Universidade Estadual Paulista
Instituto de Biociências, Letras e Ciências Exatas
Departamento de Ciência da Computação e Estatística

Fabrício Gustavo Frank

Simulador de Eventos Discretos em Python

São José do Rio Preto
2022

Fabrício Gustavo Frank

Simulador de Eventos Discretos em Python

Monografia apresentada ao Departamento de Ciência de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para obtenção do título de "Bacharel" em Ciência da Computação da UNESP

Orientadora: Prof^a Dr^a Renata Spolon Lobato

Banca avaliadora:
Prof. Dr. Kelton Augusto Pontara da Costa

Prof. Dr. Rodolfo Ipolito Meneguette

São José do Rio Preto
2022

F828s

Frank, Fabrício Gustavo
Simulador de Eventos Discretos em Python / Fabrício Gustavo Frank. -- São José do Rio Preto, 2022
40 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientadora: Renata Spolon Lobato

1. Simulação. 2. Python. 3. SimPy. 4. Software. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

Agradecimentos

Agradeço primeiramente a Deus, que me permitiu chegar até aqui. Agradeço também a minha família e amigos pelo apoio e amor incondicional. Também agradeço a minha orientadora que me deu suporte para a realização desse trabalho.

"A tecnologia move o mundo"
Steve Jobs

Resumo

A simulação de eventos discretos é uma ferramenta de fundamental importância para a análise de desempenho de sistemas reais que podem ser modelados como uma fila de espera. A implementação de um simulador, entretanto, depende do conhecimento prévio de uma área e de uma linguagem de programação específica. A linguagem Python permite que se trabalhe com uma gama de ferramentas matemáticas e analíticas e juntamente o framework SimPy, é possível criar um ambiente de simulação de modo simplificado e fácil de entender. Assim, neste trabalho foi realizada a implementação de um simulador que trata da chegada de clientes em um evento qualquer, podendo servir como base para a modelagem de vários sistemas da vida real.
Palavras-chave: simulação de eventos discretos, teoria das filas, python, simpy.

Abstract

Discrete event simulation is a critically important tool for analyzing the performance of real systems that can be modeled as a queue. The implementation of a simulator, however, depends on prior knowledge of an area and a specific programming language. The Python language allows working with a range of mathematical and analytical tools and together with the SimPy framework, it is possible to create a simulation environment in a simplified and easy-to-understand way. Thus, this work was carried out to implement a simulator that deals with the arrival of customers in any event, which can serve as a basis for modeling various real-life systems.

Keywords: discrete event simulation, queuing theory, python, simpy.

Lista de ilustrações

Lista de tabelas

Sumário

1 INTRODUÇÃO

Ambientes de simulação de eventos discretos são importantes ferramentas computacionais disponíveis para a análise de vários tipos de sistemas reais que podem ser modelados como filas de espera, com o objetivo de contribuir para a melhoria de tomada de decisões em níveis operacionais e gerenciais.

Todos nós estamos familiarizados com os vários sistemas de filas que enfrentamos em nosso dia a dia, seja em bancos, parques de diversão, supermercados etc. E é por isso que devemos tentar torná-los o mais eficientes possível. Há muita aleatoriedade envolvida nesses sistemas, o que pode causar grandes atrasos, resultar em longas filas, reduzir a eficiência e até mesmo ocasionar perdas financeiras. A aleatoriedade pode ser abordada desenvolvendo um modelo de simulação de eventos discretos, que pode ser extremamente útil para melhorar a eficiência operacional, analisando as principais medidas de desempenho.

De acordo com Chwif e Medina 2014, pode-se dizer que há uma grande complexidade em um sistema real, devido às suas mudanças de estado ao longo do tempo juntamente com a existência de variáveis aleatórias, tendo portanto um caráter dinâmico e aleatório. Neste contexto, fica claro que com a aplicação do modelo de simulação é possível reproduzir essas características com fidelidade em um computador, de modo a analisar o comportamento que o sistema apresentaria quando submetido a diferentes configurações.

A linguagem Python apresenta uma sintaxe descomplicada e aprendizado acessível, e continua sendo uma tendência entre os programadores. Ela conta com a biblioteca SimPy para a simulação de eventos discretos, que possui recursos poderosos e de fácil implementação para modelagem de sistemas. Assim, ela foi a linguagem escolhida neste trabalho para a implementação de um simulador de eventos discretos.

1.1 Objetivos

O objetivo deste trabalho é realizar a implementação de um simulador de eventos discretos que utilize a biblioteca de simulação da linguagem Python, SimPy, e que possibilite a análise de sistemas reais que possam ser modelados como filas de espera. A ferramenta a ser desenvolvida deve ser capaz de simular um sistema de filas para entrada em um evento qualquer, considerando a chegada massiva de clientes em um determinado intervalo de tempo.

1.2 Motivação

Nos dias atuais, é cada vez mais necessário o uso de modelos que simulem eventos do mundo real, para otimizar processos já estabelecidos ou até mesmo para auxiliar o desenvolvimento de um novo projeto. Com a simulação de eventos é possível, por exemplo, organizar o atendimento de um estabelecimento e planejar novas rotas de transporte. Portanto, dada a importância da simulação de eventos, aliada às facilidades da linguagem Python, surgiu a oportunidade para implementação do simulador.

1.3 Justificativa

A tomada de decisões sem um planejamento prévio é bastante arriscada e pode trazer resultados negativos. A simulação de eventos discretos possibilita a criação de estratégias para otimização de sistemas do mundo real de forma segura, por meio de testes. Portanto, essas simulações são indispensáveis para quem precisa avaliar as consequências de mudanças em um sistema e decidir quais métodos implementar e quais descartar.

1.4 Metodologia

A metodologia para realização deste trabalho consiste nas seguintes etapas: pesquisa sobre o tema, incluindo simulação de sistemas, simulação de eventos discretos, filas, Python e Simpy; implementação e especificação do funcionamento do simulador desenvolvido; e a realização de testes e análise dos resultados obtidos. O material utilizado na pesquisa foi retirado principalmente de livros, artigos científicos, documentação online e sites, todos meios gratuitos e acessíveis.

1.5 Exequibilidade

Para a realização deste trabalho será utilizada a linguagem de programação Python e os recursos de *hardware* do próprio autor. Além disso, todo o material utilizado pode ser obtido gratuitamente, assim o trabalho pode ser realizado dentro do prazo estipulado sem preocupação com recursos adicionais.

1.6 Organização do trabalho

Além desse capítulo introdutório, o trabalho está dividido nos seguintes capítulos:

- O capítulo 2 apresenta a fundamentação teórica do trabalho, envolvendo o estudo de conceitos e trabalhos relacionados a simulação, modelagem de sistemas, filas, Python

e SimPy.

- O capítulo 3 descreve todo o desenvolvimento do trabalho proposto, com a aplicação dos conceitos apresentados no capítulo anterior.
- O capítulo 4 relata os resultados obtidos, com base em testes realizados com o simulador para a validação do trabalho proposto.
- O capítulo 5 expõe as conclusões sobre trabalho, além de propostas para pesquisas futuras.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão apresentados e definidos os conceitos de filas, simulação de sistemas, simulação de eventos discretos, Python e SimPy, necessários para o entendimento e compreensão deste trabalho.

