Dissertação de Mestrado

Estudo do Tempo de Evacuação Total em Ambientes Gerais via Autômatos Finitos

Por : Leandro Alves Pereira

Orientador : Luiz Henrique Duczmal
Co-orientador : Frederico R.B. Cruz

Março de 2007

Leandro Alves Pereira

Estudo do Tempo de Evacuação Total em Ambientes Gerais via Autômatos Finitos

Dissertação apresentada ao Departamento de Estatística do Instituto de Ciências Exatas da UFMG como requisito parcial para obtenção do título de Mestre em Estatística.

Orientador: Luiz Henrique Duczmal
Co-orientador : Frederico R.B. Cruz

Universidade Federal de Minas Gerais

Belo Horizonte, Março de 2007

Folha de Aprovação

Dedico esta dissertação a minha mãe, Senhora Luzia Campinho Pereira, por ter confiado e me dado todas as condições para que este sonho fosse realizado.

Agradecimentos

Primeiramente agradeço a Deus, que me iluminou durante todo este percurso e que me ajudou a vencer mais este desafio na minha vida.

Aos meus pais, que me deram todo o apoio necessário para que este sonho se concretizasse. Sem a ajuda deles, nada disso poderia ser possível. Se que juntos enfrentamos momentos difíceis, não só nesta etapa, mas em todas as etapas da minha vida como estudante.

Ao professor Luiz Henrique Duczmal que tem sido um grande amigo. Agradeço muito por todas as lições e por ter sido o principal motivador para que todas as idéias contidas nessa dissertação fossem escritas. Obrigado pela compreensão, paciência e confiança.

Ao professor Frederico R. B. Cruz, que tem acompanhado todo essa trabalho desde o inicio e continuou ao nosso lado até os momentos finais. Obrigado por ter me ajudado principalmente durante o período de qualificação e por todos os trabalhos desenvolvidos durante a graduação.

Agradeço aos meus parentes, amigos, irmãos e todos que de alguma forma contribuíram para que este trabalho pudesse ser realizado. A todos o meu eterno agradecimento.

Resumo

Esta dissertação é baseada em uma proposta de modelo de fluxo de pessoas e evacuação baseado em simulação por autômatos finitos. Neste trabalho, estudamos os modelos anteriormente propostos e a partir deles, descrevemos um novo modelo com novas componentes para o fluxo de pessoas em situação de evacuação. Este novo modelo proposto baseia-se na idéia de que a velocidade está em função da densidade (Lotação) do ambiente. Utilizamos a teria de autômatos finitos para a construção de um programa de simulação, que compara o modelo com a nova componente frente aos modelos tradicionais. A medida estudada é o tempo de evacuação total. As simulações fornecem o tempo de evacuação total em ambientes diversos, tendo como medida final de estudo a média estimada juntamente com o histograma das simulações. As simulações demonstram a eficiência da nova abordagem em modelar situações ainda não descritas. Uma comparação entre a distribuição empírica de uma simulação e a distribuição do valor extremo mostra que esta seria uma distribuição razoável para modelar a função densidade de probabilidade do tempo de evacuação total.

Palavras-Chaves: Autômatos finitos, tempo de evacuação total, Distribuição do valor extremo.

Abstract

This dissertation presents on a proposal of modeling flow and evacuation of people based on simulation by finite automata. In this work, we studied some of the models previously proposed and described a new model including new components for people flows under evacuation situations. This new model is based on the idea that the speed is a function of the density (capacity) of the environment. We used new finite automata for constructing a simulation program, we compare the new model with the traditional models. The was studied measure the time of total evacuation. The simulations provides the time of total evacuation for several diferent situations, along with the histogram of the simulations. The simulations show the efficiency of the new approach in modeling situations not previously analysed. A comparison between the empiric distribution of a simulation and the distribution of the extreme value shows that this would be a reasonable distribution for the time of total evacuation.

Key-Words: Finite automata, Time of total evacuation, Extreme value distribution.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

K : Conjunto de estados finito, não vazio;

Σ : Alfabeto de entrada (finito);

δ : Função de transição;

i : Estado inicial;

F : Conjunto de estados finais;

M : Matriz de estados que pode ser 3x3 ou 5x5;

M1 : Matriz de estados 3x3;

M2 : Matriz de estados 5x5;

D : Matriz de força Dinâmica;

S : Matriz de força estática;

K_d: Fator de contribuição da força estática;

K_S : Fator de contribuição da força dinâmica;

p_{xy} : Probabilidade de movimentação de um autômato em um passo;

t : Tempo de simulação;

V_{xy} : Fator de diferenciação entre matrizes M1 e M2;

ξ_{xy} : Fator reconhecedor de obstáculos e barreiras;

n_{xy} : Número de pessoas em uma repartição do ambiente.

Sumário

1. Introdução

1.1. Motivação

O estudo de tráfego geral tem ganhado razoável importância nas últimas décadas devido à sua grande aplicabilidade. Os estudos de tráfego são bastante diversificados e podem utilizar metodologias completamente diferentes para descrever o tráfego de pessoas ou veículos. Os modelos terão a sua preferência de acordo com a configuração que se deseja estudar. Modelos analíticos são preferíveis, porém, apenas para casos simples. Para casos de configuração complexa, o desenvolvimento de metodologias é tópico de pesquisa em aberto. Técnicas de simulação trazem a necessidade de utilização de ferramentas estatísticas, para uma análise adequada. Em particular, há um grande interesse em se investigar um particular conjunto de modelos de simulação, o que supõe ser capaz de avaliar o congestionamento em sistemas de tráfego de pessoas para fluxo de emergência, ou seja, situações de evacuação total do domínio (edifícios, prédios, casas, etc).

Existem várias situações práticas de tráfego de pessoas que são interessantes devido a sua importância. A modelagem de tráfego pode representar economia, robustez e qualidade de serviços. Dentre as principais situações, destacam-se o problema de alocação de áreas de circulação, avaliação do fluxo de pedestres e o tempo total de evacuação do ambiente.

O problema de alocação de áreas de circulação (conhecido na literatura de língua inglesa como "buffer allocation") encontra-se presente em diversas situações, tais como em salas de espera para atendimentos, serviços telefônicos (espera em linha), sistemas de armazenamento, entre outros. É importante determinar uma configuração ótima para estes sistemas visando racionalizar sua utilização. Atualmente existem vários trabalhos que abordam o problema segundo uma rede de filas M/G/CC, dentre eles destacamos os trabalhos de Kerbache & MacGregor Smith (2000) e Cruz et all (2005).

A avaliação do fluxo de pedestres é especialmente importante no projeto de rotas de fuga de emergência e no projeto da circulação de edifícios públicos, shopping centers, terminais de transporte, etc. Esta situação também é muito explorada em trabalhos, principalmente em modelos de filas M/G/CC. Existem vários trabalhos dentre os quais destacamos os trabalhos de Cheah & McGregor Smith (1994).

O problema de tempo de evacuação trata-se de situações de evacuação em que as pessoas devem desocupar o ambiente em um menor tempo possível, devido a uma situação de emergência. Daí vem a importância de se ter um modelo que seja eficiente para descrever a realidade a fim de se determinar melhores opções ou estratégias de evacuação, como por exemplo a localização de portas, corredores ou obstáculos, além de se determinar uma quantidade segura de densidade (lotação máxima) que se deve ocupar em um ambiente para que se tenha um tempo de evacuação dentro do especificado, em caso de emergência.

Nosso estudo se refere a este caso, ou seja, a simulação de tráfego de pessoas em situação de evacuação total através de um algoritmo que utiliza o princípio de autômatos finitos não determinísticos.

1.2. Objetivos e Escopo da Dissertação

O principal objetivo deste estudo é propor um novo modelo de simulação de tráfegos de pessoas em situação de emergência, sendo que este modelo é uma adaptação do modelo de Schadschneider (2001). No novo modelo propomos uma nova componente que é a mudança de velocidade em função da densidade do domínio. A dissertação restringe-se ao estudo de comparação dos modelos segundo o tempo de evacuação total, obtido por simulação em linguagem C++.

Assim pretende-se neste trabalho:

- Apresentar uma revisão bibliográfica sobre a teoria de autômatos finitos;
- Apresentar o modelo que define as probabilidades de transição no espaço, segundo as componentes já existentes na literatura e a nova componente proposta nesta dissertação;
- Realizar as simulações do modelo e obter as medidas de tempo de evacuação total;
- Comparar os dois tipos de modelagem e fazer levantamentos finais.

1.3. Organização da Dissertação

Esta dissertação está organizada da seguinte maneira. No Capítulo 2 é feita uma revisão sobre a teoria de autômatos finitos. No Capítulo 3 apresentamos o modelo com todas as suas considerações e passos para implementação. No capítulo 4 apresentamos um estudo sobre o tempo total e parcial de evacuação através de simulações realizadas. A conclusão do estudo, algumas observações e comentários finais encerram este trabalho, no Capítulo 5.

2. Definição de Autômatos Finitos

2.1. Considerações

Neste capítulo estudaremos uma máquina (um procedimento aceitador, ou reconhecedor), chamada autômato finito (AF). A palavra finito é incluída no nome para ressaltar que um AF só pode conter uma quantidade finita e limitada de informação, a qualquer momento. Essa informação é representada por um estado da máquina, e só existe um número finito de estados. Essa restrição faz com que o AF seja severamente limitado na classe de linguagens que pode reconhecer, composta apenas pelas linguagens regulares, como mostraremos neste capítulo. Duas versões do AF são estudadas aqui: o AF determinístico (AFD) e o AF não determinístico (AFND), (Hopcroft et all, 1979).

2.2. Autômato Finito Determinístico

Como observado, a informação que um AF guarda sobre a entrada (mais precisamente sobre a parte da entrada já lida) é representada por um estado, escolhido em um conjunto finito de estados. A definição formal de autômato finito, na sua versão determinística é dada a seguir.

Definição. Um *Autômato Finito Determinístico (AFD)* M, sobre um alfabeto Σ é um sistema

$$(K, \Sigma, \delta, i, F), \tag{2.1}$$

onde

K é um *conjunto de estados* finito, não vazio;

Σ é um *alfabeto de entrada* (finito);

δ: $K \times \Sigma \rightarrow K$ é a *função de transição;*

$i \in K$ é o *estado inicial;*

$F \subseteq K$ é o conjunto de *estados finais.*

O nome *determinístico* faz referência ao fato de que δ é uma função (também chamada função *próximo-estado*), que determina precisamente o próximo estado a ser assumido quando a máquina M se encontra no estado q e lê da entrada o símbolo a: o estado $\delta(q, a)$.

De forma simplificada, podemos dizer que um AFD aceita uma cadeia se, partindo do *estado inicial*, e mudando de estado de acordo com a *função de transição*, o AFD atinge um *estado final* ao terminar de ler a cadeia. Uma das maneiras de visualizar o funcionamento de um AFD é através de um *controle finito* que lê símbolos de uma *fita* de

entrada (onde se encontra a cadeia de entrada), seqüencialmente, da esquerda para a direita. Os elementos do conjunto de estados K representam os estados possíveis do controle finito. A operação se inicia no estado inicial i, lendo o primeiro símbolo da fita de entrada. Por conveniência, considera-se que a cabeça de leitura se move sobre a fita, ao contrário do que seria de se esperar.

A Figura 1 representa um AFD cujo controle está no estado q, e que está lendo o quarto símbolo da cadeia de entrada, um b.

Figura 1: Autômato Finito

Exemplo 1: Considere o AFD M = (K, Σ, δ, i, F), onde temos

K = { q0, q1, q2, q3 },

Σ = { a, b },

i = q0,

F = { q3 },

e onde a função de transição δ: { q0, q1, q2, q3 } × { a, b } → { q0, q1, q2, q3 } é dada pela Tabela 1.

δ	a	b
q0	q1	q2
q1	q0	q3
q2	q3	q0
q3	q2	q1

Tabela 1: Função de Transição do Exemplo 1

Alternativamente, podemos representar o AFD M por um *diagrama de transições*, ou *diagrama de estados*, como o da Fig. 4.2. Note que o diagrama de transições determina completamente o autômato M, através de algumas convenções:

- os estados são os nós do grafo, ou seja, K = { q0, q1, q2, q3 };
- o estado inicial é indicado pela seta, ou seja, i = q0;
- os estados finais são indicados pelo círculo duplo: q3 é o único estado final, ou seja, F = { q3 };
- as transições são as indicadas pelas arestas: δ(q0, a) = q1, δ(q0, b) = q2, δ(q1, a) = q0, etc, ou seja, δ é a mesma função representada pela Tabela 1.

Cada estado de um AF corresponde a uma determinada informação sobre a parte da cadeia de entrada já lida. No caso do exemplo, a informação pode ser descrita em frases curtas, mas isso nem sempre acontece. Para o estado q2, por exemplo, podemos dizer:

"se o estado atingido é q2,
o número de símbolos a já lidos é par, e
o número de símbolos b já lidos é ímpar".

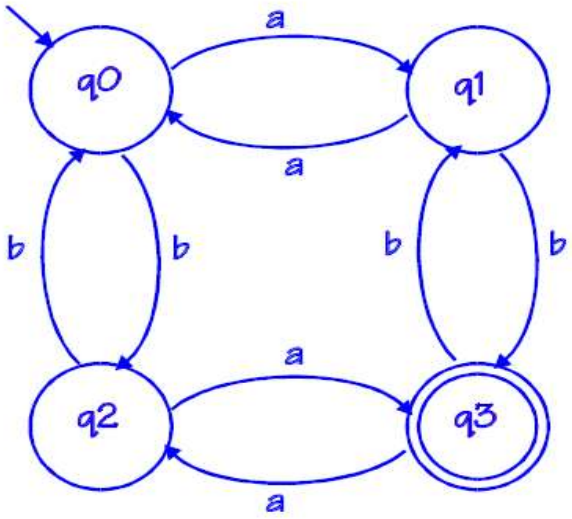

Figura 2: AFD para o Exemplo 1

2.3. Autômato Finito não Determinístico

Passaremos agora ao estudo do AF não determinístico. Em oposição ao que acontece com o AFD, a função de transição de um AFND não precisa determinar exatamente qual deve ser o próximo estado. Em vez disso, a função de transição fornece uma lista (um conjunto) de estados para os quais a transição poderia ser feita. Essa lista pode ser vazia, ou ter um número qualquer positivo de elementos. Essa possibilidade de escolha entre vários caminhos a serem seguidos nos leva a modificar a definição de aceitação. Um AFD aceita se "o último estado atingido é final"; mas um AFND aceita se "*existe uma sequência de escolhas tal que* o *último estado atingido é final*". Podemos alternativamente imaginar que o AFND "escolhe", "adivinha", o caminho certo para a aceitação, uma vez que a existência de escolhas erradas, que não levam a um estado final, é irrelevante.

