structures, the estimations are stable and accurate. Compared with HMM (DT^H), the estimation error of our algorithm, DT, is smaller.

In addition, we measure how the algorithm performs as the sequences are generated from the different w (see Fig. 4). We select 13 w ranging from $(0, 5)$ and use each of them to generate around 3000 sequences. Then our algorithm is applied to estimate the values of w. As we observe in the figure, the estimated value is close to the ground-truth value. The average value of AT, RT, and DT are 0.047, 0.020, and 0.0278, respectively. When HMM is used for estimation, the average DT^H is 3.163, which is much larger than DT's average values. When both HMM and our method start from the same initial weight and attribute priors, the estimation from HMM is significantly worse than our method's estimation. When weights and attributes are refined, our method significantly improves over HMM in learning the transition probability.

	N	Q	w	$\hat{w}$	std	AT	DT	DT^H
G1	5	12	2.5	2.493	0.001	0.006	0.019	2.476
G2	7	40	2.5	2.491	0.002	0.009	0.011	3.131
G3	8	50	2.5	2.471	0.001	0.029	0.027	4.632
G4	9	60	2.5	2.483	0.003	0.017	0.020	6.306
G5	10	80	2.5	2.511	0.003	0.011	0.008	7.171

Table 2: *Our model is trained on selected five distinct graph structures (G1 $\sim$ G5) and their generated sequences. N and Q refer to the numbers of nodes and edges respectively. The true value of w is set as 2.5, while random initial weights are assigned at the start of the algorithm, respectively. We average the estimated $\hat{w}$ and also compute AT and DT.*

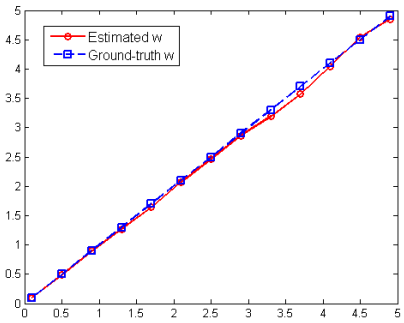


Figure 4: *Taking G2 as example, we select 13 different w in the range of (0,5) to generate data and apply our algorithm to estimate them from the data.*

6.2 Crowd Motion Patterns

Based on the provided training sequences, our algorithm can generate crowd motions with different patterns. To demonstrate the benefits of our algorithm, we set up a virtual tradeshow scene divided into four sections shown in Fig. 6. We separately train the SGS model for each section using different training data, which allows for distinct motion patterns in each section of the virtual scene.

The process of generating the training data is based on two steps: (1) the random, sequential goal selection (i.e., the transition probabilities subject to the uniform distribution); (2) manually filtering of the sequences for training.

We filter the training sequences based on the predefined crowd motion features:

- *Long sequences*. We select the agents with longer sequences, who are more likely interested in the goals in the environment.
- *Short sequences*. We select only agents with short sequences, who show less interest in goals; these agents tend to leave the scene in a shorter time.
- *Destination*. We select only the agents who move towards specific destinations. Using these agents' sequences for training can affect the main trend of the crowd motion.

We take section D of the tradeshow scene as an example. By training only the long sequences or only the short sequences, we can estimate the transition probabilities T_{short} and T_{long}, respectively. We sample 1000 sequences for each of them and measure the lengths. As shown in Fig. 5, T_{long} samples more long sequences (for the sequences whose lengths are longer than 5), while T_{short} can generate more shorter sequences. It is worth to noting that even