2.1 Redes de fila

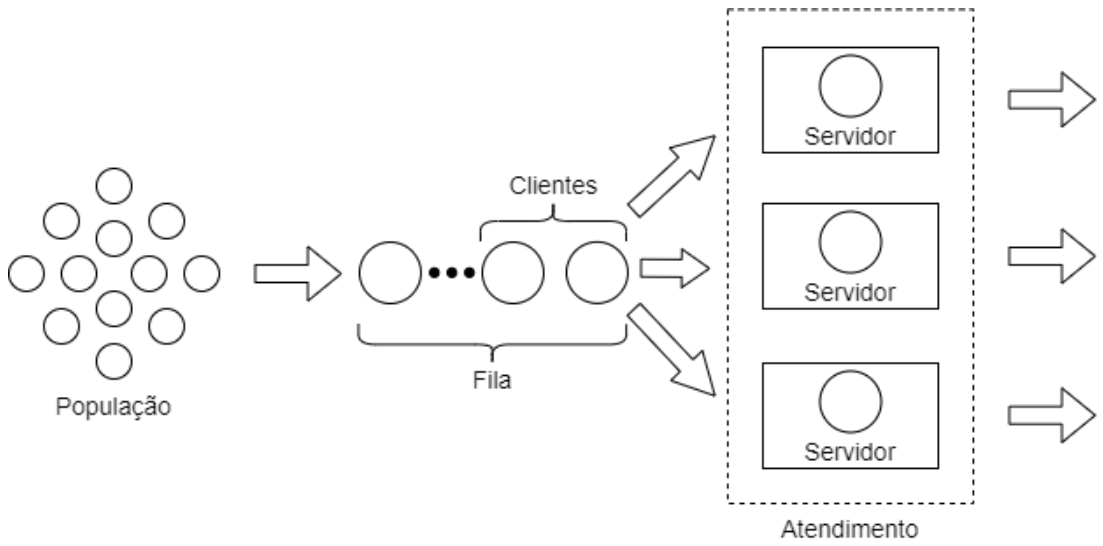
A fila trata de uma das experiências mais desagradáveis da vida, a espera. O enfileiramento é bastante comum em muitos campos, como por exemplo, em centrais telefônicas, supermercados, em um posto de gasolina, etc [Sztrik 2016]. A abordagem matemática das filas se iniciou no começo do século XX com A. K. Erlang, considerado o pai da teoria das filas. Apesar do enorme progresso alcançado pela teoria, inúmeros problemas não são adequadamente resolvidos por causa de complexidades matemáticas [Prado 2017].

Normalmente, os modelos de filas são especificados em termos do processo de chegada, processo de atendimento e a disciplina da fila.

2.1.1 Elementos de uma fila

Na figura 1 é possível observar os elementos que compõem uma fila. Nela temos que, de uma certa população, surgem clientes que formam uma fila, aguardam por algum tipo de serviço e ao serem atendidos deixam o sistema.

Figura 1 – Elementos de uma fila.



Fonte: Extraído e adaptado de [Prado 2017]

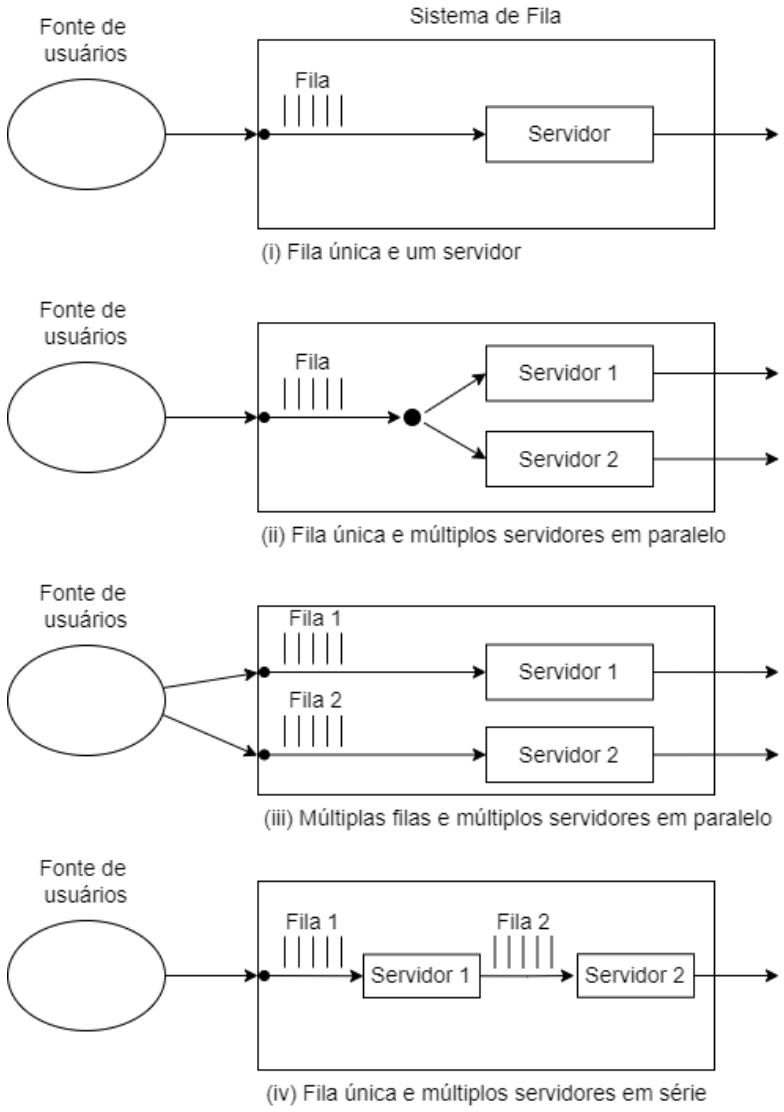
2.1.2 Tipos de filas

Segundo Arenales e Armentano 2006, existem diferentes sistemas de filas:

- (i) Fila única e um servidor;
- (ii) Fila única e múltiplos servidores em paralelo;
- (iii) Múltiplas filas e múltiplos servidores em paralelo;
- (iv) Fila única e múltiplos servidores em série.

Na figura 2 é possível os vários tipos de sistemas de filas.

Figura 2 – Tipos de sistemas de filas.



Fonte: Extraído e adaptado de [Arenales e Armentano 2006]

Como pode ser observado na figura anterior, os modelos de atendimento podem apresentar diversas configurações: fila única, múltiplas filas, servidor único, servidores múltiplos. Uma fila única é caracterizada por ter apenas um canal de atendimento, podendo ter um ou mais servidores. Já uma fila múltipla apresenta mais de um canal de atendimento em paralelo, atuando de forma independente, sendo necessário mais de um servidor.

2.1.3 Processo de chegada

O processo de chegada de usuários no sistema é descrito pelo intervalo de tempo entre chegadas sucessivas de usuários. Em geral, admitimos que não mais de um usuário pode chegar no mesmo instante; caso contrário, dizemos que pode ocorrer uma chegada em lote [Arenales e Armentano 2006].

Segundo Moreira 2017, se forem conhecidos o número de chegadas e os instantes de tempo em que elas acontecem, esse processo é denominado determinístico; caso contrário, tem-se um comportamento aleatório constituindo um processo estocástico caracterizado por uma distribuição de probabilidade. Nesse último caso, é necessária a especificação de um parâmetro denominado taxa de chegadas, que representa o número médio de usuários que chegam ao sistema por unidade de tempo. Usualmente as taxas de chegada são representadas por λ.

Tradicionalmente, há duas formas tradicionais de se falar sobre a chegada de clientes para o atendimento: número de clientes que chegam em um dado intervalo de tempo e o tempo decorrido entre duas chegadas consecutivas. É muito comum nos estudos de teoria das filas se utilizar da distribuição de Poisson para configurar a taxa de chegada à fila e atendimento [Moreira 2017].

A tabela 1 apresenta as distribuições de probabilidade utilizadas em cada uma das formas de chegada de clientes:

Tabela 1 – Processo de chegada - distribuições.

Grandeza	Tipo de distribuição	Média
Número de chegadas na unidade de tempo	Poisson	λ
Tempo decorrido entre duas chegadas consecutivas	Exponencial	$\frac{1}{\lambda}$

Fonte: [Moreira 2017]

2.1.4 Processo de atendimento

O processo de atendimento é descrito pelo tempo de atendimento por usuário. Cada servidor não precisa ser um indivíduo, mas pode ser um grupo de pessoas ou máquinas realizando simultaneamente um serviço. Em geral, admitimos que não mais de um usuário

pode ser atendido por um servidor no mesmo instante; caso contrário, dizemos que pode ocorrer atendimento em lote [Arenales e Armentano 2006].

Usualmente nos modelos de atendimento usa-se o parâmetro μ e, assim como nos modelos de chegadas, existem duas nomenclaturas comumente utilizadas associadas a este parâmetro: número de atendimentos na unidade de tempo e tempo decorrido entre dois atendimentos consecutivos [Moreira 2017].

Cada uma dessas grandezas apresenta uma distribuição de probabilidade, como é ilustrado na tabela 2.

Tabela 2 – Processo de atendimento - distribuições.