Exemplo 2: Considere o AFND dado pelo diagrama da Figura 1 e a cadeia de entrada ababa.

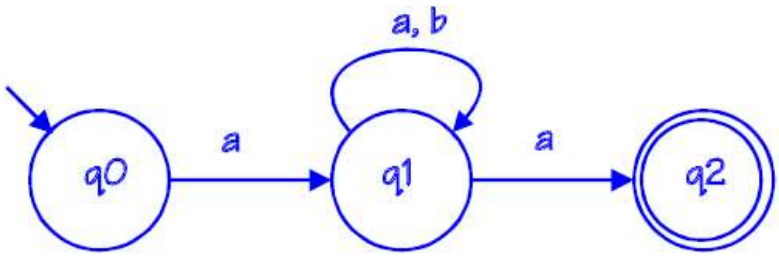

Figura 3: afnd para o exemplo 2

A cadeia ababa é aceita, porque uma das possibilidades é a sequência de estados q0, q1, q1, q1, q1, q2. Naturalmente, com a mesma cadeia, poderíamos escolher a seqüência q0, q1, q1, q1, q1, q1, que não leva a um estado final. Ou a sequência q0, q1, q1, q2, interrompida, porque q2 não prevê uma transição com o segundo b. Mas estes casos em que "o autômato adivinhou errado" não criam problemas para a aceitação, porque "existe um caminho certo".

Este AFND aceita a linguagem das cadeias (de comprimento maior ou igual a 2), cujo primeiro e último símbolos são a, sendo os restantes quaisquer. (Compare este AFND com o AFD de um o exemplo anterior, que aceita a mesma linguagem.)

Definição. Formalmente, um *Autômato Finito não Determinístico (AFND)* M, sobre um alfabeto Σ é um sistema

$$(K, \Sigma, \delta, i, F), \qquad (2.2)$$

em que:

K é um conjunto (finito, não vazio) de estados,

Σ é um *alfabeto de entrada* (finito),

δ: $K\times(\Sigma \cup \{ \varepsilon \})\rightarrow \mathsf{P}(K)$ é a função de transição,

$i \in K$ é o *estado inicial*,

$F \subseteq K$ é o *conjunto de estados finais*.

A notação $\mathsf{P}(K)$ indica o conjunto "partes" de K (conjunto potência de K, ou, ainda, "powerset" de K), o conjunto de todos os subconjuntos de K. Pela definição, portanto, δ é uma função que aceita como argumentos q e a, onde q é um estado e a pode ser um símbolo de Σ ou a cadeia vazia ε. Em qualquer caso, $\delta(q, a)$ é sempre um conjunto de estados, ou seja, um subconjunto de K. Se tivermos $\delta(q, a) = \{p_1, p_2, ..., p_k\}$, entendemos que o autômato M, a partir do estado q, pode escolher um dos estados $p_1, p_2, ..., p_k$ para ser o próximo estado. Se $a = \varepsilon$, nenhum símbolo da entrada é lido; se $a \neq \varepsilon$, o símbolo a da entrada é lido. Podemos considerar o caso $a=\varepsilon$ como correspondendo a transições espontâneas: M muda de estado sem estímulo da entrada. Se tivermos $\delta(q, a) = \varnothing$, não há transições possíveis a partir do estado q com o símbolo a.

2.4 . Conclusões e Observações Finais

Neste capítulo vimos os conceitos de autômatos finitos AFD e AFND, através de exemplos e de suas definições formais. Descrevemos a teoria de autômatos finitos de uma maneira introdutória, já que o nosso objetivo é apenas ter uma visão básica do procedimento. Esta é uma teoria muito abrangente, utilizada para estudar diversos fenômenos das mais variadas áreas. O leitor que estiver interessado em conhecer mais profundamente esta teoria pode consultar Rangel et all (2004) e Hopcroft (1979).

3. Modelo de Evacuação via AFND

3.1. Considerações

Neste capítulo, apresentaremos o modelo de interesse, descrevendo toda sua estrutura. O modelo baseia-se em descrever a probabilidade de uma pessoa ou autômato mover-se de uma posição a outra no ambiente. Aqui apresentamos todas as componentes do modelo e todo o esquema de funcionamento para implementação computacional.

3.2. Estrutura

O tráfego de pessoas ocorre geralmente em ambientes (salas, corredores, escadas, etc.) onde temos uma limitação de espaço propriamente dito. Cada indivíduo ocupa uma determinada área nesse domínio e, portanto, a movimentação de pessoas está condicionada à existência de obstáculos, à densidade de pessoas e à localização de portas. Para fazermos

uma analogia à teoria de autômatos finitos estudadas anteriormente, iremos supor que cada pessoa representa uma autômato, ou célula, que lê a informação da cadeia de entrada e toma a decisão de qual direção seguir. Na situação real, essa cadeia de entrada seria fornecida pela visão, que faz a leitura do ambiente e informa para a pessoa qual deve ser o caminho que ela provavelmente deverá seguir. Obviamente ao ver o caminho certo pela visão, como por exemplo, uma porta que dá acesso a saída de uma sala, ela terá uma "sugestão", mas isso não significa que ela irá exatamente por aquele caminho, apenas terá uma probabilidade elevada de se dirigir a porta. Imagine uma situação em que um cego é colocado em um ambiente desconhecido. Neste caso ele não possui uma cadeia de entrada, portanto não teria um sentido preferencial e neste caso demorará mais tempo para encontrar a saída.

Além disso, as outras componentes que irão influenciar o movimento de uma pessoa sobre um ambiente estão descritas em tópicos, a seguir:

O conhecimento do ambiente. Supomos que o conhecimento de uma pessoa acerca de um ambiente pode ser um fator importante no tempo de evacuação. Imaginamos que o tempo de evacuação possa ser diferente para dois ambientes muito parecidos sendo que um é povoado por pessoas que conhecem o ambiente e o outro é povoado por pessoas que estão pela primeira vez no ambiente (ou raramente). Como exemplo, supomos que o tempo de evacuação em um shopping seria mais rápido para os funcionários do que para os visitantes. Na situação de desconhecimento, a tendência é que as pessoas sigam as outras que estão a sua frente, deixando de segui-las apenas no momento em que se encontrarem num lugar que é conhecido para ela ou quando localizarem a saída. Esta idéia está bem descrita em Shadschneider (2001) e Shadschneider et all (2002).

A velocidade. Supomos que a velocidade está em função da lotação e pode ser alterada conforme a localização das pessoas no ambiente. Em uma situação de evacuação, as pessoas terão uma maior velocidade em locais menos densos. A partir do momento que se encontrarem em locais muito povoados, terão a sua velocidade diminuída. Esta é a componente proposta para este trabalho.

Imaginamos que o ambiente possa ser totalmente repartido, de forma que cada área resultante possa conter exatamente uma única pessoa. Uma estimativa razoável para esta área seria de $\pm\sqrt{0,2}$ metros quadrados (veja a Figura 5). Dessa forma temos como descrever a localização da pessoa em termos de linhas e colunas (x,y). Logo podemos representar um ambiente como sendo uma matriz n x n, em que cada espaço dessa matriz seria um espaço físico. Esta matriz será chamada de matriz ambiente. Assim quando nos referirmos a um termo da matriz estamos nos referindo a um espaço físico do ambiente. O espaço físico (x,y) estaria preenchido por uma pessoa quando a matriz na posição (x,y) assumir valor igual a 1, por exemplo, e vazio, quando a posição da matriz (x,y) for igual a zero.

Outro passo importante é determinar qual ambiente será estudado. Para isso outra matriz se vê necessária, a matriz auxiliar. Esta matriz possui as mesmas dimensões da matriz ambiente descrita anteriormente, porém apenas guardará informações sobre o ambiente tais como localização de portas, obstáculos fixos ou móveis (como cadeiras ou mesas) e limites físicos que seriam as paredes. Na Figura 4 temos um exemplo de uma matriz auxiliar. Esta matriz representa um ambiente a ser simulado. Note que em cada divisão temos um número, que é o indicador do que está contido em cada divisão. As cores na figura foram feitas para facilitar a compreensão. Os campos em branco que contém os números zero e quatro indicam áreas externas ao ambiente e as divisões (paredes),

respectivamente. Obviamente nenhum autômato poderá ocupar estas posições. Os campos em amarelo indicam posições internas do ambiente e que poderão ser ocupadas por um autômato. Os campos em amarelo podem assumir valor igual a um, que significa posições mais centrais do ambiente, ou então assumir valor igual a três, que indica uma aproximação a barreiras, como paredes e obstáculos. Os campos que contém valores iguais a 9 representam a saída do ambiente.

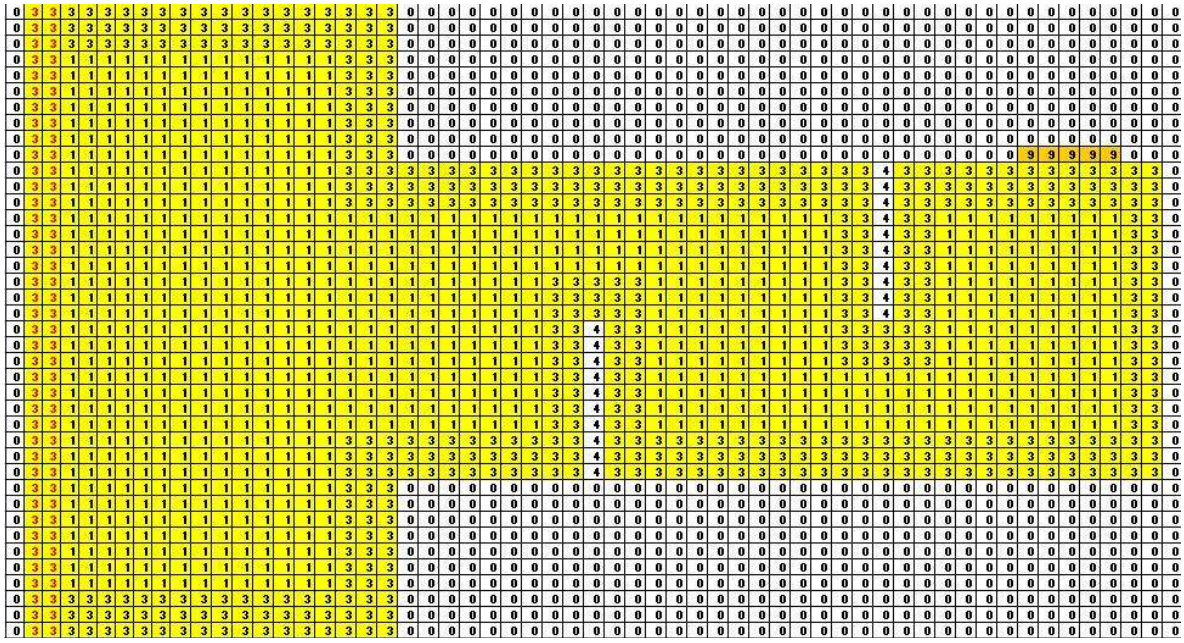

Figura 4: Matriz auxiliar para um ambiente simples

A matriz auxiliar é definida externamente ao programa de simulação. Utilizamos o Microsoft Excel® para descrever esta matriz. Ela deve ser produzida e salva como um arquivo do tipo txt que será passada como entrada no programa de simulação.

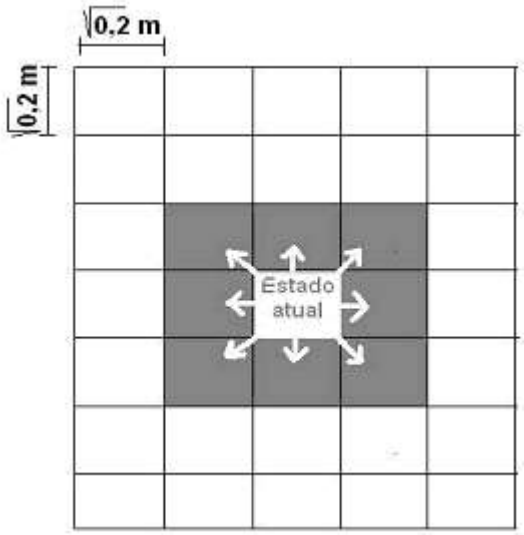

Figura 5: representação de uma pessoa no ambiente

Cada autômato representa uma pessoa e ao iniciar o processo, a pessoa ocupa uma posição no espaço (x,y) da mesma forma que a posição da matriz ambiente (x,y) é igual a 1 (autômato). Ao mover o número 1 para a posição da matriz ambiente (x,y+1) , é como se a pessoa tivesse se deslocado no ambiente um passo a frente.

Para se mover pelo espaço, deve ser definida a probabilidade de cada autômato se mover para uma vizinhança ou de ficar na mesma posição. Para definição das probabilidades de transição, foi utilizado um modelo matemático com várias componentes, que são usados para representar a interação entre as pessoas, interação entre os obstáculos, conhecimento acerca do domínio e a velocidade condicionada ao estado de lotação do domínio. Para a interação entre as pessoas, a idéia básica é ter um rastro deixado pelas pessoas que já passaram pelo mesmo caminho (Shadschneider, 2001). Isso funcionaria para modelar o movimento de acompanhamento das pessoas que não estão em um lugar totalmente conhecido no ambiente. As áreas mais visitadas terão uma maior probabilidade de serem escolhidas para o próximo passo da célula. Isto é o que chamamos de *força dinâmica*, já que esta força se altera a cada vez que um determinado espaço no ambiente é

ocupado por uma pessoa. Esta força dinâmica é mais evidente no início e tem uma queda constante, porque supomos que, ao passar do tempo, as pessoas já terão um fluxo determinado de saída, e não vão interagir mais dessa forma. A interação entre os obstáculos do domínio chama-se força estática, por permanecer a mesma em toda a simulação, já que é determinada pela configuração do ambiente estudado. Essa força é maior para os caminhos que levam para áreas livres e menor para áreas próximas a paredes, obstáculos e áreas contrárias à saída (Shadschneider, 2001). Para a velocidade condicionada ao domínio, teremos uma condição que indica que a velocidade deve ser diminuída se as vizinhanças à frente estão parcialmente ocupadas ou totalmente ocupadas. A velocidade também poderá ser aumentada em casos em que não temos ocupação à frente.

