SIMULAÇÃO DE MONTE CARLO NO PYTHON: UM ESTUDO INICIAL DA EVACUAÇÃO DE PESSOAS EM AMBIENTES FECHADOS

Cristiano Alves de Sousa (PUC Goiás)
cristianoas.gyn@gmail.com

Maria José Pereira Dantas Drª. (PUC Goiás)
mjpdantas@gmail.com

Clarimar José Coelho Dr. (PUC Goiás)
clarimarc@gmail.com

Resumo. *O fluxo de grandes multidões de pedestres é um problema muito relevante com o aumento das populações nas cidades. Existe a preocupação quanto à necessidade de segurança e infraestruturas adequadas. Nas últimas décadas esses estudos de fluxo de pedestres foram empreendidos, mas ainda não são suficientes. Um estudo ainda maior sobre esses fenômenos coletivos do comportamento de uma multidão, numa situação de emergência, é de grande importância, pois hoje ainda ocorrem acidentes de grandes proporções relacionados a multidões. O poder computacional disponível atualmente permite que as análises sejam maiores e mais complexas, fornecendo soluções de maior qualidade. Com isso este artigo apresenta uma implementação da Simulação de Monte Carlo para situações de Evacuação do Tráfego de Pedestres em ambiente fechado, com o desenvolvimento de um simulador desenvolvido na linguagem de programação Python. No ambiente simulado os pedestres estão uniformemente distribuídos, sem pânico, admitindo uma distribuição aleatória dos mesmos. Foram considerados pedestres com características inerentes sem pânico e a racionalidade. A análise dos resultados confirmou-se a tendência linear com o aumento do número de pedestres dentro da sala independente dos tamanhos analisados. E constatou-se que a linguagem de programação Python mostrou-se ágil, simples e intuitivo o que possibilitará o desenvolvimento colaborativo do simulador para futuras evoluções..*

Palavras Chave: *Tráfego de Pedestres, Monte Carlo, Python, Fluxo de Pedestres, Evacuação, Simulador.*

1. Introdução

A modelagem e simulação têm sido técnicas de apoio à decisão mais conhecidas (LUBAN, 2005). Para Saliby (1989), a simulação é uma abordagem de estudo utilizada, nas mais variadas área de conhecimento, devido a dois fatores que contribuem para isso: a crescente complexidade dos problemas e a maior disponibilidade de recursos computacionais. De acordo com Bateman et al. (2013), simulação é um processo de experimentação com um modelo detalhado de um sistema real, para determinar como um sistema responderá a mudanças em sua estrutura, ambiente ou condições de contorno. A simulação é uma reprodução de um item ou evento, seu objetivo específico é de imitar ou simular um sistema real, para que se possa explorá-lo, realizar experimentos e compreendê-lo antes da implementação de alternativas de decisão no mundo real (ALBRIGHT; WINSTON, 2007).

Segundo Luban e Hîncu (2009), um modelo de simulação pode ser uma ferramenta útil e versátil para obter percepções sobre o funcionamento do sistema. Uma das maiores vantagens da simulação está em permitir análises relacionadas a responder perguntas, como: o que aconteceria se? Permitindo olhar para o futuro, sob determinados pressupostos (BANKS et al., 2009; CHWIF; MEDINA, 2007; ROBINSON, 2008; KELTON, SADOWSKI; STURROCK, 2007).

Modelar o tráfego de pedestres tem se tornado uma prioridade nas grandes cidades, nas rodovias e vias em geral. Contudo, ainda há poucas pesquisas na área de modelagem de tráfego de pedestres e continuam ocorrendo grandes acidentes relacionados a multidões. Com a modelagem de tráfego de pedestres podemos tentar solucionar problemas como: Tornar possível a análise de estruturas para pedestres ainda não planejadas e construídas; Indicar melhorias para estruturas já existentes; Gerenciar o fluxo de pedestres em instalações já construídas, etc.

O nome "Monte Carlo" surgiu durante o projeto Manhattan na Segunda Guerra Mundial. No projeto de construção da bomba atômica, Ulam, von Neumann e Fermi consideraram a possibilidade de utilizar o método, que envolvia a simulação direta de problemas de natureza probabilística relacionados com o coeficiente de difusão do nêutron em certos materiais (Hammersley e Handscomb, 1964).