Grandeza	Tipo de distribuição	Média
Número de atendimentos na unidade de tempo	Poisson	μ
Tempo decorrido entre dois atendimentos consecutivos	Exponencial	$\frac{1}{\mu}$

Fonte: [Moreira 2017]

2.1.5 Disciplina da fila

Como bem nos assegura Prado 2017, a disciplina da fila define qual o próximo usuário a ser atendido, o mais comum nos sistemas é que o primeiro da fila é atendido ou, de uma maneira mais ampla, o "primeiro a chegar é o primeiro a ser atendido" (em inglês, FIFO: First In First Out ou FCFS: First Come First Served). Outras disciplinas podem existir, tais como "último a chegar primeiro a ser atendido" (em inglês, LIFO: Last In First Out ou ou LCFS: Last Come First Served), serviço por ordem de prioridade, serviço randômico etc.

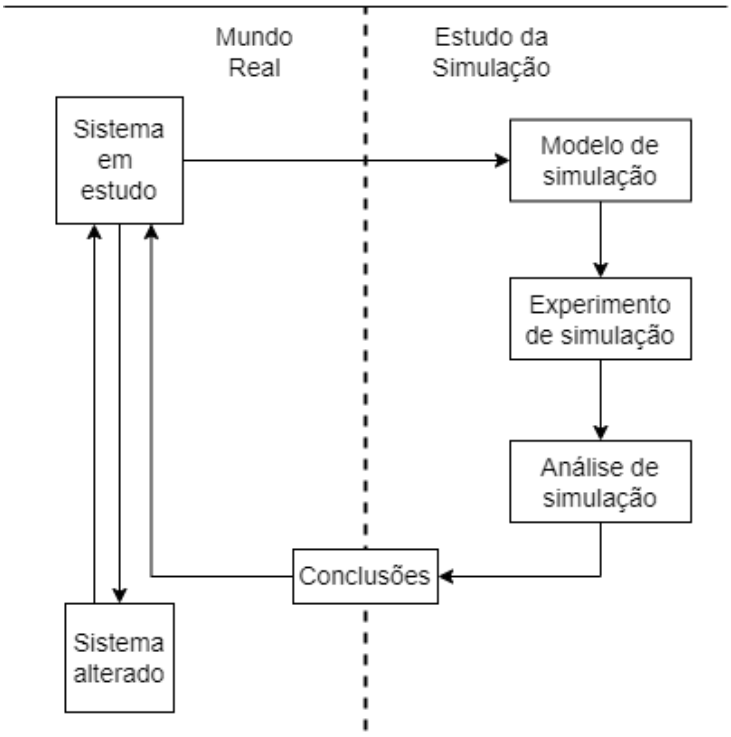
2.2 Simulação de sistemas

A técnica de simulação consiste em imitar o funcionamento de um sistema real [Prado 2017]. Segundo Chwif e Medina 2014, o termo simulação pode ser classificado em duas categorias: a simulação computacional e a simulação não computacional. A primeira necessita de um computador para ser realizada, já a segunda não. Para este trabalho, a simulação computacional é o foco do estudo.

Nesse contexto, a simulação pode ser entendida como um processo de projetar um modelo computacional de um sistema real e conduzir experimentos com este modelo com o objetivo de entender seu comportamento e/ou avaliar estratégias para sua operação [Pegden et al. 1990].

A figura 3 ilustra o processo de simulação. A natureza iterativa do processo é indicada pelo sistema em estudo que se torna o sistema alterado e o ciclo se repete.

Figura 3 – Processo de simulação de sistemas reais.



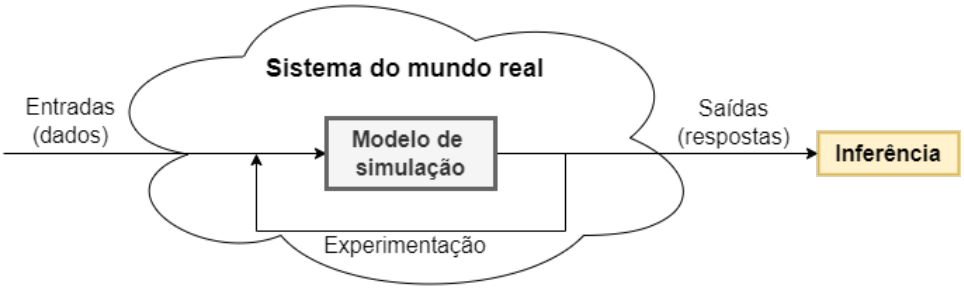
Fonte: Extraído e adaptado de [Bakenaz A. Zeidan 2015]

2.2.1 Modelos de simulação

Um modelo de simulação é especificado por meio dos elementos do sistema e suas características e a interação entre esses elementos, ocasionando alterações no estado do sistema durante a simulação.

A figura 4 mostra, de forma esquemática, a ideia do modelo e do processo experimental.

Figura 4 – Representação esquemática de um modelo de sistema.



Fonte: Extraído e adaptado de [Filho 2008]

Como pode ser observado na figura 4, o modelo de simulação pode ser utilizado como uma ferramenta para se obter respostas a sentenças do tipo: "O que ocorre se...", possibilitando que sejam obtidos *insights* que auxiliem nas tomadas de decisões antes que mudanças no sistema sejam implementadas [Chwif e Medina 2014].

Os modelos de simulação podem ser divididos em duas categorias: modelos contínuos e modelos discretos.

- Modelos contínuos: são utilizados para modelar sistemas cujo estado varia continuamente no tempo. Geralmente, modelos desse tipo são compostos por uma série de equações diferenciais, que são resolvidas numericamente para ver como o modelo se comporta. [Bratley, Fox e Schrage 2012]

- Modelos discretos: são utilizados para modelar sistemas que mudam o seu estado em momentos discretos no tempo, a partir da ocorrência de eventos. Em modelos discretos pode ser observado a disputa por recursos limitados e filas onde os elementos do sistema esperam pela liberação do recurso solicitado.

Neste trabalho, os modelos aos quais estaremos tratando são voltados a simulação discreta de sistemas.

2.2.2 Aplicações da simulação

Desde sua concepção, a simulação foi aplicada em vários setores como manufatura, serviços, defesa, saúde e serviços públicos [Jahangirian et al. 2010]. Há uma necessidade crescente de abordar as complexidades de processos operacionais e gerenciais e solucionar problemas de tomada de decisão dentro de um sistema.

Alguns exemplos de aplicações da simulação descritos por Chwif e Medina 2014 são:

- Bancos: a simulação pode ser utilizada para verificar qual é a melhor política de abertura e fechamento de caixas, número de caixas automáticos necessários, estudar problemas de layout, determinar o tempo máximo de espera em fila etc;

- Hospitais: neste caso a simulação pode ser utilizada para estudar o comportamento de UTIs, dimensionamento de ambulâncias, simulações para testar políticas de transplantes de órgãos etc;

- Aeroportos: em um aeroporto é possível fazer uma simulação para dimensionar o número de check-ins necessários ou, mesmo, para dimensionar um sistema completo de transporte de bagagens;

- Cadeias logísticas: nesse segmento a simulação pode ser utilizada para determinar qual deve ser a melhor politica de estocagem, transporte e distribuição, desde a origem das matérias-primas, passando pela fabricação até o consumidor final;

2.3 Simulação de sistemas de eventos discretos

A simulação de eventos discretos é um tipo de simulação que considera um sistema como uma coleção discreta de eventos, com cada evento tendo um efeito definido no resto do sistema. Os processos individuais que compõem um sistema podem ser definidos em termos de seu impacto no sistema, seus requisitos de recursos e seu início (ou seja, eles podem ser programados, podem ocorrer aleatoriamente ou ocorrer em resposta a outro evento do sistema).

2.3.1 Componentes de um sistema

Segundo Banks 2005, a fim de entender e analisar um sistema para a construção de um modelo de simulação por eventos discretos, alguns termos precisam ser definidos, entre eles:

- Sistema: coleção de entidades que interagem em conjunto para realização de uma ou mais ações.
  Ex: clientes e vendedores.
- Entidade: objeto de interesse no sistema.
  Ex: cliente.
- Atributo: propriedade de uma entidade.
  Ex: saldo na conta de um cliente.
- Recurso: objeto que é utilizado e concorrido pelas entidades.
  Ex: caixa da loja.
- Evento: ocorrência instantânea que muda o estado do sistema.
  Ex: chegada de clientes.