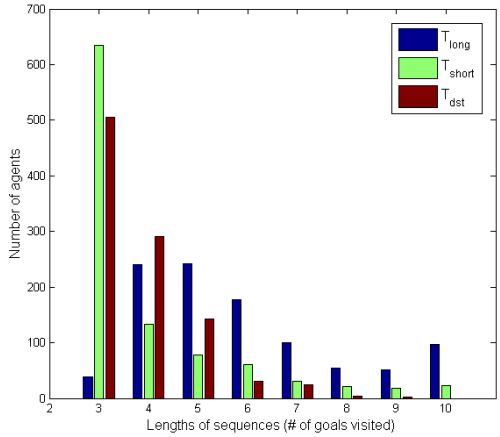


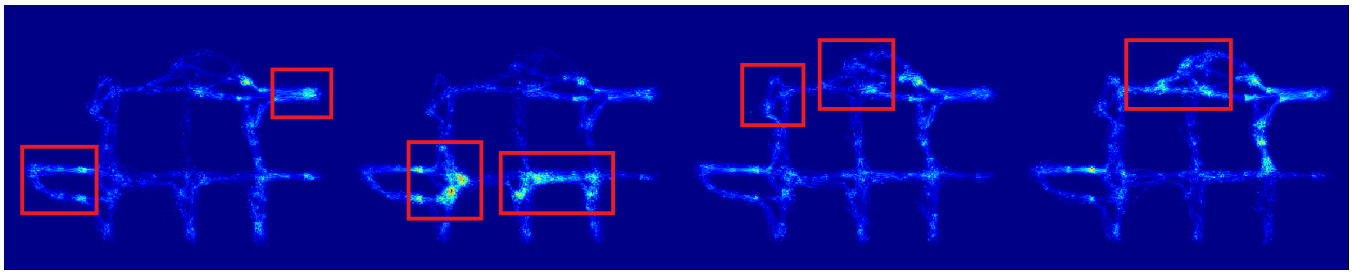
Figure 5: *We sample 1000 sequences and show the histogram of the sequence lengths (i.e. the number of the booths/goals visited).*

when only long sequences are used for training, the algorithm still generates a small number of short sequences in order to increase the diversity of the crowd motion.

We also specify the destinations (blue arrows 'X' and 'Y' in Fig. 6); all of the training data end up at these two destinations. We estimate the transition probability T_{dst} and again sample 1000 sequences. Compared with sampling sequences from the previously trained T_{short} and T_{long}, the ratios of the sequences that end at the specified destinations for T_{dst}, T_{short} and T_{long} are 0.557, 0.259, and 0.209, respectively. This indicates that applying T_{dst} to the simulation forces more agents to specific destinations.

After the filtered data from each section of the scene is used to train the SGS model separately, the corresponding transition probabilities are estimated and used to simulate the desired crowd motion. Since there is no overlapping area for any two sections of the scene, we can easily merge the estimated probabilities of the four sections to generate the global transition-probability map. Using the merged estimated transition probabilities, we generate the crowd simulation of the tradeshow tour. To show the global crowd motion pattern, we record and accumulate all of the agents' simulated trajectories and produce a heatmap (Fig. 7).

Now that we've created the global transition-probability map, we run the algorithm through tests designed to show how close it comes to ground-truth data. We first adopt 'short sequences' to train for all sections (i.e. $T^A_{short} \cup T^B_{short} \cup T^C_{short} \cup T^D_{short}$) (shown in Fig. 7(a)). As shown, more short sequences are generated; this behavior would be characteristic of agents that are not especially interested in the tradeshow. The heatmap shows that, in this situation, the hotspots are located near the portals or exits, indicating that agents leave the scene often. For comparison, for section A and D, we switch to the 'long sequences' training data (i.e. $T^A_{long} \cup T^B_{short} \cup T^C_{short} \cup T^D_{long}$) (shown in Fig. 7(b)). When filtering for the long sequences, the hotspots shift to the central areas of the section A and D - behavior that would be characteristic of agents deeply interested in the tradeshow. Then, in Fig. 7(c), we assign specific destinations for section A and D, which drives agents to move upward (i.e. $T^A_{dst-up} \cup T^B_{short} \cup T^C_{short} \cup T^D_{dst-up}$). The distributions of hotspots change under the "destinations" filtering condition; this is because agents move from sections A and D up to sections B and C. This movement is visible when comparing Figure 7(b) to Figure 7(d):



(a) (b) (c) (d)

Figure 7: *The hotspots of the agents' movement are indicated by the heatmaps. The lighter color indicates the higher frequency and vice versa. The red rectangles highlight the changes when using different crowd features.*