O método de Monte Carlo (MMC) é um método estatístico utilizado em simulações estocásticas com diversas aplicações, cita-se a possibilidade de resolver problemas usando números aleatórios, para implementar o método numericamente, é imprescindível um bom

gerador de números aleatórios. O método de Monte Carlo consiste de forma sistemática, na geração de valores aleatórios para cada distribuição de probabilidade dentro de um modelo com o objetivo de produzir vários cenários.

Algumas das vantagens são:

- Distribuição das variáveis do modelo não precisa ser aproximada;
- Correlações e outras interdependências podem ser modeladas;
- O computador realiza todo trabalho de geração dos valores aleatórios;
- O nível de precisão da simulação pode ser melhorado através de um simples aumento do número de interações calculadas;
- A validade da teoria da simulação de Monte Carlo é amplamente reconhecida, o que permite que seus resultados sejam facilmente aceitos;
- Alterações no modelo podem ser feitas rapidamente e os novos resultados podem ser comparados com os anteriores.

A fim de criar um ambiente computacional eficiente e facilmente expansível para o desenvolvimento e teste de métodos em processamento de simulação. Assim optamos por utilizar a linguagem de programação Python com ideia de utilizar as bibliotecas de código aberto e que possibilitasse a implementação de uma interface gráfica para simular a evacuação.

Python é uma linguagem de programação de alto nível, interpretada e passou a ser bastante usada na computação científica. Conta com uma grande disponibilidade de bibliotecas ou pacotes para várias áreas, sendo que esses pacotes facilitam a utilização da linguagem na computação científica. Essas bibliotecas estão reunidas sob a assim chamada pilha SciPy (Scientific Python). As principais bibliotecas que compõe a pilha SciPy [OLIPHANT, 2007, HUNTER, 2007, Pérez and Granger, 2007] são:

- Python – a própria linguagem de programação Python; ˜
- SciPy – biblioteca de funções numéricas; ´
- NumPy – biblioteca com a definição de tipos de array e matrizes, bem como suas respectivas operações; ˜
- Matplotlib – biblioteca popular para produção de gráficos de alta qualidade;
- Pandas – biblioteca de análise de dados e estrutura de dados de alta performance;
- SymPy – biblioteca matemática simbólica e álgebra computacional;

• Random - biblioteca que implementa geradores de números pseudo-aleatórios;

Vale a pena lembrar que existem muitas outras bibliotecas não listadas neste artigo que estão disponíveis para área científica.

Este artigo apresenta os resultados da simulação de Monte Carlo para o fluxo de pedestres, em uma evacuação em ambiente fechado, em que os pedestres estão uniformemente distribuídos, sem pânico, admitindo uma distribuição aleatória dos mesmos. Foram considerados pedestres com características inerentes sem pânico e a racionalidade.

2. Materiais e Métodos

Os métodos da fenomenologia foram aplicados para os pedestres, sendo que algumas características podem ser analisadas e a racionalidade é uma delas. Foram definidos os seguintes estudos comportamentais para todos os pedestres:

- os pedestres são: **agentes ativos**, ou seja, em condições normais, sem pânico, partilham o mesmo objetivo de andar com a velocidade constante até atingirem os seus objetivos (por exemplo, portas de saída), evitando possíveis obstáculos e evitando as zonas mais concorridas.
- os pedestres são **agentes inteligentes**, isto é, sua mente avalia, seleciona e/ou faz a síntese do que ele percebe de acordo com vários critérios psicológicos (por exemplo, o nível de ansiedade ou a capacidade de realizar avaliações do grupo), ou seja, ele opta por espaços menos tumultuados (com densidade menor), com o objetivo de atingir a saída.
- os pedestres não **são igualmente afetados** por estímulos vindos de todas as direções no espaço. Especificamente, eles distinguem entre frente e laterais, sendo que tem uma tendência maior a caminharem para frente;
- em condições normais externas e subjetivas, os pedestres não percebem a estrutura total de caminhada. São afetados apenas por regiões ao seu redor, dentro de seu campo de visão. (VARGAS, Mariana, 2010, p.3)

Com base nessas características simula-se a movimentação desses pedestres em uma sala fechada onde tem uma única porta de saída.

- A sala fechada é representada por uma malha retangular de $(m+1 \times n+1)$ $m2$,
  onde: m = comprimento da sala e n = largura. Supõe-se que p = pedestres distribuídos

uniformemente na malha.

- E a porta de saída localizada no ponto P(x_i, y_{n+1}) *com* $i = 1,2,...,m$ conforme ilustrado na Figura 1.