2.3.2 Programação de eventos

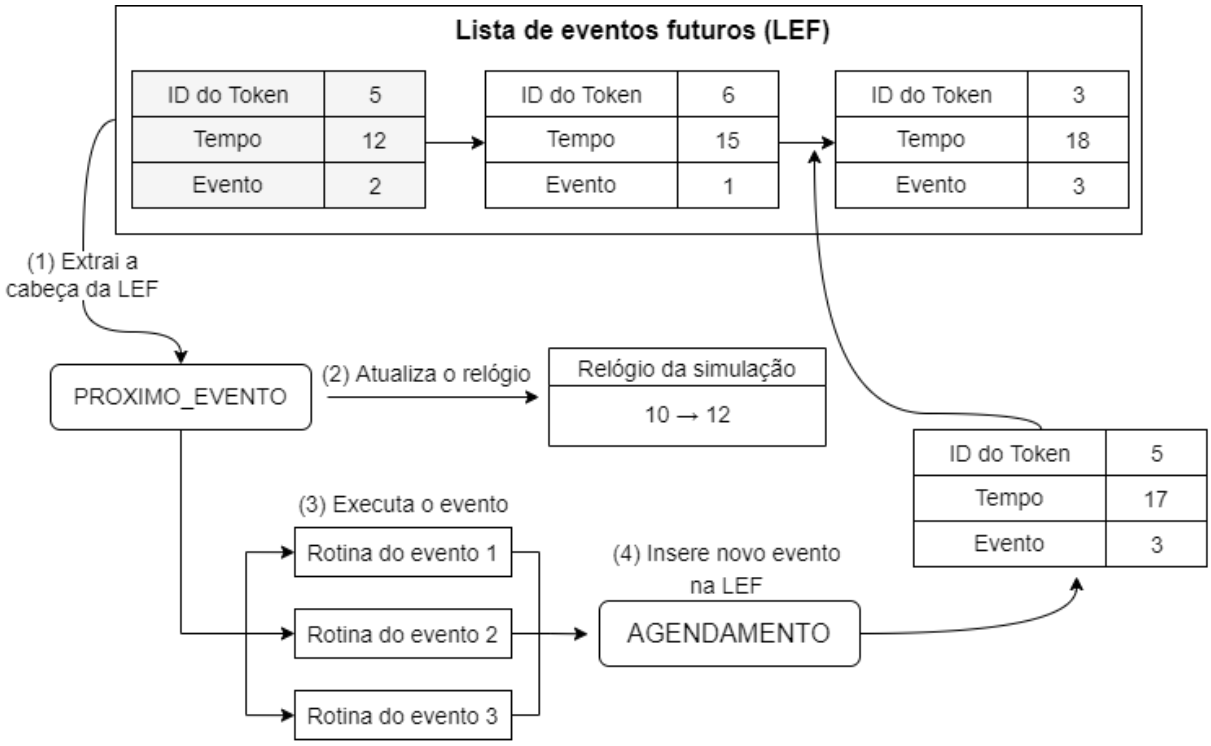
O mecanismo para avançar o tempo de simulação e garantir que todos os eventos ocorram em sua ordem cronológica é baseado no relógio de simulação e na lista de eventos futuros (LEF, ou FEL em inglês). A LEF contém todas as ocorrências de eventos para eventos que foram programados para ocorrer em um tempo futuro [Banks 2005].

Tipicamente, os eventos são agendados para execução de modo dinâmico, ao longo da própria da simulação.

Os eventos da LEF são geralmente classificados em ordem de data/hora crescente. Quando a simulação inicia, a cabeça da LEF é extraída, atualizando o relógio de simulação. O evento extraído é então enviado para uma rotina de evento, onde reproduz um novo evento após sua execução. O novo evento é inserido na LEF, e em seguida é realizada novamente a ordenação crescente da LEF em termos de data/hora. Este processo é iterado até que a simulação termine.

A figura 5 ilustra o ciclo básico para programação de eventos. Três eventos futuros são armazenados na LEF. Quando PROXIMO_EVENTO é chamado, o token #5 com carimbo de data/hora 12 é extraído da cabeça da LEF. O relógio de simulação então avança de 10 para 12. O evento é então executado na rotina de evento 2, que cria um novo evento futuro, neste caso o evento #3. Assim, o token #5 com o evento #3 é agendado e inserido na LEF. Na inserção, o token #5 é colocado na LEF entre o token #6 e o token #3 depois de comparar sua marcação de data/hora. O loop de eventos itera para chamar PROXIMO_EVENTO até que a simulação termine. [Park e Fishwick 2010]

Figura 5 – Ciclo básico utilizado para a programação de eventos.



Fonte: Extraído e adaptado de [Park e Fishwick 2010].

2.4 Python e SimPy

O desenvolvimento do simulador tem como base a linguagem Python. Ela se destaca entre as linguagens dinâmicas como uma das mais populares e poderosas. Existe uma comunidade movimentada de usuários da linguagem no mundo, o que se reflete em listas ativas de discussão e muitas ferramentas disponíveis em código aberto. [Borges 2014]

Já o SimPy é um framework baseado em processos com simulação de eventos discretos utilizando as bibliotecas padrão do Python. Os processos em SimPy são definidos através de funções *generator*. O ambiente também disponibiliza diversos tipos de recursos compartilhados para utilização de recursos propensos a congestionamento (como servidores, caixas e túneis). As simulações podem ser realizadas do modo "mais rápido possível", em tempo real (de acordo com o desenrolar do tempo) ou manualmente passo a passo por cada evento.

2.5 Trabalhos relacionados

Esta seção destina-se a apresentação de alguns trabalhos relacionados com o projeto em questão, a fim de expor o estado da arte. Dentre estes, estão presentes trabalhos de conclusão de curso e artigos em geral.

No artigo de Huling e Miles 2015, os autores apresentam um projeto de prova de conceito para uma simulação de eventos discretos de reconstrução residencial com o objetivo de avaliar seu potencial para simular a recuperação de desastres em geral. A ferramenta foi implementada em Python como um protótipo usando o *framework* SimPy. Os resultados preliminares da simulação do protótipo sugerem que o simulador é apropriado e promissor para modelar a reconstrução de casas.

No trabalho de Cordeiro 2021, o autor apresenta um estudo sobre o método de simulação computacional de dinâmica molecular, que fornece a dinâmica do sistema de partículas a partir da integração numérica das equações de movimento. Os resultados são obtidos em função da temperatura e da densidade do sistema.

No artigo de Pinho, Coelho e Boaventura-Cunha 2016, os autores propõem a modelagem e simulação de uma cadeia de abastecimento de base florestal, em particular a cadeia de abastecimento de biomassa, através do *framework* SimPy. O modelo desenvolvido foi utilizado para avaliar o impacto das mudanças no plano de trabalho diário em três situações. Os resultados obtidos validam o ambiente de simulação SimPy como um *framework* para modelagem de cadeias de suprimentos em geral e para o problema da biomassa em particular.

Assim, este trabalho segue relativamente o mesmo objetivo dos trabalhos citados anteriormente, com o enfoque em desenvolver um simulador utilizando SimPy, que permita

analisar um sistema da vida real a partir de uma sequência de eventos discretos no tempo, podendo servir como base para a modelagem de outros sistemas. O projeto considera a estimativa de propriedades importantes do sistema a ser modelado e a verificação de características relevantes sobre seu comportamento, para que tenha seu desempenho otimizado.

3 DESENVOLVIMENTO DO PROJETO

Neste capítulo serão descritas as etapas do desenvolvimento do simulador, considerando os conceitos apresentados no capítulo 2. Assim, será apresentado de que forma foi feita a construção e configuração do modelo de simulação, considerando o cenário escolhido para a modelagem do sistema, com seus principais componentes.

A ferramenta deve ser capaz de auxiliar o usuário na otimização da eficiência de um processo operacional já estabelecido ou até mesmo para auxiliar o desenvolvimento de um novo projeto por meio da análise dos resultados obtidos a partir do sistema modelado.