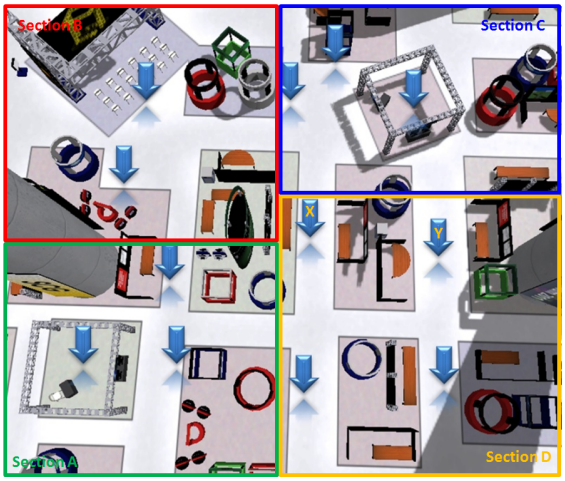


Figure 6: *We sample 1000 sequences and show the histogram of the sequence lengths (i.e. the number of the booths/goals visited).*

the area of greatest concentration has shifted, and the traces in section B and C are distributed more evenly. Lastly, in Fig. 7(d), we assign section C as the destination, driving agents to move leftward (i.e. $T^A_{dst-up} \cup T^B_{short} \cup T^C_{dst-left} \cup T^D_{dst-up}$). Comparing (c) and (d), the main difference is that there are more agents passing through the path from section C to B.

To sum up, by using different types of crowd motion filtering for training, our framework can generate correspondingly different global motion patterns for the simulated crowd.

6.3 Multi-agent Simulation

As we discuss in the last section, our method is able to learn the parameters in our sequential goal selection model. Here we apply the model to generate multi-agent simulation. We test the simulation in two scenarios: a virtual tradeshow and a campus. To evaluate the crowd simulation results, we apply a metric to grade the generated trajectories on how authentically their simulations reproduced the ground-truth data from the corresponding scenarios.

Scenarios. Since our model is suitable for a virtual world with multiple goals, we test on two virtual scenarios. One is a virtual tradeshow containing 15 booths/showcases and 6 entry/exit points as the goals; the other is a section of a campus consisting of 22 goals (see Fig. 8(a,c)).

Simulation results. Given the graph structure of the scenes (Fig. 8(a,c)), Figures 1 and 8(b) show several representative examples of the synthetic trajectories and highlight the sequentially-visited goals. Since there are many blocks in the campus scene (as shown in Fig. 8(c)), the simulation requires intermediate goals, or waypoints, to direct agents to the final destinations. We insert these waypoints, treating them as the virtual goals which only connect to the local neighboring goals (i.e. the rhombic nodes in Fig. 8(c)). Because of these waypoints, this scenario is also a multiple goal-selection scheme. Fig. 8(d) shows a screenshot of the crowd simulation in the scenario of the campus in 2D space; the red circles indicate the agents that stay static when visiting the goals (e.g., talking to other agents in the hall), and the blue circles are the agents who are moving around. The arrows indicate the moving agents' velocities.

Comparison. To verify our goal-selection algorithm for crowd simulation, we compare our model with the two types of goal-selection methods based on random choice. One type allows agents to make uniform random choices in goal selection, and the other type adds HMM-estimated transition probability to the random-selection method. We denote these two methods as UN-RND and D-RND, respectively. As a further verification step, we combine UN-RND, D-RND, and SGS with RVO and the social force model (SF), respectively, so that we have 6 total combinations. For the virtual tradeshow, we generate 200 trajectories for each combination: 1200 trajectories in total. For the campus scene, we generate 300 trajectories for each combination: 1800 trajectories in total.

Since there is no standard method for evaluation of simulation trajectories, we propose a metric based on our observations of real pedestrian behaviors. The metric consists of three terms: the sequence length (S_L), the overlapping area in the trajectories (S_O), and the planning efficiency (S_I). Firstly, we penalize the short sequences of the goal selection, so $S_L = 1 - \exp(-L/\phi)$, where L is the length of the trajectory and ϕ is constant. This is because the short sequences are not often observed, and are especially rare in the tradeshow and the exhibition worlds. Secondly, people usually do not repeatedly visit the same location in the real world, but in some scenes (such as the exhibition) agents can visit the same location multiple times because of the random goal assignment. $S_O = \exp(-\sigma G_{overlap}/G_{total})$ where $G_{overlap}$ refers to number of the grids that are visited for more than twice and G_{total} is the total number of grids the agent steps on in its travels. Lastly, S_I computes the efficiency of path planning used to reach a specific goal: $S_L = \exp(-||L_i - \hat{L}_i||^2/\gamma)$ where $\hat{L}_i$ is the shortest path to the goal i. Since a real person's goal selection is based on a hybrid of local and global decision-making, the agents in our framework are driven by the transition probability for the sake of simplicity. Since virtual goals exist, agents may get distracted by meaningless goals (see Fig. 9(a)), which is unlikely to happen in real-world scenarios. For the final metric, we add up these three terms and