Todas essas componentes fazem parte de um modelo em que a principal componente é a matriz de estados, que aqui também será chamada de matriz M (Shadschneider, 2001). Esta matriz contém as probabilidades iniciais de uma pessoa mover-se para uma determinada vizinhança ou de ficar parada. Esta matriz atribui as probabilidades de acordo com a informação da cadeia de entrada, definida antes da simulação. Dessa forma a cada passo será feita uma leitura na cadeia de entrada e teremos uma nova matriz M que definirá as probabilidades de cada vizinhança. Um exemplo dessa uma matriz M pode ser visto na Figura 6. A matriz mostra que uma pessoa situada na posição (x,y) central da figura teria probabilidades de 0.4, 0.05, 0.07 e 0,15 de se mover para a direita, para a esquerda, para acima e de ficar parada, respectivamente.

0,03	0,07	0,1
0,05	0,15	0,4
0,03	0,07	0,1

Figura 6: Matriz de estados

Essa matriz M será girada de acordo com a informação recebida pela cadeia de entrada. Para este trabalho, os códigos de entrada serão os número de um a oito, representados na Figura 7. Para cada um desses números temos todas as vizinhanças como possíveis estados, inclusive o estado atual, mas com probabilidades diferentes, que serão remanejadas de acordo com o código recebido. Este código provém de uma matriz criada especificamente para este fim, após uma leitura detalhada do ambiente a ser simulado. Dessa forma estaremos fornecendo no momento da simulação o código que descreve a melhor forma da matriz M para estado corrente do autômato, garantindo que a probabilidade de transição seja sempre maior para a posição do ambiente que sugere o caminho para a saída.

Em situações de evacuação em locais públicos como shoppings e grandes supermercados, supomos que o conhecimento das pessoas acerca do domínio é baixo, pois são ambientes não freqüentados pelas mesmas pessoas todos os dias. Nestes casos, a matriz M teria contribuição baixa na probabilidade de transição, já que uma pessoa qualquer tenderia a deixar o ambiente "seguindo" as outras pessoas, ou então apenas se conduzindo entre os corredores sem saber realmente se está no caminho certo. Em ambientes como escolas, faculdades ou empresas, o conhecimento das pessoas sobre o domínio é alta, pois são ambientes freqüentados pelas mesmas pessoas todos os dias. Nestes casos, a matriz M

teria um peso maior, já que uma pessoa qualquer seguiria para a saída do ambiente sem necessariamente seguir as outras.

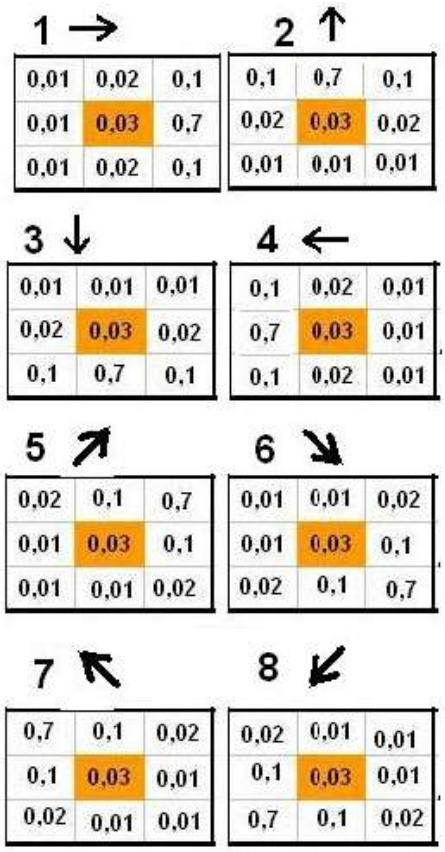

Figura 7: Direção da Matriz M para código da cadeia de entrada

Para fornecer o código da cadeia de entrada, a cada simulação é feita uma leitura em uma matriz que guarda o numero de código. Esta matriz, que aqui será chamada de matriz cadeia de entrada, é definida antes do início da simulação e tem as mesmas dimensões da matriz ambiente. Para cada posição no espaço é atribuído o código ou valor numérico de um a oito na matriz cadeia de entrada, e este valor é lido pelo autômato no momento da simulação. Veja que este número é escolhido arbitrariamente. O que é feito geralmente é uma leitura visual do ambiente e este número é escolhido de forma que ele colocará a matriz M mais adequada para a posição descrita. Na Figura 8 temos um exemplo

de uma matriz cadeia de entrada para o ambiente citado na matriz auxiliar da Figura 4. As cores na figura servem para melhor visualizar a localidade de cada código. Como exemplo, nas posições amarelas temos o número 3, que indica que a matriz M que será utilizada por algum autômato ao ocupar estas posições será aquela que possui probabilidade 0,7 (maior probabilidade) situada abaixo da posição central. Realmente ao observar o ambiente na Figura 8 notamos que os autômatos ao ocuparem esta posição teriam que ter logicamente uma maior probabilidade de andar para "baixo" para que realmente pudessem simular corretamente uma situação real. Esta matriz também é criada externamente ao programa de simulação e deve ser passada como entrada.

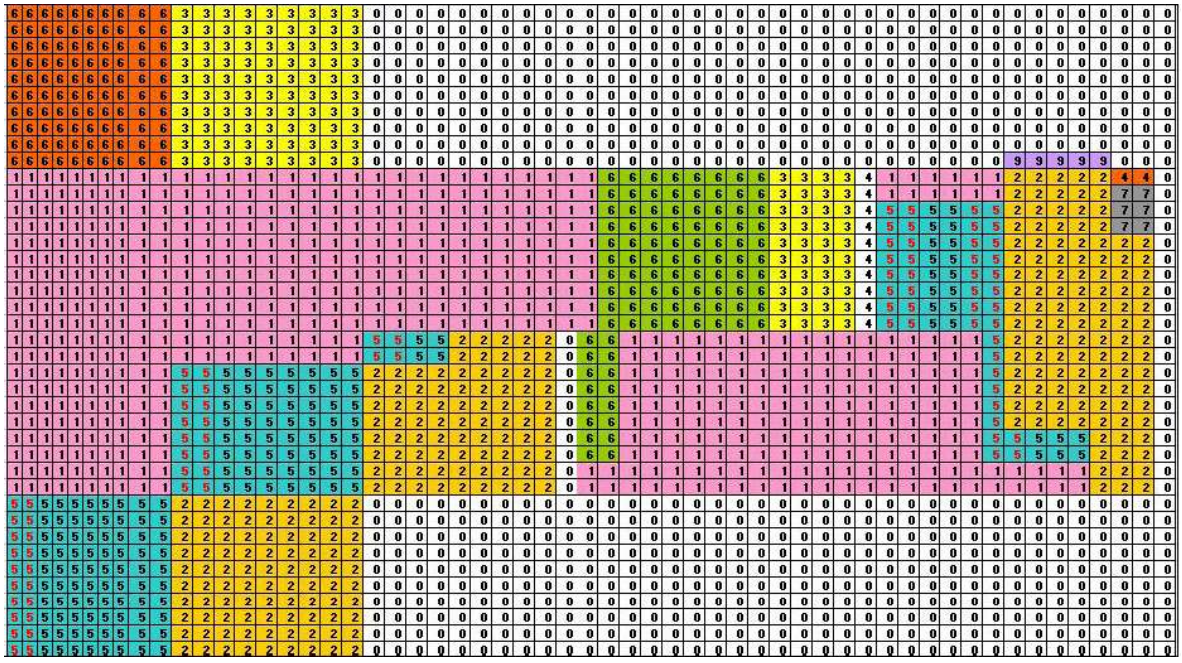

Figura 8: Matriz Cadeia de Entrada para o ambiente

Em uma simulação, supomos que um autômato teria sua velocidade aumentada se desse dois passos ao invés de um, o que seria equivalente a ter uma matriz M com números

de linhas e colunas maiores. Na matriz da Figura 6, as dimensões são iguais a 3x3, indicando que o movimento do autômato pode ser feito apenas para uma vizinhança a frente, o que seria equivalente a dar apenas um passo. Neste trabalho propomos utilizar 2 matrizes, que serão chamadas de M1 e M2, e que tem dimensões 3x3 e 5x5 respectivamente. Dessa forma teremos duas formas de movimentação e que determinará a velocidade do autômato de acordo com a lotação do ambiente. Apresentamos na Figura 9 um exemplo de matriz 5x5. Uma regra simples de decisão de qual matriz utilizar em cada passo é verificar as posições das vizinhanças posteriores. Se as duas vizinhanças a frente estiverem vazias, é um indicativo de que a localidade do autômato se dá em um local com pouca lotação, e portanto, em uma situação de emergência a sua velocidade será aumentada. Nesta situação a matriz atribuída será a M2. Numa situação em que pelo menos uma das duas posições a frente estejam ocupadas, é um indicativo de que estamos num possível início de congestionamento, ou lotação maior, e neste caso teremos uma velocidade baixa de movimentação. Neste caso a matriz a ser utilizada será a matriz 3x3.

0,01	0,01	0,02	0,04	0,07
0,01	0,01	0,03	0,06	0,1
0,01	0,01	0,05	0,08	0,13
0,01	0,01	0,03	0,06	0,1
0,01	0,01	0,02	0,04	0,07

Figura 9: Matriz M2 (5x5)

3.3. Definição do modelo

Definiremos agora o modelo estocástico proposto por Schadschneider (2001) que define as probabilidades p_{xy} de movimentação de um autômato para a posição (x,y) da matriz ambiente.

$$p_{xy} = Ne^{k_d D_{xy}} e^{k_s S_{xy}} M_{xy}(1 - n_{xy})\xi_{xy} . \tag{3.1}$$

Em que M_{xy} representa o valor numérico da posição (x,y) na matriz M. D_{xy} e S_{xy} são os valores numéricos da posição (x,y) nas matrizes D e S, que representam as forças dos campos dinâmico e estático, respectivamente. Para simular a idéia do rastro, existe um contador interno para cada posição (x,y) da matriz ambiente que irá acumular pontos cada vez que essa posição for visitada por um autômato. Podemos falar que quanto mais vezes a posição (x,y) for visitada, maior seria a probabilidade de a mesma receber uma visita de um outro autômato. Logo, este contador seria um peso relativo a posição (x,y) da matriz ambiente e que pode mudar a cada passo. Este peso é guardado em D_{xy}. Podemos então diminuir e aumentar a contribuição do peso D_{xy} através da constante K_d. Veja que se K_d for igual a zero, eliminamos a influência da força dinâmica no modelo. Para simulações em que a idéia do rastro é mais evidente, como em colônia de insetos, utilizamos valores de K_d altos. Porém para simulações de tráfego de pessoas, o rastro pode existir mais fortemente para simulações de ambientes em que o conhecimento dos freqüentadores acerca do mesmo não é tão alto, como em shoppings e feiras de eventos, por exemplo, pois as pessoas tendem a seguir a multidão em caso de urgência. Uma situação que concordamos acontecer

na maioria dos casos, seria a diminuição do rastro com o passar do tempo. Supomos que a um certo tempo, o efeito do rastro seja diminuído pelo fato de as pessoas já terem se situado no caminho certo e o fluxo de movimentação das pessoas se tornam quase que independente. No modelo, uma proposta para incluir este efeito seria a diminuição da constante K_d a cada passo, de forma que K_d convirja para zero com o passar do tempo. Isto pode ser descrito pela seguinte equação:

$$k_d^{(t)} = k_d^{(t-1)} \times 0.99 \tag{3.2}$$

em que $k_d^{(t)}$ representa o valor de K_d na simulação atual e $k_d^{(t-1)}$ representa o valor de K_d na simulação anterior, de forma que $p_{xy} \rightarrow M_{x,y}$, quando $t \longrightarrow \infty$.

S é uma matriz que guarda o peso estático, ou seja, o peso que é atribuído a cada posição (x,y) da matriz ambiente de acordo com a sua localidade. Este peso é determinado antes da simulação e permanece fixo. Geralmente são atribuídos pesos maiores para divisões da matriz ambiente que se encontram mais próximos da saída. Divisões vizinhas a obstáculos possuem pesos menores que divisões mais afastadas desses obstáculos. A constante K_S representa a contribuição deste peso. Veja que este efeito seria mais um que representa a interação do autômato com o ambiente, pois a matriz M também faz um papel parecido. N é uma constante normalizadora que garante que $\sum_{x,y} P_{xy} = 1$. O termo n_{xy} representa o numero de autômatos na posição (x,y). Como assumimos que cada divisão poderá conter apenas um autômato, então n_{xy} só poderá assumir valores zero e um, de forma que, quando n_{xy} for igual a um, p_{xy} será igual a zero. O termo ξ_{xy} é uma fator reconhecedor de obstáculos e barreiras, de forma que ξ_{xy} será igual a zero caso exista um

obstáculo ou barreira na posição (x,y). Isto garante que p_{xy} será igual a zero para uma posição ocupada por uma barreira ou obstáculo.

A partir de agora, definiremos o modelo com a substituição da matriz M pelas matrizes M1 e M2 discutidas anteriormente:

$$p_{xy} = N e^{k_d D_{xy}} e^{k_s S_{xy}} M1_{xy}^{V_{xy}} M2_{xy}^{1-V_{xy}} (1 - n_{xy}) \xi_{xy} \tag{3.3}$$

em que $M1_{xy}$ e $M2_{xy}$ representam os valores numéricos das posições (x,y) das matrizes M1 e M2 respectivamente. O termo V_{xy} é um termo que assume apenas valores iguais a zero e um e que será responsável em dizer qual matriz será utilizada para calcular p_{xy} . O termo V_{xy} assumirá valor igual a zero se as duas posições posteriores (no sentido definido pela cadeia de entrada) estiverem desocupadas. Caso exista alguma ocupação, V_{xy} assumirá valor igual a um.