3.1 Cenário da simulação

O objetivo do trabalho é desenvolver um programa capaz de simular um sistema de filas para entrada em um evento qualquer: regularmente chegarão vários clientes em um ônibus que precisarão ter seus ingressos escaneados antes de entrar no evento. Algumas pessoas já terão comprado o ingresso antecipadamente e só devem escaneá-lo para entrar, já outras necessitarão comprá-lo em guichês com vendedores, o que pode gerar uma fila e um tempo de espera. No entanto, outros exemplos que seguem um padrão semelhante, como uma fila em um supermercado ou um restaurante que recebe pedidos online, um cinema ou uma estação de trem podem ser facilmente modelados com a modificação dos parâmetros.

3.2 Componentes do modelo de simulação

Nesta seção serão apresentados os componentes do sistema em questão, considerando as entidades com seus atributos e os eventos que podem ocorrer. Durante a simulação ocorre a interação entre esses componentes, ocasionando alterações no estado do sistema, até que o tempo de simulação se esgote.

Entidade, atributo e recursos:

- Cliente: pessoa interessada no evento;
- Vendedor/Guichê de venda: responsável pela venda do ingresso (recurso);
- Scanner: objeto responsável por escanear o ingresso (recurso);
- Ingresso: atributo necessário para entrada no evento.

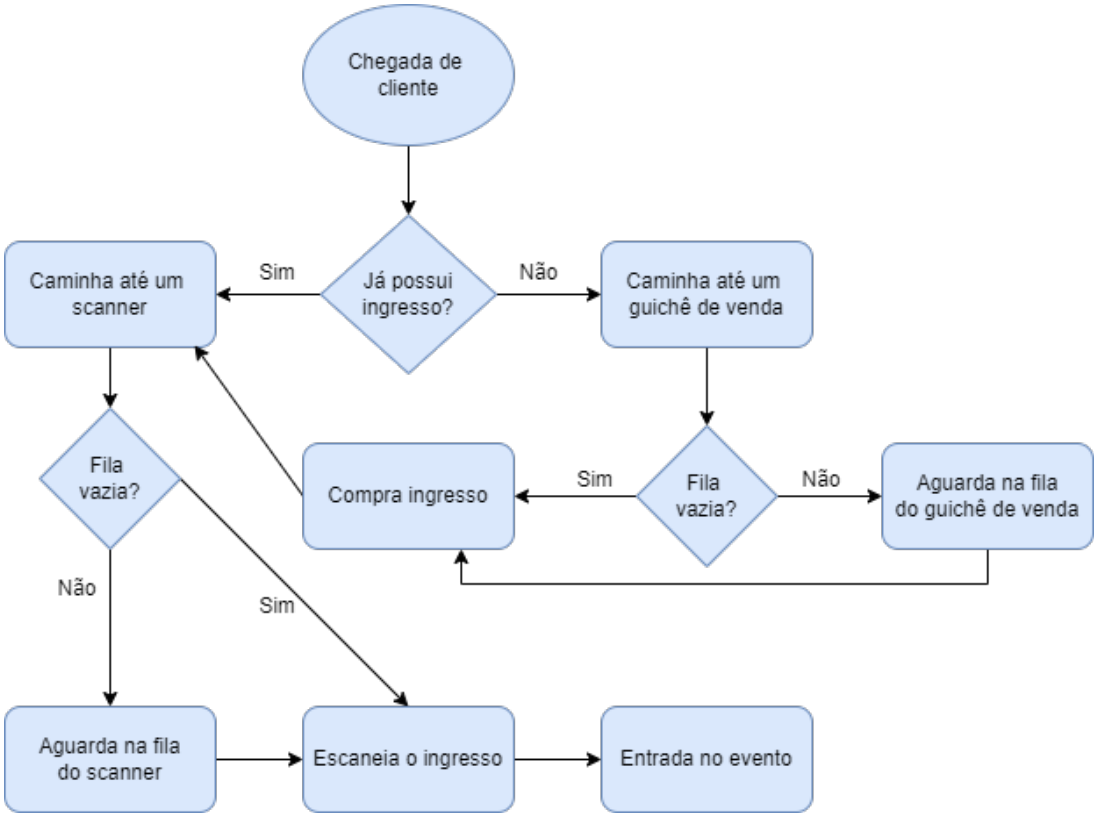
Eventos:

- Chegada de clientes;
- Cliente caminha até o scanner;
- Cliente caminha até o guichê de venda;
- Cliente espera na fila do scanner;
- Cliente espera na fila do guichê de venda;
- Cliente compra o ingresso;
- Cliente entra no evento (escaneia o ingresso).

3.3 Visualização do sistema

O fluxograma da figura 6 apresenta uma visão geral do sistema a ser modelado:

Figura 6 – Representação esquemática do sistema a ser modelado.



Fonte: Elaborado pelo autor.

É possível observar por meio do fluxograma da figura acima que, após a chegada do cliente, sua passagem pelo guichê de venda ou ida direta ao scanner e consequente entrada

no evento é condicionada pela posse do ingresso, visto que se o cliente não apresentar esse atributo, deverá obrigatoriamente comprá-lo.

Ao longo da simulação, com a chegada massiva de clientes, ambos os recursos, guichês de venda e scanners, formarão filas e assim poderemos observar o desempenho do sistema, por meio dos tempos de espera.

3.4 Construção do simulador

As etapas da construção do simulador envolvem a inicialização dos parâmetros da simulação, criação do ambiente de simulação e implementação de rotinas para visualização dos resultados a partir da plotagem de gráficos.

3.4.1 Inicialização dos parâmetros da simulação

A seguir estão listados os parâmetros que devem ser inicializados para que seja feita a simulação. Alguns parâmetros são inicializados com valor fixo, já outros, que são dados por distribuições de probabilidade, devem ser inicializados com um valor médio e um desvio padrão.

- Tempo entre chegadas de ônibus;
- Número de pessoas que chegam em cada ônibus;
- Número de pessoas que compraram o ingresso;
- Tempo para caminhar até o guichê de venda;
- Tempo para caminhar até o scanner;
- Número de guichês de venda;
- Número de scanners.

Além desses valores, o tempo de simulação, ou seja, por quanto tempo a simulação irá ocorrer deve ser fornecido. Os parâmetros podem ser inicializados via código ou obtidos por meio de *input* do usuário. Para uma maior rapidez e praticidade no desenvolvimento, nesse trabalho todos as variáveis foram inicializadas via código.

3.4.2 Criação do ambiente de simulação

Depois da inicialização dos parâmetros deve-se iniciar o SimPy com a criação de um ambiente de simulação. Para isso é necessário criar as filas (recursos), definir o tempo

de simulação, e iniciar a simulação com a chamada de uma função que controla a chegada dos ônibus.

A seguir é mostrado o trecho de código responsável por essa etapa:

```
env = simpy.Environment() #Def. do ambiente

#Config. dos guiches de venda
seller_lines = [simpy.Resource(env, capacity=1)
                for _ in range(SELLER_LINES)]
#Config. dos scanners
scanner_lines = [simpy.Resource(env, capacity=1)
                 for _ in range(SCANNER_LINES)]

#Inicio da simulacao ate o tempo definido por TIME_SIMULATION
env.process(bus_arrival(env, seller_lines, scanner_lines))
env.process(create_clock(env))
env.run(until=TIME_SIMULATION)
```

Com o ambiente configurado, o comando env.process() iniciará o processo conforme descrito na função bus_arrival() descrita abaixo. Esta função é o evento de nível superior a partir do qual todos os outros eventos são disparados. A função simula a chegada de um ônibus com pessoas a bordo, de acordo com os parâmetros fornecidos na seção 3.4.1 e, em seguida, aciona os processos de venda e escaneamento de ingressos.

```
def bus_arrival(env, seller_lines, scanner_lines):

while True:
    next_bus = random.expovariate(1 / BUS_ARRIVAL_MEAN)
    on_board = int(random.gauss(BUS_OCCUPANCY_MEAN,
    BUS_OCCUPANCY_STD))
    next_bus = ARRIVALS.pop()
    on_board = ON_BOARD.pop()

    # Espera pelo onibus
    bus_log.next_bus(next_bus)
    yield env.timeout(next_bus)
    bus_log.bus_arrived(on_board)

    while len(people_ids) > 0:
        remaining = len(people_ids)
        group_size = min(round(random.gauss(
```

```
            PURCHASE_GROUP_SIZE_MEAN, PURCHASE_GROUP_SIZE_STD)),
            remaining)

        people_processed = people_ids[-group_size:]
        people_ids = people_ids[:-group_size]



        # Determina aleatoriamente se o grupo de pessoas esta
        indo para os guiches de venda ou diretamente para os
        scanners
        if random.random() > PURCHASE_RATIO_MEAN:
            env.process(scanning_customer(env, people_processed,
            scanner_lines, TIME_TO_WALK_TO_SELLERS_MEAN +
                        TIME_TO_WALK_TO_SCANNERS_MEAN,
                        TIME_TO_WALK_TO_SELLERS_STD +
                        TIME_TO_WALK_TO_SCANNERS_STD))
        else:
            env.process(purchasing_customer(
                env, people_processed, seller_lines,
                scanner_lines))
```

Vale ressaltar que como essa é a função de evento de nível superior, vemos que todo seu trabalho está ocorrendo em um loop while infinito. No final do loop, é disparado um de dois eventos de forma aleatória: o grupo vai diretamente aos scanners (chamada da função scanning_customer()) ou é necessário comprar os ingressos primeiro (chamada da função purchasing_customer()). Em ambos os casos, a menor fila será escolhida no momento da seleção.

A função purchasing_customer() simula três eventos principais: ir para a fila, esperar na fila e, em seguida, passar o controle para o evento scanning_customer() (a mesma função que é chamada por bus_arrival() para aqueles que vão direto para os scanners).

```
def purchasing_customer(env, people_processed, seller_lines,
scanner_lines):
walk_begin = env.now
yield env.timeout(random.gauss(TIME_TO_WALK_TO_SELLERS_MEAN,
TIME_TO_WALK_TO_SELLERS_STD))
walk_end = env.now
```

```
queue_begin = env.now
# Assumimos que cada cliente vai escolher a menor fila
seller_line = pick_shortest(seller_lines)
with seller_line[0].request() as req:
    # Espera na fila
    sellers.add_to_line(seller_line[1])
    yield req
    sellers.remove_from_line(seller_line[1])
    queue_end = env.now


    # Compra ingresso
    sale_begin = env.now
    yield env.timeout(random.gauss(SELLER_MEAN, SELLER_STD))
    sale_end = env.now

    register_group_moving_from_bus_to_seller(
        people_processed, walk_begin, walk_end, seller_line[1],
        queue_begin, queue_end, sale_begin, sale_end)

    env.process(scanning_customer(env, people_processed,
    scanner_lines,
                TIME_TO_WALK_TO_SCANNERS_MEAN,
                TIME_TO_WALK_TO_SCANNERS_STD))
```

Por fim, a função scaning_customer(), que é muito semelhante a purchasing_customer(), exceto pelo fato de que embora os visitantes possam chegar e caminhar juntos em grupos, cada pessoa deve ter seu ingresso verificado de forma individual.

```
def scanning_customer(env, people_processed, scanner_lines,
walk_duration, walk_std):
# Caminha ate o guiche de venda
walk_begin = env.now
yield env.timeout(random.gauss(walk_duration, walk_std))
walk_end = env.now

# Assumimos que cada cliente vai escolher a menor fila
queue_begin = env.now
scanner_line = pick_shortest(scanner_lines)
with scanner_line[0].request() as req:
```

```
# Espera na fila
for _ in people_processed:
    scanners.add_to_line(scanner_line[1])
yield req
for _ in people_processed:
    scanners.remove_from_line(scanner_line[1])
queue_end = env.now

# Escaneia o ingresso de cada pessoa
for person in people_processed:
    scan_begin = env.now

    yield env.timeout(random.gauss(SCANNER_MEAN,
    SCANNER_STD))
    scan_end = env.now
    register_visitor_moving_to_scanner(
        person, walk_begin, walk_end, scanner_line[1],
        queue_begin, queue_end, scan_begin, scan_end)
```

3.4.3 Visualização dos dados

Para visualizar os resultados da simulação foram adicionadas algumas listas e dicionários globais para rastrear as principais métricas. Por exemplo, o dicionário de chegadas rastreia o número de chegadas por minuto e os dicionários seller_waits e scan_waits contidos nas funções registers mapeiam o minuto da simulação para uma lista de tempos de espera para aqueles que saem das filas nesses minutos. Com posse dessas informações é possível plotar os gráficos de chegadas de clientes e tempo médio de espera nas filas dos guichês de venda e scanners, utilizando a biblioteca matplotlib.

4 TESTES E RESULTADOS

Neste capítulo serão apresentados os testes realizados com o sistema em questão, de modo a validar o trabalho proposto. Assim, serão apresentados quatro possíveis configurações para execução do simulador e observaremos como cada caso de teste influencia nos resultados.

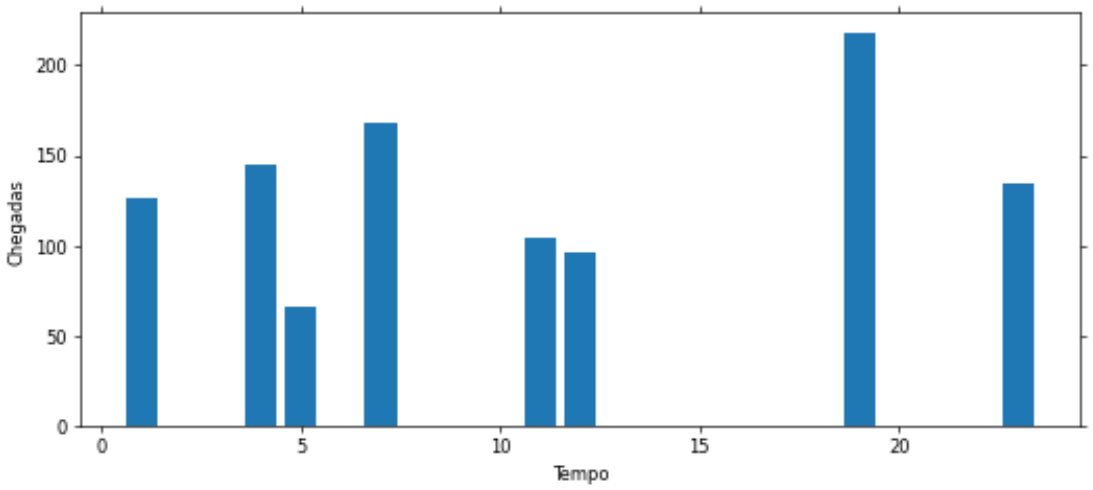
A tabela 3 a seguir mostra a configuração dos casos de teste, com a aplicação de distribuições de probabilidade. No caso dos parâmetros que utilizam a distribuição normal, são apresentados seus valores médios (μ) e o respectivo desvio padrão (σ), da forma $\mu \pm \sigma$. Além disso, vale ressaltar que todas as simulações foram realizadas em um intervalo de 30 unidades de tempo.

Tabela 3 – Valores dos parâmetros em cada caso de teste.

	1º CT	2º CT	3º CT	4º CT
Taxa de chegada de ônibus (distrib. exp)	1/3	1/3	1/3	1/3
Ocupação do ônibus (distrib. normal)	100±30	100±30	100±30	100±30
Tempo para caminhar até o guichê de venda (distrib. normal)	1±0.25	1±0.25	1±0.25	1±0.25
Tempo para caminhar até o scanner (distrib. normal)	0.5±0.1	0.5±0.1	0.5±0.1	0.5±0.1
Clientes sem ingresso (%)	40	40	40	60
Número de guichês de venda	2	4	6	5
Número de scanners	2	4	6	5

A figura 7 apresenta a distribuição das chegadas dos clientes ao longo do tempo:

Figura 7 – Chegadas de clientes ao longo do tempo em todos os casos de teste.



Por se tratar de uma simulação determinística, o número de chegadas ao longo do tempo é sempre o mesmo para todos os casos de teste.

As configurações dos quatro casos testes apresentam diferentes quantidades de guichês de venda e scanners. Um caso peculiar é a configuração do 4º caso de teste que possui uma proporção maior de clientes sem ingresso, ou seja, que devem passar pela fila dos guichês de venda.

Ao executar as simulações foram obtidos os resultados apresentados nas subseções seguintes.

4.1 1º Caso de teste: 2 guichês de venda e 2 scanners

Figura 8 – 1º Caso de teste: Média de espera na fila dos guichês de venda ao longo do tempo.

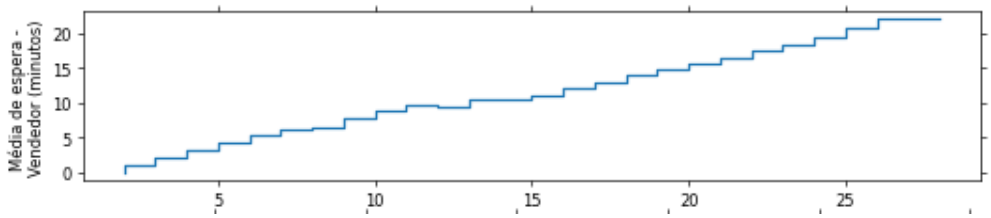
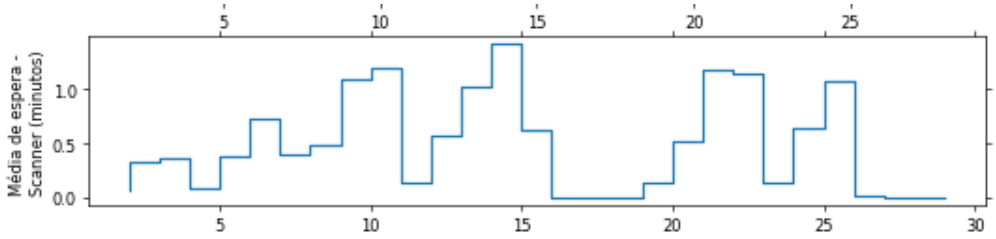


Figura 9 – 1º Caso de teste: Média de espera na fila dos scanners ao longo do tempo.



Ao observar a figura 8 é possível notar que, no primeiro caso de teste, o tempo de espera na fila dos guichês de venda segue de certo modo um padrão de crescimento linear ao decorrer da simulação e chega a atingir um pico de 20 unidades de tempo ao seu fim. Como resultado médio foi obtido uma espera de 10.9 unidades de tempo nos guichês.

A figura 9 mostra que o tempo de espera nos scanners supera a marca de 1 unidade de tempo em alguns momentos da simulação, tendo como resultado médio o tempo de 0.6 unidades de tempo.

4.2 2º Caso de teste: 4 guichês de venda e 4 scanners

Figura 10 – 2º Caso de teste: Média de espera na fila dos guichês de venda ao longo do tempo.

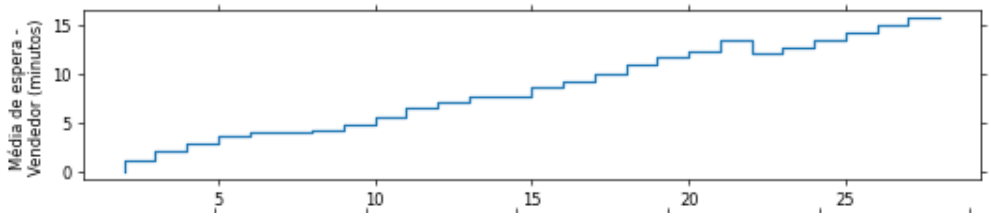
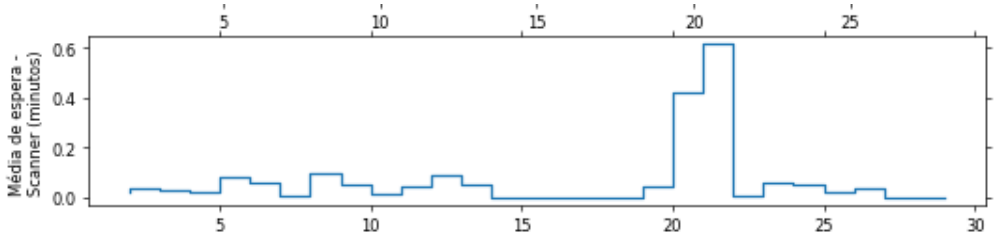


Figura 11 – 2º Caso de teste: Média de espera na fila dos scanners ao longo do tempo.



Na figura 10 podemos ver que assim como no primeiro caso de teste, o crescimento do tempo de espera nos guichês de venda no segundo caso de teste também é de certa forma linear, atingindo o seu ápice de pouco mais de 15 unidades de tempo também ao final da simulação. O tempo médio calculado foi de 8.1 unidades de tempo.

Já a figura 11 retrata que o maior pico de espera nos scanners foi quando o relógio da simulação estava por volta de 21 chegando a 0.6 unidades de tempo, além disso foi obtido uma espera média de 0.1 unidades de tempo.

4.3 3º Caso de teste: 6 guichês de venda e 6 scanners

Figura 12 – 3º Caso de teste: Média de espera na fila dos guichês de venda ao longo do tempo.

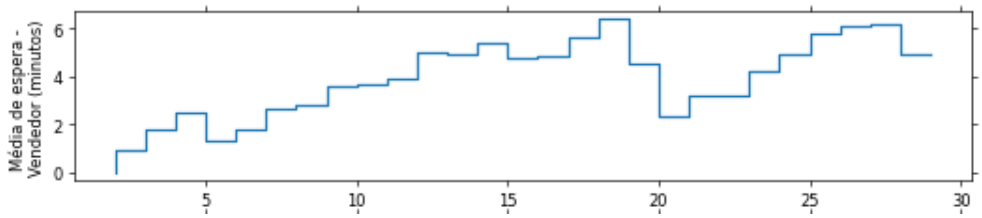
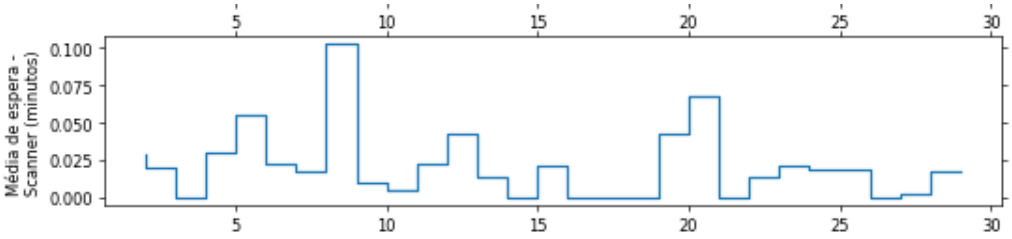


Figura 13 – 3º Caso de teste: Média de espera na fila dos scanners ao longo do tempo.



Com 6 guichês e 6 scanners pode ser notada uma redução considerável nos tempos de espera comparados aos resultados das seções 4.1 e 4.2.

Nesse caso, o tempo de espera nos guichês de venda ultrapassa ligeiramente 6 unidades de tempo em poucos momentos durante a simulação, como pode ser visto na figura 12. Isso se reflete no tempo de espera médio obtido, que foi de 3.7 unidades de tempo, sendo a menor média entre todos os casos de teste.

Os scanners tiveram tempos de espera bem baixos, tendo como pico 0.1 unidades de tempo em um único dado momento da simulação, fora isso a espera se manteve abaixo de 0.075 unidades de tempo durante quase toda a simulação. Assim, tem-se um tempo médio de espera bem próximo de 0.

4.4 4º Caso de teste: 5 guichês de venda, 5 scanners, maior proporção de clientes sem ingresso

Figura 14 – 4º Caso de teste: Média de espera na fila dos guichês de venda ao longo do tempo.

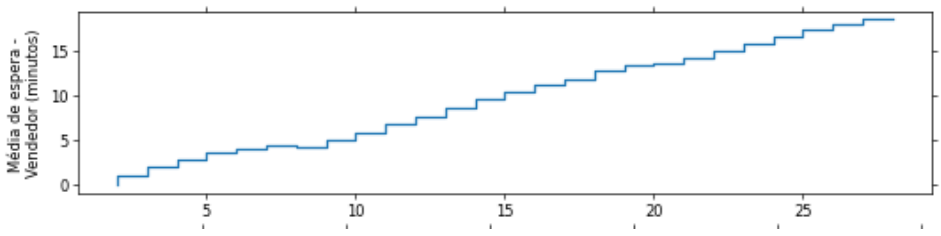
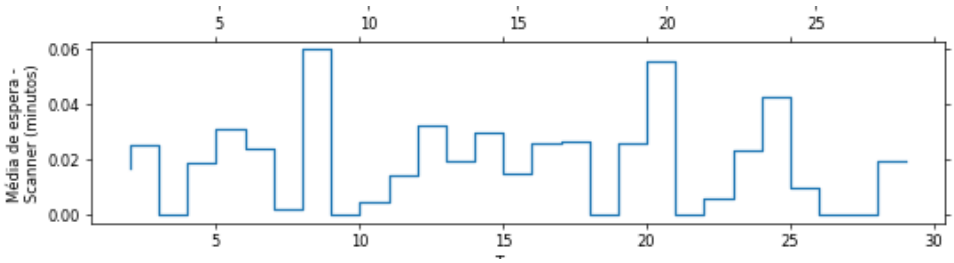


Figura 15 – 4º Caso de teste: Média de espera na fila dos scanners ao longo do tempo.



É possível observar por meio da figura 14 que, na configuração do quarto caso de teste, o crescimento do tempo de espera nos guichês pode ser considerado linear assim como nos dois primeiros casos de teste. Além disso, o tempo de espera tem seu pico bem próximo de 20 unidades de tempo ao fim da simulação, similar ao primeiro caso que conta somente com 2 guichês de venda. Nesse caso, o tempo médio de espera dos guichês foi de 9.4 unidades de tempo.

Por outro lado, a figura 15 mostra que o tempo de espera nos scanners é o menor entre todos os casos de teste, ficando sempre abaixo de 0.06 unidades de tempo, resultando em uma espera média bem próxima de 0.

4.5 Comparação dos resultados

A tabela 4 apresenta uma síntese dos resultados obtidos, mostrando a média geral de tempo de espera em cada fila para cada um dos casos de teste:

Tabela 4 – Médias dos tempos de espera nas filas em cada caso de teste.

	1º CT	2º CT	3º CT	4º CT
Tempo de espera médio - Guichês de venda	10.9	8.1	3.7	9.4
Tempo de espera médio - Scanners	0.6	0.1	≈ 0	≈ 0

Ao observar os três primeiros casos de teste, é possível notar que com o aumento do número de guichês de venda e scanners o tempo de espera médio em ambas as filas decresce consideravelmente. Por outro lado, no quarto caso de teste, que apresenta uma maior porcentagem média de clientes sem ingresso (20% maior que o restante), mesmo possuindo 5 guichês e 5 scanners o tempo médio de espera nos guichês de venda é o segundo maior.

Portanto, com a análise dos resultados obtidos, é possível concluir que para efeito de reduzir as filas de espera nesse cenário, pode ser mais interessante investir nas vendas online de ingressos do que aumentar a quantidade de guichês de venda e scanners.

5 CONCLUSÕES

5.1 Conclusão

Ao final deste trabalho, podemos concluir que um simulador como este pode ser utilizado como base para a análise de diversos processos da vida real que podem ser modelados como uma fila de espera. Também pode ser utilizado para facilitar o aprendizado da linguagem empregada e da simulação de eventos discretos.

Os estudos realizados durante a revisão bibliográfica foram de suma importância para adquirir conhecimento sobre simulação e embasar o projeto teoricamente. O contato com a linguagem Python e o ambiente de simulação SimPy também foi fundamental para o desenvolvimento do simulador. A gama de ferramentas matemáticas e analíticas disponíveis para Python é muito alta e o SimPy completa esses recursos para incluir simulações de eventos discretos, com um código limpo e fácil de entender.

Foi muito proveitoso o desafio de criar uma ferramenta com o objetivo de auxiliar os usuários na vida real. As dificuldades encontradas, considerando o aprendizado da aplicação do framework SimPy em Python e a compreensão do tema puderam ser superadas e o trabalho pôde ser concluído.

5.2 Trabalhos futuros

Como sugestão de trabalhos futuros, cita-se a importância da implementação de um mecanismo para considerar a utilização dos guichês de venda e scanners. A redução do tempo nas filas é apenas um componente da análise, já que a porcentagem de tempo que os vendedores e scanners ficam ociosos também deve ser considerada para se chegar à solução ideal. Além disso, também seria interessante adicionar uma probabilidade de que alguém optou por não entrar em determinada fila, caso ela esteja muito longa, mesmo sendo a menor. Por fim, a implementação de uma interface gráfica, exibindo os deslocamentos de clientes e a formação das filas, por exemplo, também seria de grande auxílio para monitoramento da simulação.

Referências

ARENALES, M.; ARMENTANO, V. **Pesquisa operacional**. Elsevier Brasil, 2006. ISBN 9788535251937. Disponível em: <https://books.google.com.br/books?id=sB_ _Fi8rprEC>.

Bakenaz A. Zeidan. Mathematical modeling of environmental problems. 2015. Disponível em: <http://rgdoi.net/10.13140/RG.2.1.2423.5362>.

BANKS, J. (Ed.). **Discrete-event system simulation**. 4th ed. ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2005. (Prentice-Hall international series in industrial and systems engineering). ISBN 9780131446793.

BORGES, L. E. **Python para desenvolvedores: aborda Python 3.3**. [S.l.]: Novatec Editora, 2014.

BRATLEY, P.; FOX, B.; SCHRAGE, L. **A Guide to Simulation**. Springer New York, 2012. ISBN 9781468401677. Disponível em: <https://books.google.com.br/books?id=bh_ lBwAAQBAJ>.

CHWIF, L.; MEDINA. **Modelagem e simulação de eventos discretos, 4a edição: Teoria e aplicações**. Elsevier Brasil, 2014. ISBN 9788535279337. Disponível em: <https://books.google.com.br/books?id=wZmoBQAAQBAJ>.

CORDEIRO, I. d. C. Simulação computacional de sistemas de partículas via dinâmica molecular. 2021. Disponível em: <http://www.repositorio.ufc.br/handle/riufc/57701>.

FILHO, P. J. d. F. **Introdução a modelagem e simulação de sistemas com aplicações em Arena**. Florianópolis, SC: Visual Books, 2008. OCLC: 817027367. ISBN 9788575022283.

HULING, D.; MILES, S. B. Simulating disaster recovery as discrete event processes using python. In: **2015 IEEE Global Humanitarian Technology Conference (GHTC)**. Seattle, WA, USA: IEEE, 2015. p. 248–253. ISBN 9781467365611. Disponível em: <http://ieeexplore.ieee.org/document/7343980/>.

JAHANGIRIAN, M.; ELDABI, T.; NASEER, A.; STERGIOULAS, L. K.; YOUNG, T. Simulation in manufacturing and business: A review. **European Journal of Operational Research**, v. 203, n. 1, p. 1–13, 2010. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221709004263>.

MOREIRA, D. **Pesquisa operacional: curso introdutório**. São Paulo: Cengage Learning Edições Ltda, 2017. ISBN 9788522110513.

PARK, H.; FISHWICK, P. A gpu-based application framework supporting fast discrete-event simulation. **Simulation**, v. 86, p. 613–628, 10 2010.

PEGDEN, C. D.; SHANNON, R. E.; SADOWSKI, R. P.; PEGDEN, C. D. **Introduction to simulation using SIMAN**. New York: McGraw-Hill, 1990. ISBN 9780070492172.

PINHO, T. M.; COELHO, J. P.; BOAVENTURA-CUNHA, J. Forest-based supply chain modelling using the simpy simulation framework. **IFAC-PapersOnLine**, v. 49, n. 2, p. 90–95, 2016. ISSN 2405-8963. 7th IFAC Conference on Management and Control of Production and Logistics MCPL 2016. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405896316300167>.

PRADO, D. **Teoria das Filas e da Simulação**. Nova Lima: Falconi Editora, 2017. OCLC: 1229919887. ISBN 9788555560200.

SZTRIK, J. **Basic queueing theory: foundations of system performance modeling**. [S.l.: s.n.], 2016. OCLC: 1237181173. ISBN 9783639734713.