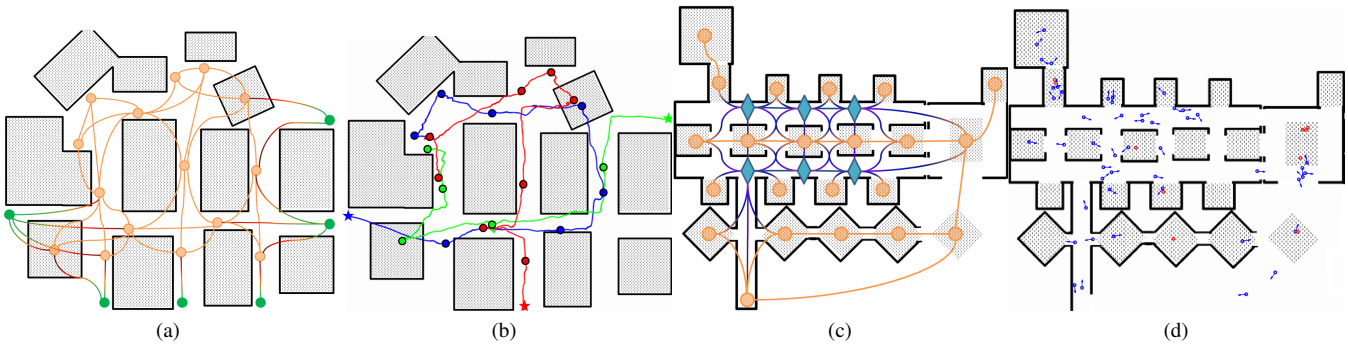


Figure 8: *(a) Overview of the tradeshow scene. The golden circles indicate the location of booths/goals, while the small green circles near the boundary are portals (entry/exit). The shaded rectangles are regions of interests in Fig. 6. (b) Some representative traces. The stars indicate the beginning position of the agents, while the small circles are the positions where the agents stop and interact with the goals. (c) The graph structure of the campus scene. In the scene, the blocks with green circles refer to the buildings or the 'portals' connected to other buildings. The simulated trajectories can possibly begin or end at these locations. The rest of the blocks are resting zones, classrooms, or lecture rooms, where the agents will temporarily stop for a while. There is also one rhombic node: this represents a virtual goal that connects to neighboring goals. (d) The 2D multi-agent simulation in the campus.*

	UN-RND		D-RND		SGS	
	SF	RVO	SF	RVO	SF	RVO
Tradeshow	1.455	1.645	1.945	2.05	2.785	3.230
Campus	2.330	2.263	3.060	2.973	4.047	4.097
Avg.	1.980	2.016	2.614	2.604	3.542	3.750

Table 3: *The average scores of the different methods.*

normalize the value into the scope of $[0, 5]$.

$$S = \text{Norm}(S_L + S_O + S_I) \tag{28}$$

The results of the scores in both scenes are summarized in Fig. 9(b,c). The average scores are provided in Table 3. Our method generates more quality trajectories than other methods; as the average scores indicate, our method significantly outperforms both SF and RVO. The anomalous low scores come from artifacts in the simulation results in Fig. 9(a). The blue and purple trajectories in this scenario are generated from UN-RND, where agents move back and forth between goals, making their S_O values small; the green and red trajectories are generated from D-RND, which also get relatively low scores because their planning efficiency is far lower than that of real behavior. This creates artifacts especially in the red trajectory, since the agents can directly reach the destinations instead of detouring through the arbitrary virtual goals.