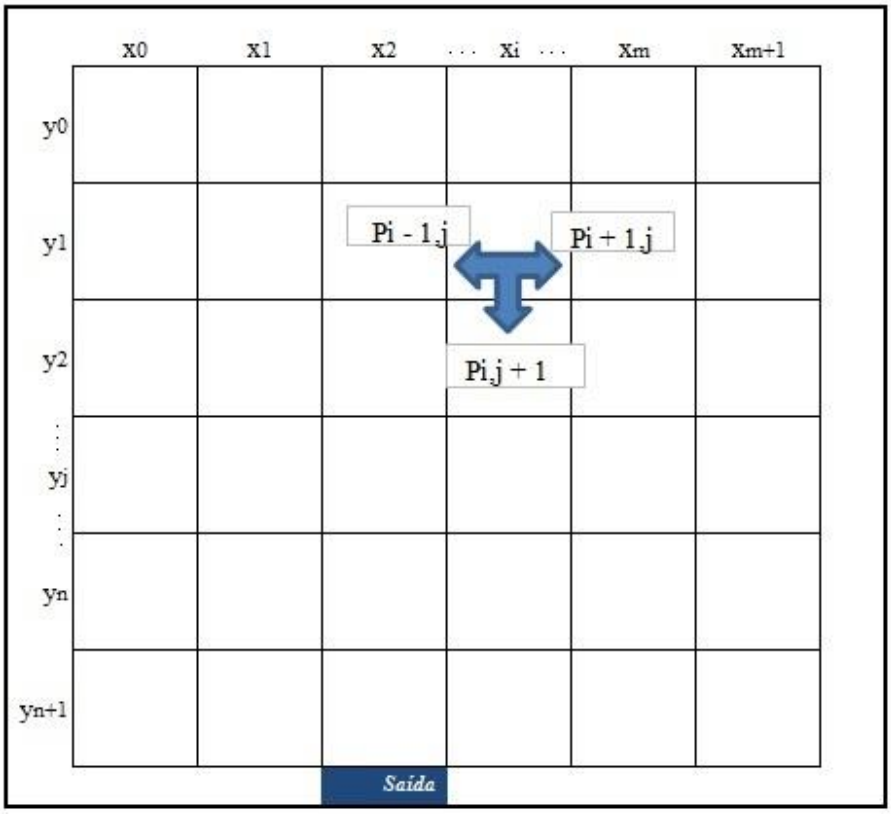


Figura 1: Malha retangular de tamanho (*m+1 x n+1*) com *p* pedestres distribuídos uniformemente na malha. Os pedestres não ocupam as bordas, exceto na posição da porta, quando saem. Os pedestres só caminham no sentido das setas.

O pseudocódigo de Vargas R.P.G. et. al. (2015, p.6) que resulta na saída de todos os pedestres inicialmente nesta malha é dado por:

> **Entrada**: a dimensão da malha: *m* e *n*; o número de pedestre: *p*; a localização da porta*: P*(x_i, y_{n+1})*, com* $i = 1,2,...,m$. **Saída**: o tempo *t* decorrido para a saída de todos os pedestres. **Passo 1**: $t = 0$; **Passo 2**: escolhe-se aleatoriamente um pedestre p_{ij}. **Passo 3**: testando as possíveis posições do pedestre p_{ij}. **Passo 3a**: se $(x_i, y_{i+1}) = P$, a posição a frente do pedestre é igual a da porta, então p_{ij} sai. **Passo 3b**: se $(x_i, y_{j+1}) = 0$, a posição a sua frente está vazia, então p_{ij} ocupa esta posição. **Passo 3c**: se $(xi-$

$_{1,yj}) = 0$ e $_{(xi+1,yj)} = 0$, as posições laterais estão vazias, então pij ocupa uma destas posições aleatoriamente. **Passo 3d**: se as posições $(x_{i-1},y_j) = (x_{i+1},y_j) = (x_i,y_{j+1}) = 1$, ou seja todas as posições laterais e a frente estão ocupadas, então pij permanece na mesma posição. **Passo 4**: repetir o passo 3 até que todos os p_{ij} tenham se movimentado um única vez. **Passo 5**: $t = t + 1$, incrementa-se o tempo. **Passo 6**: repetir os Passos 2 ao 5 até que todos tenham saído. **Passo7**: SAÍDA: o tempo t (VARGAS, Mariana, 2010, p.3).

Pseudocódigo: Descrição do movimento dos pedestres na malha para o esvaziamento da sala.

2.1. – Protótipo do Simulador.