Abaixo temos um resumo da seqüência correta em que cada item deve ser definido para a simulação:

(1) Definição do ambiente. Neste item, criamos um ambiente geométrico através do software Microsoft Excel® em formato de matriz, com as codificações necessárias. Essa matriz deve ser retangular porém o ambiente não precisa ter este formato. Como exemplo, temos a Figura 4, que mostra um ambiente criado como exemplo. Os códigos têm a seguinte função:

• 1: espaço que pode ser ocupado por uma pessoa no domínio;

• 3: espaço que também pode ser ocupado por uma pessoa no domínio. Este se diferencia do 1 por estar próximo aos limites do domínio;

• 4: representa barreiras físicas dentro do domínio como paredes;

• 9: representa a saída do domínio;

• 0: espaço que não faz parte do domínio. Representa a fronteira do domínio e é usado para completar a matriz retangular, caso o domínio não tenha este formato.

(2) Definição das matrizes de estados M1 e M2. Neste item definimos os valores das matrizes M1 e M2, ou seja, as probabilidades iniciais de cada vizinhança tornar o estado posterior. Esta probabilidade é um valor inicial e será alterado conforme o modelo (3.3).

(3) Definição da Matriz cadeia de entrada. Neste item, criamos a cadeia de entrada apropriada para o domínio geométrico definido. Ela também será criada no programa Microsoft Excel® no formato de planilha, com codificações próprias. Como exemplo temos a Figura 8 que mostra uma cadeia de entrada criada para o ambiente apresentado na Figura 4. Os números em cada área têm a função de girar a matriz de preferência M1 ou M2 e passar essa informação para a célula que ali se encontra no momento da simulação. As cores são simplesmente para facilitar a visualização das áreas que possuem o mesmo código.

(4) Definição da Matriz S. Neste item atribuímos um peso (força estática) a cada posição (x,y) do ambiente descrito e guardamos em S_{xy}. A matriz S também é criada com a ajuda do Microsoft Excel e passada como entrada no programa de simulação.

(5) Definição dos parâmetros adicionais. Estes parâmetros são a densidade inicial do domínio, os valores de K_d , K_S e o número de simulações que serão realizadas.

(6) Definição das estatísticas a serem colhidas na simulação. Neste trabalho estudaremos apenas o tempo de evacuação total e de metade do ambiente. Evidentemente várias outras estatísticas podem ser colhidas, desde que sejam implementadas no algoritmo de simulação.

3.4. Implementação em C++

Um dos objetivos finais deste algoritmo é apresentar o tempo total de evacuação e verificar o quanto esta medida pode ser sensível aos modelos (3.1) e (3.3). A linguagem utilizada para implementação foi a linguagem C++ pelo fato de ser uma linguagem poderosa e flexível. Em particular, o compilador DEV C++ (copyright(c) Bloodshed software, versão 4.9.9.2).

A Figura 10 mostra uma foto da simulação do ambiente da Figura 4. Cada ponto branco representa uma célula ou pessoa, as linhas verticais representam as barreiras ou paredes e o pequeno traço a direita superior representa a saída do ambiente.

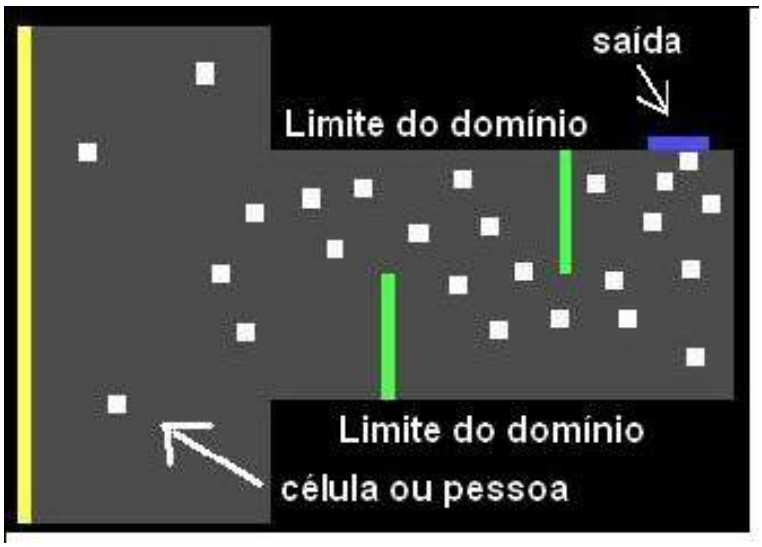

Figura 10: Foto da simulação do ambiente descrito

3.5. Conclusões e Principais Contribuições

Neste capítulo descrevemos o modelo que determina a probabilidade p_{xy} de movimentação por uma pessoa em um ambiente. A descrição detalhada dos procedimentos é importante para entender o esquema de funcionamento e como cada componente real, tais como a interação entre as pessoas e a interação com ambiente, é incorporada ao modelo. Além disso, uma das principais contribuições desse capítulo é a apresentação do modelo (3.3) que propõe um esquema com mais de uma matriz de estado, o que seria uma forma de explicar a alteração de velocidade devido à lotação e congestionamentos.

4. O Tempo de Evacuação Total

4.1. Considerações

Neste capítulo estudaremos o tempo de evacuação total de ambientes. O tempo de evacuação total consiste no tempo até que todas as pessoas se retirem do ambiente estudado, principalmente em situações de emergência. Esta medida é de extrema importância para definição de estruturas de ambientes, tais como localização de obstáculos, localização de portas, dimensões de corredores, etc. Kelvin (2005) apresenta um dos principais trabalhos a abordar esta medida, através de uma metodologia chamada de "Agentes determinísticos". Por ser uma medida não tão simples de obter analiticamente, devido à complexidade existente em cada ambiente, as simulações se fazem necessárias. Utilizando os modelos descritos no capítulo anterior, serão feitas simulações comparando o modelo tradicional e o modelo proposto.

4.2. Experimentos Computacionais

Para a simulação, primeiramente definimos qual seria o ambiente estudado. Este ambiente foi definido seguindo todos os passos apresentados no capítulo 3 e está representado nas figuras 11, 12 e 13. Note que este ambiente contém todas as características de um ambiente real, exceto os obstáculos móveis (mesas, armários etc), que, em um estudo mais detalhado, também poderão ser incluídos. Os pontos em branco representam os autômatos (pessoas), as linhas verdes e amarelas representam barreiras (parede) e as linhas em azul representam as portas do ambiente. Note que em um ambiente qualquer podemos ter várias saídas, com várias dimensões. Nas figuras 11, 12 e 13 temos três fotos de uma simulação em três momentos diferentes. Além disso definimos na figura as medidas em metros do ambiente em uma situação real. Para esta simulações utilizamos o ambiente em duas situações de lotação: a primeira situação compreende a simulações realizadas com 5% do ambiente preenchido e a segunda compreende a 10%.

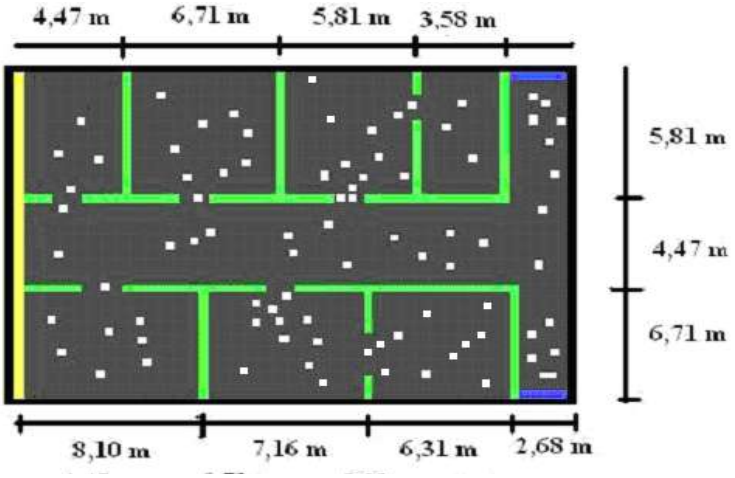

Figura 11: Ambiente estudado nas simulações (foto do inicio da simulação)

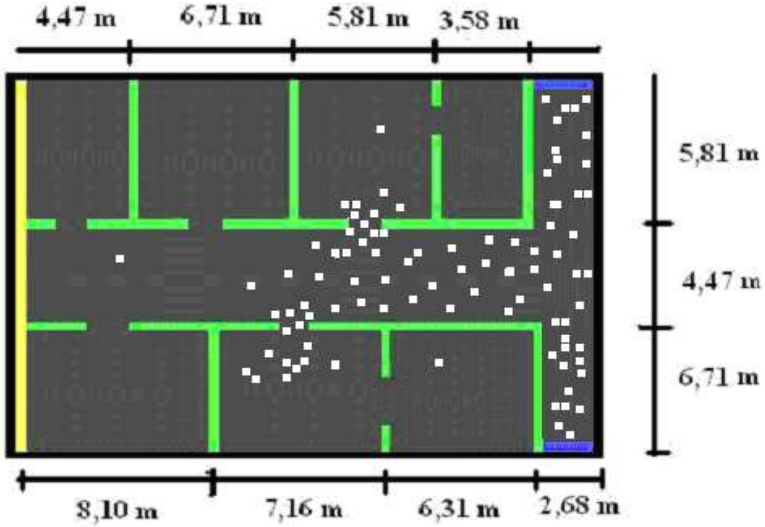

Figura 12: Ambiente estudado nas simulações (metade da simulação)

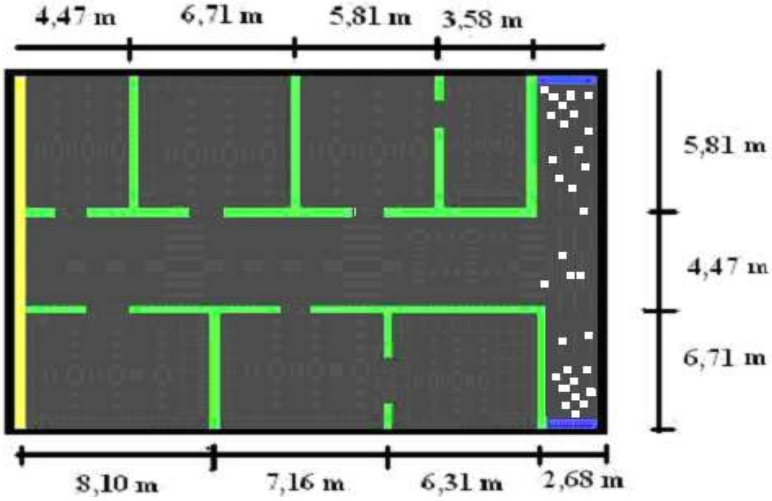

Figura 13: Ambiente estudado nas simulações (foto do fim da simulação)

Para estudar o tempo de evacuação total, iremos estabelecer uma relação entre o tempo em minutos e o número de simulações realizadas. Poderíamos estudar o tempo de evacuação simplesmente como sendo o número de simulações até que a o último autômato abandone o ambiente, porem é mais interessante fornecemos a medida aproximada em minutos, pois senão perderemos o nosso referencial de uma situação real. Segundo Tregenza (1976), a velocidade média de deslocamento de uma única pessoa em um ambiente é aproximadamente 1,5 m/s. Então, para que uma pessoa se desloque em uma distância de $\sqrt{0,2}$ metros (deslocamento entre as divisões do espaço), ela demoraria cerca de 0,298 segundos. Com isso temos que:

$$\text{tempo de evacução total (em minutos)} = \frac{\text{nº de simulações} \times 0,298}{60} \tag{4.1}$$

As simulações serão realizadas até que todas as pessoas saiam do ambiente e então este valor é anotado e utilizado na expressão (4.1) para se obter o tempo de evacuação total em minutos. Um caso particular em que temos o tempo de evacuação de metade das pessoas também foi analisado. Foram realizadas 1000 simulações para cada situação e obtido os valores de média e desvio padrão. Também foi gerado o histograma que permite verificar as características da distribuição do tempo de evacuação.

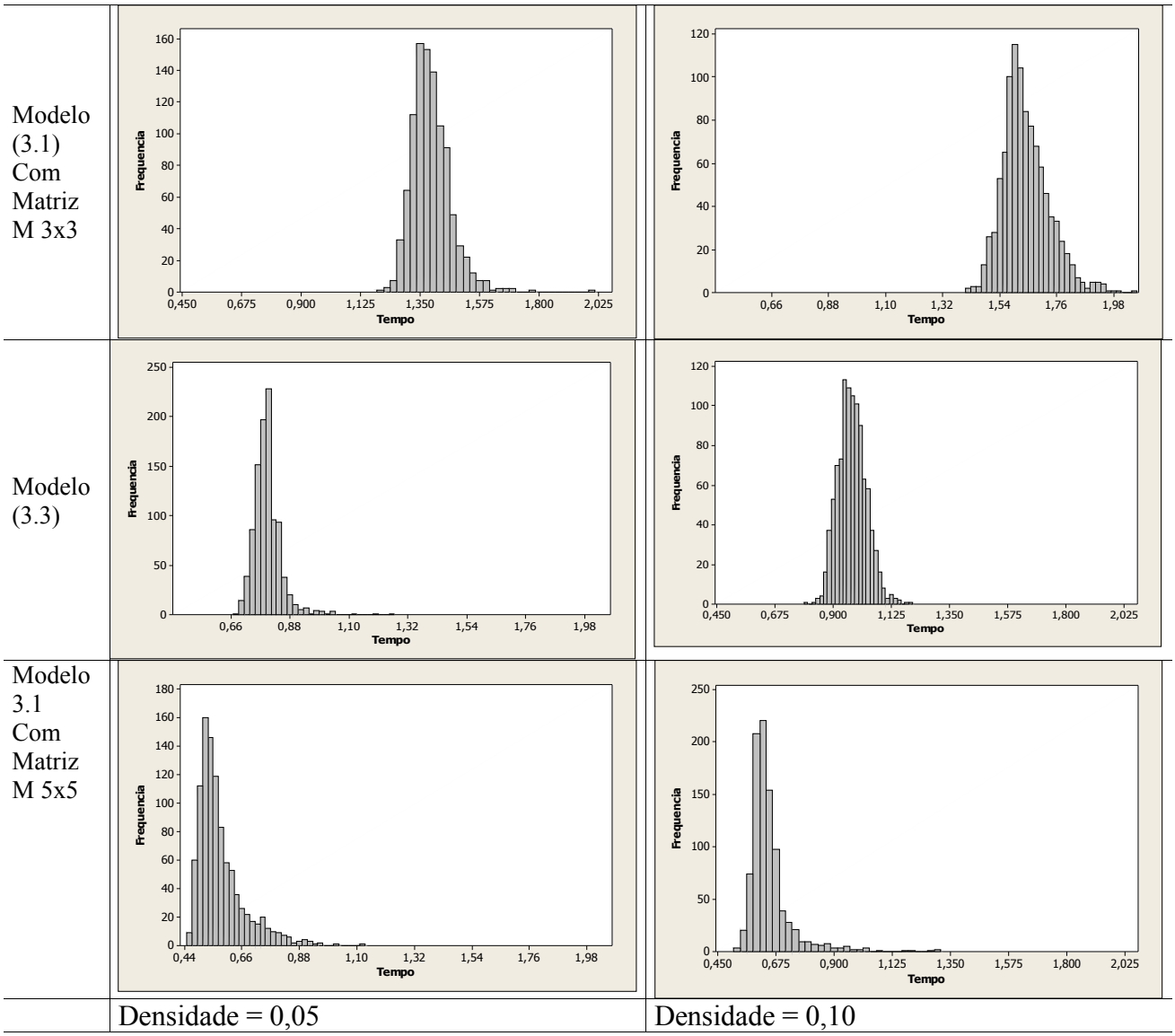

Figura 14: Histograma do tempo de evacuação total (em minutos)

	Média	Desvio Padrão	Média	Desvio Padrão
Modelo (3.1) Matriz M 3x3	1,39270	0,07310	1,6401	0,09144
Modelo (3.3)	0,79550	0,05360	0,97210	0,05460
Modelo (3.1) Matriz M 5x5	0,57740	0,09000	0,61530	0,08520
	Densidade = 0,05		Densidade = 0,1	

Tabela 2: Média e desvio padrão para as simulações do tempo de evacuação total

A Figura 14 mostra as simulações realizadas para o modelo (3.1) utilizando a matriz M com dimensão 3x3 e 5x5 e para o modelo (3.3) que utiliza as matrizes M1 e M2.