7 Conclusion and Future Works

In this paper, we have presented a goal-selection method for crowd simulation that allows virtual agents to navigate around virtual worlds by sequentially and automatically selecting their goals. Our framework, which is based on a combination of the DCM algorithm and HMM, can learn the pattern of goal selection from the training sequences. We have demonstrated that our SGS model outperforms the random-choice model in the crowd simulation.

There are still limitations in our work. First, because we lack real sequences for training, we needed to manually filter the random sequences. In the future, we would like to capture real-world or online players' sequences to use for training. Another limitation lies in the input of the scene prior, which is still tedious to obtain for a large virtual world; in future work, we would like to simplify the prior input process.

Acknowledgements

This research is partially supported by National Science Foundation and a grant from the Boeing Company.

References

BARRAQUAND, J., KAVRAKI, L., LATOMBE, J., MOTWANI, R., LI, T., AND RAGHAVAN, P. 1997. A random sampling scheme for path planning. *Int'l Journal of Robotics Research* **16**, 6.

BRAUN, A., MUSSE, S., DE OLIVEIRA, L., AND BODMANN, B. 2003. Modeling individual behaviors in crowd simulation. In *Proc. CASA*.

CURTIS, S., BEST, A., AND MANOCHA, D. 2014. Menge: a modular framework for simulating crowd movement. *Technical Report, University of North Carolina at Chapel Hill*.

FUNGE, J., TU, X., AND TERZOPOULOS, D. 1999. Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In *Proc. ACM SIGGRAPH*.

GAYLE, R., AND MANOCHA, D. 2008. Navigating virtual agents in online virtual worlds. In *Proc. Web3D Symposium*.

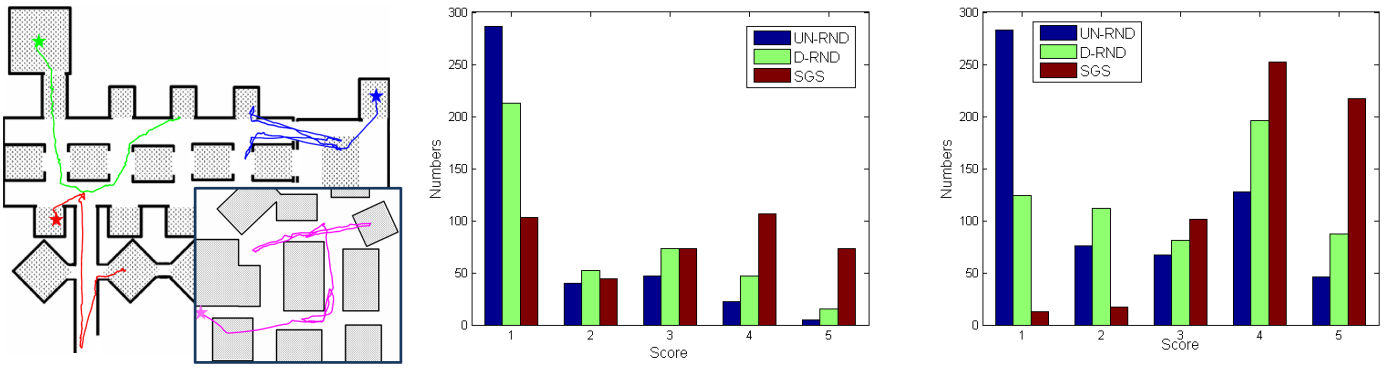
GUY, S., KIM, S., LIN, M. C., AND MANOCHA, D. 2011. Simulating heterogeneous crowd behaviors using personality trait theory. In *Proc. ACM SCA*.

HELBING, D., AND MOLNAR, P. 1995. Social force model for pedestrian dynamics. *Physical Review E*.

IKEDA, T., CHIGODO, Y., REA, D., ZANLUNGO, F., SHIOMI, M., AND KANDA, T. 2013. Modeling and prediction of pedestrian behavior based on the sub-goal concept. *Robotics*.

JAKLIN, N., VAN TOLL, W., AND GERAERTS, R. 2013. Way to go-a framework for multi-level planning in games. In *Proc. Planning in Games Workshop*.