Construir um protótipo de aplicativo pode poupar tempo, trabalho de desenvolvimento do design, aparência e usabilidade (SANTOS, 2004). A interface gráfica do usuário (GUI - Graphic User Interface), que permite a interação com o simulador por meio de elementos gráficos onde o usuário é capaz de informar os dados de entrada e gerar dos gráficos de Distribuição dos tempos e Distribuição das médias.

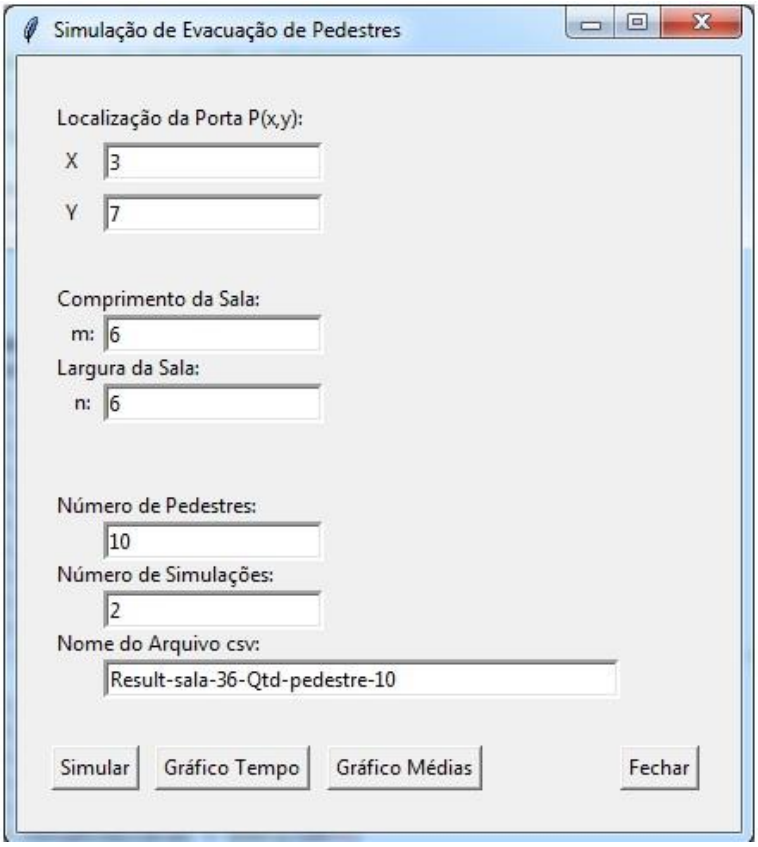


Figura 2: A interface gráfica do usuário para a interação com o simulador.

2.2, Fluxograma gerado a partir do pseudocódigo para o esvaziamento da sala.