Além disso, as simulações foram realizadas para densidade de ambiente iguais a 0,05 (5%) do domínio ocupado e 0,10 (10% do domínio ocupado). Utilizamos K_d e K_S iguais a 1. A Tabela 2 mostra as medidas de média e desvio padrão de cada situação estudada. Pelo histograma, vemos que existe uma queda no tempo de evacuação entre os modelos. O modelo utilizando a matriz M com dimensão 3x3 teve um maior tempo médio, além de ter um histograma mais disperso e com faixa de valores bem acima dos outros. O modelo utilizando uma matriz M com dimensão 5x5 teve o menor tempo médio, com a faixa de valores bem abaixo dos demais. O modelo (3.3) obteve um histograma com faixa de valores situada entre os outros dois histogramas para o modelo (3.1). Isto sugere que o tempo de evacuação total para o primeiro caso seria maior pelo fato dos autômatos estarem desenvolvendo uma velocidade menor, já que apenas a movimentação para a posição (x,y) imediatamente a frente pode ser feita. O caso em que temos a matriz 5x5 é o caso em que os autômatos desenvolvem uma maior velocidade pelo fato de poderem movimentar-se para uma posição duas unidades à frente. Neste caso supomos que a viscosidade será baixa, o que determinará um fluxo mais "solto" das autômatos. Para o caso do modelo (3.1), a velocidade será influenciada pelo tráfego. Utilizar dois tipos de densidade é interessante para mostrar que se existir uma densidade maior, o modelo (3.3) tende a se aproximar do modelo (3.1) com uma matriz 3x3, já que se supõe que quanto maior for a densidade, menor será a velocidade, logo, internamente o modelo (3.3) tende a utilizar mais vezes a matriz 3x3. Todavia, se diminuirmos a densidade, vemos que o modelo (3.3) tende a se aproximar do modelo (3.1) com uma matriz 5x5, já que o ambiente está mais "vazio" e logo, o modelo (3.3) tende a utilizar com mais freqüência a matriz 5x5. Isto fica claro quando analisamos as diferenças dos valores médios obtidos nas densidades 0,05 e 0,10.

Quanto à variabilidade, vemos que ela é a menor no caso em que utilizamos o

modelo (3.3). Isso pode ser explicado pelo fato do modelo condicionar a informação de que as pessoas que estão nas partes mais afastadas do ambiente e que estão com caminho livre irão desenvolver uma velocidade maior ao ponto de alcançar as pessoas que estão "congestionadas", próximos à saída do ambiente e desenvolvem uma velocidade menor. Isto faria com que, mesmo estando em um lugar afastado da saída no início da evacuação, estas pessoas teriam um tempo de saída muito próximo daqueles que já estavam a uma distância não tão afastada da saída. Este fenômeno pode ser notado claramente quando vemos a simulação em tempo real.

Na Figura 15 temos o histograma do tempo de evacuação para metade do domínio em cada uma das situações descritas. Para esta situação, novamente observamos que o tempo de evacuação é menor para o modelo (3.1) utilizando a matriz M 3x3, e maior para o mesmo modelo, utilizando a matriz M 5x5. Vemos que o formato da distribuição desse tempo difere significativamente do formato das distribuições do tempo de evacuação total, o que já era esperado. Como não temos valores extremos, esse tempo não sofrerá influência alguma dos autômatos com tempo de evacuação muito superior aos demais. Alem disso o tempo sofrerá pouco efeito da densidade. Veja que houve um aumento no valor médio não tão alto quanto o aumento obtido para o tempo de evacuação total

Notamos também que a variabilidade agora passa a ser menor para o modelo (3.1) utilizando a matriz M 5x5, e maior para o mesmo modelo utilizando a matriz M 3x3. Note também que as variabilidades caem consideravelmente em comparação ao tempo de evacuação total.

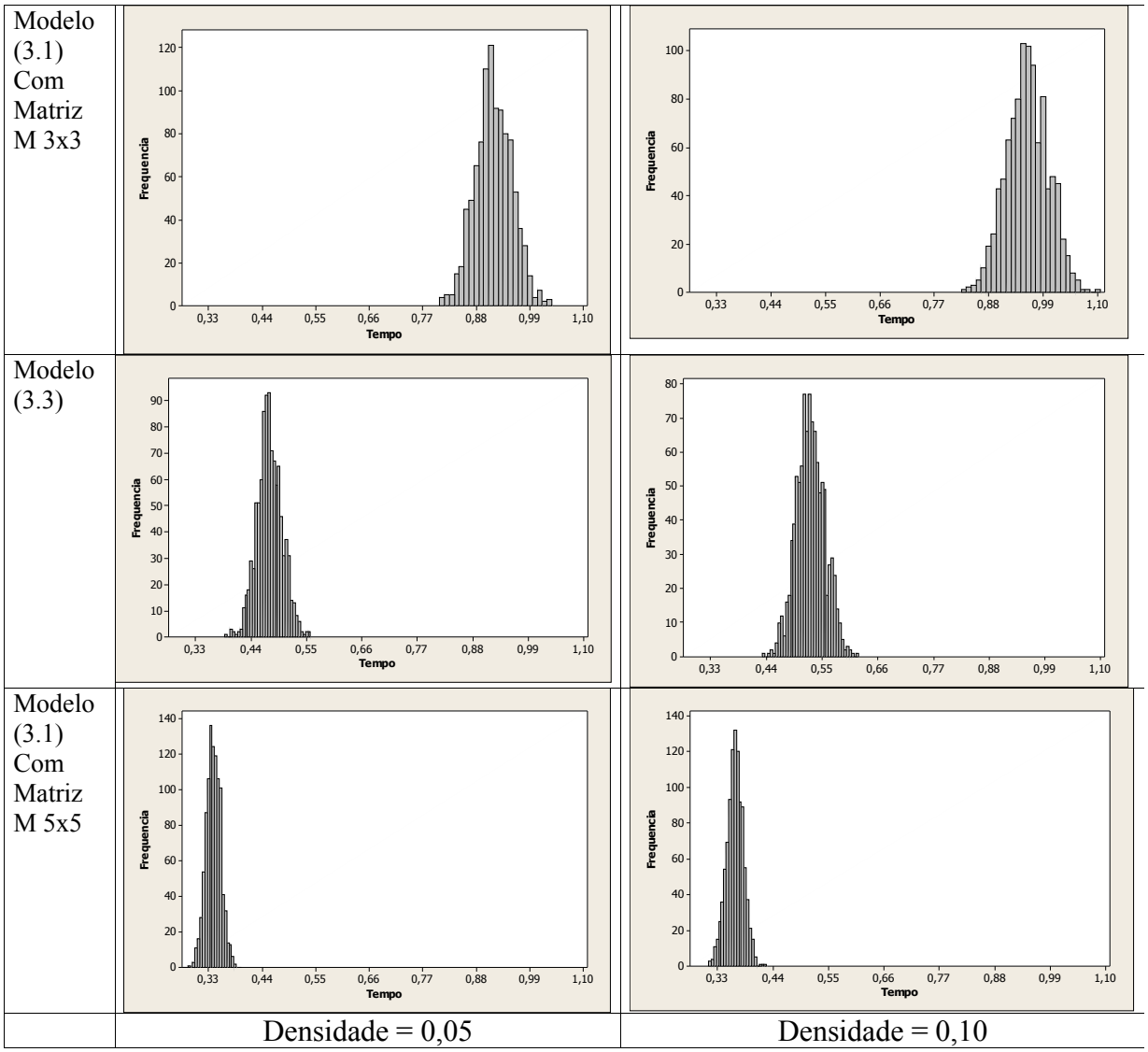

Figura 15: Histograma do tempo de evacuação parcial (metade)

	Média	Desvio Padrão	Média	Desvio Padrão
Modelo (3.1) Matriz M 3x3	0,91750	0,03730	0,95980	0,04143
Modelo (3.3)	0,47830	0,02480	0,52850	0,02810
Modelo (3.1) Matriz M 5x5	0,33803	0,01480	0,36260	0,01673
	Densidade = 0,05		Densidade = 0,1	

Tabela 3: Média e desvio padrão do tempo de evacuação de metade do ambiente

4.3. Comparação entre a Distribuição do Valor Extremo e a Distribuição Empírica dos Dados de Tempo de Evacuação Total Obtidos pela Simulação

A seguir apresentaremos a distribuição do valor extremo. Esta distribuição é utilizada para descrever a distribuições dos valores de máximo em diversas situações práticas, tais como altura de inundações, altas temperaturas, velocidade do vento, idade da pessoa mais velha falecida em uma cidade etc. O nosso objetivo aqui é verificar se a distribuição empírica do tempo de evacuação total obtida através das simulações aproxima-se da distribuição do valor extremo. Para mais informações sobre a distribuição do valor extremo e uma de suas mais importantes aplicações, consulte Achcar (1996).

Uma variável aleatória que segue a distribuição do máximo valor extremo tem como f.d.p (função densidade de probabilidade) a seguinte função:

$$f(x) = \frac{1}{\theta} \exp\left(\frac{-(x-\varepsilon)}{\theta} \right) \exp\left(-\exp\left(\frac{-(x-\varepsilon)}{\theta} \right) \right), \qquad (4.1)$$

em que:

θ = parâmetro de escala,

ξ = parâmetro de locação,

média = $\xi + \nu\theta$,

ν = constante de Euler (ν ~0,57722),

variância = $(\pi\theta)/6$

A fdp pode assumir vários formatos de acordo com os seus parâmetros de escala e locação. Na Figura 16 temos vários formatos da distribuição do valor extremo. Note que, apesar dos diversos fomatos, essa distribuição possui características marcantes: a assimetria à direita, o declínio abrupto na cauda esquerda e o declínio lento e suave na cauda direita. Essa característica esteve evidente em praticamente todos os histogramas do tempo de evacuação total da Figura 14. Apesar dessa distribuição ser bastante utilizada em casos em que assume-se independência entre as variáveis aleatórias em que foi obtido o valor de máximo, nada impede de verificarmos se esta distribuição é uma aproximação adequada para o tempo de evacuação total, apesar da independência entre as variáveis aleatórias (que seria o tempo de evacuação de cada pessoa) não existir. Para isso, utilizamos o modelo (3.3) para realizar 2000 simulações e estimar os parâmetros θ e ξ. Na Figura 17 temos o histograma do tempo de evacuação total dessas simulações e a curva estimada pela distribuição do valor extremo.

Notamos que essa distribuição parece adequar-se muito bem aos dados simulados, indicando que esta seria uma boa aproximação para a função densidade de probabilidade do tempo de evacuação total. Note que as características principais da distribuição dos dados parecem adequar-se muito bem ao modelo proposto. Sabemos que, apesar dos dados simulados visualmente se adequarem bem à curva, é inviável pensarmos em estudar o tempo de evacuação simplesmente por uma modelagem deste tipo. Isso porque temos infinitas formas de descrever um ambiente e verificar como cada configuração pode afetar os parâmetros de locação e escala torna-se uma tarefa quase impossível. Este item serviria apenas como um indicativo de que o modelo de evacuação segundo autômatos finitos seria uma boa aproximação da realidade, já que a f.d.p. que é comumente utilizada para descrever fenômenos semelhantes a estes se aproxima muito bem aos dados simulados.