JU, E., CHOI, M., PARK, M., LEE, J., LEE, K., AND TAKAHASHI, S. 2010. Morphable crowds. In *Proc. ACM SIGGRAPH Asia*.



(a) (b) (c)

Figure 9: *(a) Four low-score examples. (b) The score histogram for the simulated traces from tradeshow scene. (c) The score histogram for the simulated traces from the campus scene.*

KARAMOUZAS, I., GERAERTS, R., AND OVERMARS, M. 2009. Indicative routes for path planning and crowd simulation. In *Proc. Int'l Conf. on Foundations of Digital Games*.

KAVRAKI, L., SVESTKA, P., LATOMBE, J., AND OVERMARS, M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*.

KOREN, Y., AND BORENSTEIN, J. 1991. Potential field methods and their inherent limitations for mobile robot navigation. In *Proc. IEEE Int'l Conf. on Robotics and Automation*.

MCFADDEN, D. 1974. Conditional logit analysis of qualitative choice behavior. *Frontiers and Econometrics*.

NARAIN, R., GOLAS, A., CURTIS, S., AND LIN, M. 2009. Aggregate dynamics for dense crowd simulation. In *Proc. ACM SIGGRAPH Asia*.

ONDREJ, J., PETTRE, J., OLIVIER, A., AND DONIKIAN, S. 2010. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. on Graphics*.

ORKIN, J. 2005. Agent architecture considerations for real-time planning in games. In *Proc. AIIDE*.

PATIL, S., VAN DEN BERG, J., CURTIS, S., LIN, M., AND MANOCHA, D. 2011. Directing crowd simulations using navigation fields. *IEEE Trans. on Visualization and Computer Graphics*.

PELECHANO, N., O'BRIEN, K., SILVERMAN, B., AND BADLER, N. 2005. Crowd simulation incorporating agent psychological models, roles and communication. In *Proc. Int'l Workshop on Crowd Simulation*.

PELECHANO, N., ALLBECK, J., AND BADLER, N. 2008. *Virtual crowds: methods, simulation and control*. Morgan and Claypool Publishers.

PETTRE, J., LAUMOND, J., AND THALMANN, D. 2005. A navigation graph for real-time crowd animation on multilayered and uneven terrain. In *Proc. Int'l Workshop on Crowd Simulation*.

PETTRE, J., ONDREJ, J., OLIVIER, A., CRETUAL, A., AND DONIKIAN, S. 2009. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proc. ACM SCA*.

PITTMAN, D., AND GAUTHIERDICKEY, C. 2010. Characterizing virtual populations in massively multiplayer online role-playing games. In *Advances in Multimedia Modeling*. Springer.

RABINER, L. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*.

REYNOLDS, C. 1987. Flocks, herds and schools: A distributed behavioral model. In *Proc. ACM SIGGRAPH*.

REYNOLDS, C. 1999. Steering behaviors for autonomous characters. In *Proc. Game Developers Conference*.

SUD, A., GAYLE, R., ANDERSEN, E., GUY, S., LIN, M., AND MANOCHA, D. 2007. Real-time navigation of independent agents using adaptive roadmaps. In *Proc. ACM VRST*.

THALMANN, D., O'SULLIVAN, C., CIECHOMSKI, P., AND DOBBYN, S. 2006. *Populating virtual environments with crowds*. Eurographics Tutorial Notes.

TRAIN, K. 1978. A validation test of a disaggregate mode choice model. *Transportation Research*.

TRAIN, K. 2003. *Discrete choice methods with simulation*. Cambridge University Press.

TREUILLE, A., COOPER, S., AND POPOVIC, Z. 2006. Continuum crowds. In *Proc. ACM SIGGRAPH*.

VAN DEN BERG, J., LIN, M., AND MANOCHA, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE Int'l Conf. on Robotics and Automation*.

WOLINSKI, D., GUY, S., OLIVIER, A., LIN, M., MANOCHA, D., AND PETTRE, J. 2014. Parameter estimation and comparative evaluation of crowd simulations. In *Proc. Eurographics*.

YU, Q., AND TERZOPOULOS, D. 2007. A decision network framework for the behavioral animation of virtual humans. In *Proc. ACM SCA*.