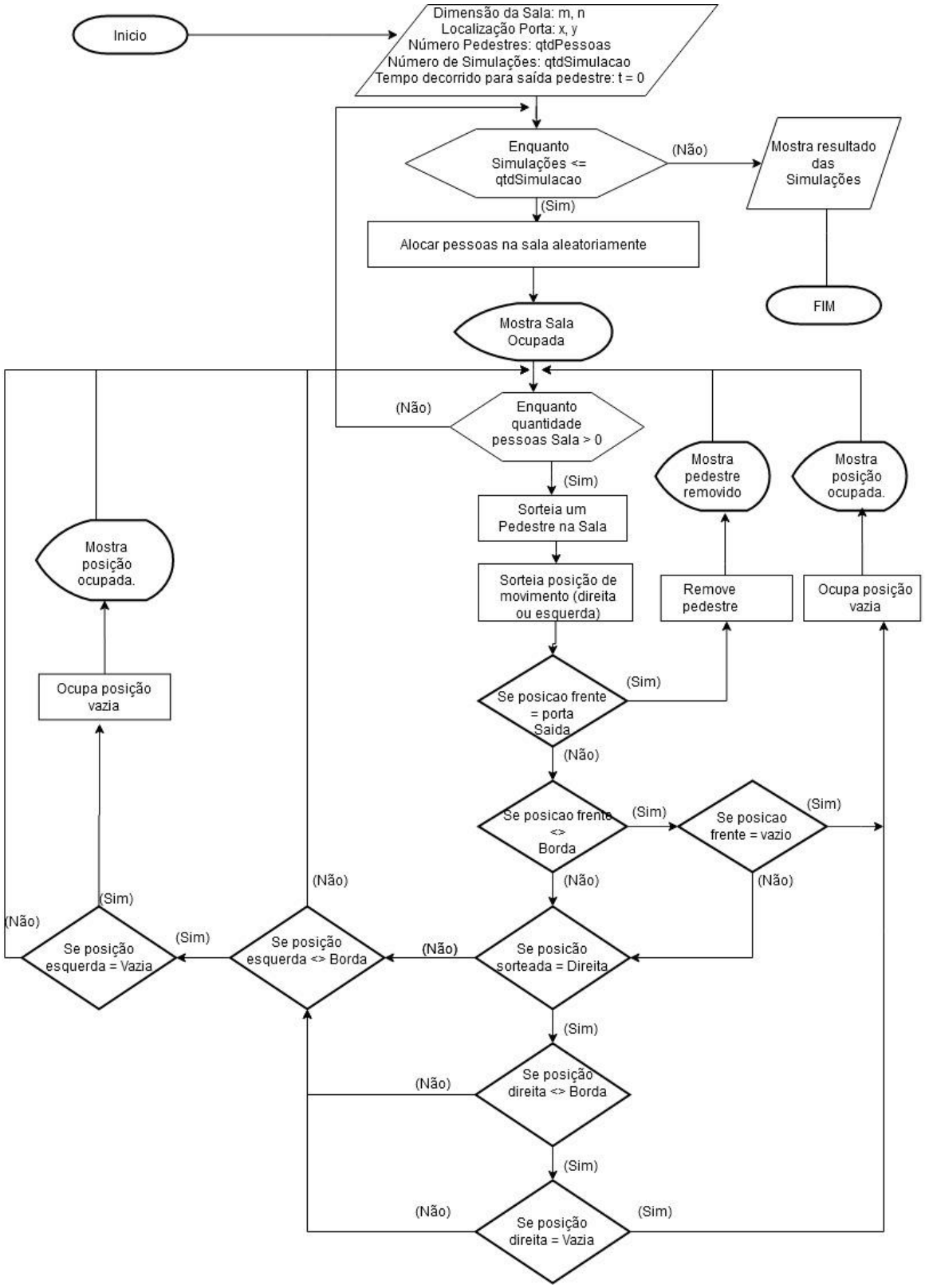


Figura 3: Fluxograma gerado a partir do pseudocódigo para movimentação dos pedestres em uma sala fechada onde tem uma única saída.

2.1. Desenvolvimento do Algoritmo em Python

Para implementação do algoritmo em Python foram utilizadas as seguintes bibliotecas: Pandas, Matplotlib, Random, csv, Tkinter e Time. O comportamento do algoritmo foi divido em vários métodos onde cada um determina as possíveis movimentações dos pedestres na sala.

```
def testaBorda(parmlimLi, parmlimCo, parmLi, parmCo):
    if (parmLi == 0 or parmLi == parmlimLi):
        return "BORDA"
    elif (parmCo == 0 or parmCo == parmlimCo):
        return "BORDA"
    else:
        return "NAO_BORDA"
```

Algoritmo 1: Método "**testaBorda**": método para testar se é o limite da borda, o pedestre não pode ocupar esses limites da borda;

No algoritmo 1, encontra-se o método testa borda é utilizado para testar se a posição que será ocupada é o limite da sala. Os parâmetros são: limites de linha e coluna juntamente com a posição da linha e coluna atual.

```
def testaPortaSaida(parmY, parmX, parmLi, parmCo):
    if (parmLi == y and parmCo == x):
        return "PORTA"
    else:
        return "PROXIMO"
```

Algoritmo 2: Método "**testaPortaSaida**": método para testar se a posição é a porta de saída;

No algoritmo 2, encontra-se o método que testa se a posição que ocupada é a porta de saída da sala. Os parâmetros são: Posição da porta (parmY e parmX) e posição ocupada (parmLi e parmCo).

```
def removePessoa(parmMa, parmLi, parmCo):
    parmRemovida = parmMa[parmLi,parmCo]
    parmMa[parmLi,parmCo] = 0
    return parmMa, parmRemovida
```

Algoritmo 3: Método "**removePessoa**": método para remover a pedestre da sala;

No algoritmo 3, encontra-se o método para remover o pedestre da sala. É passado como parâmetros a matriz (parmMa) que representa a sala com as posições de linha e coluna (parmLi e parmCo).

```
def geraNumeroAleatorio(parmLimMalha):
    parmLi = 0
    parmCo = 0
    parmLi = random.randint(1,parmLimMalha)
    parmCo = random.randint(1,parmLimMalha)
    return parmLi, parmCo
```

Algoritmo 4: Método "**geraNumeroAleatorio**": método para gerar uma posição aleatória do pedestre na sala.

No algoritmo 4, encontra-se o método para gerar números aleatórios. Na linha 4 e 5 do código temos a função "random.randint" usada para gerar um número inteiro aleatório importada do módulo random.

```
def sorteiaPessoa(parmMa):
    myListaPessoas = []
    myListaPessoas = np.concatenate(parmMa)
    myListaPessoas_valida = []
    for elem in myListaPessoas:
        if elem > 0:
            myListaPessoas_valida.append(elem)

    return random.choice(myListaPessoas_valida)
```

Algoritmo 5: Método "**sorteiaPessoa**": método para fazer o sorteio aleatório de um pedestre para se movimentar.

No algoritmo 5, encontra-se o método sortear um pedestre aleatoriamente. Na linha 8 do código apresenta a função "random.choice" usada para selecionar um elemento aleatório de uma sequência não vazia

```
def testaPosicaoOcupada(parmMa, parmLi, parmCo):
  vlrPosicao = parmMa[parmLi, parmCo]
  if vlrPosicao == 0:
     return "VAZIO"
  else:
     return "OCUPADO"
```

Algoritmo 6: Método "**testaPosicaoOcupada**": método para testar se a próxima posição está ocupada por algum pedestre.

No algoritmo 6, encontra-se o método para testar se a próxima posição a movimentar está ocupada por outro pedestre. Recebe como parâmetro a matriz (sala) com a linha e coluna.

```
def geracaoDados(ma, qt, limMalha):
   li = 0
   co = 0
   res = (0,)
   num = 1

   while len(res) <= qt:
     li, co = geraNumeroAleatorio(limMalha)
     if testaPosicaoOcupada(ma, li, co) == "VAZIO":
        ma[li, co] = num
        if num not in res:
           res += (num,)

        num = num + 1
   return ma
```

Algoritmo 7: Método "geracaoDados": método para distribuir os pedestres aleatoriamente na sala. É gerada uma posição aleatória para cada pessoa na malha.

No algoritmo 7, encontra-se o método geração de dados é usado para colocar os pedestres aleatoriamente na sala. Na linha 8 do código é feito uma chamada para o método gera número aleatório (algoritmo 4).

3. Resultados

Nesta seção, descreve-se as respostas obtidas pelas análises das simulações executadas. Primeiramente a simulação foi executada com os seguintes parâmetros:

Localização da porta *P*(x,y) *onde x = 3 e y = 7;*

Comprimento da malha(sala) de 36 m² onde *m = 6 e n = 6*;

Número de pedestres na sala foi de *1 a 36;*

Número de simulações *1000* para cada caso;

Nome do arquivo CSV "Result-sala-36-Qtd-pedestre-XX" onde XX = 1 ... 36.

Em seguida os parâmetros foram alterados e executadas novas simulações com:

Localização da porta *P*(*x,y*) *onde x = 3 e y = 8;*

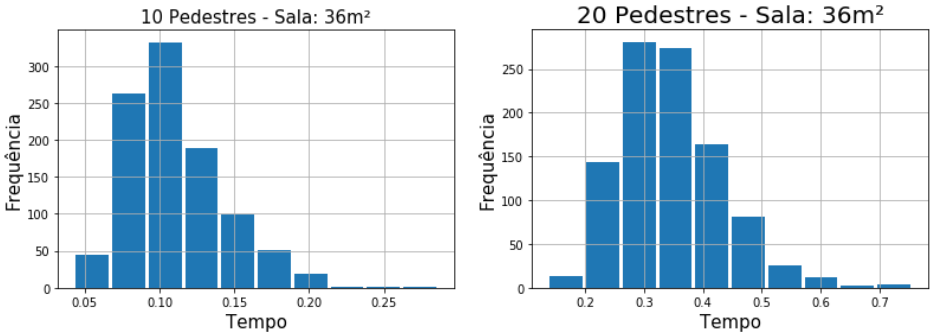
Comprimento da malha(sala) de 49 m² onde *m = 7 e n = 7*;

Número de pedestres na sala foi de *1 a 36;*

Número de simulações 1000 para cada caso;

Nome do arquivo CSV "Result-sala-49-Qtd-pedestre-XX" onde XX=1...36.

Através dos gráficos da figura 4, pode-se observar o histograma com a distribuição dos tempos, em que foram analisados 3 casos de cada sala. Na figura 3 é observado as médias do tempo para evacuação, de acordo com a quantidade de pedestres.



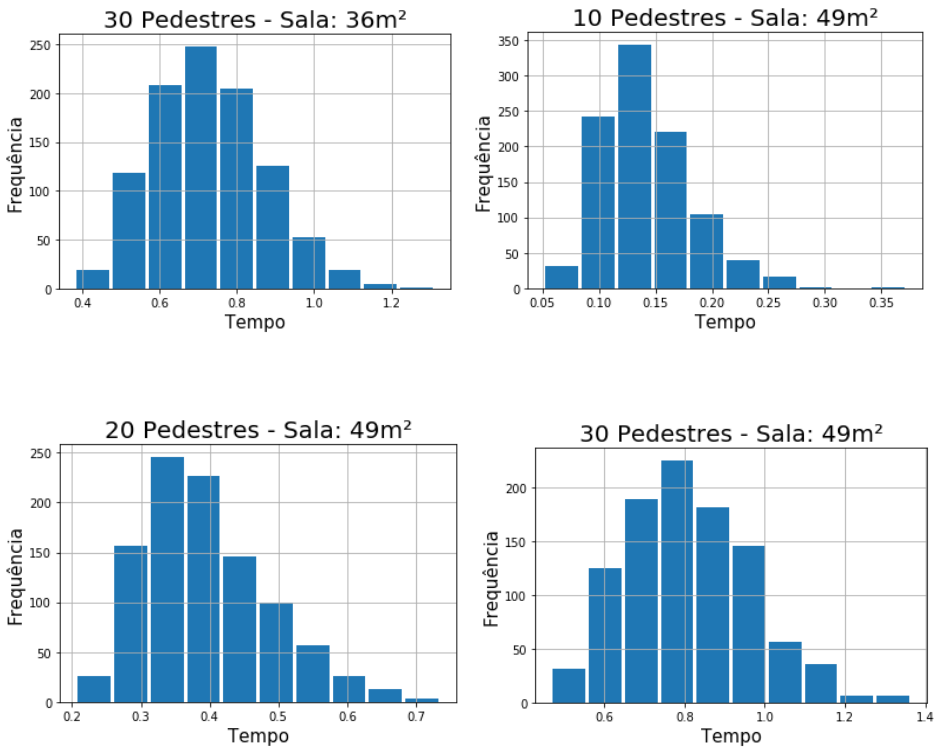


Figura 4: Histograma da distribuição dos tempos de retirada dos pedestres da sala de 36 e 49 m², considerando salas com 10, 20 e 30 pedestres.

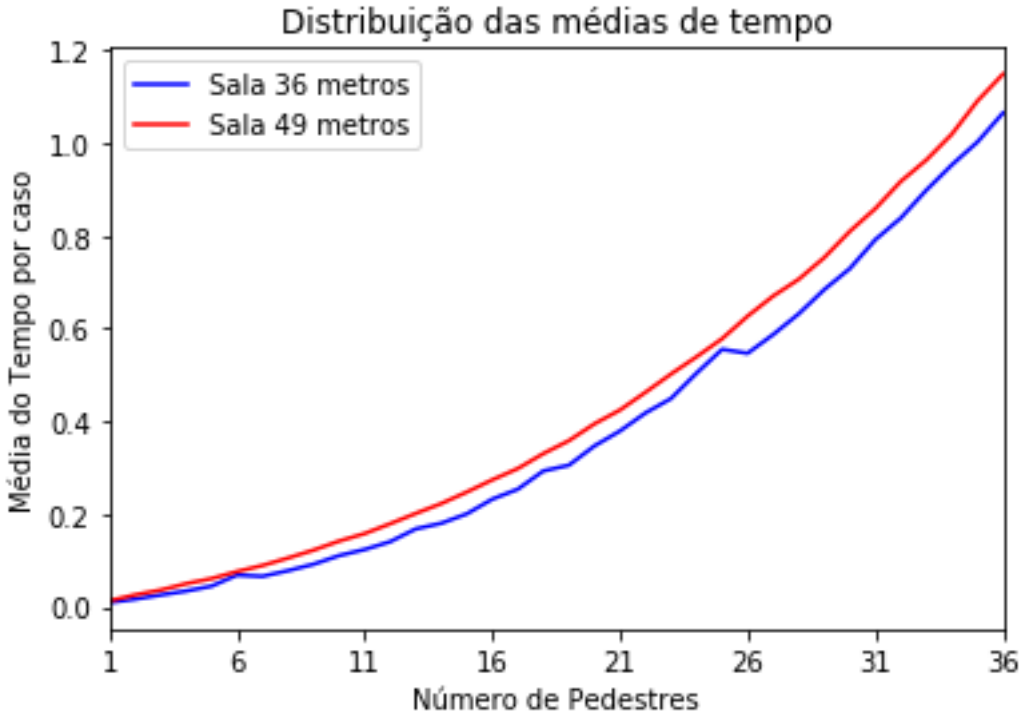


Figura 5: Distribuição das médias de tempo pelo tamanho da sala de acordo com o número de pedestres.

Na Figura 5 pode-se observar que o aumento das médias de tempo para o esvaziamento do ambiente, conforme o aumento do número de pedestres na sala, confirmando a tendência linear. Além disso, foi possível observar que quando temos até 06 (seis) pedestres na sala quase não há diferença entre o ambiente de 36 e 49 m^2. Contudo, a partir de 25 pedestre esta diferença sofre aumento, mas mantém a tendência linear. Ressalta-se que foi definida uma estrutura fenomenológica caracterizada pela racionalidade, onde tem-se pedestres sem pânico, e todos com o mesmo objetivo de atingir a porta de saída.

4. Conclusão

A implementação do pseudocódigo de Vargas R.P.G. et. al. (2015, p.6) da Simulação de Monte Carlo para situações de Evacuação do Tráfego de Pedestres em ambiente fechado, desenvolvido com a linguagem de programação Python resultou nas seguintes observações.

Apesar da simplicidade o simulador desenvolvido mostrou-se satisfatório com os resultados nas simulações, mostrando que os modelos de evacuação poderão fornecer valiosas informações sobre o movimento de pessoas quando na evacuação de ambientes fechados. Destaca-se também que a linguagem de programação Python mostrou-se ágil, simples e intuitivo o que possibilitará o desenvolvimento colaborativo do simulador para futuras evoluções possibitando trabalhos futuro.

REFERÊNCIAS

BANKS, J. et al. Discrete-event Simulation. 5. ed. New Jersey: PrenticeHall, 2009.

BATEMAN, R. E. et al. Sistemas de simulação: aprimorando processos de logística, serviços e manufatura. 1. ed. Rio de Janeiro: Elsevier, 2013.

CHWIF, L.; MEDINA, A. C. Modelagem e simulação de eventos discretos, teoria & aplicações. 2. ed. São Paulo: [s.n.], 2007.

HAMMERSLEY J. e HANDSCOMB D. Monte Carlo Methods. Chapman and Hall, 1964.

LUBAN, F. Sisteme bazate pe cunostinte in management. Bucuresti, Editura ASE. 2006.

LUBAN, F. Methods for evaluating economics of knowledge management systems. In: INTERNATIONAL CONFERENCE ON INFORMATICS IN ECONOMY, Romênia, 2005. Proceedings... Bucarest, Romênia, 2005.

LUBAN, F.; HÎNCU, D. Interdependency between simulation model development and knowledge management. Theoretical and Empirical Researches in Urban Management, v. 1, n. 10, 2009.

OLIPHANT, "Python for Scientific Computing", Computing in Science & Eng, vol. 9, no. 3, pp. 10-20, 2007.

KELTON, W. D.; SADOWSKI, R. P. E.; STURROCK, D. T. Simulation with ARENA. 4. ed. New York: McGraw-Hill, 2007.

ROBINSON, S. Conceptual modelling for simulation Part I: definition and requirements. Journal of the Operational Research Society, v. 59, p. 278-290, 2008.

SALIBY, E. Repensando a simulação: a amostragem descritiva. São Paulo: Atlas; Rio de Janeiro: Editora de UFRJ, 1989.

SANTOS N. A. S, DINIZ R. C. A utilização da técnica de prototipação no desenvolvimento de sistemas de informações contábeis. 2004. Disponível em <https://www.researchgate.net/publication/304730392_A_utilizacao_da_tecnica_de_prototipacao_no_desenvolvimento_de_sistemas_de_informacoes_contabeis/citation/download>

VARGAS, Mariana, 2010, SIMULAÇÃO MONTE CARLO PARA O TRÁFEGO DE PEDESTRES OBSERVADOS COMO UM FLUÍDO NÃO CONVENCIONAL.

.