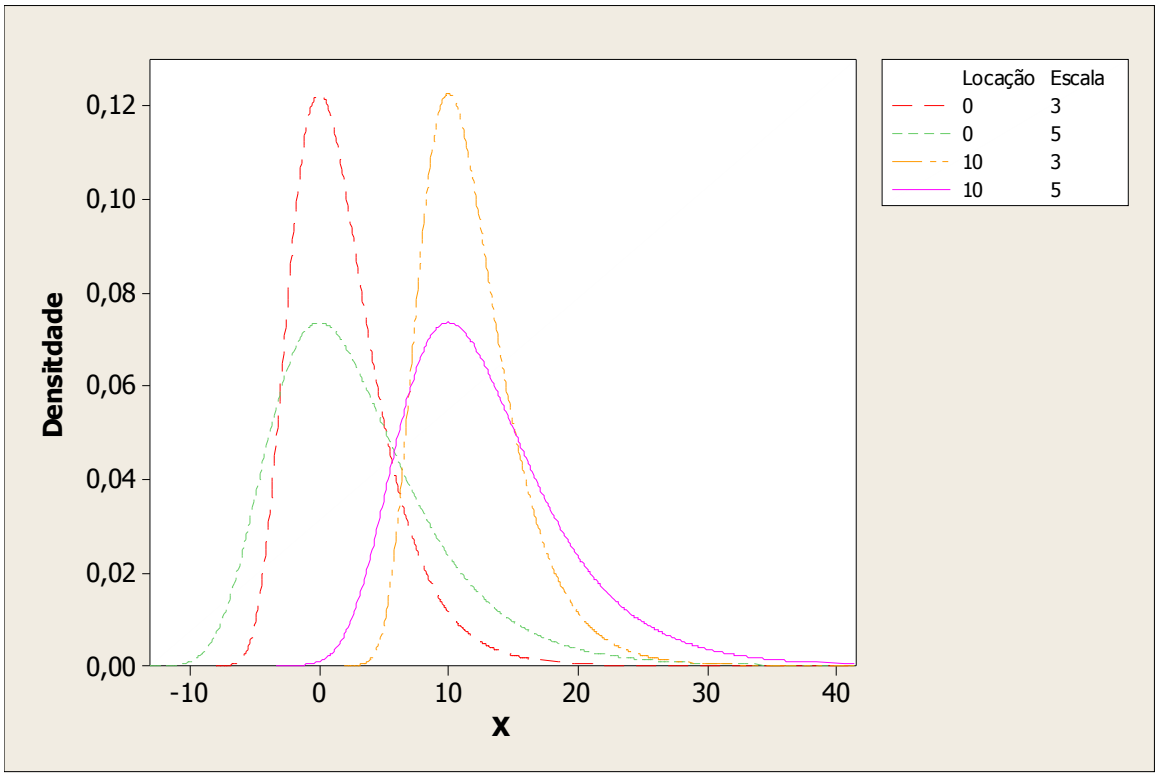

Figura 16: Distribuição do valor extremo

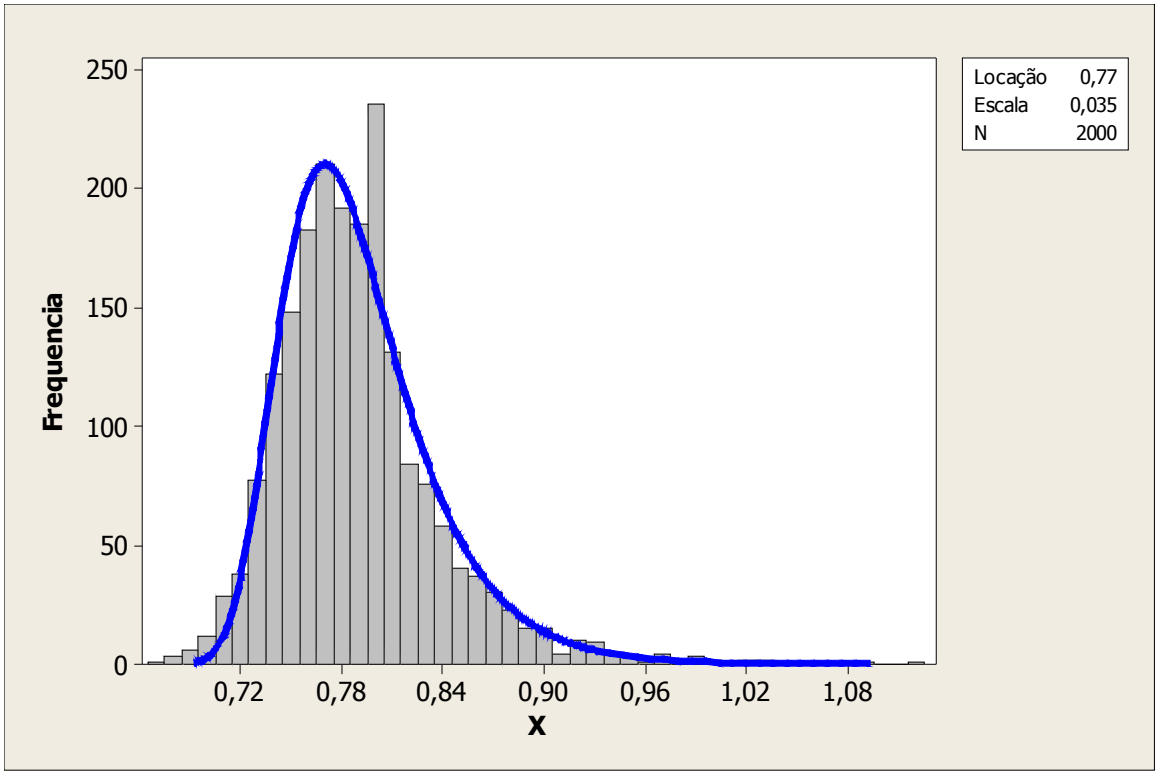

Figura 17: Curva estimada da f.d.p da distribuição do valor extremo a partir das simulações

4.4. Conclusões e Principais Contribuições

Neste capítulo estudamos o tempo de evacuação total em ambientes segundo os modelos descritos no capítulo 3. Foi realizada a comparação entre o modelo tradicional e o modelo proposto com a inclusão da nova componente, ficando claro, através das simulações, o efeito significativo da nova componente no tempo médio e na forma da distribuição. Este estudo é essencial para compreendermos na prática algumas das diferenças entre os modelos e aliar este conhecimento a situações reais. Um estudo particular foi realizado para o tempo de evacuação de metade do ambiente. A distribuição empírica do tempo de evacuação total foi comparada diretamente com a distribuição do valor extremo, uma distribuição muito usada para modelar valores de máximos em situações reais. Esta comparação é um indicativo razoável de que o modelo proposto seria uma aproximação consistente de um fenômeno real.

5. Conclusões

5.1. Observações Finais

Neste trabalho apresentamos um modelo de simulação de fluxo de pessoas em situação de emergência, que é derivado do modelo tradicional de Schadschneider (2001). Mostramos detalhadamente todo o procedimento de modelagem, a equação do modelo com a nova componente de velocidade (matriz 5x5), e os resultados das simulações que comparam o modelo tradicional e o modelo com a nova componente. Vimos que os resultados de tempo médio para o novo modelo seria um valor intermediário entre as simulações do modelo tradicional utilizando a matriz M com dimensões 3x3 e a matriz M com dimensões 5x5. Podemos descrever essas conclusões como sendo efeito da viscosidade, que deixa o fluxo mais "solto" quando temos o modelo (3.1) com a matriz M 5x5 e o fluxo mais "preso" quando temos o modelo (3.3). Uma conclusão interessante foi que a variação do tempo de evacuação para o novo modelo parece ser a menor quando comparada ao modelo tradicional para as matrizes M com dimensões 3x3 ou 5x5.

A principal contribuição desse trabalho foi aumentar a eficácia de um modelo bastante robusto e flexível, pois o modelo apresentado, além de modelar praticamente

qualquer configuração de ambiente e interações entre pessoas, agora também possui uma outra principal componente, a mudança da velocidade segundo a densidade. Os resultados da comparação entre a f.d.p. empírica para o tempo de evacuação e a f.d.p. do valor extremo é um bom indicativo de que o modelo seria uma boa aproximação de uma situação real.

5.2. Propostas de Continuidade

Visando dar continuidade ao trabalho apresentado aqui, uma das opções seria descrever o modelo com mais de duas matrizes de estados, ou seja, utilizar n matrizes de estados. Outra opção seria descrever o modelo com a componente de compressibilidade, ou seja, para situações de congestionamento, as pessoas tenderiam a ocupar um menor espaço no ambiente. Esta idéia poderia ser implementada com a idéia principal de que dois autômatos poderiam ocupar a mesma posição no espaço em caso de super lotação ou congestionamento. Outra idéia interessante seria colocar a ocupação de cada autômato em mais de uma posição do ambiente, como por exemplo, ocupar quatro posições da matriz ambiente, para que o movimento fosse mais refinado.

