UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS
ENGENHARIA DE PRODUÇÃO ELÉTRICA

Maria Giuliana Occhioni Martins Villela

MODELAGEM E SIMULAÇÃO DE TERMINAIS AEROPORTUÁRIOS UTILIZANDO SOFTWARE LIVRE

Florianópolis
2022

Maria Giuliana Occhioni Martins Villela

MODELAGEM E SIMULAÇÃO DE TERMINAIS AEROPORTUÁRIOS UTILIZANDO SOFTWARE LIVRE

Trabalho Conclusão do Curso de Graduação em Engenharia de Produção Elétrica do Centro de Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro Eletricista, habilitado em Produção
Orientador: Prof. Dr. Ricardo Villarroel Dávalos.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Villela, Maria Giuliana Occhioni Martins
Modelagem e simulação de terminais aeroportuários utilizando software livre / Maria Giuliana Occhioni Martins Villela ; orientador, Ricardo Villarroel Dávalos, 2022.
92 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Produção Elétrica, Florianópolis, 2022.

Inclui referências.

1. Engenharia de Produção Elétrica. 2. Simulação a eventos discretos. 3. Terminais de embarque aeroportuários. 4. Adiantamento em relação à chegada. 5. Método de Monte Carlo. I. Dávalos, Ricardo Villarroel. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Produção Elétrica. III. Título.

Maria Giuliana Occhioni Martins Villela

MODELAGEM E SIMULAÇÃO DE TERMINAIS AEROPORTUÁRIOS UTILIZANDO SOFTWARE LIVRE

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de Engenheira Eletricista com habilitação em Produção e aprovado em sua forma final pelo Curso de Engenharia de Produção Elétrica

Florianópolis, 01 de dezembro de 2022.

_____

Prof. Mônica Maria Mendes Luna, Dr.
Coordenadora do Curso

Banca Examinadora:

Documento assinado digitalmente
Ricardo Villarroel Davalos
Data: 19/12/2022 19:56:27-0300
CPF: ***.328.619-**
Verifique as assinaturas em https://v.ufsc.br

_____

Prof. Ricardo Villarroel Dávalos, Dr.
Orientador
Universidade Federal de Santa Catarina

_____

Prof. Guilherme Ernani Vieira, Dr.
Avaliador
Universidade Federal de Santa Catarina

_____

Prof. Sérgio Fernando Mayerle, Dr.
Avaliador
Universidade Federal de Santa Catarina

Este trabalho é dedicado a todos que me incentivaram e me ajudaram a seguir a carreira de engenharia.

AGRADECIMENTOS

Agradeço a minha família e amigos por todo o suporte durante o curso e sempre acreditar que eu poderia alcançar meus objetivos.

Agradeço ao Eduardo Rauh Müller por todo o apoio, encorajamento e auxílio ao longo dos anos.

Agradeço aos professores da UFSC, responsáveis por enormes ensinamentos e principalmente ao meu orientador Ricardo Villarroel Dávalos, por toda a dedicação despendida nesse trabalho.

“Todos os modelos são errados, porém alguns são úteis.” (BOX, 1976)

RESUMO

Dentro do sistema aeroportuário, o terminal de passageiros é um setor com uma grande quantidade de processos, e também é tipicamente o local que apresenta o maior potencial para melhora do nível de serviço. O padrão de chegada de passageiros é um componente crítico da análise do fluxo de passageiros. Dadas as características não usuais desse padrão de chegadas, um modelo foi proposto para capturar o comportamento observado por diversos autores. Um modelo de simulação de terminais aeroportuários foi implementado em linguagem de programação Python, levando em conta o modelo de chegadas proposto, e o paradigma de simulação orientada a eventos. O sistema desenvolvido utiliza o método de Monte Carlo para realizar diversas simulações estocásticas e coletar estatísticas. Como resultado, obteve-se um sistema flexível para simulação de terminais aeroportuários, facilmente adaptável, utilizando apenas tecnologias livres, e que produz relatórios relativos aos indicadores-chave de desempenho (Key performance Indicators - KPIs) do sistema.

Palavras-chave: Simulação a eventos discretos. Terminais de embarque aeroportuários. Adiantamento em relação à chegada. Método de Monte Carlo.

ABSTRACT

Within the airport system, the passenger terminal is a section with a large number of processes, and it is typically the place that has the greatest potential for improving the level of service. The passenger arrivals pattern is a critical component of passenger flow analysis. Given the unusual characteristics of this arrivals pattern, a model was proposed to capture the behavior observed by several authors. An airport terminal simulation model was implemented in Python programming language, considering the proposed arrivals model and the event-driven simulation paradigm. The developed system uses the Monte Carlo method to perform several stochastic simulations and collect statistics. As a result, a flexible and easily adaptable system for simulating airport terminals was obtained, using only free technologies, producing reports related to the system's Key Performance Indicators (KPIs).

Keywords: Discrete event simulation. Airport departure terminal. Earliness of arrival. Monte Carlo method.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ANAC Agência Nacional de Aviação Civil

DES Simulação de eventos discretos

IATA Associação Internacional de Transportes Aéreos

Infraero Empresa Brasileira de Infraestrutura Aeroportuária

KPI Indicador-chave de desempenho

MCM Método de Monte Carlo

SUMÁRIO

1 INTRODUÇÃO

Atualmente o transporte aéreo de passageiros vem aumentando paulatinamente. Segundo pesquisas realizadas pela ANAC (2009, 2019, 2020), enquanto em 2009 as empresas brasileiras do setor apresentaram uma receita de 16,5 bilhões de reais, o valor apresentado em 2019 foi de 46 bilhões. Além dos resultados na receita, a quantidade de passageiros transportados aumentou de 69,7 milhões para 119,4 milhões, o que representa uma alta de 71% no período de 10 anos. A quantidade de passageiros transportados em 2020 teve seu valor reduzido para 52,0 milhões, por causa principalmente das medidas sanitárias para evitar a disseminação de COVID-19. Apesar disso, a pesquisa feita pela IATA (2022) aponta que a perspectiva para a área é de recuperação nos períodos seguintes à restrição.

Thiebaut (2013) aponta que, embora a indústria de aviação seja considerada grande, esta opera com baixa margem de lucro, tendo em 2010 em torno de 3,3%, tornando essencial a busca por operações os mais eficientes possíveis.

A operação e projeto de aeroportos envolve a tomada de decisões em relação ao gerenciamento de diversos tipos de recursos, como por exemplo a quantidade de balcões disponíveis para *check-in*. Para o operador, é desejável garantir a manutenção do nível de serviço utilizando a menor quantidade possível de recursos. Embora possa ser feita uma análise qualitativa do impacto de diversas decisões no nível de serviço, alguns indicadores, como tempo de espera em filas e quantidade de passageiros que perdem seus voos mesmo chegando dentro do período estipulado, podem ser adquiridos através de análises quantitativas.

Para Marcos e Ferreira (2015), a administração dos aeroportos é um elemento estratégico e é necessário oferecer um serviço de qualidade para os passageiros e a gestão de sua capacidade precisa ser tratada com prioridade.

1.1 PROBLEMA

O terminal de passageiros é uma área importante do transporte aéreo de passageiros, que envolve os processos de *check-in*, despacho de bagagens, inspeção de segurança e controle de documentação.

Como o sistema conta com uma quantidade considerável de processos envolvendo os passageiros, um erro de dimensionamento neste setor pode impactar o nível de serviço, ocasionar perdas econômicas e gerar uma perda de embarque por passageiros que chegaram no período estipulado. Por isto é de vital importância que estes processos tenham um

funcionamento adequado, dados os níveis de serviço definidos pelas agências reguladoras do transporte aéreo de passageiros.

Entre os métodos possíveis para definir e calcular os indicadores-chave de desempenho (KPIs) está a Simulação a Eventos Discretos (DES), que envolve definir um modelo suficientemente representativo do sistema para que as conclusões tiradas de sua análise sejam relevantes para o sistema real. Com um sistema aproximado da realidade, é possível analisar diferentes cenários e obter uma resposta próxima do sistema real, sem a necessidade de arcar com os custos e riscos inerentes de se atuar diretamente na realidade.

A técnica mais utilizada em conjunto com a DES é o Método de Monte Carlo, que se apresenta como uma boa opção para sistemas com componentes estocásticos consideráveis e relações complexas, tais como pode ser observado em filas de *check-in* em aeroportos.

Limitações comuns de estudos sobre terminais aeroportuários são (i) a simplificação do modelo de chegada de passageiros considerando-a como independente do horário do voo do passageiro, (ii) não considerar o voo do passageiro ao alocar o mesmo em diferentes processos, e (iii) desconsiderar que processos abrem e fecham de acordo com os voos aos quais estão associados.

Tanto no contexto acadêmico quanto no industrial a DES frequentemente é aplicada utilizando softwares comerciais ou proprietários de custo elevado. Apesar de proporcionarem certas facilidades, o uso dessas ferramentas também traz desvantagens. Por exemplo, o uso de diferentes padrões e formatos fechados dificulta a disseminação de conhecimento, e pode impor dificuldades à continuidade dos trabalhos caso se torne necessário trocar de ferramenta.

Diferentes softwares também têm ferramentas distintas, podendo não existir uma ferramenta comercial ideal para um problema específico. King e Harrison (2010) apontam que softwares comerciais de simulação podem se apresentar inadequados principalmente em modelos complexos ou modelos que não apresentem as ferramentas adequadas desenvolvidas, dada a limitação no ambiente de programação destes.

Dagkakis e Heavey (2016) observam o alto custo relacionado a softwares comerciais de simulação, que envolvem também a compra de serviços adicionais, como treinamento para o uso da ferramenta e compra de pacotes. Isso se torna extremamente importante à medida que uma pesquisa desenvolvida em um software comercial se torna de difícil reprodução para outros pesquisadores sem o treinamento dessa ferramenta.

Atualmente, grande parte das pesquisas na DES utilizam softwares comerciais. Além das desvantagens mencionadas, segundo Wiedemann (2002), a falta de padronização de licenças nas organizações dificulta a continuidade do trabalho por outros pesquisadores e

compartilhar conhecimento, visto que frequentemente é preciso refazer a modelagem em outros softwares para criar um modelo similar, com a mesma estrutura funcional, com a compra de licenças de todos os softwares sendo inviável.

No contexto acadêmico, é preferível o uso de tecnologias livres e/ou de código aberto que permitam a outros pesquisadores saber exatamente como tudo é implementado, facilitando a reprodução de resultados independentemente de ferramentas.

1.2 OBJETIVOS

O objetivo principal deste trabalho é implementar a partir de tecnologias livres um modelo de simulação a eventos discretos do embarque de passageiros em terminais aeroportuários, modelando aspectos comumente negligenciados, como momento de abertura dos processos e processo de chegada dos passageiros.

A fim de garantir que o problema fosse abordado adequadamente, foram definidos os seguintes objetivos específicos:

- Levantar características de terminais aeroportuários e suas possíveis estruturas.
- Estruturar uma ferramenta flexível que seja replicável em terminais aeroportuários diferentes e permita que um usuário adapte o modelo com uma interface simples.
- O modelo deverá considerar os seguintes aspectos:
    - os tempos de chegada dos passageiros dependem do seu voo;
    - o voo de um passageiro determina a quais processos ele pode ser alocado; e
    - o momento de abertura e fechamento dos processos depende dos voos aos quais esses os mesmos estão alocados.
- O modelo de simulação deverá ser capaz de calcular indicadores chave de desempenho do embarque de passageiros em terminais aeroportuários visando avaliar o nível de serviço do sistema.

1.3 JUSTIFICATIVA

O terminal de passageiros é a área do complexo aeroportuário com a qual o passageiro possui mais contato e, portanto, mais afeta a sua percepção do nível de serviço, como

demonstrado por Almeida (1998). Mesmo que ineficiências nesse setor não gerem atraso no voo, fatores como um longo tempo em filas podem prejudicar a percepção de nível de serviço.

Almeida (1998) observa que o processamento rápido do passageiro no terminal aeroportuário contribui fortemente para a melhoria da visão do passageiro perante a companhia aérea.

Manataki e Zografos (2006, 2009) definem o terminal aeroportuário como um sistema complexo, por causa de fatores como a grande quantidade de entidades e processos envolvidos, a complexidade entre as fases de processamento e eventos estocásticos.

Shannon (1998) aponta a principal força da simulação como a possibilidade de simular tanto sistemas já existentes quanto sistemas possíveis. Essa característica se torna extremamente importante quando alterações no sistema são caras ou impactam o nível de serviço percebido ou até quando as alterações no sistema real não são possíveis no momento da análise.

As características da simulação a tornam adequada para a análise de cenários hipotéticos ou inusitados, podendo prever o que aconteceria em casos específicos com base nas informações do sistema na condição usual.

Gatersleben e Van der Weij (1999) defendem a simulação como ferramenta para análise de terminais aeroportuários, citando a dependência entre os processos do sistema como relevante para a análise e a necessidade dessa consideração para evitar subotimização.

Terminais aeroportuários são relativamente padronizados, podendo ser descritos razoavelmente bem por uma estrutura comum. Apesar disso, desconhece-se a existência de uma ferramenta simples para sua simulação.

Gatersleben e Van der Weij (1999) observam que, como o comportamento humano é difícil de definir em modelos, a simulação de processos envolvendo pessoas são menos comuns do que simulações industriais, o que explicaria a carência de ferramentas para a simulação de terminais aeroportuários.

Nesse cenário, percebe-se que seria benéfico para a comunidade acadêmica e indústria, a existência de um código de simulação de terminais aeroportuários de livre acesso e adaptável, que inclua funcionalidades comumente necessárias nesse tipo de aplicação e propicie maior facilidade para um trabalho colaborativo e eficiente.

Com isso, esse trabalho visa implementar um modelo do embarque de passageiros em terminais aeroportuários numa ferramenta DES baseada em tecnologias livres ou de código aberto.

1.4 ESTRUTURA DO TCC

Este TCC encontra-se estruturado por cinco capítulos, referências bibliográficas e oito apêndices.

No capítulo 1 o problema é apresentado, junto com um panorama geral sobre o setor e a oportunidade de pesquisa, seguido por uma breve explicação de como este trabalho visa auxiliar nos problemas encontrados.

O capítulo 2 apresenta a fundamentação teórica do trabalho, tendo como principais temas os terminais aeroportuários, a modelagem, a simulação e modelos de chegada, contando também com considerações finais sobre o tema.

O capítulo 3 apresenta a metodologia, onde é feita a classificação da pesquisa e definidos os procedimentos metodológicos e as delimitações.

O capítulo 4 apresenta o desenvolvimento, onde é descrito o trabalho desenvolvido.

O capítulo 5 apresenta as conclusões principais e recomendações para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 TERMINAIS AEROPORTUÁRIOS

O terminal aeroportuário é a área do aeroporto onde passageiros são transportados da sua chegada até o terminal de embarque da aeronave. Além do processamento de passageiros, são executadas outras operações, que envolvem o processamento de bagagem, movimentação de aeronaves, prestação de serviços, como limpeza da aeronave, e movimentação de insumos, como lanches consumidos durante o voo etc. Esses processamentos podem ser afetados por congestionamentos no terminal de passageiros, que têm impacto direto no atraso dos voos.

Horonjeff (2010) descreve o terminal de passageiros como a principal interface entre os transportes terrestre e aeronáutico. Com isso, caso ocorram falhas nessa área, os processos seguintes são afetados significativamente.

Rolim et al. (2015) mencionam que o terminal de passageiros deve receber atenção especial, visto que este pode limitar a capacidade do aeroporto por inteiro.

Dada a importância do terminal de passageiros para a funcionalidade do sistema, se torna crítico garantir que os processos sejam feitos de forma eficiente, assegurando que nenhum passageiro que chegou dentro do período estipulado perca seu voo.

A operação do terminal de passageiros impacta diretamente no nível de serviço prestado. Enquanto o processamento da bagagem pode acarretar em atrasos para o cliente, o serviço prestado no terminal de passageiros tem impacto direto na satisfação do mesmo. Um tempo de espera longo em filas tem impacto na satisfação, mesmo que não haja atrasos no voo.

Dentro dos aspectos que afetam o nível de serviço no terminal de passageiros, Rolim et al. (2015) cita a rapidez e efetividade das operações como agregadoras de valor para o passageiro, além de aumentarem valor para o aeroporto, visto que aumentam a disponibilidade do passageiro para consumo e exploração das dependências do aeroporto.

A satisfação dos clientes em aeroportos é definida por Joustra (2001) como o objetivo primário, o que, para o *check-in*, significa manter os tempos de fila limitados. O processamento rápido e eficiente também beneficia as companhias aéreas, visto que, segundo Almeida (1998), contribuem fortemente para a imagem da empresa perante a seus clientes.

Para garantir um nível de satisfação mínimo, a IATA (2014) apresenta padrões para o nível de serviço, que inclui tempo de espera e tempo máximo em filas, categorizando cada indicador em 5 categorias, classificando se o nível de serviço do sistema está abaixo ou acima do esperado.

Embora seja importante melhorar a eficiência e manter o nível de serviço desejado, Manataki e Zografos (2009) afirmam que a tomada de decisão para gerenciar terminais aeroportuários é uma tarefa desafiadora, visto que este é um sistema complexo.

Para Manataki e Zografos (2006), as principais causas do comportamento complexo em terminais aeroportuários são a grande quantidade de entidades e processos envolvidos, a relação complexa entre as fases do processamento, a relação entre recursos utilizados e nível de serviço oferecido e a variabilidade e eventos estocásticos.

Alodhaibi, Burdett e Yarlagadda (2017) descrevem os processos de terminais aeroportuários como tendo a chegada como primeiro componente, seguida do *check-in* e despacho de bagagens (podendo ser realizado por diversos meios), sendo seguido pela inspeção de segurança e controle de imigração, a depender do tipo do voo.

Maia et al. (2010) apontam os tipos de *check-in* utilizados em aeroportos brasileiros como: *check-in* tradicional realizado nos balcões, *check-in* de autoatendimento realizado nos totens e *check-in online*, considerando também o despacho realizado no balcão de atendimento e o despacho realizado a partir de estações de despacho.

2.2 INDICADORES – KPI PARA TERMINAIS AEROPORTUÁRIOS

Gkritza, Niemeier e Mannering (2006) observam que o entendimento da dinâmica de satisfação dos passageiros é crítico, visto que o sucesso dos processos e a viabilidade financeira da indústria de transporte aéreo é afetada.

Entre os fatores que impactam a satisfação dos passageiros, Sankaranarayanan et al. (2016) observam a importância do congestionamento e tempo de espera em aeroportos no nível de serviço percebido.

A IATA (2014) define diversos indicadores para o nível de serviço de terminais aeroportuários. Entre eles Al-Sultan (2018) apresenta como importantes para o nível de serviço do aeroporto fatores quantitativos como tamanho das filas ou quantidade de passageiros por área. Além disso, esses indicadores são dinâmicos, variando conforme a quantidade de voos e o horário do dia.

Outros indicadores também podem ser empregados, como citado por Correia e Wirasinghe (2004), como o custo da passagem aérea e dos demais serviços e confiabilidade e previsibilidade do tempo de processamento. Também podem ser considerados indicadores derivados de dados qualitativos, como tratamento oferecido nos processos e no aeroporto.

2.3 MODELAGEM

Dada a complexidade do problema, a modelagem se torna essencial para garantir que o sistema seja retratado como esperado e as devidas considerações sejam feitas.

A modelagem pode ser segmentada em diferentes etapas. Para Fishman (2001), a primeira etapa para estudar um sistema é utilizar do conhecimento acumulado para construir um modelo, que pode ser baseado em informações formais, empíricas ou, normalmente, ambos.

Um modelo, para Ortúzar e Willumsen (2011), é uma representação de uma parte de um sistema real, no qual alguns elementos são de interesse para a análise. Com isso, modelos são específicos para um ponto de vista.

Embora seja possível, Witelski e Bowen (2020) argumentam que tentar incluir todos os detalhes possíveis sobre um sistema com o intuito de obter uma descrição completa do mesmo seria intratável. Porém, modelos com muitas simplificações se tornam triviais e perdem a sua utilidade.

Um problema recorrente na modelagem observado por Fishman (2001) é a necessidade de equilíbrio entre detalhamento e a necessidade de simplicidade para a resolução do modelo. Essa observação traz à tona a necessidade de definir o nível de detalhamento desejado e as técnicas coerentes com esse objetivo.

Entre os modelos, podem ser citados modelos matemáticos que, segundo Ortúzar e Willumsen (2011), são modelos que visam replicar um sistema de interesse e seu comportamento utilizando expressões matemáticas. Essa definição vai ao encontro da observação de Raychaudhuri (2008), que descreve os modelos matemáticos como dependentes de uma série de entradas, com os quais o modelo realiza operações e retorna resultados.

A modelagem matemática pode ser aplicada em diferentes áreas, como o sistema aeroportuário. Segundo Horonjeff (2010), no sistema aeroportuário, a modelagem matemática possibilita analisar fatores importantes para avaliar o nível de serviço de aeroportos, como tempos de fila em diferentes processos, tempo total de processamento, entre outros.

2.4 SIMULAÇÃO

Saliby (1989) descreve a simulação como uma técnica que apoia a realização de experimentos numéricos com modelos lógicos/matemáticos, envolvendo geralmente grande volume de cálculos repetitivos e fazendo uso intensivo do computador.

Já Pegden (1990) define simulação como o processo de projetar um modelo computacional de um sistema real para entender seu comportamento ou avaliar estratégias para sua operação. Com isso, um modelo de simulação pode ser obtido que, segundo Almeida (1998), utiliza de representações lógicas e matemáticas para estimar a resposta do sistema a uma dada entrada.

Marcos e Ferreira (2015) observam que na construção de modelos de simulação, no qual se deve abstrair e reduzir a complexidade do problema, deve-se representá-lo formalmente utilizando um programa de computador e analisá-lo através de experimentos.

Segundo as observações de Rolim et al. (2015), a simulação é algo de grande utilidade para a administração do nível de serviço, visto que ela permite conhecer o comportamento do sistema e gerar cenários alternativos.

Modelos de simulação para terminais aeroportuários são citados por Horonjeff (2010) como úteis para uma análise detalhada do comportamento do sistema ou para a análise de um período mais longo. Além disso, a simulação permite medir indicadores do nível de serviço, como o congestionamento do sistema e o tempo que os passageiros esperam em filas.

Law e Kelton (1991) definem três dimensões como principais para classificar uma simulação, que são:

1. Simulação estática ou dinâmica. A simulação estática é adequada quando o sistema não se altera durante o tempo, já a simulação dinâmica tem alterações no seu estado durante o tempo.
2. Simulação determinística ou estocástica. A simulação determinística apresenta apenas parâmetros pré-definidos, sem considerar aleatoriedade no sistema, ao contrário da simulação estocástica, em que nem todos os parâmetros são conhecidos de antemão, visto que apresentam componentes aleatórios.
3. Simulação contínua ou discreta. Na simulação contínua as alterações de estado são feitas de forma contínua. Já a simulação discreta apresenta alterações no sistema apenas em um conjunto definido de momentos.

Dadas essas classificações, existem oito formas possíveis de classificar uma simulação perante essas características. Onde cada combinação de fatores leva ao uso de diferentes métodos para simulação, podendo inclusive utilizar mais de um método se esses tiverem compatibilidade.

O problema do terminal aeroportuário pode ser classificado como dinâmico, visto que o sistema apresenta alterações no decorrer da simulação. A quantidade de pessoas na fila se altera, a quantidade de voos, a abertura de processos é realizada dependendo do tempo.

Outra classificação a ser realizada é quanto à aleatoriedade do sistema. Como Manataki e Zografos (2006, 2009) definem o terminal aeroportuário como um sistema complexo, com e eventos estocásticos, o uso de simulação estocástica se apresenta como uma boa escolha.

Como o sistema possui grande parte de seus componentes discretos e pode ser descrito por alterações de estado em um momento específico, a simulação discreta se apresenta como uma boa opção quando comparada com a simulação contínua.

2.4.1 Método de Monte Carlo

A simulação de Monte Carlo se apresenta como uma técnica para simulação dinâmica e estocástica, podendo ser adaptada para simulação contínua ou discreta.

O método de Monte Carlo é útil para sistemas em que o componente estocástico é considerável ou o sistema é suficientemente complexo ao ponto que soluções analíticas se tornam muito custosas. Kroese et al. (2014) e Raychaudhuri (2008) apresentam o método de Monte Carlo como uma das abordagens mais úteis, dada a sua simplicidade e aplicabilidade, sendo especialmente úteis em cenários com componentes aleatórios. Amar (2006) descreve o método de Monte Carlo como algoritmos que usam de números aleatórios ou pseudo aleatórios para resolver problemas. A simulação de Monte Carlo é definida por Raychaudhuri (2008), como muito parecida com experimentos aleatórios, onde o resultado não é conhecido de antemão, podendo ser considerada como uma forma metódica de analisar cenários hipotéticos.

O método de Monte Carlo consiste em repetidas avaliações de um sistema estocástico utilizando conjuntos distintos de *inputs*, obtidos através de uma geração de números pseudo aleatórios respeitando as distribuições de probabilidade definidas, e subsequente análise estatística dos resultados obtidos de cada repetição.

Raychaudhuri (2008) aponta diferentes passos para utilizar a simulação de Monte Carlo, começando pela identificação das distribuições de probabilidade associadas aos *inputs*, onde é gerado um conjunto de valores a ser utilizado em cada repetição. Realizando a simulação com cada conjunto definido, é obtido um conjunto de resultados, que passam por uma análise estatística.

A complexidade do sistema e os eventos estocásticos fazem com que o método de Monte Carlo seja um bom candidato, visto que este permite que os eventos sejam representados com fatores estocásticos e que, provavelmente, uma solução analítica precisaria de uma grande simplificação para ser realizada.

2.4.2 Simulação discreta

A simulação discreta é definida por Banks et al. (2004), como a simulação de sistemas que têm suas mudanças feitas em um conjunto discreto no tempo, tendo sua resolução de forma numérica, ao invés de uma resposta analítica.

Existem duas formas principais de realizar a simulação discreta de acordo com a forma como o tempo é avançado, sendo a primeira a passos fixos e a segunda orientada a eventos.

A simulação a passos fixos realiza esse avanço no tempo em passos fixos, tendo o período constante durante toda a simulação. Nesse tipo de avanço, o relógio começa no tempo inicial, tendo um incremento fixo. Toda vez em que esse incremento é realizado, o sistema checa se eventos ocorreram no período transcorrido. Com isso, as alterações de estado são realizadas no momento corrente, e não no momento no qual os eventos foram agendados.

Dado esse funcionamento, a percepção do momento em que os eventos ocorrem fica distorcida, visto que é considerado pelo sistema o primeiro instante simulado após o momento agendado.

A simulação orientada a eventos apresenta passos com períodos variáveis durante a simulação, isso ocorre pois o simulador avança para o próximo momento em que um evento ocorre, ao invés de avançar no tempo em passos fixos.

Dado esse funcionamento, o sistema considera os eventos como ocorrendo no momento agendado, sem distorções nos indicadores ou no sistema.

A simulação orientada a eventos é, segundo Law e Kelton (1991), a mais utilizada. Isso ocorre pois ela apresenta uma grande quantidade de benefícios em relação a simulação a passos fixos, visto que apenas os momentos em que são realizadas alterações no sistema são simulados e não ocorrem alterações de estado nos períodos não simulados, visto que não ocorrem eventos neles.

Law e Kelton (1991) descrevem a simulação de eventos discretos como a modelagem de um sistema que se desenvolve em relação ao tempo utilizando uma representação em que os estados das variáveis mudam instantaneamente em um ponto definido no tempo.

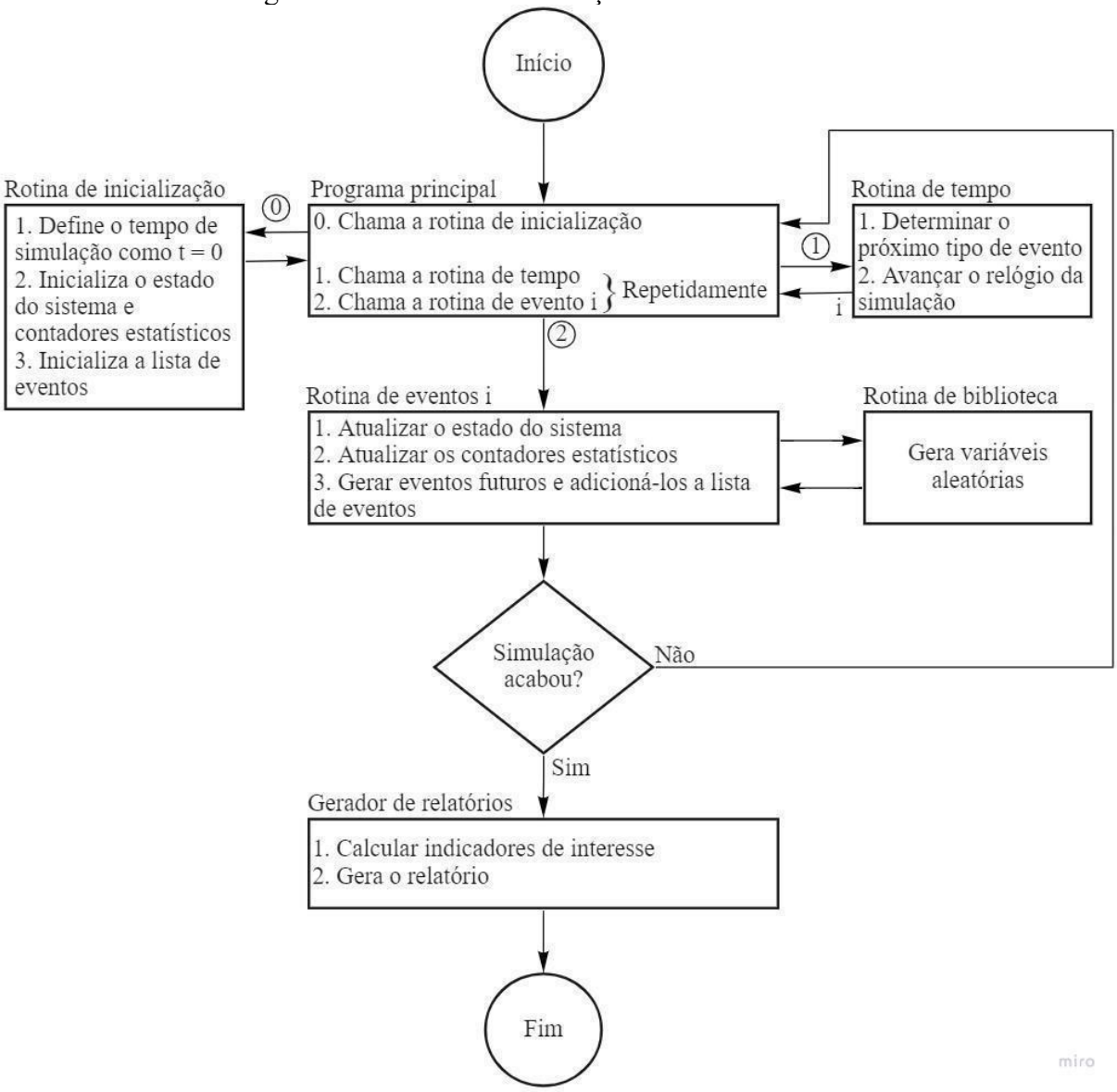
A simulação de eventos discretos se mostra como uma boa opção para a simulação dinâmica e discreta, podendo ser adaptada para simulação determinística ou estocástica. Nela, considera-se um conjunto contável de pontos no tempo, que podem ser chamados de eventos. Essa forma de modelagem leva a ausência de alterações do sistema entre um momento e outro do conjunto.

Law e Kelton (1991) apresentam os principais componentes de uma simulação de eventos discretos como os seguintes:

1. Estado do sistema. O conjunto de variáveis de estado necessárias para descrever o sistema em um momento da simulação.
2. Relógio da simulação. Uma variável responsável por representar o momento da simulação.
3. Lista de eventos. Uma lista que apresenta os eventos e o momento em que estes vão ocorrer.
4. Contadores estatísticos. Variáveis responsáveis por armazenar informações sobre a performance do sistema.
5. Rotina de inicialização. Um subprograma que inicializa o sistema no tempo zero.
6. Rotina de tempo. Um subprograma que determina o próximo evento a partir da lista de eventos e avança o tempo no relógio para o momento de ocorrência deste.
7. Rotina de evento. Um subprograma que atualiza o estado do sistema quando um evento acontecer.
8. Rotina de biblioteca. Subprogramas utilizados para gerar informações a partir de distribuições de probabilidade encontradas na simulação.
9. Gerador de relatórios. Um subprograma que compila os indicadores do sistema.
10. Programa principal. Um subprograma que determina o próximo evento e transfere controle para a rotina de evento correspondente realizar as alterações no estado do sistema.

Com isso, pode-se montar um sistema para a simulação de eventos discretos, onde os componentes interagem de forma a simular o sistema proposto, como pode ser visto na Figura 1.

Figura 1– Estrutura da simulação de eventos discretos



Fonte: Adaptado de Law e Kelton (1991)

A simulação começa com o programa principal chamando a rotina de inicialização, que define o tempo de simulação como zero, inicializa o estado do sistema, os contadores e a lista de eventos. Após a rotina de inicialização ser completa, o controle retorna para o programa principal, que chama a rotina de tempo, que determina o próximo evento e avança o relógio da simulação para o momento de ocorrência deste. Com isso, o controle volta para o programa principal, que chama a rotina do próximo evento, que atualiza o estado do sistema baseado no tipo de evento e em seguida atualiza os contadores estatísticos. Após a atualização, caso exista um componente aleatório nessa rotina de eventos, será chamada a rotina de biblioteca, que gera variáveis aleatórias e retornará um valor para a geração dos novos eventos e devolverá o controle para esta. Com isso, a rotina do evento gera novos eventos e coloca na lista de eventos.

Esse sistema se repete, com exceção da rotina de inicialização, até que a condição de término da simulação seja satisfeita e essa seja terminada. Com o término da simulação, será utilizado o gerador de relatórios, que calcula os indicadores e gera o relatório da simulação, finalizando o processo do simulador.

Dada a estrutura da simulação de eventos discretos, segundo Fishman (2001), são possíveis oito benefícios do uso deste método, que incluem:

1. Permite que o desenvolvedor organize seus conhecimentos sobre o sistema e faça deduções lógicas a partir disso.
2. Melhora o entendimento do sistema.
3. Evidencia a relevância de detalhes do sistema.
4. Diminui o tempo de análise.
5. Provê um ambiente para testar as modificações do sistema.
6. Facilita a manipulação quando comparado ao sistema real.
7. Permite controle de fontes de incerteza.
8. É normalmente menos custoso do que o estudo direto do sistema.

Como um terminal de passageiros pode ser representado por um sistema de eventos discretos, a simulação de eventos discretos se torna adequada nesse caso, visto que podem ser definidos apenas alguns pontos para a mudança no sistema, como início de atendimento, momento de chegada, final de atendimento etc.

2.5 MÉTODOS DE CHEGADA

Na simulação, é importante a escolha dos métodos de chegadas, que definem como uma entidade entra no sistema em relação ao tempo.

Segundo a Landrum & Brown (2010), a escolha do modelo de chegada de passageiros no aeroporto afeta significativamente os indicadores de nível de serviço de passageiros na simulação. Postorino et al. (2019) observam que a modelagem das chegadas de passageiros geralmente é mais complexa do que para outros tipos de entidade, pois envolve o comportamento do passageiro.

O modelo de chegadas mais comumente utilizado em simulações é o chamado tempo entre chegadas, no qual são gerados números aleatórios a partir de uma distribuição de probabilidade para definir o tempo transcorrido entre chegadas para cada par de entidades consecutivas.

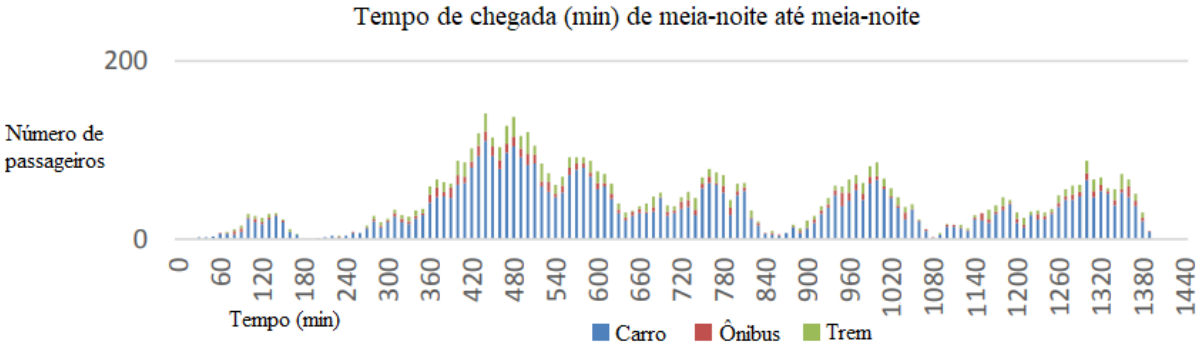
Fishman (2001) apresenta dois modelos de chegadas como os mais utilizados em simulação de eventos discretos, sendo o primeiro o tempo entre chegadas de um item e o segundo utiliza do tempo entre chegadas de um lote de itens. Embora apenas esses dois modelos sejam apresentados, Fishman (2001) deixa claro que esses possuem limitações.

Uma maneira comum para lidar com a dependência das chegadas em relação ao tempo é a mudança da distribuição de probabilidade do tempo entre chegadas no decorrer da simulação. Embora essa prática seja útil em diversos eventos, ela possui limitações quando se considera a simulação de terminais aeroportuários.

Outra forma de representar a chegada dos passageiros é pela curva de *check-in*, onde a chegada dos passageiros é representada por uma curva de adiantamento em relação ao voo. Ou seja, modela-se o tempo de chegada de um passageiro como um adiantamento aleatório em relação ao momento agendado para a partida do voo.

Alguns autores exploraram as características da chegada de passageiros em terminais aeroportuários, comentando sobre a correlação entre momento de chegada e horário do voo. Alodhaibi, Burdett e Yarlagadda (2017) observam que o padrão de chegada dos passageiros de terminais aeroportuários é afetado pelo horário de saída do voo e destino do voo, e apresentam o momento de chegadas de passageiros em um aeroporto em um dia, dividindo essas chegadas também por meio de transporte. Na Figura 2 pode ser visto o padrão de chegadas obtido nesse estudo, que apresenta grande variação conforme o horário do voo.

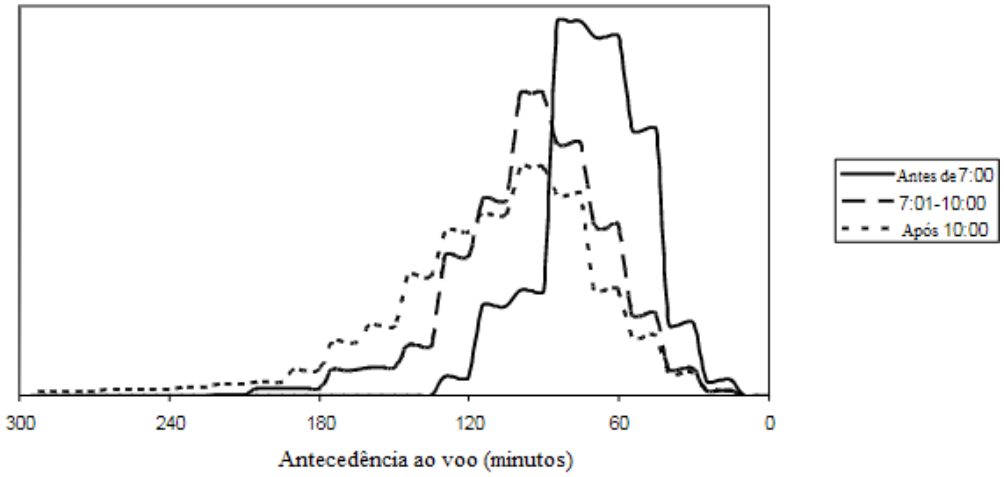
Figura 2 – Padrão de chegadas para ocupação total das aeronaves



Fonte: Adaptado de Alodhaibi, Burdett e Yarlagadda (2017)

Barros e Tomber (2007) observam que a distribuição de chegada dos passageiros depende do horário do voo e fatores como o tipo de voo e o formato de distribuição. Além dessa análise, foi observado também que essa distribuição se altera dependendo do horário de saída do voo, como pode ser visto na Figura 3.

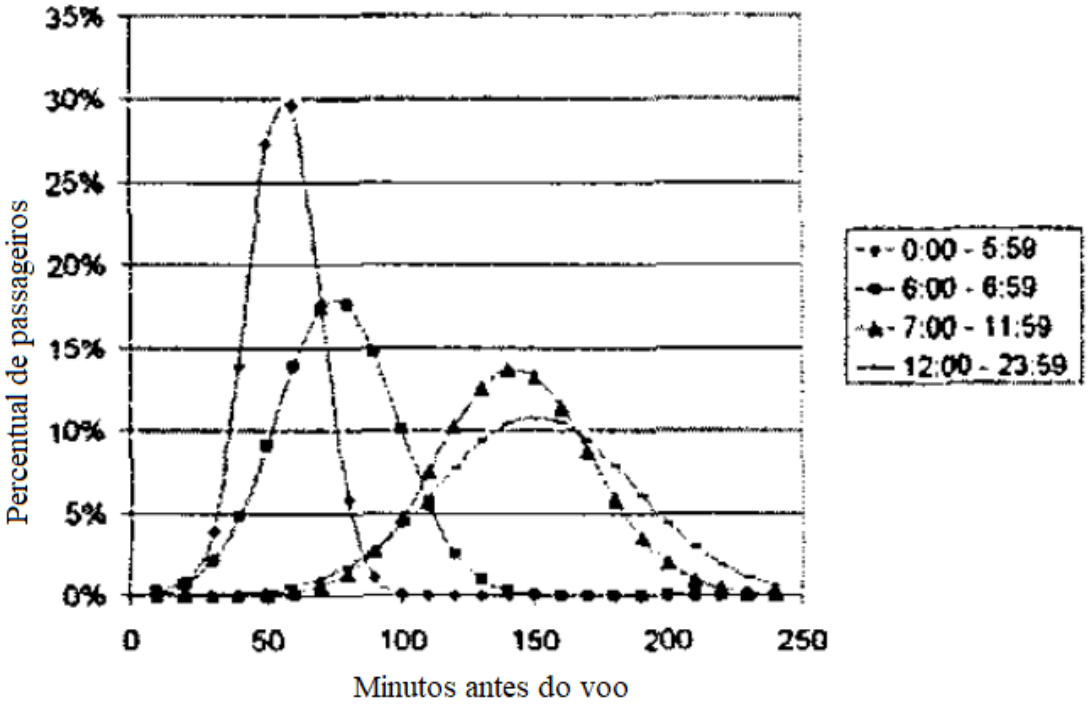
Figura 3 – Antecedência de chegada em relação ao horário do voo



Fonte: Adaptado de Barros e Tomber (2007)

Robertson et al. (2002) também utilizam da antecedência em relação ao voo para sua análise e apresentam as curvas de chegadas de passageiros em diferentes períodos do dia, seguindo uma distribuição normal, com os passageiros chegando com menor antecedência no período da manhã e com a antecedência aumentando ao longo do dia, como pode ser visto na Figura 4.

Figura 4 – Antecedência de chegada em relação ao horário do voo



Fonte: Adaptado de Robertson et al. (2002)

Postorino et al. (2019) também observam as diferentes distribuições de adiantamento em relação ao voo durante o decorrer do dia em aeroportos. Justificando essa diferença pelos diferentes perfis de trânsito durante o dia.

Pode ser observado que o processo de chegada afeta a utilização de diversos recursos e os tempos de espera em filas, afetando todo o terminal e sendo essencial para uma análise correta.

O uso de valores médios e não dependentes do horário do voo fazem com que os picos de uso de recursos sejam mascarados, já que o sistema não se altera ao longo do dia.

Embora a forma mais comum de simular sistemas seja utilizando o tempo entre chegadas como método de chegadas, o terminal aeroportuário é um sistema atípico, com uma quantidade pré-definida de passagens vendidas para cada voo e dependência entre o tempo de chegada de um passageiro e o horário do seu voo.

Além disso, o tempo de antecedência em relação ao horário do voo varia, podendo ser observado que só existirá a chegada de passageiros no terminal se houver um voo próximo. Se o primeiro voo do dia for as 8 da manhã, não haverá a mesma chegada de passageiros nas primeiras horas do dia quando comparado com as horas que antecedem esse voo.

A quantidade de passageiros que chega para um determinado voo também é importante, visto que não ocorre a chegada de uma quantidade de passageiros para um voo maior do que a quantidade de passagens vendidas.

Para representar a dependência do momento de chegada do passageiro e o horário de partida do voo, diversos autores implementaram como solução um sistema dinâmico de tempo entre chegadas, que se altera a partir dos horários dos voos agendados.

Stolletz (2011) utiliza do tempo entre chegadas para fazer uma análise do terminal aeroportuário, alterando esta a cada 30 minutos.

Embora Barros e Tomber (2007) explorem a distribuição de adiantamento em relação à saída do voo, na implementação da simulação estes optam por dividir o tempo em períodos de 15 minutos, com cada período com um tempo entre chegadas, baseado na análise da distribuição de chegada de passageiros.

Embora tenha uma análise semelhante, Robertson et al. (2002) opta por realizar a simulação de uma forma diferente. Baseando-se nas distribuições de chegada, são definidas as quantidades de passageiros a chegar simultaneamente a cada período de 10 minutos.

Ambas as análises focam em partes diferentes do problema. Enquanto Barros e Tomber (2007) apresentam um modelo de chegadas que desconsidera a quantidade máxima de passageiros que chega para determinado voo, Robertson et al. (2002) dividem as chegadas em

intervalos de 10 minutos, o que faz com que todos os passageiros cheguem ao mesmo instante nesse período.

Com isso, levando em conta as características do sistema apresentadas, percebe-se os seguintes requisitos para um modelo de chegada de passageiros adequado:

a) O tempo de chegada dos passageiros deve estar associado ao momento de seu voo, obedecendo uma distribuição de adiantamento.
b) A quantidade de passageiros que chegam para um determinado voo deve obedecer a quantidade de passagens vendidas e a taxa de ocupação esperada da aeronave.

2.6 FERRAMENTAS DES COM TECNOLOGIAS LIVRES

King e Harrison (2010) apontam as dificuldades de configurar softwares comerciais de simulação, visto que, quando alguma interface de customização é oferecida, essa geralmente é de baixo nível e carece de ferramentas de programação e *debugging*.

King e Harrison (2010) citam alguns benefícios de DES em linguagens de propósito geral, como a maior facilidade de interface com bancos de dados e sistemas de controle e a possibilidade de criar interfaces de fácil uso.

Dagkakis et al. (2013) apresenta a linguagem Python como uma opção ideal para o desenvolvimento de DES, visto que esta apresenta uma grande capacidade de processamento de listas quando comparado com outras opções, uma característica essencial para modelar sistemas de filas.

A linguagem Python disponibiliza um pacote para simulação de eventos discretos, chamado Simpy, desenvolvido originalmente por Klaus Müller e Tony Vignaux, e mantido e atualizado por vários outros desenvolvedores. Segundo Dagkakis et al. (2013) a ferramenta Simpy oferece duas principais classes, que representam as entidades a serem simuladas e os recursos, além de funções auxiliares na lógica de sistemas. O Simpy também apresenta outros benefícios, como observado por Matloff (2008), pela facilidade do uso de corrotinas (funções que podem ceder o controle do fluxo de execução, permitindo que sua execução seja suspensa ou retomada) quando comparado com outras formas de simulação.

A presença de documentação, guias e exemplos práticos para o uso do Simpy foi observado por Guerrero et al. (2022) e é considerada uma característica importante, visto que o uso de ferramentas DES com tecnologias livres ainda é baixo.

Existem também outros pacotes desenvolvidos por pesquisadores para a modelagem utilizando DES em linguagem Python, como o Manpy, desenvolvido por Dagkakis et al. (2013) e o Ciw, desenvolvido por Palmer et al. (2019). Além disso, existem iniciativas programadas em outras linguagens, como King e Harrison (2010), que utilizaram Java para a simulação de eventos discretos.

Embora não seja a ferramenta mais rápida disponível para DES, a linguagem Python apresenta uma grande comunidade e uma interface de fácil uso, o que permite que mais pesquisadores utilizem tecnologias desenvolvidas nessa linguagem. Com isso, para problemas em que uma resposta rápida não é a prioridade, a linguagem se apresenta como uma boa alternativa.

2.7 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Para a análise do nível de serviço oferecido em terminais aeroportuários, a simulação de eventos discretos e a utilização do método de Monte Carlo se mostram favoráveis, visto que a estrutura do sistema é compatível com ambos os métodos.

O sistema é altamente complexo e possui filas não convencionais, o que o torna um bom candidato para a utilização do método de Monte Carlo. Além disso, o sistema é principalmente discreto, o que indica que a simulação de eventos discretos é uma forma adequada de se simular o sistema, ao invés de caracterizá-lo como um sistema contínuo.

Os métodos de chegadas de passageiros encontrados na literatura foram diversos, sendo selecionados alguns para a análise. Porém foram encontrados problemas em ambos os métodos analisados.

A utilização de uma quantidade fixa de passageiros chegando em períodos determinados é insuficiente no sentido de criar ondas de passageiros e desconsiderar o componente estocástico do momento de chegada, considerando esse fator como determinístico.

Já a utilização de diferentes tempos entre chegadas durante o dia é insuficiente no sentido em que não garante que a quantidade de passageiros gerada será compatível com a quantidade de passagens vendidas. Limitar a criação de entidades utilizando esse método criaria outro problema, pois a distribuição de chegadas de passageiros seria truncada a partir de um certo tempo, resultando numa distribuição que não representa o sistema analisado.

3 METODOLOGIA

3.1 CLASSIFICAÇÃO DA PESQUISA

A pesquisa pode ser classificada como aplicada, visto que objetiva gerar conhecimentos para a aplicação prática de problemas.

A abordagem pode ser classificada como quantitativa, visto que os indicadores analisados são puramente quantitativos.

Os procedimentos técnicos são classificados como estudo de caso, pois o trabalho foca no estudo profundo da simulação de terminais de passageiros aeroportuários.

3.2 PROCEDIMENTOS METODOLÓGICOS

As principais metodologias utilizadas para simulação a eventos discretos apresentam poucas diferenças entre si, sendo compostas pelos mesmos passos principais, com algumas adições ou simplificações.

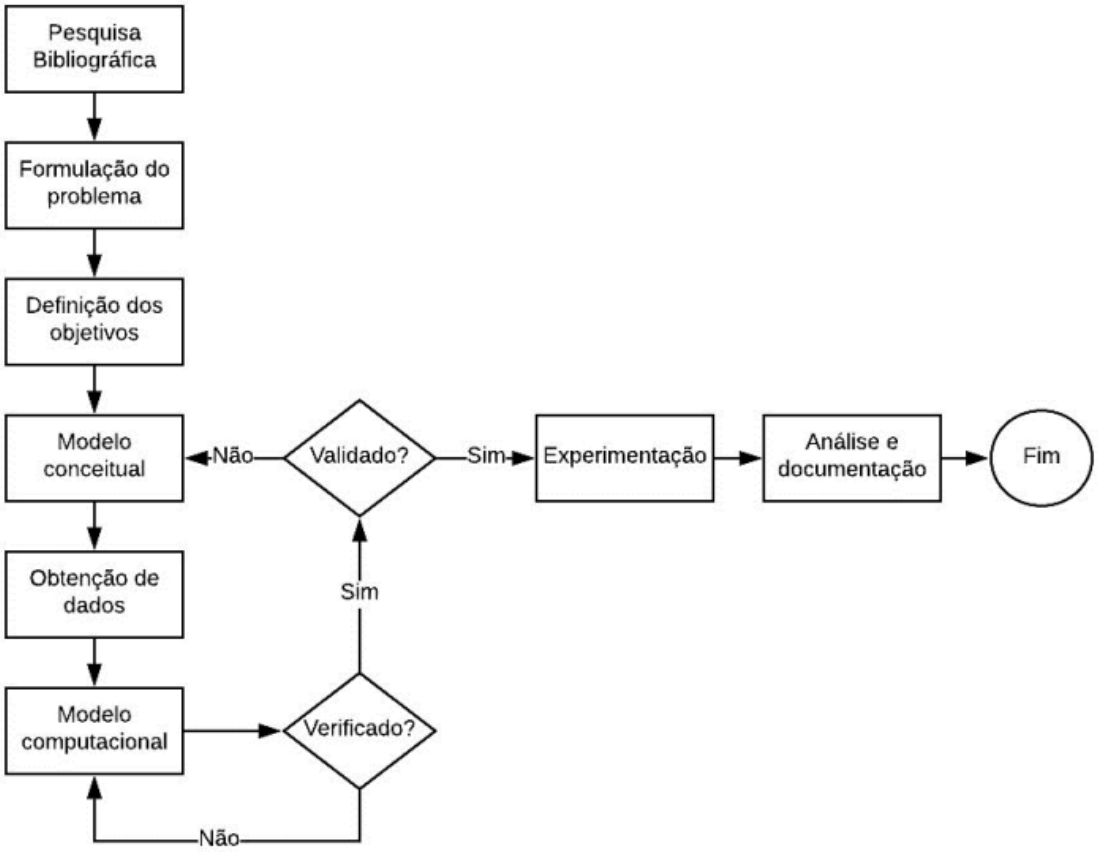
Shannon (1998) propõe uma metodologia para a realização de simulações de eventos discretos, com os seguintes passos:

1. Definição do problema. Definir claramente os objetivos do estudo.
2. Planejamento do projeto. Garantir que existem recursos suficientes para realizar a análise.
3. Definição do sistema. Definir as limitações e restrições que serão utilizadas na definição e investigação do sistema.
4. Formulação do modelo conceitual. Desenvolver um modelo preliminar que constitua o sistema, podendo utilizar de diferentes representações.
5. Design experimental preliminar. Selecionar os indicadores de efetividade e planejar como estes serão obtidos e utilizados.
6. Preparação dos dados de entrada. Identificar e coletar os dados de entrada do sistema.
7. Tradução do modelo. Traduzir o modelo conceitual para a linguagem de simulação a ser utilizada.

8. Verificação e validação. Garantir que o modelo opera da forma planejada pelo modelo conceitual e que o sistema funciona de maneira representativa do modelo real.
9. Design experimental final. Projetar um experimento que obtenha indicadores e determinar como os testes serão executados.
10. Experimentação. Executar a simulação para gerar os indicadores desejados e realizar análise de sensibilidade.
11. Análise e interpretação. Inferir sobre o sistema a partir dos resultados obtidos na simulação.
12. Implementação e documentação. Reportar resultados, colocá-los em uso e documentar as descobertas.

Banks et al. (2004) propõe uma metodologia específica para simulação de eventos discretos, esquematizada na Figura 5.

Figura 5 – Metodologia de Banks et al. (2004)



Fonte: Adaptado de Banks et al. (2004)

Analisando as metodologias descritas, optou-se por utilizar a metodologia de Banks et al. (2004), por ser específica para simulação de eventos discretos e amplamente utilizada.

A primeira etapa do método seria a pesquisa bibliográfica, onde se pode ter um entendimento mais profundo do problema e dos métodos mais utilizados recentemente.

Em seguida, deve ser realizada a formulação do problema e dos objetivos, identificando os indicadores, as variáveis envolvidas e suas relações.

O próximo passo é a criação de um modelo conceitual, onde se encontra um recorte da realidade que seja útil para a análise utilizada e define-se a estrutura da simulação.

Em seguida deve ser realizada a obtenção de dados, para definir a alimentação do modelo, seguida da elaboração do modelo computacional.

O modelo computacional deve ser verificado, a garantir que este esteja de acordo com o modelo conceitual. Se este não estiver de acordo, o procedimento deve ser repetido de forma iterativa, até que essa condição seja realizada.

Assim que o modelo estiver verificado, o modelo deve ser validado, onde deve-se julgar se o modelo conceitual está tendo um recorte da realidade compatível com o desejado para alcançar os objetivos definidos.

Caso a validação não ocorra, deve-se repetir o procedimento, possivelmente retornando à etapa de criação do modelo conceitual, até que a validação seja satisfatória.

Com o sistema verificado e validado, podem ser realizados experimentos, avaliando-se diferentes cenários e obtendo-se os indicadores correspondentes.

Com a experimentação realizada e os cenários desejados explorados, a etapa final é de análise dos resultados e documentação.

A primeira etapa do trabalho consistiu em realizar a pesquisa bibliográfica, que foi realizada previamente e as referências mais relevantes foram apresentadas no capítulo 2.

Com isso, foi realizada a formulação do problema, utilizando como base as principais características abordadas na literatura e relatos sobre o funcionamento do sistema.

Foram definidos então os objetivos do trabalho, baseando-se nos problemas encontrados na literatura e as áreas com maior espaço para melhoria.

Com isso, foi criado um modelo conceitual e foram obtidos dados, que se encontram no capítulo 4 do trabalho.

Utilizando esse modelo, foi feita a modelagem computacional utilizando a linguagem Python e realizada a verificação até que o modelo computacional estivesse de acordo com o esperado.

Dadas as delimitações do trabalho, a etapa da validação será recomendada para trabalhos futuros.

3.3 DELIMITAÇÕES

O trabalho não visa sugerir alterações em qualquer sistema real, apenas representá-lo e oferecer uma ferramenta para sua análise.

O trabalho irá tratar apenas da estrutura aeroportuária no Brasil, logo divergências de protocolos em outros países não serão consideradas.

O trabalho visa avaliar apenas os processamentos relativos ao terminal aeroportuário.

Não são considerados diferentes pontos de chegada para os setores dentro do aeroporto.

O modelo considera que os passageiros irão passar pelos processamentos assim que possível, utilizando a estrutura do aeroporto (lojas, restaurantes etc.) apenas enquanto estiverem esperando a abertura do processo seguinte.

É considerado que o único fator que afeta a probabilidade de realização de *check-in online* e presença de bagagem é o tipo de voo (Internacional ou Nacional) selecionado.

Não são consideradas possíveis intervenções policiais ou recusa de documentação no modelo.

Esse trabalho não irá realizar a etapa de validação do modelo e experimentação.

4 DESENVOLVIMENTO

Apresenta-se a estrutura geral do fluxo de passageiros em terminais aeroportuários, abrangendo desde a chegada dos passageiros até a inspeção de segurança (voos nacionais) ou checagem da documentação (voos internacionais). Com isso, elaborou-se uma estrutura que permite diferentes processamentos e é capaz de abranger os mais diferentes aeroportos.

4.1 ESTRUTURA DO PROBLEMA

4.1.1 Indicadores

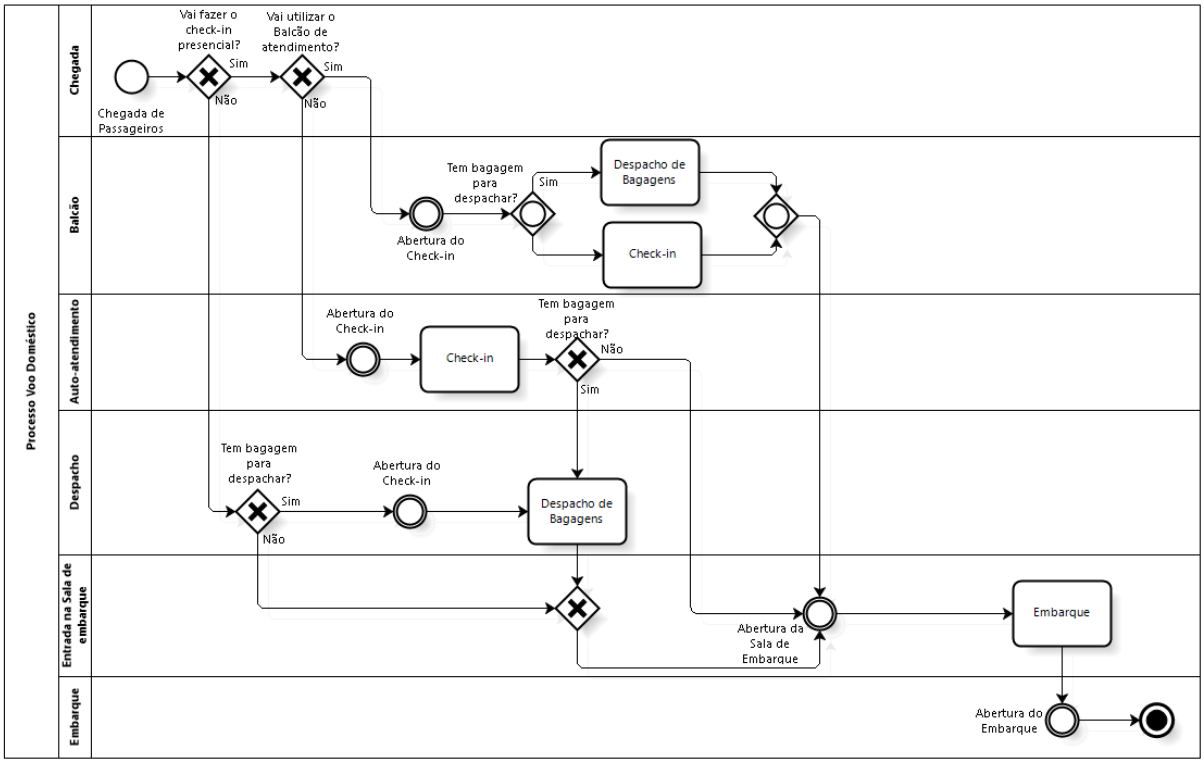
Como o nível de serviço é determinado pela IATA utilizando o espaço por passageiro e tempo máximo em filas para cada processamento, esses foram definidos como os indicadores mais relevantes para a simulação.

Além desses indicadores, foram definidos indicadores auxiliares que ajudam a ter uma maior noção da operação do sistema, assim como a quantidade de passageiros a perder o voo e a distribuição de chegadas ao longo do tempo.

4.1.2 Modelo conceitual

No terminal, são realizados os processos de *check-in* presencial, despacho de bagagens, e inspeção de segurança. Na Figura 6 podem ser vistos os processamentos possíveis, suas condições e a localização de cada processamento para o caso de um voo doméstico.

Figura 6 – Processos realizados no terminal de passageiros



Fonte: Villela, Dávalos e Cavaco (2019)

O processamento inicia quando o passageiro chega no terminal, podendo ou não ter feito o *check-in online*. Caso o *check-in* não tenha sido realizado de forma *online*, existe a necessidade de fazê-lo utilizando a estrutura do aeroporto, que pode ter balcões de atendimento e autoatendimento. Existe também a possibilidade de o passageiro ter ou não bagagem para despachar, podendo realizá-la junto ao *check-in* no balcão de atendimento, ou a parte, no caso do *check-in online* ou autoatendimento. O único processo obrigatório é a inspeção de segurança, que ocorre após a abertura da sala de embarque.

No caso de um voo internacional, as mesmas etapas acontecem, embora possam ser feitas de forma diferente, como no caso da inspeção de segurança, que é mais rigorosa para o voo internacional.

Para o processo para voos internacionais, o controle de documentação é inserido e eventuais divergências, como intervenções da polícia no processamento de um passageiro, estão sendo desconsideradas.

Após os processamentos relacionados ao terminal, o passageiro espera o voo na sala de pré-embarque, seguido do processo de embarque na aeronave.

4.1.3 Distribuição de chegadas

Levando em consideração que o modelo de adiantamento em relação ao horário do voo se encontra como uma opção pesquisada e que representa o sistema de forma satisfatória, foi optado por utilizar um método de chegadas com esse conceito. Utilizando os achados de Robertson et al. (2002), foi definido que as distribuições de chegada deveriam ser específicas para cada voo.

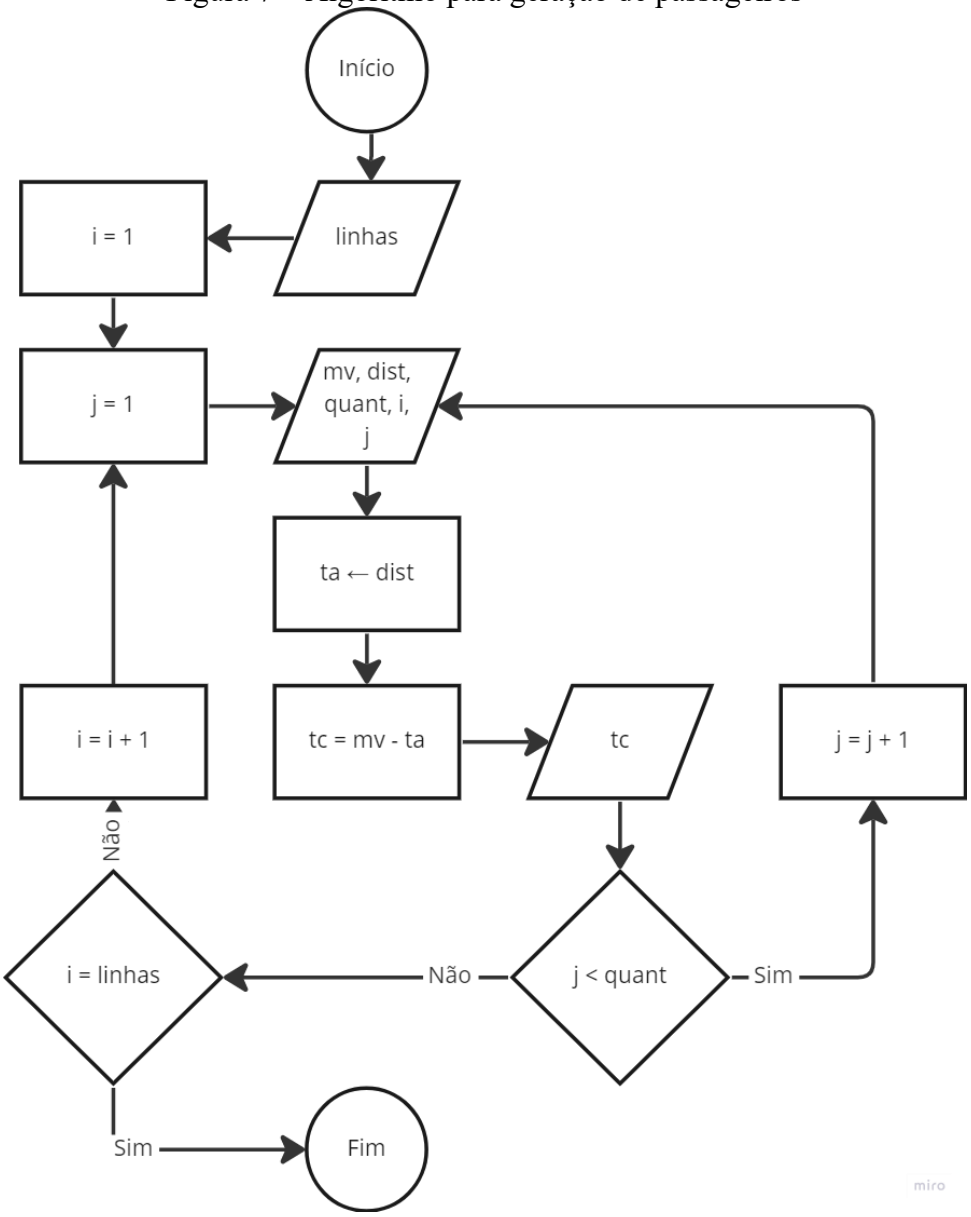
Com isso, foi criado o algoritmo de geração de passageiros demonstrado na Figura 7 para simulação de eventos discretos, podendo ser utilizado para Monte Carlo ou não.

O início da geração dos passageiros envolve guardar a quantidade de entradas (linhas) de voos, o que é equivalente a quantidade de voos a serem simulados. O identificador do voo e o identificador do passageiro são definidos como 1 e são associados ao momento do voo, a distribuição de chegadas relativa ao voo, a quantidade de passageiros e os identificadores do voo e passageiro às respectivas variáveis. Para cada passageiro, com base na distribuição de chegadas definida, é gerado o seu tempo de adiantamento, e calculado o seu tempo de chegada no sistema.

Após o processamento de cada passageiro, caso o algoritmo não tenha gerado os tempos de chegada para todos os passageiros do voo corrente, o identificador do passageiro é incrementado em um e se repete o processamento por passageiro, até que todos os passageiros daquele voo sejam gerados.

Após a geração dos tempos de chegada de todos os passageiros para determinado voo, o gerador passa para o próximo voo, incrementando o identificador do voo e repetindo o processo, até que todos os voos tenham sido analisados.

Figura 7 – Algoritmo para geração de passageiros



Fonte: Elaborado pela autora

Em que:

- dist: distribuição de probabilidade do voo;
- i: sequência do voo analisado;
- j: número do passageiro do voo analisado;
- linhas: quantidade de voos a serem analisados;
- mv: momento do voo;
- quant: quantidade de passageiros alocados ao voo;
- ta: tempo de antecedência em relação ao horário do voo; e
- tc: tempo de chegada do passageiro.

4.1.4 Dados de entrada

Para facilitar a configuração do sistema, foi decidido permitir a entrada de dados através do preenchimento de uma planilha, com os dados agrupados em diversas abas ou seções. Ao longo desta seção, nos quadros mostrados com exemplos de dados de entrada, os campos editáveis pelo usuário para a inserção de dados são destacados por um fundo cinza.

A primeira seção de dados de entrada compreende as informações gerais da simulação, como quantidade de repetições a serem realizadas e momento inicial e final da simulação, como ilustrado no Quadro 1.

Quadro 1 – Exemplo de configurações gerais de entrada

Configurações gerais	
Repetições	30
Dia inicial	2-set.
Horário Inicial	12:00
Dia final	3-set.
Horário final	20:00

As informações gerais de entrada também incluem dados padrão para voos nacionais e internacionais, que não vão depender de informações do voo, como a distribuição padrão de chegadas para o tipo de voo, que vai ser utilizada em caso de falta de informações sobre a distribuição do voo em específico, podendo ser escolhida entre distribuição uniforme, normal, triangular e um valor constante, tendo seus valores em minutos de antecedência ao voo. Além disso, pode-se especificar a probabilidade de um passageiro ter realizado *check-in online* e de ter bagagem de acordo com o tipo de voo, como pode ser visto no Quadro 2.

Quadro 2 – Exemplo de dados de entrada gerais para voos

Voos nacionais	
Distribuição de chegadas [min]	Normal (μ, σ)
μ =	90
σ =	15
Probabilidade de *check-in online*	50%
Probabilidade de passageiro com bagagem	50%
Voos internacionais	
Distribuição de chegadas [min]	Triangular (a, b, c)
a =	30
b =	180
c =	120
Probabilidade de *check-in online*	40%
Probabilidade de passageiro com bagagem	80%

Como alguns processos no terminal de embarque são separados por companhias aéreas ou localização do portão de embarque, foi utilizado o conceito de grupos de processamento para modelar esse comportamento. O usuário pode definir grupos para cada tipo de atendimento, cada um com características próprias de tempo de processamento, horário de abertura etc. Cada grupo desses pode ser associado a um conjunto de voos distintos. Por exemplo, pode-se definir um grupo de processamento de balcão chamado “Grupo 1” com dadas características, referente ao balcão de atendimento de determinada companhia aérea, e associar os voos desta companhia a esse grupo de processamento.

O Quadro 3 ilustra a definição básica dos nomes de grupos utilizados na simulação.

Quadro 3 – Exemplo de conjunto de grupos a serem utilizados na simulação

Grupos				
Balcões	Autoatendimento	Despacho	Segurança	Documentos
Grupo B 1	Grupo A 1	Grupo De 1	Grupo S 1	Grupo Doc 1
Grupo B 2	Grupo A 2 e 3	Grupo De 2	Grupo S 2, 3 e 4	Grupo Doc 2, 3 e 4
Grupo B 3 e 4	Grupo A 4	Grupo De 3 e 4		

Dada a dimensão de alguns aeroportos, a distância entre processamentos não é desconsiderável, e foi percebida uma necessidade de ter a distância aproximada entre estações que podem ter fluxo de passageiros. Com isso, foi criada uma seção de inputs para a distância entre processamentos, onde deve ser colocado, para cada par de processos que podem ser realizados em sequência, o processo e o grupo de cada possível origem e destino, devendo ser descritos todos os processos que podem ter fluxo de passageiros, incluindo o ponto de chegada, como exemplificado no Quadro 4.

Quadro 4 – Exemplo de dados de entrada de fluxo entre setores e distância

Entrada	Grupo entrada	Saída	Grupo saída	Distância [m]
Chegada		Autoatendimento	Grupo A 1	40
Chegada		Balcões	Grupo B 1	60
Chegada		Despacho	Grupo De 1	70
Chegada		Segurança	Grupo S 1	60
Autoatendimento	Grupo A 1	Balcões	Grupo B 1	60
Autoatendimento	Grupo A 1	Despacho	Grupo De 1	120
Autoatendimento	Grupo A 1	Segurança	Grupo S 1	50
Balcões	Grupo B 1	Segurança	Grupo S 1	60
Despacho	Grupo De 1	Segurança	Grupo S 1	80
Segurança	Grupo S 1	Documentos	Grupo Doc 1	60

Foi definido também um formato para as informações de entrada de cada voo, que incluem identificadores, como o identificador do voo e a companhia aérea responsável. Assim como informações gerais do voo, como data e horário de partida, tipo de voo (Internacional ou Nacional) e a quantidade de passageiros alocados ao voo, como pode ser visto no Quadro 5.

Quadro 5 – Exemplo de dados de entrada específicos do voo

Identificadores		Informações gerais de voo			
Voo	Companhia	Data	Horário	Categoria	Quantidade de Passageiros
Voo 1	Companhia 1	2-set.	18:30	Internacional	300
Voo 2	Companhia 1	2-set.	19:30	Nacional	200

Nas informações do voo também estão inclusos os grupos de processamento para cada voo, além do momento de abertura do *check-in* e a inspeção de segurança em relação ao momento de partida do voo. Além disso, é informada a antecedência com a qual o balcão de prioridade para voos próximos abre e a antecedência com a qual não é mais aceito *check-in* para cada voo, como pode ser visto no Quadro 6.

Quadro 6 – Exemplo de dados de entrada de processos específicos do voo

Grupos					Abertura *check-in* [min]	Abertura da Prioridade de *check-in* [min]	Fechamento *check-in* [min]	Abertura Segurança [min]
Balcões	Autoatendimento	Despacho	Segurança	Documentos				
Grupo B 1	Grupo A 1	Grupo De 1	Grupo S 1	Grupo Doc 1	120	45	30	90
Grupo B 1	Grupo A 1	Grupo De 1	Grupo S 1	Grupo Doc 1	120	45	30	90

As informações finais a serem inseridas são a distribuição de chegadas, que é de preenchimento opcional, visto que caso não declarada, o sistema irá utilizar a distribuição padrão da categoria à qual esse voo pertence. No Quadro 7 pode-se ver na primeira linha uma distribuição de chegadas diferente da definida pela categoria, enquanto a segunda linha não está preenchida, o que significa que será utilizada a distribuição padrão. A escolha do tipo de distribuição automaticamente altera os nomes dos parâmetros da mesma na planilha.

Quadro 7 – Exemplo de dados de entrada para distribuição de chegadas do voo

Distribuição de chegadas [min]			
Triangular (a, b, c)	a = 30	b = 200	c = 120

As informações necessárias relacionadas aos recursos são a distribuição de probabilidade do tempo de atendimento e a capacidade para cada processo de cada grupo definido, como pode ser visto no Quadro 8. Entende-se por estações o local onde o processamento do passageiro é realizado.

Quadro 8 – Exemplo de dados de entrada para processamento em estações

Autoatendimento					
Grupo	Capacidade	Tempo de atendimento [segundos]			
Grupo A 1	5	Triangular (a, b, c)	a = 40	b = 150	c = 80

Para o caso do processamento dos balcões, é adicionada a informação da capacidade dos balcões de atraso, como pode ser visto no Quadro 9.

Quadro 9 – Exemplo de dados de entrada para o processamento no balcão

Balcões						
Grupo	Capacidade	Capacidade balcões de atraso	Tempo de atendimento [segundos]			
Grupo B 1	3	2	Triangular (a, b, c)	a = 60	b = 120	c = 90

4.2 ESTRUTURA DO MODELO COMPUTACIONAL

O modelo computacional foi feito de acordo com a estrutura de simulação de eventos discretos e foram utilizados o estado do sistema, o relógio da simulação, a lista de eventos e a rotina de tempo do pacote Simpy. As partes desenvolvidas foram os contadores estatísticos, a rotina de inicialização, a rotina de evento, a rotina de biblioteca, o gerador de relatórios e o programa principal.

4.2.1 Programa principal

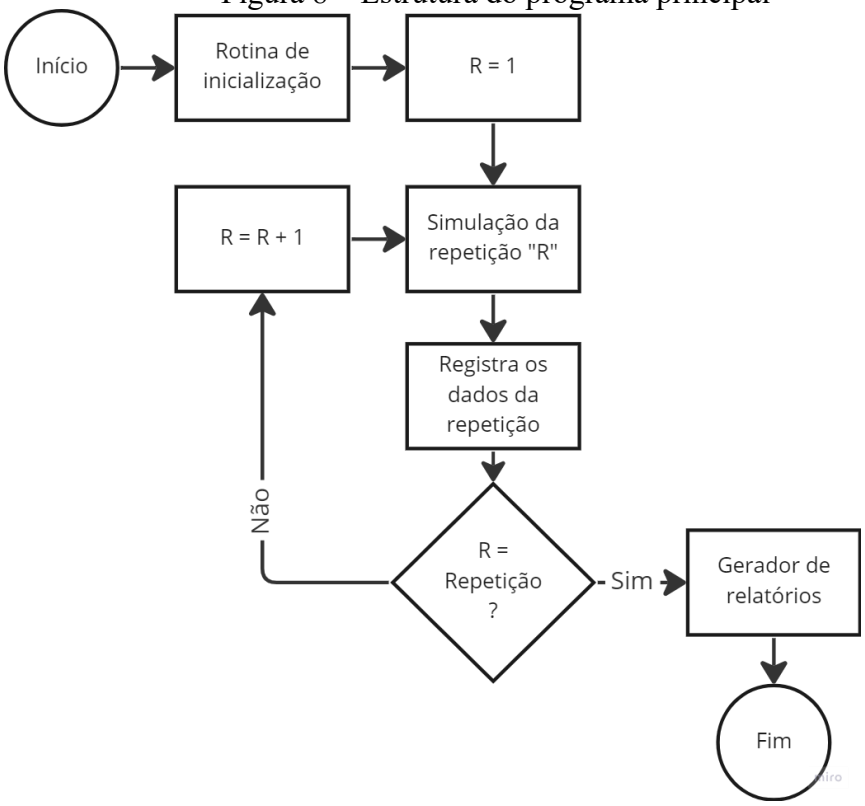
A estrutura do programa principal pode ser vista na Figura 8, onde o programa principal começa chamando a rotina de inicialização do sistema, e inicializando a estrutura de dados para armazenamento temporário dos indicadores de interesse de cada replicação.

Com isso, a quantidade de repetições obtida na rotina de inicialização é utilizada para gerar um laço com a quantidade de repetições desejadas, que realiza a operação de simulação para um conjunto de valores gerados. Após a simulação de todas as repetições, a rotina de geração de relatórios é executada.

Entende-se que a simulação da repetição “R” executa as rotinas:

- Rotina de biblioteca
- Rotina de inicialização
- Rotina de evento.

Figura 8 – Estrutura do programa principal



Fonte: Elaborado pela autora

4.2.2 Rotina de biblioteca

A rotina de biblioteca gera variáveis aleatórias baseadas na distribuição desejada e seus parâmetros. Dentro das distribuições de probabilidade possíveis, foram escolhidas quatro para serem implementadas em todo o modelo.

As opções de geração são a partir de uma constante, distribuição uniforme, distribuição normal e distribuição triangular. A partir da distribuição e seus parâmetros, é sorteado um valor a ser retornado, ou retornado o valor constante, se esse for selecionado.

Na rotina de biblioteca foi também implementada a função de obter o valor médio de uma distribuição de probabilidade, onde as entradas são o tipo de função e os parâmetros, e têm como saída o valor médio da função.

4.2.3 Rotina de inicialização

A rotina de inicialização é chamada no começo do código principal e inicializa o sistema, coletando as informações preenchidas pelo usuário e sendo executada antes de qualquer simulação e apenas uma vez em todo o procedimento.

Antes que a primeira repetição seja feita, o sistema lê os dados da planilha de entrada, preenche e manipula os valores para definir a condição inicial do sistema.

São lidas primeiro as informações mais básicas do sistema: quantidade de repetições, a data e horário de início e final da simulação.

Com as datas de início e final, é calculada a duração da simulação, visto que o relógio do sistema começa com tempo zero. Após os parâmetros gerais da simulação serem definidos, o sistema define a distribuição de chegadas padrão para voos nacionais e internacionais, visto que esses são utilizados quando a distribuição específica do voo não é definida.

São definidas probabilidades de *check-in online e* de passageiro com bagagens para despachar para voos nacionais e internacionais, que serão utilizadas na simulação para todos os voos do mesmo tipo.

Após isso, são processadas as informações sobre os recursos, como os grupos de recursos, sua capacidade e distribuição de probabilidade de tempo de atendimento. No caso do balcão de atendimento para *check-in*, existe ainda a informação adicional da quantidade de balcões destinados a passageiros atrasados. Com isso, os recursos são internamente inicializados com as informações dadas.

Nesse momento, os dados de cada voo são lidos, sendo as informações de identificador do voo, companhia aérea, data e horário do voo, sua categoria, a quantidade de passageiros, a curva de probabilidade da chegada dos passageiros (podendo ser deixada em vazio para usar curva padrão) e os grupos de cada recurso utilizados naquele voo definidos no sistema.

A última parte a ser inicializada é a referente às distâncias entre estações, onde é definido o processo e o nome de grupo da origem e sua distância em relação a cada possível destino.

Entre repetições, o Simpy possui uma rotina de inicialização própria para cada repetição, fazendo as tarefas de resetar o tempo inicial, inicializar o sistema e a lista de eventos.

4.2.4 Rotina de evento

Foram definidas quatro funções padrão para a rotina de eventos, que envolvem a caminhada de um ponto ao outro do aeroporto, à espera da abertura de processos, a escolha da fila e o processamento do passageiro

A função de caminhada no aeroporto recebe o local atual e o destino do passageiro, busca essas localizações nos dados de entrada e calcula o tempo esperado de caminhada. Foi

considerado que cada passageiro tem velocidade de 1,0m/s. Após essa passagem de tempo, o local do passageiro é atualizado para seu local de destino.

A função de espera de abertura recebe o processo solicitado pelo passageiro, o tempo de simulação e o tempo de abertura do processo e checa se o processo já está aberto, caso esteja, o passageiro prossegue normalmente, caso não esteja, o passageiro espera até que o processo abra para prosseguir.

A função de escolha de fila é processada de forma que recebe informações sobre quais são as opções de processamento e retorna a opção com o menor valor esperado.

Para o cálculo do valor esperado em fila, é feita a projeção para o autoatendimento e o balcão de atendimento para todos os passageiros que não fizeram *check-in online*, onde na projeção é considerado o tempo médio de cada processo e a quantidade de pessoas na fila e a quantidade de máquinas de autoatendimento, como descrito pela Equação 1.

$$Ta = \frac{Fa}{Qa}.Tma \quad (1)$$

Em que:

- Ta: tempo estimado de espera na máquina de autoatendimento;
- Fa: quantidade de passageiros na fila da máquina de autoatendimento;
- Qa: quantidade de máquinas de autoatendimento; e
- Tma: duração média do processo na máquina de autoatendimento.

Para o balcão de atendimento, a projeção de tempo na fila é feita de forma análoga, utilizando o tempo médio do processo, a quantidade de pessoas na fila e a quantidade de balcões abertos, como descrito na Equação 2.

$$Tb = \frac{Fb}{Qb}.Tmb \quad (2)$$

Em que:

- Tb: tempo estimado de espera no balcão;
- Fb: quantidade de passageiros na fila do balcão;
- Qb: quantidade de balcões abertos; e
- Tmb: duração média do processo no balcão.

Caso o passageiro possua bagagem, é calculado o tempo estimado de espera para despacho, que utiliza o tempo médio do processo, a quantidade de pessoas na fila e a quantidade de estações abertas, como descrito na Equação 3.

$$Td = \frac{Fd}{Qd}.Tmd \quad (3)$$

Em que:

- Td: tempo estimado de espera na estação de despacho;
- Fd: quantidade de passageiros na fila da estação de despacho;
- Qd: quantidade de estações de despacho; e
- Tmd: duração média do processo na estação de despacho.

Finalmente, é adicionado um erro de estimativa de 10% utilizando de uma distribuição de probabilidade uniforme, e assim é comparado qual processamento teria o menor tempo de espera de acordo com a estimativa do passageiro, utilizando apenas o balcão ou o autoatendimento e, caso o passageiro tenha bagagem, a estação de despacho.

A função de processamento recebe as informações do passageiro e o processamento a ser realizado.

Caso o processamento seja de *check-in*, a função começa um contador para o momento de abertura do *check-in* prioritário e checa se o balcão de *check-in* prioritário está aberto para o voo do passageiro. Caso ainda não esteja aberto, o passageiro entra na fila do processo escolhido pela função de escolha de fila. Caso o passageiro seja atendido antes da abertura do *check-in* prioritário, o passageiro permanece na fila de espera até ser atendido. Caso a abertura do *check-in* prioritário ocorra antes do processamento do passageiro, é checado se o passageiro está no autoatendimento ou balcão, e é definido o tempo de deslocamento até o balcão de *check-in* prioritário utilizando a função de caminhada e o passageiro entra na fila e espera seu processamento.

Caso o balcão de check-*in* prioritário esteja aberto no momento de chegada do passageiro, este se desloca diretamente a esse e entra na fila para processamento e é processado.

Caso o processamento não seja de *check-in*, é recebido o processo e o grupo e o passageiro entra na fila e espera seu processamento.

Após a espera, o passageiro é processado utilizando do tempo de atendimento gerado pela rotina de biblioteca, que usa a distribuição de probabilidade de tempos do respectivo processo e grupo.

4.2.5 Estrutura de repetição

Inicialmente são gerados os passageiros e definidos seus momentos de chegada, utilizando uma função de atraso não relacionada a nenhum recurso para garantir que o passageiro chegará no momento determinado.

Para cada passageiro, é realizada a mesma estrutura de processos, que começa com o sorteio baseado nas probabilidades definidas pelo usuário em relação ao *check-in online* e à presença ou não de bagagens.

A Figura 9 ilustra a estrutura de processos à qual cada passageiro está sujeito. Após a entrada do passageiro no sistema, é checado o tempo de simulação e o tempo de fechamento do *check-in*. Caso esse tenha sido encerrado, o passageiro não é processado e é considerado como atrasado. Caso esse ainda não tenha encerrado, o passageiro continua o fluxo.

Nesse ponto, é checado se o passageiro realizou *check-in online*. Caso tenha feito e tenha bagagem, ele irá caminhar até o processo de despacho e irá fazer o despacho da bagagem. Após o despacho, irá caminhar até a inspeção de segurança e realizá-la. Ao final, se o seu voo for internacional, irá caminhar até a estação de checagem de documentos e ser processado.

Caso o passageiro tenha feito *check-in online* e não tenha bagagem, ele irá caminhar diretamente até a inspeção de segurança e realizá-la e, ao final, se o seu voo for internacional, irá caminhar até a estação de checagem de documentos e ser processado.

Caso o passageiro não tenha feito *check-in online*, o passageiro espera até o *check-in* estar aberto e é verificado se o balcão prioritário está aberto para o voo. Caso o balcão prioritário esteja aberto, o passageiro caminha até o mesmo e é processado por ele. Após isso, o passageiro caminha até a inspeção de segurança, a realiza e, se o voo for internacional, caminha até a estação de checagem de documentos e é processado.

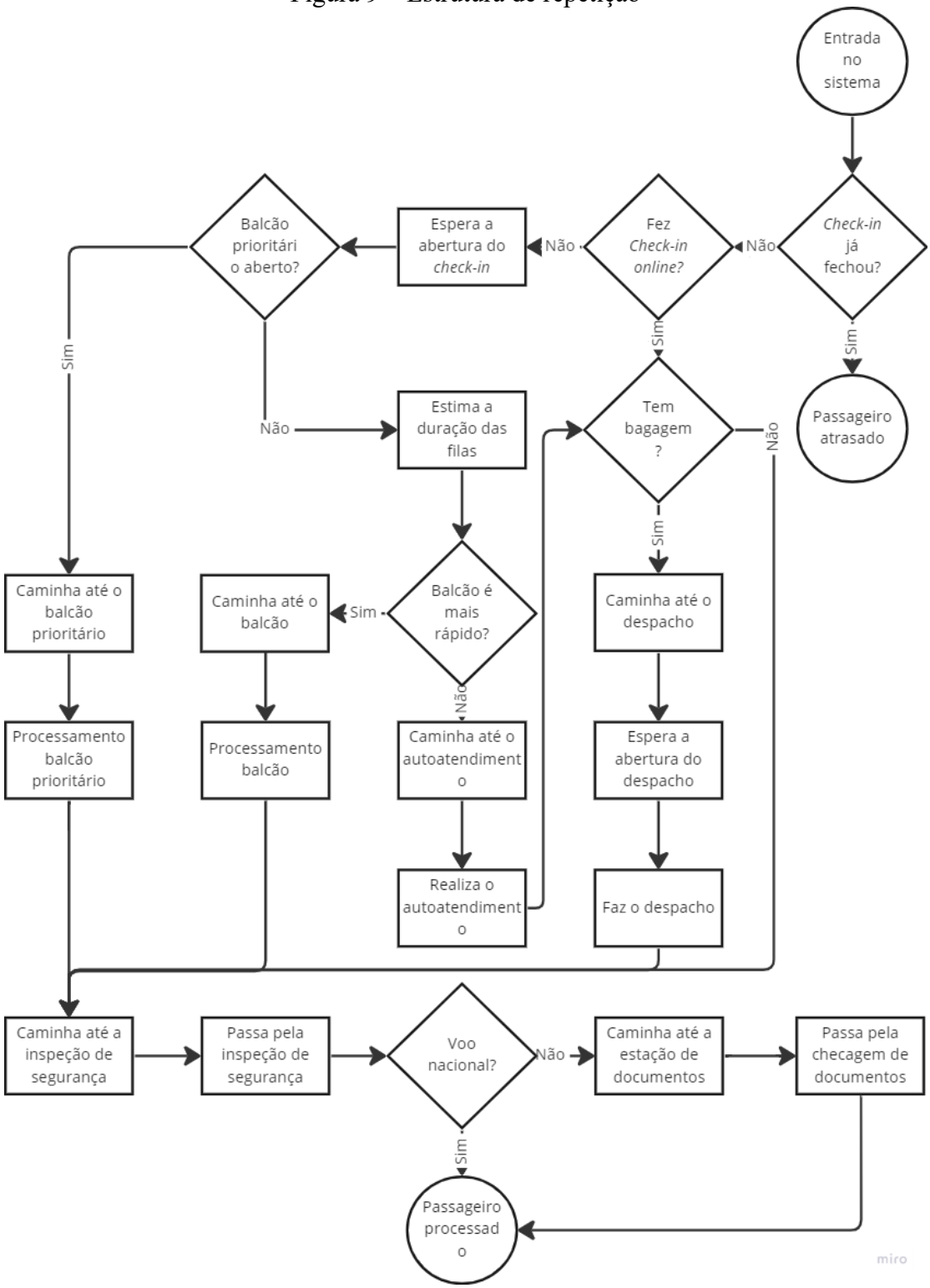
Caso o balcão prioritário não esteja aberto, é estimada a duração de espera na fila para o balcão e o autoatendimento e o despacho (caso o passageiro tenha bagagem). É definido então o caminho que o passageiro irá seguir.

Caso o tempo esperado no balcão seja menor, o passageiro irá caminhar até o balcão, ser processado no balcão, caminhar até a inspeção de segurança e realizá-la e, ao final, se o voo for internacional, irá caminhar até a estação de checagem de documentos e ser processado.

Caso o tempo esperado no balcão seja maior, o passageiro irá caminhar até o autoatendimento e ser processado. Caso tenha bagagens, ele irá caminhar até o despacho e realizá-lo, depois irá caminhar até a inspeção de segurança e realizá-la, ao final, se o voo for internacional, irá caminhar até a estação de checagem de documentos e ser processado.

Além desse fluxo, caso o passageiro esteja na fila do balcão ou autoatendimento quando o balcão prioritário abrir, ele irá se deslocar até esse e ser processado no balcão prioritário.

Figura 9 – Estrutura de repetição



Fonte: Elaborado pela autora

4.2.6 Geração de relatórios

Após cada iteração, são armazenados os indicadores daquela repetição, junto com o identificador da repetição. Esses indicadores são divididos entre indicadores do passageiro e dos recursos.

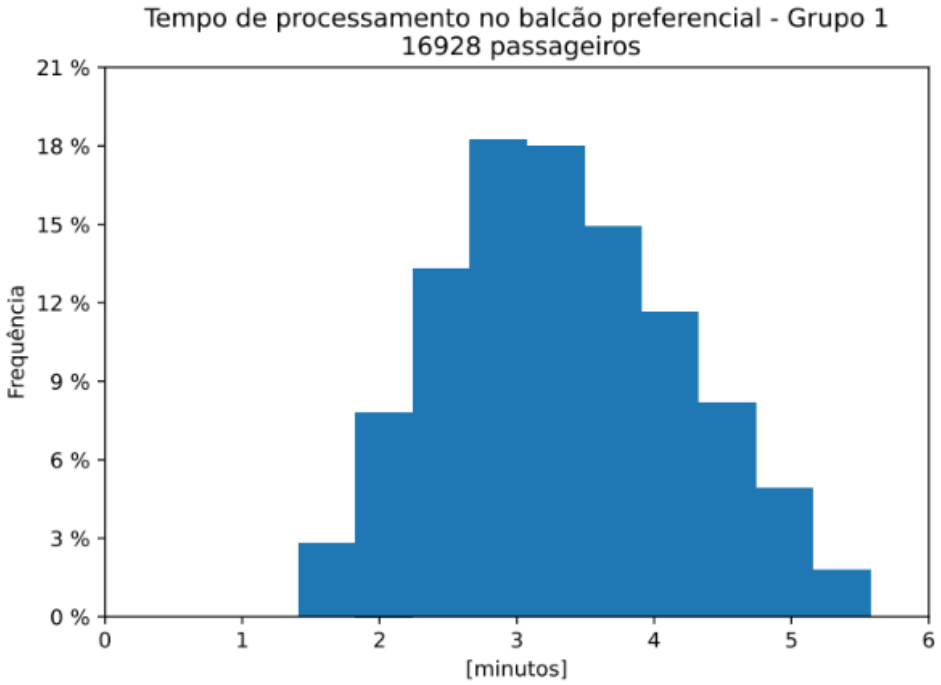
O gerador de relatórios realiza três processamentos distintos, e produz um total de 4 tipos de gráficos e análises estatísticas.

O primeiro processamento faz uma análise por estação, onde os valores de interesse são computados para cada processo e grupo correspondente, gerando uma estrutura de dados com os tempos em fila e de processamento de cada passageiro que passou por aquela estação.

Assim, são feitas análises para obter o valor máximo, mínimo, a média e a mediana de tempo de espera em fila entre todas as replicações, além da média do máximo e mínimo de cada replicação.

Também utilizando esses dados, são gerados 4 gráficos para cada estação. Primeiro são feitos histogramas do tempo de processamento e tempo de fila, como ilustrado pela Figura 10.

Figura 10 – Histograma do tempo de processamento em uma estação

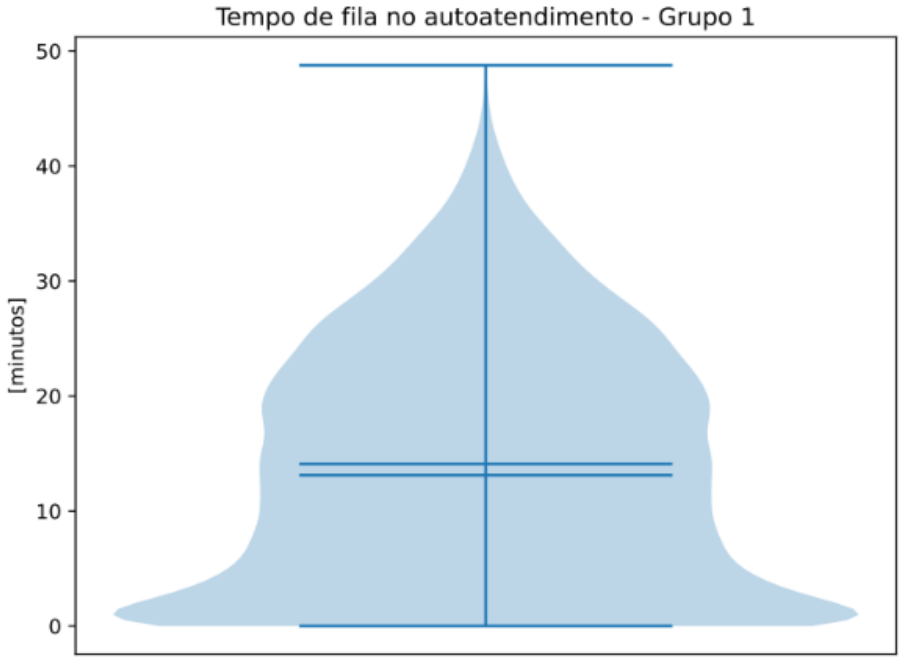


Fonte: Elaborado pela autora

Os outros dois gráficos são gráficos de violino do tempo de processamento e tempo de fila, como ilustrado pela Figura 11.

O gráfico de violino apresenta as características de um box plot, porém apresenta também uma curva de densidade de probabilidade. No gráfico de violino podem ser vistos os valores mínimo e máximo, a média e a mediana, que são marcados por um traço.

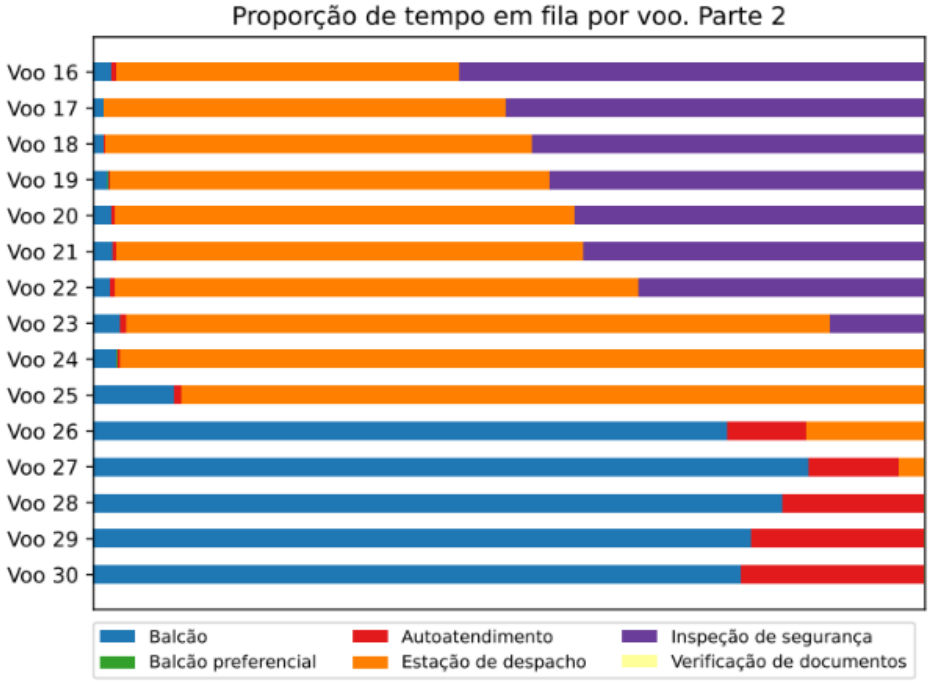
Figura 11 – Gráfico de violino do tempo de atendimento em uma estação



Fonte: Elaborado pela autora

O segundo processamento consiste em agregar os dados por voo, onde é calculado, para todos os passageiros de cada voo, o tempo total gasto em fila e o tempo total em cada processo. É gerado um gráfico mostrando, para cada voo e para o aeroporto no geral, a distribuição proporcional do tempo gasto em filas de cada processo em relação ao tempo total em fila, como pode ser visto na Figura 12. Os gráficos foram divididos de forma a comportar no máximo 15 comparações por figura. Caso esse valor seja excedido, será criada uma figura com os resultados dos próximos voos, e assim por diante.

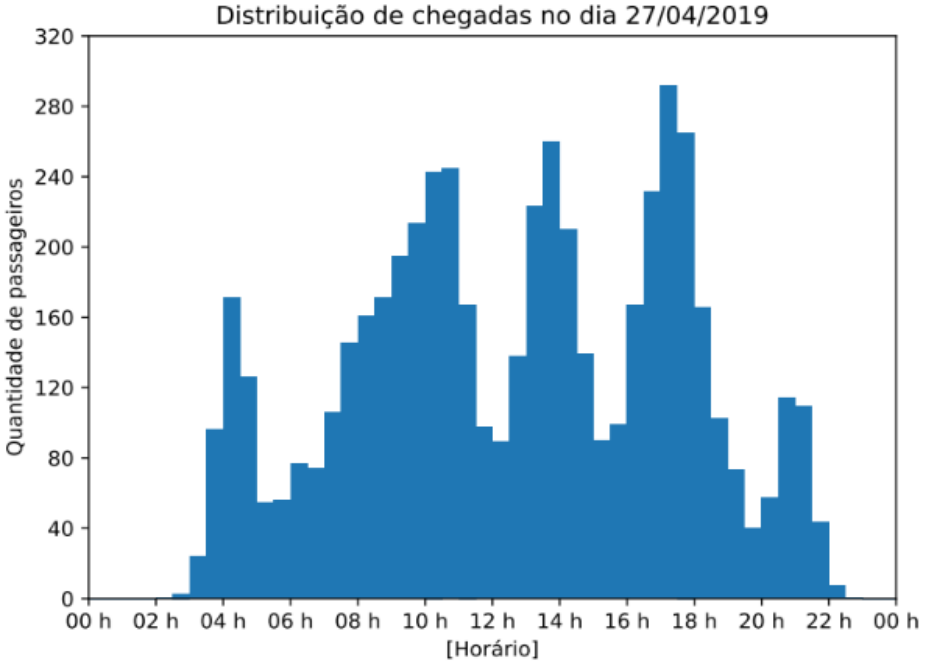
Figura 12 – Gráfico de proporção de tempo em fila em cada voo



Fonte: Elaborado pela autora

O último processamento consiste na checagem da distribuição de chegadas de todos os passageiros em relação ao tempo. As informações são agrupadas por dia de simulação e apresenta-se o número médio de chegadas de passageiros ao longo do tempo, como ilustrado na Figura 13.

Figura 13 – Gráfico de distribuição de chegadas em determinado dia



Fonte: Elaborado pela autora

Os gráficos são salvos em formato pdf na pasta de saídas com nomes gerados automaticamente pela nomeação dos processos, grupos, voos e dias da simulação.

4.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Considerando os métodos e conceitos expostos ao longo deste capítulo, o modelo de simulação proposto foi implementado de maneira concreta em linguagem Python, utilizando a biblioteca Simpy para facilitar o uso do paradigma de simulação orientada a eventos, e a biblioteca Pandas para auxiliar na coleta de estatísticas. O sistema foi executado utilizando dados provenientes de Villela, Dávalos e Cavaco (2019), obtendo-se resultados consistentes. A análise dos mesmos foge do escopo deste trabalho, mas cabe observar que o tempo de execução foi significativamente menor do que no trabalho citado, no qual a implementação havia sido feita usando o software comercial Rockwell Arena. O código completo é apresentado nos apêndices A-H e foi disponibilizado em repositório online.

5 CONCLUSÕES E RECOMENDAÇÕES

Este capítulo apresenta as conclusões e recomendações do trabalho.

5.1 CONCLUSÕES

Neste trabalho foi realizado um estudo de modelagem e simulação de terminais aeroportuários, levando em conta as particularidades do padrão de chegada de passageiros tipicamente observado, e as características das filas e abertura de processos. Foi implementado um modelo no qual os tempos de chegada dos passageiros são sorteados com base num adiantamento em relação à partida do voo.

O modelo implementado é flexível, utiliza apenas ferramentas de uso livre, permite a inserção de dados através de planilhas, e exporta os resultados na forma de gráficos, utilizando os principais indicadores, conforme recomendado pela IATA (2014). Assim, o sistema pode ser facilmente utilizado por pessoas sem conhecimento profundo do modelo ou de simuladores ou linguagens de programação específicas.

O sistema foi testado com dados utilizados em estudos anteriores, obtendo-se resultados compatíveis com Villela, Dávalos e Cavaco (2019).

Benefícios do modelo desenvolvido incluem:

- Permite a distinção de filas e processos aos quais cada passageiro é alocado dependendo do voo do passageiro.
- Permite inserção de dados e obtenção de resultados sem necessidade de alteração ou inspeção de código.
- Permite uma fácil configuração das características do aeroporto.
- Permite um grande detalhamento das características de cada voo.
- Permite uma grande flexibilidade.
- Utiliza apenas ferramentas de uso livre.

5.2 RECOMENDAÇÕES FUTURAS

Embora o modelo proposto represente um avanço do ponto de vista de detalhamento na simulação de terminais aeroportuários, ainda há espaço para melhorias que tornem o modelo mais realista, como:

- Melhorar a lógica para escolha de filas.
- Incluir mais escolhas possíveis de distribuições de probabilidade.
- Calcular estatísticas de saída adicionais.

Além disso, a ferramenta pode ser aprimorada com a inclusão de interfaces gráficas, a validação do modelo utilizando um sistema real e sua comparação com um modelo equivalente utilizando outros métodos de chegada de passageiros.

REFERÊNCIAS

ALMEIDA, Paulo Marcos Santo de. **Utilização de simulação na análise de componentes de terminais de passageiros de aeroportos brasileiros**. 1998. 1 v. Dissertação (Mestrado) - Curso de Engenharia, Instituto Tecnológico de Aeronáutica, São José dos Campos, 1998.

ALODHAIBI, Sultan; BURDETT, Robert L.; YARLAGADDAA, Prasad Kdv.. Framework for airport outbound passenger flow modelling. **Procedia Engineering**. Zhengzhou, p. 1100-1109. nov. 2017. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877705817302631. Acesso em: 10 out. 2022.

AL-SULTAN, Ahmad T.. Simulation and Optimization for Modeling the Passengers Check-in System at Airport Terminal. **Review Of Integrative Business And Economics Research**. [S. L.], p. 44-53. jan. 2018.

AMAR, Jacques. The Monte Carlo method in science and engineering. **Computing In Science And Engineering**. [S. L.], p. 9-19. mar. 2006.

ANAC - Agência Nacional de Aviação Civil. **Painel de Indicadores do Transporte Aéreo 2009**. [S. L.]. 2021. Disponível em: https://www.gov.br/anac/pt-br/assuntos/dados-e-estatisticas/mercado-do-transporte-aereo/painel-de-indicadores-do-transporte-aereo/painel-de-indicadores-do-transporte-aereo-2009 . Acesso em: 06 de julho de 2022.

ANAC - Agência Nacional de Aviação Civil. **Painel de Indicadores do Transporte Aéreo 2019**. [S. L.]. 2021. Disponível em: https://www.gov.br/anac/pt-br/assuntos/dados-e-estatisticas/mercado-do-transporte-aereo/painel-de-indicadores-do-transporte-aereo/painel-de-indicadores-do-transporte-aereo-2019 . Acesso em: 06 de julho de 2022.

ANAC - Agência Nacional de Aviação Civil. **Painel de Indicadores do Transporte Aéreo 2020**. [S. L.]. 2021. Disponível em: https://www.gov.br/anac/pt-br/assuntos/dados-e-estatisticas/mercado-do-transporte-aereo/painel-de-indicadores-do-transporte-aereo/painel-de-indicadores-do-transporte-aereo-2020 . Acesso em: 06 de julho de 2022.

BANKS, Jerry et al. **Discrete event system simulation**. New York: Pearson, 2004.

BARROS, Alexandre G. de; TOMBER, David D.. Quantitative analysis of passenger and baggage security screening at airports. **Journal Of Advanced Transportation**. [S. L.], p. 171-193. jan. 2007.

CORREIA, Anderson; WIRASINGHE, S. C.. Evaluating Level of Service at Airport Passenger Terminals: review of research approaches. **Transportation Research Record**. [S. L.], p. 1-6. jan. 2004.

DAGKAKIS, G; HEAVEY, Cathal. A review of open source discrete event simulation software for operations research. **Journal Of Simulation**. [S. L.], p. 193-206. ago. 2016.

DAGKAKIS, Georgios et al. ManPy: an open-source layer of des manufacturing objects implemented in simpy. In: EUROSIM, 8., 2013, Cardiff. **Proceedings [...]** . [S. L.]: Ieee, 2013. p. 357-363.

FISHMAN, G. S. **Discrete-event simulation**: modeling, programming, and analysis. New York: Springer, 2001.

GATERSLEBEN, Michel R; WEIJ, Simon W. van Der. Analysis and simulation of passenger flows in an airport terminal. In: CONFERENCE ON WINTER SIMULATION: SIMULATION---A BRIDGE TO THE FUTURE, 31., 1999, Phoenix. **Proceedings** [...] . Phoenix: Ieee, 1999. p. 1226-1231.

GKRITZA, Konstantina; NIEMEIER, Debbie; MANNERING, Fred. Airport security screening and changing passenger satisfaction: an exploratory assessment. **Journal Of Air Transport Management**. [S. L.], p. 213-219. maio 2006.

GUERRERO, Ruddy; SERRANO-HERNANDEZ, Adrian; PASCUAL, Jose; FAULIN, Javier. Simulation Model for Wire Harness Design in the Car Production Line Optimization Using the SimPy Library. **Sustainability**, [S.L.], v. 14, n. 12, p. 7212, 13 jun. 2022. MDPI AG. http://dx.doi.org/10.3390/su14127212.

HORONJEFF, Robert et al. **Planning and design of airports**. McGraw-Hill Education, 2010.

IATA - International Air Transport Association. **Travel Recovery Rebuilding Airline Profitability - Resilient Industry Cuts Losses to $9.7 billion**. [S. L.]. 2022. Disponível em: https://www.iata.org/en/pressroom/2022-releases/2022-06-20-02/ . Acesso em: 06 de julho de 2022.

IATA - International Air Transport Association. **Airport Development Reference Manual**. Montreal-Geneva. 2014

JOUSTRA, Paul E.; VAN DIJK, Nico M.. Simulation of check-in at airports. In: CONFERENCE ON WINTER SIMULATION, 33., 2001, Arlington. **Proceedings [...]** . Arlington: Ieee, 2001. p. 1023-1028.

KING, D; HARRISON, H. Discrete-event simulation in Java: a practitioner's experience. In: CONFERENCE ON GRAND CHALLENGES IN MODELING & SIMULATION, 33., 2010, Ottawa. **Proceedings [...]** . Vista: Ieee, 2010. p. 436-447.

KROESE, D. P. et al. Why the Monte Carlo method is so important today. **Wiley Interdisciplinary Reviews: Computational Statistics**. [S. L.], p. 386-392. jun. 2014.

LANDRUM & BROWN. **Airport Passenger Terminal Planning and Design**. Washington: Transportation Research Board, 2010.

LAW, A.M.; KELTON, W.D. . **Simulation Modeling and Analysis**. New York: McGraw-Hill, 1991.

MAIA, M. C. et al. Cenários alternativos para melhoria do nível de serviço no check-in de importantes aeroportos Brasileiros. In: SITRAER, 2010, Manaus. **Simpósio**. [S. L.]: Sitraer, 2010. p. 642-654.

MANATAKI, Ioanna E.; ZOGRAFOS, Konstantinos G.. A simulation model for airport terminal planning and operational performance assessment. In: INTERNATIONAL CONGRESS ON TRANSPORTATION RESEARCH, 3., 2006, Thessaloniki. **Proceedings [...]** . [S. L.]: International Congress On Transportation Research, 2006. p. 432-441.

MANATAKI, Ioanna E.; ZOGRAFOS, Konstantinos G.. A generic system dynamics based tool for airport terminal performance analysis. **Transportation Research**. [S. L.], p. 428-443. fev. 2009.

MARCOS, Antonio Rodolfo Araujo; FERREIRA, Luciano. Um modelo de simulação para gestão da capacidade dos aeroportos brasileiros. **Revista Eletrônica de Administração**, [S. L.], v. 21, n. 1, p. 1-26, abr. 2015.

MATLOFF, Norm. **Introduction to Discrete-Event Simulation and the SimPy Language**. [S. L.]: N.s. Matloff, 2008.

ORTÚZAR, Juan de Dios; WILLUMSEN, Luis G. **Modelling transport**. John wiley & sons, 2011.

PALMER, Geraint I. et al. Ciw: an open-source discrete event simulation library. **Journal Of Simulation**. [S. L.], p. 68-82. out. 2019.

PEGDEN, C.D.; SHANNON, R.e.; SADOWSKI, R.P.. **Introduction to Simulation Using SIMAN**. New York: McGraw-Hill, 1990.

POSTORINO, M. N. et al. Airport passenger arrival process: estimation of earliness arrival functions. **Transportation Research Procedia**. [S. L.], p. 338-345. 2019.

RAYCHAUDHURI, S.. Introduction to monte carlo simulation. In: WINTER SIMULATION CONFERENCE, 40., 2008, Miami. **Proceedings [...]** . [S.L.]: Ieee, 2008. p. 91-100.

ROBERTSON, Craig V. et al. The role of modeling demand in process re-engineering.In: WINTER SIMULATION CONFERENCE, 34., 2002, San Diego. **Proceedings [...]** . [S.L.]: Ieee, 2002. p. 1454-1458.

ROLIM, Paula Sutherland Wallauer. **Fatores determinantes no nível de serviço oferecido no check-in de voos internacionais**. 2015. Dissertação (Mestrado) - Curso de Engenharia de Infraestrutura Aeronáutica, Instituto Tecnológico de Aeronáutica, São José dos Campos, 2015.

SALIBY, E.. **Repensando a Simulação**: a amostragem descritiva. [S. L.]: Ed. Atlas, 1989.

SANKARANARAYANAN, H. B. Agarwal. V. Rathod: an exploratory data analysis of airport wait times using big data visualisation techniques. In: INTERNATIONAL CONFERENCE ON COMPUTATION SYSTEM AND INFORMATION TECHNOLOGY

FOR SUSTAINABLE SOLUTIONS, 1., 2016, Bangalore. **Proceedings [...]** . [S. L.]: Ieee, 2016. p. 324-329.

SHANNON, R. E. Introduction to the art and science of simulation. In: WINTER SIMULATION CONFERENCE, 30., 1998,Washington. **Proceedings [...]** . [S.L.]: Ieee, 1998. p. 7-14.

STOLLETZ, R.. Analysis of passenger queues at airport terminals. **Research In Transportation Business & Management**. [S. L.], p. 144-149. ago. 2011.

THIEBAUT, Alexandre Angelo. **Análise das operações aeroportuárias sob as óticas de demanda e oferta de serviços ao passageiro**: estudo de caso aeroporto internacional de guarulhos. 2013. 130 f. TCC (Graduação) - Curso de Engenharia de Produção, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013.

VILLELA, Maria Giuliana Occhioni Martins; DÁVALOS, Ricardo Villarroel; CAVACO, Marco Antonio Martins. Modelagem e simulação do embarque de passageiros em terminais aeroportuários. In: CONGRESSO DE PESQUISA E ENSINO EM TRANSPORTES, 33., 2019, Balneário Camboriú. **Anais [...]** . [S. L.]: Associação Nacional de Pesquisa e Ensino em Transportes, 2019. p. 2614-2623.

WIEDEMANN, T. Next generation simulation environments founded on open source software and XML-based standard interfaces.In: WINTER SIMULATION CONFERENCE, 34., 2002, San Diego. **Proceedings [...]** . [S.L.]: Ieee, 2002. p. 623–628.

WITELSKI, Thomas; BOWEN, Mark. **Methods of mathematical modeling**: continuous systems and differential equations. [S. L.]: Springer, 2020.

APÊNDICE A – CÓDIGO PRINCIPAL

```
# coding: utf-8

from config.config import Configs
from airport_sim import Airport_Simulation
from analysis.data_analysis import analyze_data

import pandas as pd
import logging

logging.basicConfig(level=logging.INFO)
logging.getLogger("matplotlib").setLevel(logging.ERROR)
logging.getLogger("PIL").setLevel(logging.ERROR)

logging.info(f"Processando dados de entrada")
config = Configs('config.xlsx')

df_pas = pd.DataFrame()
df_res = pd.DataFrame()

for rep_id in range(1, config.num_reps+1):
    logging.info(f"Executando replicação {rep_id} de {config.num_reps}")
    my_simulation = Airport_Simulation(config, rep_id=rep_id)
    my_simulation.run()
    df_pas_part, df_res_part = my_simulation.export_data()
    df_pas = pd.concat([df_pas, df_pas_part], ignore_index=True)
    df_res = pd.concat([df_res, df_res_part], ignore_index=True)

logging.info(f"Simulações finalizadas. Processando resultados")
analyze_data(df_res, df_pas, config=config)
logging.info(f"Processamento concluído")
```

APÊNDICE B – CÓDIGO DE CONFIGURAÇÕES

```
# coding: utf-8

import os
import pandas as pd
from datetime import datetime
import warnings
from numpy import isnan

class Configs:
    RESOURCE_MAP = {
        'cheg': 'Chegada',
        'balc': 'Balcões',
        'auto': 'Autoatendimento',
        'balc_atraso': 'Balc Atraso',
        'desp': 'Despacho',
        'seg':  'Segurança',
        'doc':  'Documentos'
    }
    REVERSE_RESOURCE_MAP = {v: k for k, v in RESOURCE_MAP.items()}

    def __init__(self, config_filename=None):
        self.resources = {r: [] for r in Configs.RESOURCE_MAP if r != 'cheg'}
        self.flights = {}
        self.distances = {}
        self.load_configs(config_filename)

    def load_configs(self, filename=None):
        if filename is None:
            filename = 'config.xlsx'

        filename = 'config.xlsx'
        config_file = os.path.join(os.path.realpath(__file__), '..', filename)
```

```
        # Lê tempos
        warnings.simplefilter(action='ignore', category=UserWarning)
        main_configs = pd.read_excel(config_file)
        self.num_reps = main_configs.iloc[1,1]
        self.sim_start = datetime.combine(main_configs.iloc[2,1].date(), main_configs.iloc[3,1])
        self.sim_end = datetime.combine(main_configs.iloc[4,1].date(), main_configs.iloc[5,1])
        self.duration = (self.sim_end - self.sim_start).total_seconds()
        self.param_dist_nac = [
            main_configs.iloc[8, 1],
            main_configs.iloc[9, 1]*60,
            main_configs.iloc[10, 1]*60,
            main_configs.iloc[11, 1]*60,
        ]
        self.param_dist_int = [
            main_configs.iloc[16, 1],
            main_configs.iloc[17, 1]*60,
            main_configs.iloc[18, 1]*60,
            main_configs.iloc[19, 1]*60,
        ]
        self.prob_online_checkin_nac = main_configs.iloc[12, 1]
        self.prob_online_checkin_int = main_configs.iloc[20, 1]
        self.prob_luggage_nat = main_configs.iloc[13, 1]
        self.prob_luggage_int = main_configs.iloc[21, 1]

        # Lê informações de recursos
        for res_name, res_label in Configs.RESOURCE_MAP.items():
            if res_name  in ['cheg', 'balc_atraso']:
                continue
            df1 = pd.read_excel(config_file,
sheet_name=res_label,converters={'Grupo':str,'Capacidade':int})
            df = df1.drop(df1.columns[[-2, -4, -6]], axis=1)
            df.columns.values[-1] = "par_3"
            df.columns.values[-2] = "par_2"
```

```
            df.columns.values[-3] = "par_1"
            df.columns.values[-4] = "Distribuição"
            res_name = str(res_name)
            for _, row in df.iterrows():
                dict_row = row.to_dict()
                self.resources[res_name].append(dict_row)
                if res_name == 'balc':
                    # Balcão de atraso usa mesma configuração do balcão, porém com capacidade = 1
                    dict_row['Capacidade'] = dict_row['Capacidade balcões de atraso']
                    self.resources['balc_atraso'].append(dict_row)

        # Lê informações de voo
        flights_t = pd.read_excel(config_file, sheet_name='Voos')
        flights_p =  flights_t.drop(flights_t.columns[[-2, -4, -6]], axis=1)
        flights = flights_p.drop([0])
        flights.columns = ['Voo', 'Companhia','Data', 'Horário', 'categoria' , 'Quantidade de Passageiros', 'Grupo de balcões', 'Grupo de auto atendimento','Grupo de despacho' , 'Grupo de segurança', 'Grupo de documentos', 'Abertura checkin', 'Fechamento checkin', 'Abertura Segurança','Prioridade checkin', 'Distribuição chegadas', 'par 1', 'par 2', 'par 3']
        for _, row in flights.iterrows():
            flight_config = row.to_dict()
            flight_config["par 1"] = 60 * flight_config["par 1"]
            flight_config["par 2"] = 60 * flight_config["par 2"]
            flight_config["par 3"] = 60 * flight_config["par 3"]
            flight_config['partida'] = datetime.combine(flight_config['Data'].date(), flight_config['Horário'])
            self.flights[flight_config['Voo']] = flight_config


        # Lê distâncias
        conversoes = {'Entrada': str, 'Saída': str,
                      'Grupo entrada': str, 'Grupo saída': str, 'Distância [m]': float}
        distances = pd.read_excel(config_file, sheet_name='Deslocamento', converters=conversoes)
```

```
for _, row in distances.iterrows():
    grupo_entrada = row['Grupo entrada']
    if not isinstance(grupo_entrada, str) and isnan(grupo_entrada):
        grupo_entrada = None
    grupo_saida = row['Grupo saída']
    if not isinstance(grupo_saida, str) and isnan(grupo_saida):
        grupo_saida = None
    origin = (self.REVERSE_RESOURCE_MAP[row['Entrada']], grupo_entrada)
    destination = (self.REVERSE_RESOURCE_MAP[row['Saída']], grupo_saida)
    dist = row['Distância [m]']
    self.distances[origin, destination] = dist
```

APÊNDICE C – CÓDIGO DO AEROPORTO

```
# coding: utf-8

import typing
import simpy
import numpy as np
import logging
import pandas as pd

from config.config import Configs
from componentes.voo import Voo
from componentes.passageiro import Pas
from componentes.distribuicoes import Distribuicao
from componentes.processo import Proc



class Airport_Simulation:
    """Classe que armazena dados referentes ao aeroporto"""
    teste = 1
    last_id = 0

    def __init__(self, config, rep_id=None, seed=None):
        # Inicialização
        self.rep_id = rep_id
        self.config = config
        self.env = simpy.Environment()

        if seed is not None:
            np.random.seed(seed=233423)

        self.dist_chegadas_nac = Distribuicao(*self.config.param_dist_nac)
        self.dist_chegadas_int = Distribuicao(*self.config.param_dist_int)
```

```
        self.processes = {}
        for res_name in self.config.resources.keys():
            self.processes[res_name] = {}
            for resource_group_config in self.config.resources[res_name]:
                res = Proc(self.env, resource_group_config, res_name)
                self.processes[res_name][res.group_name] = res

        self.flights = {}
        self.passageiros = {}
        for flight_params in self.config.flights.values():
            flight = Voo(flight_params, self.config.sim_start)
            self.flights[flight.name] = flight

            qtd_pass_voo = flight.qtd_pass
            arrivals = self.get_num_arrivals(qtd_pass_voo)

            for i in range(arrivals):
                pas = Pas(self.env, self, flight)
                self.passageiros[pas.id] = pas

    def get_id(self):
        self.last_id = self.last_id + 1
        return self.last_id

    def get_distancia(self, origin, destination):
        #Pega a distância entre dois locais
        try:
            return self.config.distances[(origin, destination)]
        except KeyError:
            logging.exception(f"par {origin}, {destination} não encontrado na planilha de
distâncias")
            raise KeyError
```

```
    def get_proc(self, proc_name: str, group_name: str):
        """
        Método que recebe uma especificação de tipo de grupo de recurso e retorna o recurso correspondente registrado
        no aeroporto
        :param proc_name: nome do processo
        :param group_name: nome do grupo
        :return: processo correspondente
        Pode gerar uma exceção se o recurso não for encontrado
        """
        try:
            procs = self.processes[proc_name]
            try:
                return procs[group_name]
            except KeyError:
                logging.exception(f"Grupo {group_name} não existente para o processo {procs}")
                raise KeyError(f"Grupo {group_name} não existente para o processo {procs}")
        except KeyError:
            logging.exception(f"Processo {proc_name} não encontrado")
            raise KeyError(f"Processo {proc_name} não encontrado")

    def get_num_arrivals(self, qtd):
        return qtd

    def run(self):
        self.env.run(until=self.config.duration)

    def export_data(self, print_on_console=False):

        pass_rows_list = []
        for pas in self.passageiros.values():
            row = pas.tempos
            row['pas_id'] = pas.id
            row['rep_id'] = self.rep_id
```

```
            row['flight'] = pas.flight.name

            for attr_name in ['balc', 'balc_atraso', 'auto', 'desp', 'seg', 'doc']:
                row[f'group_{attr_name}'] = getattr(pas.flight, attr_name)

            pass_rows_list.append(row)
        df_pas = pd.DataFrame(pass_rows_list)

        res_rows_list = []
        for res_name in self.processes:
            for group_name in self.processes[res_name]:
                proc = self.processes[res_name][group_name]
                process_data = proc.res.data
                for data_point in process_data:
                    row = data_point
                    row['rep_id'] = self.rep_id
                    row['process'] = res_name
                    row['process_group'] = group_name
                    res_rows_list.append(row)
        df_res = pd.DataFrame(res_rows_list)

        return df_pas, df_res
```

APÊNDICE D – CÓDIGO DO PASSAGEIRO

```
# coding: utf-8

import simpy
from numpy import random, nan
from scipy import stats
import logging



class Pas:
    atrasados = 0

    def __init__(self, env, airport_simulation, flight):
        self.id = airport_simulation.get_id()

        self.env = env
        self.air_sim = airport_simulation
        self.flight = flight

        self.tempos = {}
        for cat in ['processamento', 'fila']:
            for process_name in airport_simulation.config.RESOURCE_MAP.keys():
                key = f'{cat}_{process_name}'
                self.tempos[key] = nan

        self.walk_speed = 1.0
        self.bagagens = self.generate_luggage()
        self.checkInOnline = self.generate_onlineCheckin()
        self.tempos['chegada'] = self.generate_arrivalTime()

        self.action = self.env.process(self.run())
        self.despachou = False
```

```
        self.local = 'cheg'
        self.embarcou = False

    def run(self):
        # Chega no aeroporto
        yield self.env.timeout(self.tempos['chegada'])
        logging.debug(f"Passageiro {self.id} chegou as {self.env.now}, agendado para {self.tempos['chegada']} para o voo {self.flight.name}")

        if self.env.now > self.flight.fechamento_checkin:
            logging.debug(f"Passageiro {self.id} chegou atrasado. Não será processado")
            self.tempos['fim'] = self.env.now
            Pas.atrasados = Pas.atrasados +1
            return

        # Ponto de decisão: Checkin online?
        if self.checkInOnline:
            if self.bagagens > 0:
                yield self.env.process(self.caminha('desp'))
                yield self.env.process(self.espera_abertura('desp'))
                yield self.env.process(self.processa('desp'))
            else:
                logging.debug(f"passageiro {self.id} não tem bagagens")

        else:
            logging.debug(f"passageiro {self.id} não fez checkin online")
            # Decide qual tipo de checking fazer
            yield self.env.process(self.espera_abertura('auto'))
            if self.flight.prioridade_checkin < self.env.now:
                auto_atendimento = False
            else:
                estimativa_auto = (self.get_tempo_estimado_em_fila('auto', self.flight.auto))*random.uniform(0.9, 1.1)
```

```
            estimativa_balcao = (self.get_tempo_estimado_em_fila('balc',
self.flight.balc))*random.uniform(0.9, 1.1)
            if self.bagagens > 0:
                estimativa_despacho = self.get_tempo_estimado_em_fila('desp', self.flight.desp)
                estimativa_auto = estimativa_auto + estimativa_despacho
            if estimativa_balcao < estimativa_auto:
                auto_atendimento = False
            else:
                auto_atendimento = True

        if auto_atendimento:
            # Caminha até próximo processo
            yield self.env.process(self.caminha('auto'))
            yield self.env.process(self.espera_abertura('auto'))
            yield self.env.process(self.processa('auto'))
            if self.bagagens > 0 and not self.despachou:
                yield self.env.process(self.caminha('desp'))
                yield self.env.process(self.processa('desp'))
        else:
            yield self.env.process(self.caminha('balc'))
            yield self.env.process(self.espera_abertura('balc'))
            yield self.env.process(self.processa('balc'))

    # Inspeção de segurança
    yield self.env.process(self.caminha('seg'))
    yield self.env.process(self.espera_abertura('seg'))
    yield self.env.process(self.processa('seg'))

    # Checagem de documentos internacionais
    if self.flight.internacional:
        yield self.env.process(self.caminha('doc'))
        yield self.env.process(self.processa('doc'))

    self.tempos['fim'] = self.env.now
```

```
        self.embarcou = True
        logging.debug(f"Passageiro {self.id} terminou sua passagem pelo terminal as {self.env.now}")

    def get_walking_time(self, origin, destination):
        o = (self.local, self.flight.get_process_group(self.local))
        d = (destination, self.flight.get_process_group(destination))
        distance = self.air_sim.get_distancia(o, d)
        walking_time = distance / self.walk_speed
        return walking_time

    def espera_abertura(self, res):
        if res in ['balc', 'auto', 'desp']:
            if self.env.now < self.flight.abetura_checkin:
                diff = self.flight.abetura_checkin - self.env.now
                yield self.env.timeout(diff)
        elif res == 'seg':
            if self.env.now < self.flight.abetura_seg:
                diff = self.flight.abetura_seg - self.env.now
                yield self.env.timeout(diff)
        else:
            logging.error("Não deveria ser possível chegar aqui")

    def get_tempo_estimado_em_fila(self, res_name, group_name):
        proc = self.air_sim.get_proc(res_name, group_name)
        return proc.get_tempo_estimado_para_processar_fila()

    def processa(self, process_name):
        start = self.env.now

        group_name = self.flight.get_process_group(process_name)
        proc = self.air_sim.get_proc(process_name, group_name)

        if process_name in ['balc', 'auto']:
```

```
            group_atraso = self.flight.get_process_group('balc_atraso')
            proc_atraso = self.air_sim.get_proc('balc_atraso', group_atraso)
        else:
            proc_atraso = None

        processamento_no_balcao_de_prioridade = False
        with proc.res.request() as req:
            logging.debug(f"Passageiro {self.id} esperando acesso ao recurso {process_name} em {self.env.now}")

            if proc_atraso is None:
                results = yield req
            else:
                tempo_ate_abertura_de_balcao_prioritario = self.flight.prioridade_checkin - self.env.now
                if tempo_ate_abertura_de_balcao_prioritario > 0:
                    results = yield req | self.env.timeout(self.flight.prioridade_checkin - self.env.now)
                else:
                    results = []

            if results is None or req in results:
                # Fomos atendidos antes de abrir o balcão prioritário ou outro processo
                self.tempos[f'fila_{process_name}'] = self.env.now - start
                logging.debug(f"Passageiro {self.id}, conseguiu acesso ao recurso {process_name} em {self.env.now}")
                tempo = proc.sorteia_tempo_de_processamento()
                yield self.env.timeout(tempo)
                self.tempos[f'processamento_{process_name}'] = tempo
                logging.debug(f"Passageiro {self.id} foi processado por {process_name} em {self.env.now}")
            else:
                processamento_no_balcao_de_prioridade = True

        # O balcão de prioridade abriu antes de sermos atendidos
```

```
        if processamento_no_balcao_de_prioridade:
            start_prioridade = self.env.now
            self.tempos[f'fila_{process_name}'] = start_prioridade - start
            if process_name == 'auto':
                walking_time = self.get_walking_time('auto', 'balc')
                yield self.env.timeout(walking_time)
                self.local = 'balc'

            with proc_atraso.res.request() as req_balc_atraso:
                logging.debug(f"Passageiro {self.id} foi para a fila do balcão prioritário em {self.env.now}")

                yield req_balc_atraso

                self.tempos[f'fila_balc_atraso'] = self.env.now - start_prioridade

                logging.debug(f"Passageiro {self.id}, conseguiu acesso ao recurso balc_atraso em {self.env.now}")
                tempo = proc_atraso.sorteia_tempo_de_processamento()
                yield self.env.timeout(tempo)

                self.tempos[f'processamento_balc_atraso'] = tempo

                logging.debug(f"Passageiro {self.id} foi processado por balc_atraso em {self.env.now}")
                self.despachou = True

    def caminha(self, destino):
        walking_time = self.get_walking_time(self.local, destino)
        yield self.env.timeout(walking_time)
        self.local = destino

    def generate_companions(self):
        if self.flight.internacional:
```

```
        probs = self.air_sim.config.probs_acomp_int
    else:
        probs = self.air_sim.config.probs_acomp_nac

    # Calcula probabilidades acumuladas
    keys = sorted(list(probs.keys()))
    acumuladas = {k: 0  for k in keys}
    for k in keys:
        for i in acumuladas:
            if i >= k:
                acumuladas[i] = acumuladas[i] + probs[k]

    r = random.random()
    for num_accomp, prob in acumuladas.items():
        if r <= prob:
            return num_accomp

def generate_luggage(self):
    if self.flight.internacional:
        t = self.air_sim.config.prob_luggage_int
    else:
        t = self.air_sim.config.prob_luggage_nat

    r = random.random()
    if r <= t:
        return 1
    return 0

def generate_onlineCheckin(self):
    if self.flight.internacional:
        t = self.air_sim.config.prob_online_checkin_int
    else:
        t = self.air_sim.config.prob_online_checkin_nac
```

```
    r = random.random()
    if r <= t:
        return True
    return False

def generate_arrivalTime(self):
    if self.flight.dist_chegadas is not None:
        dist = self.flight.dist_chegadas
    else:
        if self.flight.internacional:
            dist = self.air_sim.dist_chegadas_int
        else:
            dist = self.air_sim.dist_chegadas_nac

    adiantamento = dist.sorteia()
    t = self.flight.partida - adiantamento
    return t
```

APÊNDICE E – CÓDIGO DO PROCESSO

```
# coding: utf-8

import simpy
from componentes.distribuicoes import Distribuicao



class Proc:
    def __init__(self, env, resource_group_config, res_name=None):
        self.env = env
        self.res_name = res_name
        self.group_name = resource_group_config.get("Grupo")

        dist_params = [
            resource_group_config.get("Distribuição"),
            resource_group_config.get("par_1"),
            resource_group_config.get("par_2"),
            resource_group_config.get("par_3"),
        ]
        cap = resource_group_config.get("Capacidade")

        self.dist = Distribuicao(*dist_params)
        self.res = Recurso(self.env, capacity=cap)

    def sorteia_tempo_de_processamento(self):
        return self.dist.sorteia()

    def get_tempo_estimado_para_processar_fila(self):
        tempo_medio = self.dist.media()
        estimativa = (len(self.res.queue) * tempo_medio) / self.res.capacity
        return estimativa

    def get_usage_data(self):
```

```
        return self.res.data


class Recurso(simpy.Resource):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.data = []

    def request(self, *args, **kwargs):
        res = super().request(*args, **kwargs)
        self.add_data_point()
        return res

    def release(self, *args, **kwargs):
        res = super().release(*args, **kwargs)
        self.add_data_point()
        return res

    def add_data_point(self):
        self.data.append(
            {
                'time': self._env.now,
                'queue': len(self.queue),
                'processing': self.count
            }
        )
```

APÊNDICE F – CÓDIGO DO VOO

```
# coding: utf-8

import simpy
from componentes.distribuicoes import Distribuicao



class Voo:
    def __init__(self, flight_params, sim_start_as_datetime):
        self.name = flight_params['Voo']
        self.companhia = flight_params['Companhia']
        self.qtd_pass = flight_params['Quantidade de Passageiros']
        self.partida_datetime = flight_params['partida']
        self.partida = (self.partida_datetime - sim_start_as_datetime).total_seconds()

        if flight_params.get('categoria', '').lower() == 'internacional':
            self.internacional = True
        elif flight_params.get('categoria', '').lower() == 'nacional':
            self.internacional = False
        else:
            raise RuntimeError

        self.cheg = None        # Chegadas não tem grupo
        self.balc = flight_params['Grupo de balcões']
        self.balc_atraso = flight_params['Grupo de balcões']
        self.auto = flight_params['Grupo de auto atendimento']
        self.desp = flight_params['Grupo de despacho']
        self.seg  = flight_params['Grupo de segurança']
        self.doc  = flight_params.get('Grupo de documentos')
        if self.doc == '':
            self.doc = None

        # Tempos em segundos transcorridos desde o início da simulação
```

```
        self.abetura_checkin = self.partida - int(flight_params['Abertura checkin'])*60
        self.prioridade_checkin = self.partida - int(flight_params['Prioridade checkin']) * 60
        self.fechamento_checkin = self.partida - int(flight_params['Fechamento checkin'])*60
        self.abetura_seg = self.partida - int(flight_params['Abertura Segurança'])*60
        self.fechamento_seg = self.partida

        self.dist_chegadas = None
        try:
            conf_chegadas = [
                flight_params['Distribuição chegadas'],
                flight_params['par 1'],
                flight_params['par 2'],
                flight_params['par 3'],
            ]
            self.dist_chegadas = Distribuicao(*conf_chegadas)
        except:
            pass

    def get_process_group(self, process_name):
        group_name = getattr(self, process_name)
        return group_name
```

APÊNDICE G – CÓDIGO DAS DISTRIBUIÇÕES

```
# coding: utf-8

from numpy import random, isnan


class Constante:
    def __init__(self, x):
        self.x = x

    def sorteia(self):
        return self.x

    def media(self):
        return self.x


class Uniforme:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def sorteia(self):
        return max(0.0, random.uniform(self.a, self.b))

    def media(self):
        return (self.a + self.b)*0.5


class Normal:
    def __init__(self, mu, sigma):
        self.mu = mu
        self.sigma = sigma
```

```
    def sorteia(self):
        return max(0.0, random.normal(self.mu, self.sigma))

    def media(self):
        return self.mu


class Triangular:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def sorteia(self):
        return max(0.0, random.triangular(left=self.a, mode=self.c, right=self.b))

    def media(self):
        return (self.a + self.b + self.c)/3.0


class Distribuicao:
    distribution_map = {
        'Constante (x)': Constante,
        'Uniforme (a, b)': Uniforme,
        'Normal (μ, σ)': Normal,
        'Triangular (a, b, c)': Triangular
    }

    def __init__(self, label, *args):
        dist = Distribuicao.distribution_map[label]
        args = [a for a in args if not isnan(a)]
        self._dist = dist(*args)
```

```
def sorteia(self):
    return self._dist.sorteia()

def media(self):
    return self._dist.media()
```

APÊNDICE H – CÓDIGO DA ANÁLISE DAS SAÍDAS

```
# coding: utf-8

import math
import pandas as pd
from numpy import isnan
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.dates import HourLocator, DateFormatter
import datetime

PAS_COLUMNS_TO_LABELS_MAP = [
    # 'pas_id',
    # 'rep_id',
    # 'flight',
    # 'balc', 'balc_atraso', 'auto', 'desp', 'seg', 'doc'
    'chegada',
    'fim',
    'fila_balc',
    'processamento_balc'
    'fila_balc_atraso',
    'processamento_balc_atraso',
    'fila_auto',
    'processamento_auto',
    'fila_desp',
    'processamento_desp',
    'fila_seg',
    'processamento_seg',
    'fila_doc',
    'processamento_doc',
]

PROCESSES_NAMES = ['balc', 'balc_atraso', 'auto', 'desp', 'seg', 'doc']
```

```
my_dict = {'balc':'no balcão', 'balc_atraso':'no balcão preferencial', 'auto':'no autoatendimento', 'desp':'na estação de despacho', 'seg':'na inspeção de segurança', 'doc':'na verificação de documentos'}
my_dict2 = {'balc':'Balcão', 'balc_atraso':'Balcão preferencial', 'auto':'Autoatendimento', 'desp':'Estação de despacho', 'seg':'Inspeção de segurança', 'doc':'Verificação de documentos'}

class Resultado:
    def __init__(self):
        pass

def mean(series):
    if series.isnull().values.all():
        return np.nan
    else:
        return series.mean()

def median(series):
    if series.isnull().values.all():
        return np.nan
    else:
        series.median()


def analyze_data(df_res, df_pas, config):
    results = Resultado()
    results.proc_data = {}


    # Agrega por tipo de processo e grupo, e plota histograma e violinos


    for proc_name in PROCESSES_NAMES:
        for cat in ['fila', 'processamento']:
            column = f"{cat}_{proc_name}"
```

```
        group_column = f"group_{proc_name}"
        groups = df_pas[group_column].unique().tolist()
        for group in groups:
            df_aux = df_pas.loc[df_pas[group_column] == group]
            df_by_rep = df_aux.groupby(['rep_id'])[column].agg(['min', 'max'])

            if column not in results.proc_data:
                results.proc_data[column] = {}
            if group not in results.proc_data[column]:
                results.proc_data[column][group] = {}

            # Considerando agregação por replicação
            results.proc_data[column][group]['min_rep'] = mean(df_by_rep['min'])  # Média dos
valores mínimos de cada replicação
            results.proc_data[column][group]['max_rep'] = mean(df_by_rep['max'])

            # Considerando todos os dados de todas as replicações
            results.proc_data[column][group]['min_value'] = df_aux[column].min()
            results.proc_data[column][group]['max_value'] = df_aux[column].max()
            results.proc_data[column][group]['mean'] = mean(df_aux[column])
            results.proc_data[column][group]['median'] = median(df_aux[column])



            amostras = df_aux[column]
            amostras_limpas_em_min = [a/60.0 for a in amostras.to_list() if not isnan(a)]

            plt.clf()
            fig, ax = plt.subplots()
            ax.hist(amostras_limpas_em_min)
            plt.title(f'Tempo de {cat} {my_dict.get(proc_name)} - {group}\n
{len(amostras_limpas_em_min)} passageiros')

            current_ticks = ax.get_yticks()
            if len(amostras_limpas_em_min) > 0:
```

```
            new_labels = [f"{val * 100:.0f} %" for val in
current_ticks/len(amostras_limpas_em_min)]
            ax.set_yticks(ticks=current_ticks, labels=new_labels)
            ax.set_ylim([0, min(max(current_ticks), len(amostras_limpas_em_min))])
            ax.set_xlim([0, math.ceil(max(amostras_limpas_em_min))])
        plt.xlabel("[minutos]")
        plt.ylabel("Frequência")

        #plt.show()
        plt.tight_layout()
        plt.savefig(f"out\\hist_{proc_name}_{cat}_{group}.pdf")

        plt.close()
        fig, ax = plt.subplots()
        if len(amostras_limpas_em_min) > 0:
            plt.violinplot(amostras_limpas_em_min, showmeans=True, showmedians=True)
        plt.tick_params(
            axis='x',  # changes apply to the x-axis
            which='both',  # both major and minor ticks are affected
            bottom=False,  # ticks along the bottom edge are off
            top=False,  # ticks along the top edge are off
            labelbottom=False)  # labels along the bottom edge are off
        plt.ylabel("[minutos]")
        plt.title(f'Tempo de {cat} {my_dict.get(proc_name)} - {group}')
        plt.tight_layout()
        plt.savefig(f"out\\violin_{proc_name}_{cat}_{group}.pdf")


# Agrega por Vôo e tipo de processo, e plota ditribuição do tempo despendido entre processos
tempo_total_em_fila = {'Total': []}
flights = df_pas['flight'].unique().tolist()
for flight in flights:
    tempo_total_em_fila[flight] = []
    df_aux = df_pas.loc[df_pas['flight'] == flight]
```

```
    for proc_name in PROCESSES_NAMES:
        column = f"fila_{proc_name}"
        tempo_total = df_aux[column].sum()
        tempo_total_em_fila[flight].append(tempo_total)
    soma = sum(tempo_total_em_fila[flight])
    for i in range(len(tempo_total_em_fila[flight])):
        tempo_total_em_fila[flight][i] = tempo_total_em_fila[flight][i] / soma

for proc_name in PROCESSES_NAMES:
    column = f"fila_{proc_name}"
    tempo_total = df_pas[column].sum()
    tempo_total_em_fila['Total'].append(tempo_total)

for key in tempo_total_em_fila.keys():
    soma = sum(tempo_total_em_fila[key])
    for i in range(len(tempo_total_em_fila[key])):
        tempo_total_em_fila[key][i] = tempo_total_em_fila[key][i] / soma



labels = flights + ['Total']
data = np.array(list(tempo_total_em_fila.values()))
data_cum = data.cumsum(axis=1)
processes_colors = plt.colormaps['Paired']([i/12+0.01 for i in [1, 3, 5, 7, 9, 10]])

flights_per_figure = 15
num_figures = math.ceil(len(data) / flights_per_figure)
for k in range(num_figures):
    first_index = k*flights_per_figure
    last_index = (k+1) * flights_per_figure
    if last_index >= len(flights) - 1:
        last_index = None
    figure_data = data[first_index:last_index, :]
    figure_data_cum = data_cum[first_index:last_index, :]
    fig_labels = labels[first_index:last_index]
```

```
        plt.close()
        fig, ax = plt.subplots()
        ax.invert_yaxis()
        ax.xaxis.set_visible(False)
        ax.set_xlim(0, np.sum(data, axis=1).max())
        for i, (colname, color) in enumerate(zip(PROCESSES_NAMES, processes_colors)):
            widths = figure_data[:, i]
            starts = figure_data_cum[:, i] - widths
            rects = ax.barh(fig_labels, widths, left=starts, height=0.5,
                            label=my_dict2.get(colname), color=color)

            r, g, b, _ = color
            text_color = 'white' if r * g * b < 0.5 else 'darkgrey'
        ax.legend(ncol=3, loc = [0, -0.12], fontsize='small')
        plt.title(f"Proporção de tempo em fila por voo. Parte {k+1}")
        plt.tight_layout()
        plt.savefig(f"out\\dist_tempos_{k+1}.pdf")

    #Distribuição de chegadas geral no tempo
    plt.close()
    fig, ax = plt.subplots()

    chegadas = [config.sim_start + datetime.timedelta(seconds=d) for d in
df_pas['chegada'].to_list()]
    hora_de_chegada_por_dia = {}
    for cheg in chegadas:
        date = cheg.date()
        if date not in hora_de_chegada_por_dia:
            hora_de_chegada_por_dia[date] = []
        hora_de_chegada_por_dia[date].append(cheg)

    for date, chegadas_dia in hora_de_chegada_por_dia.items():
        plt.close()
```

```
fig, ax = plt.subplots()
min_x = datetime.datetime.combine(date, datetime.time())
max_x = min_x + datetime.timedelta(days=1)
ax.set_xlim([min_x, max_x])

ax.xaxis.set_major_locator(HourLocator(byhour=range(0, 24, 2)))
ax.xaxis.set_major_formatter(DateFormatter('%H h'))

bins = [min_x + datetime.timedelta(minutes=30*x) for x in range(49)]
plt.hist(chegadas_dia, bins=bins)

current_yticks = ax.get_yticks()
new_labels = [f"{val/config.num_reps:.0f}" for val in current_yticks]
ax.set_yticks(ticks=current_yticks, labels=new_labels)

plt.title(f'Distribuição de chegadas no dia {date.strftime("%d/%m/%Y")}')
plt.xlabel("[Horário]")
plt.ylabel("Quantidade de passageiros")
plt.tight_layout()

plt.savefig(f"out\\hist_chegadas_{date.strftime('%d_%m_%Y')}.pdf")
```