Apêndice A

Programa de Simulação

O programa de simulação em linguagem C++ foi totalmente escrito por nós. Este é a principal ferramenta deste trabalho. Começou a ser desenvolvido em 26 de março 2004 e durou quase 3 anos para ser este programa que hoje implementa o modelo descrito com todas as suas componentes, além de ser flexível ao tipo de ambiente que se quer estudar e ter a opção de exibir a simulação para o usuário. Para utilizá-lo, é necessário que seja definido (desenhado) um ambiente no Excel® e passado como parâmetro, juntamente com a matriz cadeia de entrada, matriz D, matriz S e respectivas dimensões.

```
// Simulacao de fluxo de pessoas
// Autor: Luiz Duczmal e Leandro Alves
// 2004/04/26
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <windows.h>
#include <gl/glut.h>
#define IA 16807
#define IM 2147483647
#define AM (1.0/IM)
#define IQ 127773
#define IR 2836
#define MASK 123459876
long  nx,ny, pixsize;
int depth;
long n2, peoples, auxi, auxj, trocak, u,u2, nsimsmax, nsims;
long g=0,replicacoes , lance,lancemax,lancelen,lancelenmax;
long aux,pause1,pause2;
int printflag,exibesimulacao, r1, r2;
int somastatus, t1,t2,i,j,k,a,b1,b3,i00,j00,i000,j000,ntries;
char layout[50], trajetoria[50], campo[50];
long idum; //Semente da geracao de numeros aleatorios
double maximo = -1, pEntra, temp, somatemp, somataux , somataux2, mataux[3][3], mataux2[5][5],
matw2[5][5], kd=1,00,ks=1,00 // pEntra=PInicial*pSai
     pSai,media,    //
     pInicial; //
double auxr;
int  ocup[100][100], tabn[100][100],tipo[100][100], traj[100][100], dur[100][100],mark[100][100], auxw
[100][100];
int ni,nj;
long tab[100][100];
long orderi[10000],orderj[10000],smat;
double D[200][200], matw[3][3], mat3[3][3],mat5[5][5],mat7[7][7],soma,acc3[9],acc5[25],r,S[200][200];
```

```
float rnd(long idum0);
double logpow( double x1, double x2 );
double power( double x1, double x2 );
double sqr( double x );
int movever(int i);
int movehor(int i);
int mov(int range, int i0, int j0, int *ni, int *nj);
void plote(int x, int y, int t ,int c);
void finner(int i000, int j000, int *i00, int *j00);
void fbound(int i000, int j000, int *i00, int *j00);
FILE *par;
FILE *sai;
FILE *leia;
FILE *leiadir;
FILE *leiacampo;
// Função callback chamada para fazer o desenho
void redefine (int pa)
{
 int t;
  soma = 0.0;
  smat=9;
  switch (pa){
   case 1:
    if ( ( tab[orderi[k]][orderj[k+1]] + tab[orderi[k]][orderj[k+2]]) > 0 ){
    smat=9;
    }else
    {smat=25;}
    matw[0][0]=0.03; matw[1][0]=0.07; matw[2][0]=0.10;
    matw[0][1]=0.05; matw[1][1]=0.15; matw[2][1]=0.40;
    matw[0][2]=0.03; matw[1][2]=0.07; matw[2][2]=0.10;
    matw2[0][0]=0.00; matw2[1][0]=0.00; matw2[2][0]=0.00; matw2[3][0]=0.06; matw2[4][0]=0.10;
    matw2[0][1]=0.00; matw2[1][1]=0.00; matw2[2][1]=0.00; matw2[3][1]=0.05; matw2[4][1]=0.15;
    matw2[0][2]=0.00; matw2[1][2]=0.00; matw2[2][2]=0.01; matw2[3][2]=0.07; matw2[4][2]=0.20;
    matw2[0][3]=0.00; matw2[1][3]=0.00; matw2[2][3]=0.00; matw2[3][3]=0.05; matw2[4][3]=0.15;
    matw2[0][4]=0.00; matw2[1][4]=0.00; matw2[2][4]=0.00; matw2[3][4]=0.06; matw2[4][4]=0.10;
    break;
```

```
case 2:
 if ( ( tab[orderi[k-1]][orderj[k-1]] + tab[orderi[k-2]][orderj[k]]) > 0 ){
 smat=9;
 }else
 {smat=25;}
 matw[0][0]=0.10; matw[1][0]=0.40; matw[2][0]=0.10;
 matw[0][1]=0.07; matw[1][1]=0.15; matw[2][1]=0.07;
 matw[0][2]=0.03; matw[1][2]=0.05; matw[2][2]=0.03;
 matw2[0][0]=0.10; matw2[1][0]=0.15; matw2[2][0]=0.20; matw2[3][0]=0.15; matw2[4][0]=0.10;
 matw2[0][1]=0.06; matw2[1][1]=0.05; matw2[2][1]=0.07; matw2[3][1]=0.05; matw2[4][1]=0.06;
 matw2[0][2]=0.00; matw2[1][2]=0.00; matw2[2][2]=0.01; matw2[3][2]=0.00; matw2[4][2]=0.00;
 matw2[0][3]=0.00; matw2[1][3]=0.00; matw2[2][3]=0.00; matw2[3][3]=0.00; matw2[4][3]=0.00;
 matw2[0][4]=0.00; matw2[1][4]=0.00; matw2[2][4]=0.00; matw2[3][4]=0.00; matw2[4][4]=0.00;
 break;
case 3:
  if ( ( tab[orderi[k+1]][orderj[k]] + tab[orderi[k+2]][orderj[k]]) > 0 ){
 smat=9;
 }else
 {smat=25;}
 matw[0][0]=0.03; matw[1][0]=0.05; matw[2][0]=0.03;
 matw[0][1]=0.07; matw[1][1]=0.15; matw[2][1]=0.07;
 matw[0][2]=0.10; matw[1][2]=0.40; matw[2][2]=0.10;
 matw2[0][0]=0.00; matw2[1][0]=0.00; matw2[2][0]=0.00; matw2[3][0]=0.00; matw2[4][0]=0.00;
 matw2[0][1]=0.00; matw2[1][1]=0.00; matw2[2][1]=0.00; matw2[3][1]=0.00; matw2[4][1]=0.00;
 matw2[0][2]=0.00; matw2[1][2]=0.00; matw2[2][2]=0.01; matw2[3][2]=0.00; matw2[4][2]=0.00;
 matw2[0][3]=0.06; matw2[1][3]=0.05; matw2[2][3]=0.07; matw2[3][3]=0.05; matw2[4][3]=0.06;
 matw2[0][4]=0.10; matw2[1][4]=0.15; matw2[2][4]=0.20; matw2[3][4]=0.15; matw2[4][4]=0.10;
 break;
case 4:
  if ( ( tab[orderi[k]][orderj[k-1]] + tab[orderi[k]][orderj[k-2]]) > 0 ){
 smat=9;
 }else
 {smat=25;}
 matw[0][0]=0.10; matw[1][0]=0.07; matw[2][0]=0.03;
 matw[0][1]=0.40; matw[1][1]=0.15; matw[2][1]=0.05;
 matw[0][2]=0.10; matw[1][2]=0.07; matw[2][2]=0.03;
```

```
 matw2[0][0]=0.10; matw2[1][0]=0.06; matw2[2][0]=0.00; matw2[3][0]=0.00; matw2[4][0]=0.00;
 matw2[0][1]=0.15; matw2[1][1]=0.05; matw2[2][1]=0.00; matw2[3][1]=0.00; matw2[4][1]=0.00;
 matw2[0][2]=0.20; matw2[1][2]=0.07; matw2[2][2]=0.01; matw2[3][2]=0.00; matw2[4][2]=0.00;
 matw2[0][3]=0.15; matw2[1][3]=0.05; matw2[2][3]=0.00; matw2[3][3]=0.00; matw2[4][3]=0.00;
 matw2[0][4]=0.10; matw2[1][4]=0.06; matw2[2][4]=0.00; matw2[3][4]=0.00; matw2[4][4]=0.00;
 break;
 case 5:
if ( ( tab[orderi[k-1]][orderj[k+1]] + tab[orderi[k-2]][orderj[k+2]]) > 0 ){
 smat=9;
 }else
 {smat=25;}
 matw[0][0]=0.07; matw[1][0]=0.10; matw[2][0]=0.40;
 matw[0][1]=0.03; matw[1][1]=0.15; matw[2][1]=0.10;
 matw[0][2]=0.05; matw[1][2]=0.03; matw[2][2]=0.07;
 matw2[0][0]=0.00; matw2[1][0]=0.06; matw2[2][0]=0.10; matw2[3][0]=0.15; matw2[4][0]=0.20;
 matw2[0][1]=0.00; matw2[1][1]=0.00; matw2[2][1]=0.05; matw2[3][1]=0.07; matw2[4][1]=0.15;
 matw2[0][2]=0.00; matw2[1][2]=0.00; matw2[2][2]=0.01; matw2[3][2]=0.05; matw2[4][2]=0.10;
 matw2[0][3]=0.00; matw2[1][3]=0.00; matw2[2][3]=0.00; matw2[3][3]=0.00; matw2[4][3]=0.06;
 matw2[0][4]=0.00; matw2[1][4]=0.00; matw2[2][4]=0.00; matw2[3][4]=0.00; matw2[4][4]=0.00;
 break;

case 6:
    if ( ( tab[orderi[k+1]][orderj[k+1]] + tab[orderi[k+2]][orderj[k+2]]) > 0 ){
 smat=9;
 }else
 {smat=25;}
 matw[0][0]=0.05; matw[1][0]=0.03; matw[2][0]=0.07;
 matw[0][1]=0.03; matw[1][1]=0.15; matw[2][1]=0.10;
 matw[0][2]=0.07; matw[1][2]=0.10; matw[2][2]=0.40;
 matw2[0][0]=0.00; matw2[1][0]=0.00; matw2[2][0]=0.00; matw2[3][0]=0.00; matw2[4][0]=0.00;
 matw2[0][1]=0.00; matw2[1][1]=0.00; matw2[2][1]=0.00; matw2[3][1]=0.00; matw2[4][1]=0.06;
 matw2[0][2]=0.00; matw2[1][2]=0.00; matw2[2][2]=0.01; matw2[3][2]=0.05; matw2[4][2]=0.10;
 matw2[0][3]=0.00; matw2[1][3]=0.00; matw2[2][3]=0.05; matw2[3][3]=0.07; matw2[4][3]=0.15;
 matw2[0][4]=0.00; matw2[1][4]=0.06; matw2[2][4]=0.10; matw2[3][4]=0.15; matw2[4][4]=0.20;
 break;
case 7:
```

```
    if ( ( tab[orderi[k-1]][orderj[k-1]] + tab[orderi[k-2]][orderj[k-2]]) > 0 ){
  smat=9;
  }else
  {smat=25;}
  matw[0][0]=0.40; matw[1][0]=0.10; matw[2][0]=0.07;
  matw[0][1]=0.10; matw[1][1]=0.15; matw[2][1]=0.03;
  matw[0][2]=0.07; matw[1][2]=0.03; matw[2][2]=0.05;
  matw2[0][0]=0.20; matw2[1][0]=0.15; matw2[2][0]=0.10; matw2[3][0]=0.06; matw2[4][0]=0.00;
  matw2[0][1]=0.15; matw2[1][1]=0.07; matw2[2][1]=0.05; matw2[3][1]=0.00; matw2[4][1]=0.00;
  matw2[0][2]=0.10; matw2[1][2]=0.05; matw2[2][2]=0.01; matw2[3][2]=0.00; matw2[4][2]=0.00;
  matw2[0][3]=0.06; matw2[1][3]=0.00; matw2[2][3]=0.00; matw2[3][3]=0.00; matw2[4][3]=0.00;
  matw2[0][4]=0.00; matw2[1][4]=0.00; matw2[2][4]=0.00; matw2[3][4]=0.00; matw2[4][4]=0.00;
  break;
  case 8:

  if ( ( tab[orderi[k+1]][orderj[k-1]] + tab[orderi[k+2]][orderj[k-2]]) > 0 ){
  smat=9;
  }else
  {smat=25;}
  matw[0][0]=0.07; matw[1][0]=0.03; matw[2][0]=0.05;
  matw[0][1]=0.10; matw[1][1]=0.15; matw[2][1]=0.03;
  matw[0][2]=0.40; matw[1][2]=0.10; matw[2][2]=0.07;
  matw2[0][0]=0.00; matw2[1][0]=0.00; matw2[2][0]=0.00; matw2[3][0]=0.00; matw2[4][0]=0.00;
  matw2[0][1]=0.06; matw2[1][1]=0.00; matw2[2][1]=0.00; matw2[3][1]=0.00; matw2[4][1]=0.00;
  matw2[0][2]=0.10; matw2[1][2]=0.05; matw2[2][2]=0.01; matw2[3][2]=0.00; matw2[4][2]=0.00;
  matw2[0][3]=0.15; matw2[1][3]=0.07; matw2[2][3]=0.05; matw2[3][3]=0.00; matw2[4][3]=0.00;
  matw2[0][4]=0.20; matw2[1][4]=0.15; matw2[2][4]=0.10; matw2[3][4]=0.06; matw2[4][4]=0.00;
  break;
 }
  kd = kd*0.999;
  ks = ks*0.999;
  if (smat==9) {
  somataux = 0.0;
   mataux[0][0]= matw[0][0] * exp(kd*D[orderi[k-1]][orderj[k-1]]) * exp(ks*S[orderi[k-1]][orderj[k-
1]]);
  mataux[0][1]= matw[0][1] * exp(kd*D[orderi[k]][orderj[k-1]]) * exp(ks*S[orderi[k]][orderj[k-1]]);
```

```
   mataux[0][2]= matw[0][2] * exp(kd*D[orderi[k+1]][orderj[k-1]]) * exp(ks*S[orderi[k+1]][orderj[k-1]]);
  mataux[1][0]= matw[1][0] * exp(kd*D[orderi[k-1]][orderj[k]]) * exp(ks*S[orderi[k-1]][orderj[k]]);
  mataux[1][1]= matw[1][1] * exp(kd*D[orderi[k]][orderj[k]]) * exp(ks*S[orderi[k]][orderj[k]]);
  mataux[1][2]= matw[1][2] * exp(kd*D[orderi[k+1]][orderj[k]]) * exp(ks*S[orderi[k+1]][orderj[k]]);
    mataux[2][0]= matw[2][0] * exp(kd*D[orderi[k-1]][orderj[k+1]]) * exp(ks*S[orderi[k-1]][orderj[k+1]]);
  mataux[2][1]= matw[2][1] * exp(kd*D[orderi[k]][orderj[k+1]]) * exp(ks*S[orderi[k]][orderj[k+1]]);
    mataux[2][2]= matw[2][2] * exp(kd*D[orderi[k+1]][orderj[k+1]]) * exp(ks*S[orderi[k+1]][orderj[k+1]]);

  }
  else{
  somataux2= 0.0;
   mataux2[0][0]= matw2[0][0]* exp(kd*D[orderi[k-2]][orderj[k-2]]) * exp(ks*S[orderi[k-2]][orderj[k-2]]);
   mataux2[0][1]= matw2[0][1]* exp(kd*D[orderi[k-1]][orderj[k-2]]) * exp(ks*S[orderi[k-1]][orderj[k-2]]);//
  mataux2[0][2]= matw2[0][2]* exp(kd*D[orderi[k]][orderj[k-2]]) * exp(ks*S[orderi[k]][orderj[k-2]]);
  mataux2[0][3]= matw2[0][3]* exp(kd*D[orderi[k+1]][orderj[k-2]]) * exp(ks*S[orderi[k+1]][orderj[k-2]]);
  mataux2[0][4]= matw2[0][4]* exp(kd*D[orderi[k+2]][orderj[k-2]]) * exp(ks*S[orderi[k+2]][orderj[k-2]]);
   mataux2[1][0]= matw2[1][0]* exp(kd*D[orderi[k-2]][orderj[k-1]]) * exp(ks*S[orderi[k-2]][orderj[k-1]]);
   mataux2[1][1]= matw2[1][1]* exp(kd*D[orderi[k-1]][orderj[k-1]]) * exp(ks*S[orderi[k-1]][orderj[k-1]]);
  mataux2[1][2]= matw2[1][2]* exp(kd*D[orderi[k]][orderj[k-1]]) * exp(ks*S[orderi[k]][orderj[k-1]]);
  mataux2[1][3]= matw2[1][3]* exp(kd*D[orderi[k+1]][orderj[k-1]]) * exp(ks*S[orderi[k+1]][orderj[k-1]]);
  mataux2[1][4]= matw2[1][4]* exp(kd*D[orderi[k+2]][orderj[k-1]]) * exp(ks*S[orderi[k+2]][orderj[k-1]]);
  mataux2[2][0]= matw2[2][0]* exp(kd*D[orderi[k-2]][orderj[k]]) * exp(ks*S[orderi[k-2]][orderj[k]]);
    mataux2[2][1]= matw2[2][1]* exp(kd*D[orderi[k-1]][orderj[k]]) * exp(ks*S[orderi[k-1]][orderj[k]]);//
  mataux2[2][2]= matw2[2][2]* exp(kd*D[orderi[k]][orderj[k]]) * exp(ks*S[orderi[k]][orderj[k]]);
```

```
    mataux2[2][3]= matw2[2][3]* exp(kd*D[orderi[k+1]][orderj[k]]) * exp(ks*S[orderi[k+1]][orderj
[k]]);
    mataux2[2][4]= matw2[2][4]* exp(kd*D[orderi[k+2]][orderj[k]]) * exp(ks*S[orderi[k+2]][orderj
[k]]);
   mataux2[3][0]= matw2[3][0]* exp(kd*D[orderi[k-2]][orderj[k+1]]) * exp(ks*S[orderi[k-2]][orderj
[k+1]]);//
   mataux2[3][1]= matw2[3][1]* exp(kd*D[orderi[k-1]][orderj[k+1]]) * exp(ks*S[orderi[k-1]][orderj
[k+1]]);
     mataux2[3][2]= matw2[3][2]* exp(kd*D[orderi[k]][orderj[k+1]]) * exp(ks*S[orderi[k]][orderj
[k+1]]);
   mataux2[3][3]= matw2[3][3]* exp(kd*D[orderi[k+1]][orderj[k+1]]) * exp(ks*S[orderi[k+1]][orderj
[k+1]]);
   mataux2[3][4]= matw2[3][4]* exp(kd*D[orderi[k+2]][orderj[k+1]]) * exp(ks*S[orderi[k+2]][orderj
[k+1]]);//
    mataux2[4][0]= matw2[4][0]* exp(kd*D[orderi[k-2]][orderj[k+2]]) * exp(ks*S[orderi[k-2]][orderj
[k+2]]);
    mataux2[4][1]= matw2[4][1]* exp(kd*D[orderi[k-1]][orderj[k+2]]) * exp(ks*S[orderi[k-1]][orderj
[k+2]]);
     mataux2[4][2]= matw2[4][2]* exp(kd*D[orderi[k]][orderj[k+2]]) * exp(ks*S[orderi[k]][orderj
[k+2]]);
   mataux2[4][3]= matw2[4][3]* exp(kd*D[orderi[k+1]][orderj[k+2]]) * exp(ks*S[orderi[k+1]][orderj
[k+2]]);//
   mataux2[4][4]= matw2[4][4]* exp(kd*D[orderi[k+2]][orderj[k+2]]) * exp(ks*S[orderi[k+2]][orderj
[k+2]]);
 }
   if (smat == 9){
  for (t1=0;t1<=2;t1++)
   for (t2=0;t2<=2;t2++){
    somataux += mataux[t1][t2];
   }
  for (t1=0;t1<=2;t1++)
   for (t2=0;t2<=2;t2++){
    mat3[t1][t2]= mataux[t1][t2]/somataux;
    }
   for(t=0;t<9;t++){soma+=mat3[t/3][t%3];acc3[t]=soma;}
  else{
```

```
    for (t1=0;t1<=4;t1++)
   for (t2=0;t2<=4;t2++){
     somataux2 += mataux2[t1][t2];
   }
  for (t1=0;t1<=4;t1++)
   for (t2=0;t2<=4;t2++){
     mat5[t1][t2]= mataux2[t1][t2]/somataux2;
     }
  for(t=0;t<25;t++){soma+=mat5[t/5][t%5];acc5[t]=soma;}
 }
void Desenha(void)
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
  // Limpa a janela de visualização com a cor de fundo especificada
    glClear(GL_COLOR_BUFFER_BIT);
   // Especifica que a cor corrente é vermelha
    //        R     G     B
    glColor3f(0.7f, 0.7f, 0.8f);
    // Desenha n pontos com a cor corrente
g=0;
refaz:;
temp=0;
kd=100;
for(r1=0;r1<=200;r1++){
       for(r2=0;r2<=200;r2++){
        D[r1][r2] = 0;
      //  S[r1][r2] = 0;
       }
    }
g++;
n2=0; //Numero total de celulas ativas
for(i=0;i<nx;i++){ //Inicializa os estados das celulas
  for(j=0;j<ny;j++)
   if(rnd(idum)<pInicial && (tipo[i][j] & 1)==1){
      tab[i][j]=1;
```

```
      orderi[n2]=i;//Inicializa a ordem de processamento
      orderj[n2]=j;
      n2++;
    }
    else tab[i][j]=0;
}
peoples = n2;
nsims=0;
iniciodes:;
nsims++;
for(k=0;k<n2;k++){ //embaralha a ordem de processamento
  auxi=orderi[k];
  auxj=orderj[k];
  trocak=(int)(rnd(idum)*n2);
  orderi[k]=orderi[trocak];
  orderj[k]=orderj[trocak];
  orderi[trocak]=auxi;
  orderj[trocak]=auxj;
}
for(k=0;k<n2;k++){//varredura
  iniciovarr:;
  i00=orderi[k];
  j00=orderj[k];
  //if(tipo[i00][j00] & 4){}// in
  if(tipo[i00][j00] & 8){ // out
    if(rnd(idum)<pSai){
      tab[i00][j00]=0;
      n2--;
      if(n2>k){
        orderi[k]=orderi[n2];
        orderj[k]=orderj[n2];
        goto iniciovarr;
      }
    }
  }
  else
```

```
  {
    redefine(traj[orderi[k]][orderj[k]]);
    D[orderi[k]][orderj[k]]+=0.1;
   if (D[orderi[k]][orderj[k]]> maximo){
          maximo = D[orderi[k]][orderj[k]];
    }
    mov(2, i00, j00, &ni, &nj);
    orderi[k]=ni;
    orderj[k]=nj;
  }
  if (exibesimulacao) plote(i00,j00,pixsize,8);
}
/*********************************/
if (exibesimulacao){
 for(i=0;i<ny;i++){
  for(j=0;j<nx;j++)
   plote(i,j,pixsize,tab[i][j]);
   glFlush();
  for(i=0;i<nx;i++){
  for(j=0;j<ny;j++){
   aux = tipo[i][j];
   if (aux ==3) aux =1;
   plote(i,j,pixsize,aux);
  }
 }
 }
}
/*********************************/
 if (n2>0) goto iniciodes;
 temp = (nsims*0.298142)/60;
 //fprintf(sai,"numero de simulações %d\n ",nsims);
 fprintf(sai," %d tempo total de evacuação e tempo de metade : %f minutos / %d",g,temp, nsims);
 fprintf(sai,"          numero de pessoas no dominio: %d pessoas\n",peoples);
  somatemp += temp;
 if (g<replicacoes) goto refaz;
 media = somatemp/replicacoes;
```

```
 fprintf(sai,"tempo medio de evacuação total: %f minutos\n",media);
 fprintf(sai," ");
/*  for(i=0;i<=256;i++){
       for(j=0;j<=256;j++){
        fprintf(sai," %d ",D[j][i]);
       }
    fprintf(sai,"\n");
    }   */
fim:;


}
// Inicializa parâmetros de rendering
void Inicializa (void)
{
   // Define a cor de fundo da janela de visualização como preta
   glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
   //glClearColor(100, 20, 180, 1.0f);
}

// Função callback chamada quando o tamanho da janela é alterado
void AlteraTamanhoJanela(GLsizei w, GLsizei h)
{
                // Evita a divisao por zero
                if(h == 0) h = 1;
                // Especifica as dimensões da Viewport
                glViewport(0, 0, w, h);
                // Inicializa o sistema de coordenadas
                glMatrixMode(GL_PROJECTION);
                glLoadIdentity();
                // Estabelece a janela de seleção (left, right, bottom, top)
                if (w <= h)
                        gluOrtho2D (0.0f, 250.0f, 0.0f, 250.0f*h/w);
                else
                        gluOrtho2D (0.0f, 250.0f*w/h, 0.0f, 250.0f);
}
// Programa Principal
```

```
int main(void)
{
    par=fopen("parametros.txt","r");
    fscanf(par,"%s",layout);
    fscanf(par,"%s",trajetoria);
    fscanf(par,"%s",campo);
    fscanf(par,"%d",&nx);
    fscanf(par,"%d",&ny);
    fscanf(par,"%d",&pixsize);
    fscanf(par,"%d",&depth);
    fscanf(par,"%d",&nsimsmax);
    fscanf(par,"%d",&lancemax);
    fscanf(par,"%d",&lancelenmax);
    fscanf(par,"%d",&pause1);
    fscanf(par,"%d",&pause2);
    fscanf(par,"%d",&printflag);
    fscanf(par,"%d",&ntries);
    fscanf(par,"%d",&idum);
    fscanf(par,"%lg",&pEntra);
    fscanf(par,"%lg",&pSai);
    fscanf(par,"%lg",&pInicial);
    fscanf(par,"%d",&smat);
    fscanf(par,"%d",&replicacoes);
    fscanf(par,"%d",&exibesimulacao);
    fclose(par);
   leia=fopen(layout,"r");
    for(i=0;i<ny;i++)
       for(j=0;j<nx;j++)
        fscanf(leia,"%d",&tipo[j][i]);
    fclose(leia);
    leiadir=fopen(trajetoria,"r");
    for(i=0;i<ny;i++)
       for(j=0;j<nx;j++)
        fscanf(leiadir,"%d",&traj[j][i]);
    fclose(leiadir);
    leiacampo=fopen(campo,"r");
```

```
   for(i=0;i<ny;i++)
     for(j=0;j<nx;j++)
       fscanf(leiacampo,"%lg",&S[i][j]);
    fclose(leiacampo);
    for(j=0;j<ny;j++)
      for(i=0;i<nx;i++)
        ocup[i][j]=1;

    //inicialização aqui (coisas que só são acionadas uma única vez)
    sai=fopen("saida.txt","w");

    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(700,700);
    glutInitWindowPosition(10,10);
    glutCreateWindow("Simulação de tráfego de pessoas ");
     glutDisplayFunc(Desenha);
     glutReshapeFunc(AlteraTamanhoJanela);
     Inicializa();
     glutMainLoop();
     fclose(sai);
     printf("concluido");
}
double logpow(double x1, double x2){ /* power(2,3)=8 */
  return( (log(x1)*x2) );
}
double power(double x1, double x2){ /* power(2,3)=8 */
  return( exp(log(x1)*x2) );
}
double sqr( double x ){
  return (x*x);
}
int movever(int j){
  double auxr;
  auxr=rnd(idum);
  if(auxr<0.25)return(j-1);
  else if(auxr<0.75)return(j);
```

```
  else return(j+1);
}
int movehor(int i){
  double auxr;
  auxr=rnd(idum);
  if(auxr<0.10)return(i-1);
  else if(auxr<0.20)return(i);
  else return(i+1);
}
int mov(int range, int i0, int j0, int *ni, int *nj){
  double aux1,aux2;
  int p,q,i,j,k,cont=0,i00,j00,i0min,i0max,j0min,j0max;
  i0min=i0-2*range; if(i0min<0)i0min=0;
  i0max=i0+2*range; if(i0max>=nx)i0max=nx-1;
  j0min=j0-2*range; if(j0min<0)j0min=0;
  j0max=j0+2*range; if(j0max>=ny)j0max=ny-1;
  for(i=i0min;i<=i0max;i++)// Inicializa mark[][]
    for(j=j0min;j<=j0max;j++)
      mark[i][j]=0;
  i0min=i0-range; if(i0min<0)i0min=0;
  i0max=i0+range; if(i0max>=nx)i0max=nx-1;
  j0min=j0-range; if(j0min<0)j0min=0;
  j0max=j0+range; if(j0max>=ny)j0max=ny-1;
  for(i=i0min;i<=i0max;i++){//Localiza as ativas proximas
    for(j=j0min;j<=j0max;j++){
      if(tab[i][j]==1 && i!=i0 && j!=j0)
      for(lance=0;lance<lancemax;lance++){
        i000=i00=i;
        j000=j00=j;
        for(lancelen=0;lancelen<lancelenmax;lancelen++){
          if(tipo[i00][j00] & 2 ){//fronteira do dominio
            fbound(i000,j000,&i00,&j00);
            mark[i00][j00]++;
          }
          else{//dentro do dominio
            finner(i000,j000,&i00,&j00);
```

```
            mark[i00][j00]++;
          }
        }
      }
    }
}
if(printflag){
fprintf(sai,"%d %d:\n",i0,j0);
for(i=i0min;i<=i0max;i++){
  for(j=j0min;j<=j0max;j++)fprintf(sai,"%d ",mark[i][j]);
  fprintf(sai,"\n");
}
fprintf(sai,"\n");
}
r=rnd(idum);
k=0;
if(smat==9){
 while(k<smat && r>acc3[k])k++;
 *ni=k/3-1+ i0;
 *nj=k%3-1+ j0;
 if(*ni!=i0 || *nj!=j0){
   tab[i0][j0]=0;
   if(*ni<0 || *ni>=nx || *nj<0 || *nj>=ny || mark[*ni][*nj]>0 ||
      tab[*ni][*nj]>0 || (tipo[*ni][*nj]&1)==0){
     *ni=i0;
     *nj=j0;
     tab[i0][j0]=1;
   }
   else{
     tab[*ni][*nj]=1;
   }
 }
}
else{//smat==25
 while(k<smat && r>acc5[k])k++;
 *ni=k/5-2+ i0;
```

```
   *nj=k%5-2+ j0;
   if(*ni!=i0 || *nj!=j0){
     tab[i0][j0]=0;
     if(*ni<0 || *ni>=nx || *nj<0 || *nj>=ny || mark[*ni][*nj]>0 ||
        tab[*ni][*nj]>0 || (tipo[*ni][*nj]&1)==0){
       *ni=i0;
       *nj=j0;
       tab[i0][j0]=1;
     }
     else{
       tab[*ni][*nj]=1;
     }
   }
  }
}
void finner(int i000, int j000, int *i00, int *j00){
  int i,j,k;
  double r;
  r=rnd(idum);
  k=0;
  if(smat==9){
    while(k<smat && r>acc3[k])k++;
    *i00=k/3-1+ *i00;
    *j00=k%3-1+ *j00;
  }  else{//smat==25
    while(k<smat && r>acc5[k])k++;
    *i00=k/5-2+ *i00;
    *j00=k%5-2+ *j00;
  }
}
void fbound(int i000, int j000, int *i00, int *j00){
 int i,j,k,tent=0,tentmax=5;
  double r;
  do{
     r=rnd(idum);
     k=0;
```

```
    if(smat==9){
      while(k<smat && r>acc3[k])k++;
      i=k/3-1+ *i00;
      j=k%3-1+ *j00;
    }
    else{//smat==25
      while(k<smat && r>acc5[k])k++;
      i=k/5-2+ *i00;
      j=k%5-2+ *j00;
    }
    tent++;
  }while((tab[i][j] & 1)!=1 && tent<tentmax);
  if(tent<tentmax){
    *i00=i;     *j00=j;
  }
}
float rnd(long idum0){
  long k;
  float ans;
  do{
    idum0 ^= MASK;
    k=(idum0)/IQ;
    idum0=IA*(idum0-k*IQ)-IR*k;
    if(idum0<0) idum0+=IM;
    ans=AM*(idum0);
    idum0^=MASK;
    idum=idum0;
  }while(ans>0.999999);
  return(ans);
}
void plote(int x, int y, int t ,int c){
  int i,j,a,b;
  double R=0.0,G=0.0,B=0.0;
  if(c==8)glColor3f(1.0,1.0,1.0);
  else{
    if(c>0){R=G=B=0.3;}
```

```
    c/=2; if(c%2)R+=0.7;
    c/=2; if(c%2)G+=0.7;
    c/=2; if(c%2)B+=0.7;
    glColor3f(R,G,B);
  }
  a=x*t;  b=y*t;
  glBegin(GL_QUADS);
            glVertex2i(a+10,b+10);
            glVertex2i(a+t+10,b+10);
            glVertex2i(a+t+10,b+t+10);
            glVertex2i(a+10,b+t+10);
  glEnd();
}
```

Referências Bibliográficas

Achcar, J. A., Damasceno, V. L. (1996). Extreme value regression models: an uselful reparametrization for the survival function. *J. Appl. Stat*., 23: 59-68,

Cheah, J. & MacGregor Smith, J. (1994). Generalized M/G/c/c state dependent queueing models and pedestrian traffic flows. *Queueing Systems*, 15:365-386.

Cruz, F. R. B., MacGregor Smith, J. & Medeiros, R. O. (2005a). An M/G/c/c satedependent network simulation model, *Computers & Operations Research*, 32(4), 919-941.

Cruz, F. R. B., MacGregor Smith J, & Queiroz, D.C. (2005b). Service and capacity allocation in *M/G/c/c* state dependent queueing networks, *Computers & Operations Research*, 32(6), 1545-1563.

Hopcroft, J., & Ullman, J. (1979). Introduction to Automata Theory, Languages, and Computation, Addison-Wesley 80(3) 232-298

Kelvin, H.L.W., Luo, M. (2005). Computational Tool in Infrastructure Emergency Total Evacuation Analysis. International Conference on Intelligence and Security Informatics 92 (3): 536–542.

Kerbache, L. & MacGregor Smith, J. (2000). Multi-objective routing within large scale facilities using open finite queueing networks, European Journal of Operational Research 121(1): 105–123.

Rangel, J.L., Guedes, L.C. (2004). Autômatos Finitos e Expressões Regulares. Linguagens Formais 36 (4), 60–75.

Shadschneider, A. (2001) Bionics-inspired cellular automaton model for pedestrians dynamics, Trafic and Granular Flow 90(3): 110-114.

Schadschneider, A., Burstedde, C., Kirchner, A., Klauck, K. & Zittartz J., (2002) Pedestrian and Evacuation Dynamics, ed. by M. Schreckenberg, S. D. Sharma (Springer, Berlin) pp87

Tregenza, P. R. (1976). *The Design of Interior Circulation*, Van Nostrand Reinhold Company, New York, USA.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo