Universidade de Brasília
Faculdade de Tecnologia

Simulação de controle por lógica Fuzzy de um sistema de elevadores

Maria Clara Oliveira Fortes

PROJETO FINAL DE CURSO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília

2023

Universidade de Brasília
Faculdade de Tecnologia

Simulação de controle por lógica Fuzzy de um sistema de elevadores

Maria Clara Oliveira Fortes

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Orientador: Prof. Dr. Jones Yudi Mori Alves da Silva
Coorientador: Prof. Dr. Guilherme Caribé de Carvalho

Brasília
2023

Oliveira Fortes, Maria Clara.
O48s Simulação de controle por lógica Fuzzy de um sistema de elevadores / Maria Clara Oliveira Fortes; orientador Jones Yudi Mori Alves da Silva; coorientador Guilherme Caribé de Carvalho. -- Brasília, 2023.
90 p.

Projeto Final de Curso (Engenharia de Controle e Automação) -- Universidade de Brasília, 2023.

1. Lógica nebulosa. 2. Sistemas de Controle. 3. Sistema de elevadores. 4. Simulador Arena. I. Mori Alves da Silva, Jones Yudi, orient. II. Caribé de Carvalho, Guilherme, coorient. III. Título

Universidade de Brasília
Faculdade de Tecnologia

Simulação de controle por lógica Fuzzy de um sistema de elevadores

Maria Clara Oliveira Fortes

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Trabalho aprovado. Brasília, 20 de dezembro de 2023:

Prof. Dr. Jones Yudi Mori Alves da Silva,
UnB/FT/ENM
Orientador

Prof. Dr. Guilherme Caribé de Carvalho,
UnB/FT/ENM
Coorientador

Prof. Dr. Carlos Humberto Llanos Quintero, UnB/FT/ENM
Examinador interno

Prof. Dr. Adolfo Bauchpiess,
UnB/FT/ENE
Examinador interno

Brasília
2023

Este trabalho é dedicado aos meus pais,
que sempre priorizaram a minha educação e a da minha irmã,
e é graças aos seus esforços que hoje posso concluir o meu curso.
Dedico também esta conquista a todos aqueles
que me apoiaram ao longo desta jornada.

Agradecimentos

Primeiramente gostaria de agradecer aos meus pais, Ivone e Antonio, e à minha irmã, Ana Letícia, por todo o apoio e amor dedicado a mim em cada passo que trilhei em minha vida.

Agradeço também ao meu namorado por me incentivar, por todo o carinho e por sempre acreditar em mim; e aos meus amigos que compartilharam comigo esses desafiadores anos de curso, os quais, certamente, impactaram na minha formação acadêmica.

Por fim, gostaria de agradecer ao professor Guilherme e Jones a orientação e o apoio prestados durante a pesquisa; e a todos que de alguma forma colaboraram para a realização deste trabalho.

Resumo

Sistemas de controle de grupos de elevadores (EGCS - do inglês, *Elevator Group Control System*) possuem diversas formas de se realizar o controle do atendimento aos chamados, mas com o advento dos contantes avanços na área de inteligência artificial pode se obter formas mais eficazes de realizar este controle. Nesse contexto, este trabalho tem como objetivo principal apresentar uma aplicação de técnicas de controle nebuloso como solução para os desafios de tráfego nos sistemas de elevadores. A metodologia adotada envolve a implementação de uma simulação utilizando o software Arena Simulator, considerando os sistemas modernos, nos quais há conhecimento prévio do destino de cada chamada. Propõe-se, então, a implementação de um algoritmo que utiliza um controlador nebuloso, ao qual se incorporam funções de pertinência relacionadas a três variáveis: distância, quantidade de pessoas aguardando o elevador e tempo de espera dos passageiros. A saída do sistema nebuloso consiste no grau de prioridade atribuída a cada elevador, o que permite selecionar qual o melhor elevador atenderá o chamado, ou seja, que tenha o maior valor de prioridade. O caso em estudo foi conduzido pensando em um prédio comercial, simulando um cenário de tráfego up-peak. Os resultados obtidos a partir da simulação foram detalhados no desfecho deste trabalho.

Palavras-chave: Lógica nebulosa. Sistemas de Controle. Sistema de elevadores. Simulador Arena.

Abstract

Elevator Group Control Systems (EGCS) offer various methods for handling call requests, but with the constant advancements in the field of artificial intelligence, more effective approaches for managing this control can be achieved. In this context, the primary objective of this work is to present an application of fuzzy control techniques as a solution to the traffic challenges in elevator systems. The adopted methodology involves implementing a simulation using Arena Simulator software, considering modern systems where there is prior knowledge of the destination for each call. The proposed approach includes the implementation of an algorithm that employs a fuzzy controller incorporating membership functions related to three variables: distance, the quantity of people waiting for the elevator, and passenger waiting time. The output of this fuzzy sistem is the priority assigned to each elevator, enabling the selection of the elevator that will best respond to the call, i.e., the one with the highest priority value. The case study was conducted in a commercial building, simulating an up-peak traffic scenario. The results obtained from the simulation are detailed in the conclusion of this study.

Keywords: Fuzzy Logic. Control system. Elevator system. Arena simulator.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

ABC	Artificial Bee Colony	14
BPSO	Binary Swarm Paricle Optimization)	14
CLP	Controlador Lógico Programável	14
COA	Método do Centro da Área	14
DCS	Destiny System Control	14
EGC	Elevator Group Control	14
EGCS	Elevator Group Control System	14
FA	Firefly Algorithm	14
IDE	Ambiente de Desenvolvimento Integrado	14
MOM	Método da Média do Máximo	14
PDF	Pseudo Differential Feedback	14
PSO	Particle Swarm Optimization	14
RPSO	Real-time Particle Swarm Optimization	14
SED	Sistema a Eventos Discretos	14
SOM	Método do Primeiro Máximo	14
VBA	Virtual Basic for Applications	14

Sumário

1 Introdução

1.1 Contextualização

A relevância dos sistemas de elevadores cresce no cenário atual, impulsionada pelo aumento da tendência da verticalização das construções, ou seja, prédios cada vez mais altos, para otimizar os espaços das cidades (Cortés *et al.*,). Dentre os tipos de transportes verticais (escadas, escadas rolantes, rampas, elevadores), os elevadores são considerados os mecanismos mais eficientes por proporcionar o deslocamento das pessoas entre os diversos andares com o mínimo esforço e com tempo reduzido, demonstrando assim o motivo do aumento de sua relevância conforme o aumento da verticalização (Forero, 2010).

Paralelamente à crescente verticalização, questões como eficiência, produtividade e segurança assumem um papel cada vez mais crucial no mundo moderno. Desta forma, esses conceitos, se tornam fundamentais para o desenvolvimento de projetos que visam atender as demandas do mundo contemporâneo (Rodríguez, 2016).

Existem várias tecnologias estabelecidas e comercialmente em uso, porém ainda é uma área que se mostra como sendo um campo com grandes possibilidades para desenvolvimentos inovadores, com o intuito de aprimorar o desempenho destes (Forero, 2010). Esses avanços tecnológicos possuem como objetivos principais o aumento da capacidade de transporte, redução do consumo de energia e redução do tempo de espera, elevando assim o conforto dos usuários. Nesse contexto, este trabalho deseja explorar essa lacuna, visando criar um sistema com tempo de espera reduzido a partir de tecnologias de inteligência artificial.

1.2 Definição do problema

Conforme destacado por Rodríguez (2016), o problema associado ao controle do tráfego de um sistema de grupos de elevadores é classificado, do ponto de vista computacional, como difícil, ou seja, com alto grau de complexidade. Essa complexidade torna este um grande desafio para as técnicas modernas de controle. Adicionalmente, a implementação deste tipo de problema requer investimentos significativos, sendo assim é necessário que a solução ofereça um bom serviço.

Em resumo, o foco deste trabalho é realizar o controle do tráfego de um conjunto de dois elevadores, com o intuito de reduzir o tempo de espera, buscando aprimorar o conforto dos usuários. Para alcançar um controle mais eficiente, é necessário possuir informações prévias do destino dos usuários por meio do DCS (Destiny System Control), modelo desenvolvido pela Thyssenkrup entre os anos 2000 e 2018 (Thyssenkrupp…,), um exemplo deste tipo de sistema pode ser visto na Figura 1.1. Essas informações prévias permitem evocar apenas

um elevador para atender a chamada, reduzindo o gasto de energia ao evitar situações em que o elevador chega em um andar onde a chamada já foi atendida e diminuindo o tempo de espera ao minimizar paradas desnecessárias. (Ávila, 2019)

Figura 1.1 – Sistema de grupo de elevadores com a implementação de DSC.

Fonte: Ávila (2019)

No escopo deste trabalho, a abordagem adotada envolve a implementação de algoritmos de lógica nebulosa como solução para o problema descrito, visando otimizar o tempo de espera dos usuários através do uso eficiente dos elevadores. Essa otimização requer uma análise do tempo de espera nos pavimentos (tempo desde a solicitação da chamada até um elevador abrir as portas no andar de origem); e do tempo de voo (tempo que demora o elevador para levar o usuário do andar da chamada de origem ao andar de destino) (Rodríguez, 2016).

1.3 Objetivos

O objetivo principal deste trabalho é a implementação de uma técnica de EGC (Elevator Group Control) por meio da implementação de uma lógica nebulosa. Sendo assim, tendo como objetivo central, a minimização do tempo de espera dos usuários em um sistema de transporte vertical. Para viabilizar essa implementação, desenvolveu-se o simulador dedicado à geração e ao atendimentos de chamadas, adotando o padrão up-peak de um prédio comercial. Ao aplicar essa abordagem, busca-se otimizar o desempenho e a eficiência do controle de elevadores, proporcionando uma experiência mais ágil e satisfatória aos usuários.

Uma observação importante é que neste trabalho não se tem o intuito de otimizar a energia, pois a otimização simultânea entre energia e tempo de espera constitui um problema de otimização multi-objetivo que não será considerado neste (Rodríguez, 2016).

1.4 Estrutura do documento

Este documento está organizado em cinco capítulos, incluindo esta introdução, que aborda a apresentação do problema e sua importância, além de estabelecer os objetivos deste trabalho.

O segundo capítulo aborda os fundamentos teóricos relacionados às soluções de controle de elevadores, incluindo informações sobre os métodos de modelagem de sistemas, o conceito de lógica nebulosa, um detalhamento dos módulos do software Arena e a descrição de dois projetos similares a proposta de solução.

O terceiro capítulo apresenta uma proposta de solução para a otimização do controle e monitoramento de conjuntos de elevadores, além de detalhar a metodologia utilizada para a confecção da simulação.

O quarto capítulo apresenta os resultados obtidos a partir das simulações das técnicas de controle implementadas com a finalidade de avaliar o desempenho do sistema.

Por fim, o quinto capítulo consiste na conclusão deste trabalho, onde serão abordadas as considerações finais e os possíveis direcionamentos para trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, a fundamentação teórica do tema é apresentada em cinco seções distintas. Inicialmente, discutem-se as informações relacionadas ao funcionamento dos elevadores, abrangendo um breve histórico e uma análise dos padrões de tráfego. Na sequência, abordam-se conceitos cruciais de técnicas de modelagem de problemas, fundamentais para facilitar o desenvolvimento de projetos. Em sequência, são fornecidas informações sobre a lógica nebulosa. Na penúltima seção, são detalhados alguns módulos do software Arena que são utilizados na simulação. Finalmente, na última seção, elaborou-se um resumo de projetos similares ao que será implementado neste projeto.

2.1 Sistemas de Elevadores

Com o crescimento populacional e os avanços tecnológicos na construção civil, observa-se o surgimento de edifícios cada vez mais altos. Nesse contexto, a utilização de elevadores se tornou fundamental para atender à demanda de transporte vertical de forma eficiente. No entanto, não apenas a demanda por elevadores tem aumentado, mas também a evolução tecnológica desses dispositivos é impactada pelas exigências do aumento de produtividade no mundo atual. Este aumento de produtividade pode ser afetado negativamente pelo tempo gasto de locomoção, o que enfatiza a necessidade de melhorar eficiência e funcionalidade dos conjuntos de elevadores.(Forero, 2010)

Os sistemas de grupos de elevadores são aqueles que possuem três ou mais elevadores como o principal meio de transporte vertical. Esses sistemas são caracterizados dependendo do tipo de edifício em que são instalados, como residencial, comercial, hospitalar ou shopping centers. A definição das características específicas para cada tipo de prédio é crucial para desenvolver um sistema de controle mais adequado para atender as necessidades de transporte, proporcionando um menor tempo de espera e aumentando o conforto dos usuários desses sistemas. (Rodríguez, 2016)

Para poder compreender a evolução dos elevadores e suas tecnologias, será apresentado a seguir um breve histórico, desde o seu surgimento até as tecnologias mais atuais. Detalhou-se o funcionamento desse tipo de sistema e os tipos de padrão de tráfego de pessoas, o que permite compreender como devem ser operados de acordo com o horário e o tipo de estabelecimento em questão.

2.1.1 Breve histórico

A necessidade de transporte vertical existe desde séculos antes de Cristo, com os primeiros registros desse tipo de transporte desenvolvidos pelos egípcios que os utilizavam

para construir as suas icônicas pirâmides. Apesar de o sistema de transporte vertical dos egípcios ser um grande enigma, sabe-se que utilizavam de tração animal, e até mesmo, humana, por meio de escravos. Além do Egito, a Grécia também empregava sistemas de roldanas e manivelas para transportar materiais pesados durante a construções de alguns monumentos. (Paula, 2014)

Com o decorrer dos anos e das inovações tecnológicas, a tração humana e animal foi sendo substituída, primeiramente pela energia a vapor, sendo utilizado para o transporte de cargas. Posteriormente, com os avanços dos mecanismos de segurança, tornou-se possível realizar o transporte de pessoas. (Forero, 2010)

Apesar dos avanços, o transporte de pessoas permanecia perigoso, cenário este que só mudou em 1853, quando a americana Elis Graves Otis inventou um mecanismo de freio que interrompe o movimento dos carrinhos dos elevadores caso ocorra o rompimento dos cabos, aumentando significativamente a segurança. Essa inovação foi fundamental para popularizar os elevadores como meio de transporte para passageiros. (Forero, 2010)

Atualmente, os elevadores contam com sistemas modernos de eletroeletrônica, controle e mecânica, o que proporciona conforto e segurança para os usuários e para os responsáveis pela instalação e manutenção (Paula, 2014). Além disso, os elevadores visam: a rapidez de chegada no destino, podendo chegar à velocidade de 550m/min; a redução do tempo de espera; e a seleção do melhor carro por meio da utilização de algoritmos de controle. Adicionalmente, nos dias de hoje, os aspectos estéticos estão ganhando destaque, como por exemplo, melhorias na decoração dos elevadores, adotando linhas mais modernas para valorizar a arquitetura dos prédios. (Forero, 2010).

2.1.2 Funcionamento

O funcionamento padrão de sistema de grupo de elevadores, que pode ser de apenas uma cabine ou um grupo com várias, pode ser explicado ao se considerar a necessidade de transportar um passageiro do andar A a um andar B. (Gustin; Deifan, 1999)

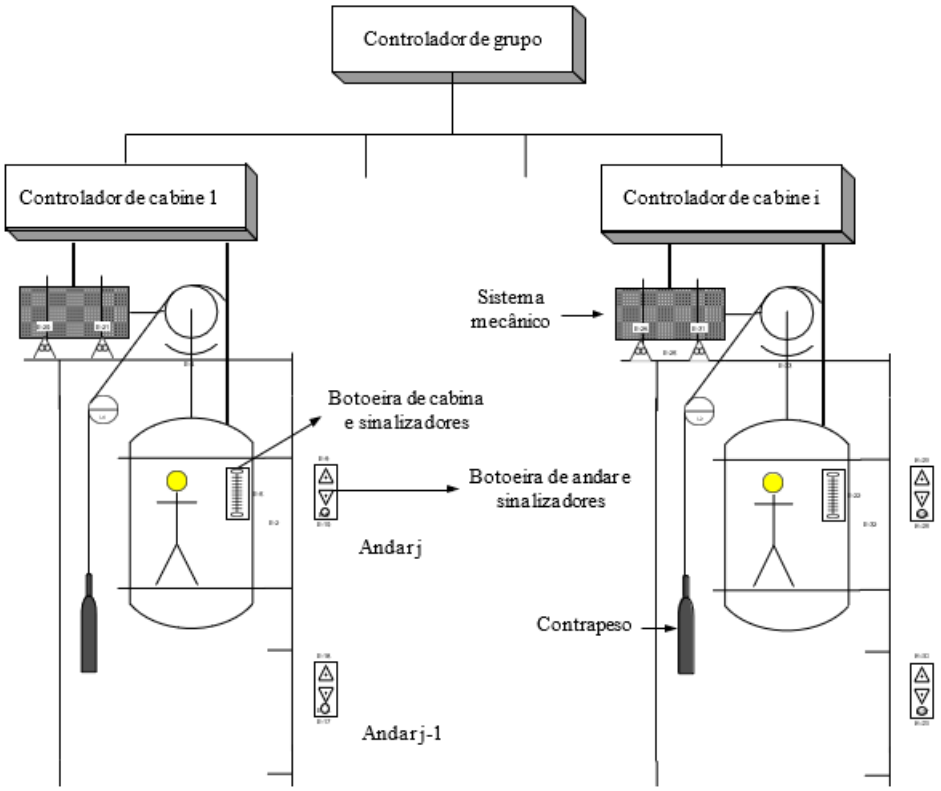
O passageiro, localizado no andar A, pressiona um botão na botoeira deste andar, sinalizando se deseja subir ou descer. O sistema de controle registra a chamada realizada pelo usuário, indicando o recebimento por meio do acionamento de um sinal luminoso, e seleciona qual cabine será utilizada pelo usuário. (Gustin; Deifan, 1999)

Para que o passageiro possa observar qual a posição e direção do movimento de um carro do sistema, normalmente, se adicionam sinalizadores luminosos que indicam o andar por meio de numeração e a direção por meio de setas. O sinalizador referente ao andar, normalmente, apresenta uma modificação para expressar quando o elevador está no andar solicitado e as portas estão abertas, estas modificações, pode ser piscá-los ou até mesmo desligá-los. (Gustin; Deifan, 1999)

Após a chegada do elevador no andar do passageiro, o mesmo pode adentrá-lo e

acionar um outro botão dentro deste para referenciar qual o seu andar de destino, isto é o andar B, o que será também registrado pelo controle do sistema para em seguida realizar o fechamento das portas e se locomover até o andar B. Por fim, ao se aproximar do andar B, destino, deve diminuir a sua velocidade de forma que o mesmo tenha uma parada mais suave ao chegar; e no andar selecionado, o mesmo para e abre as portas para que o usuário possa sair. (Gustin; Deifan, 1999)

Figura 2.1 – Sistema de controle de grupo de elevadores convencional.



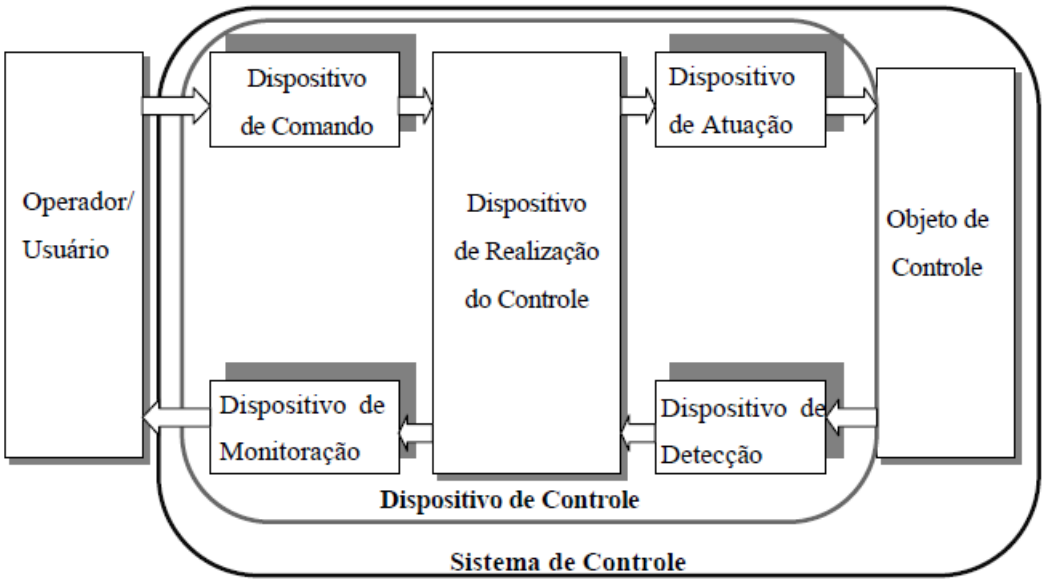
Fonte: Forero (2010)

Na Figura 2.1 são mostrados os elementos que estão envolvidos no funcionamento do sistema, sendo eles, o controle, o motor, os freios de segurança, os sensores de proximidade, o sistema de tração e o sistema da cabine, que é composto pelas portas e pelas sinalizações. (Forero, 2010)

O sistema de controle de elevadores pode ser considerado um sistema a eventos discretos (SED)(definição 2.1), por isto, pode ser decomposto no esquema apresentado na Figura 2.2. Segundo este mesmo esquema, podem se definir os principais diapositivos a serem utilizados no controle, conforme exposto na Tabela 2.1. (Montezano, 2009)

Definição 2.1. Segundo Montezano (2009), "um sistema a eventos discretos é um sistema com espaço de estados discreto e dirigido por eventos, isto é, a evolução de estados do sistema depende inteiramente da ocorrência de eventos discretos assíncronos."

Figura 2.2 – Diagrama conceitual básico do sistema de controle de SED



Fonte: Montezano (2009)

Tabela 2.1 – Dispositivos usados no controle de sistemas a elevadores

Classificação	**Dispositivos**
De Comando	Botoeiras
De Atuação	Motores das cabines e atuadores das portas
De Detecção	Fotocélulas, detetores de peso e posição
De Realização	Controlador programável, computador
De Monitoramento	Sinalizadores de direção, posição e registro, e alarmes sonoros

Ainda quanto ao controle, o mesmo pode ser feito: de um único elevador, que a sua forma mais simples é o controle automático por chamada; e de um grupo de elevadores (EGCS), que pela forma mais recente vem se utilizando, principalmente, uma evolução do DCS (Thyssenkrup AGILE).

Quanto ao controle automático de chamadas, os casos mais comuns são: (a) controle automático coletivo, caracterizado pela existência de botões de destino instalados na cabina e um único em cada pavimento, e realiza paradas em sequenciais na direção de seu movimento, independente da ordem de chamadas; (b) controle automático seletivo na descida, caracteriza-se pelo fato de as chamadas registradas só serem atendidas quando o elevador está descendo

partindo da chamada de andar mais elevado; e (c) controle automático coletivo seletivo na subida e na descida, que caracteriza-se por apresentar dois botões (subida e descida) nos andares intermediários e um nos extremos, e neste todas as chamadas de subidas são registradas separadamente das de descida, de forma a permitir atender primeiro todas as chamadas registradas em um sentido para depois atender as de sentido oposto. (Montezano, 2009)

Os EGCS são sistemas capazes de gerenciar três ou mais cabines de elevadores com o objetivo de otimizar os custos. Nos elevadores mais modernos, têm sido implementados os "Sistemas de Controle de Destino", que permitem conhecer o andar de destino do passageiro antes mesmo de ele entrar no carro. Esses sistemas apresentam um desempenho superior a outros porque possuem informações prévias sobre o destino dos passageiros, o que permite agrupar aqueles que têm o mesmo andar de destino. Isso resulta em menor tempo de espera e menos paradas, otimizando a eficiência do transporte vertical. (Forero, 2010)

O controle de um grupo de elevadores é um desafio complexo, porém, atualmente, já existem diversas técnicas avançadas disponíveis para abordar essa questão. Alguns exemplos dessas técnicas são o controle ótimo (Closs et al., 1970), a lógica nebulosa (Ho e Robertson, 1994), a programação dinâmica (Chan e So, 1996), os algoritmos genéticos (Miravete et al., 1999), sistemas especialistas (Qun et al., 2001) e as redes neurais (Barney e Imrak, 2001). Cada uma dessas abordagens oferece soluções inovadoras para otimizar o funcionamento e a eficiência do transporte vertical em edifícios. (Forero, 2010)

Uma lista de padrões mais comuns e os valores de seus respectivos parâmetros pode ser observado em Hummet, Moser e Powell (1978), o qual pode ser observados na tabela 2.2. Os três parâmetros de atribuição de andares são os seguintes:

- **A:** Porcentagem do total da população em viagem no térreo (Up-peak).
- **B:** Porcentagem do total da população em viagem que tem como andar de destino o térreo (Down-peak).
- **C:** Porcentagem do total da população em viagem que tem como andar de origem e de destino diferentes do térreo (Interfloor).

Tabela 2.2 – Padrões de tráfego de elevadores (adaptado de Hummet, Moser e Powell (1978))

	Parâmetros		
Padrão	**A**	**B**	**C**
Up-peak	90	5	5
Interfloor	45	45	10
Lunch (Saída Almoço)	20	60	20
Lunch (Entrada Almoço)	70	10	10
Down-peak	5	90	5

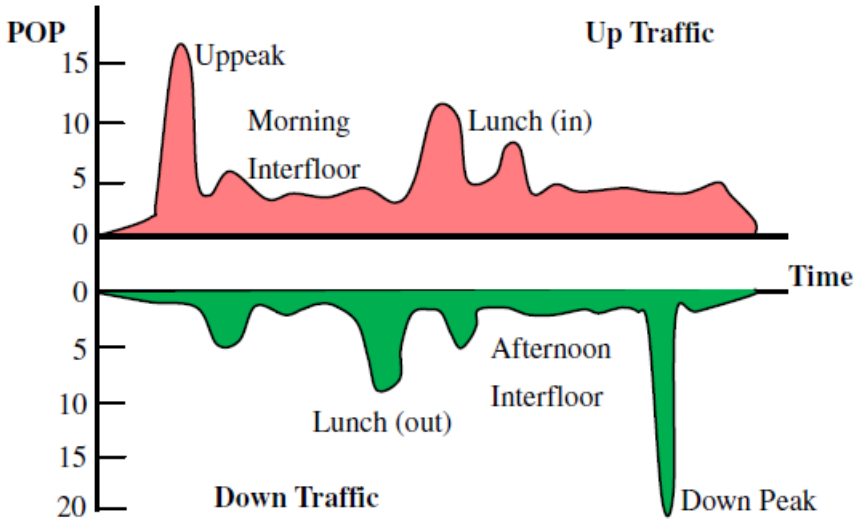
Fonte: Forero (2010)

2.1.3 Padrões de tráfego

Os padrões de tráfego são baseados nas características e distribuição demográfica dos prédios, pois podem possuir populações de tamanho diferente em cada andar, além de existir a possibilidade de ter regiões de acesso restrito, o que torna o tráfego de pessoas desigual (Amaral; Costa, 2015). Atualmente, é possível que as empresas realizam um dimensionamento dos elevadores por meio de simuladores, de forma a analisar possíveis congestionamentos. (Forero, 2010)

As diferentes demandas de usuário impões que o sistema de grupo de elevadores necessite responder de forma diferente a cada padrão de tráfego registrado ao longo do dia. Na Figura 2.3, mostra-se a demanda de passageiros de um edifício comercial de escritórios representando o número de chamadas, agrupadas pelo sentido da chamada (subindo ou descendo). (Forero, 2010)

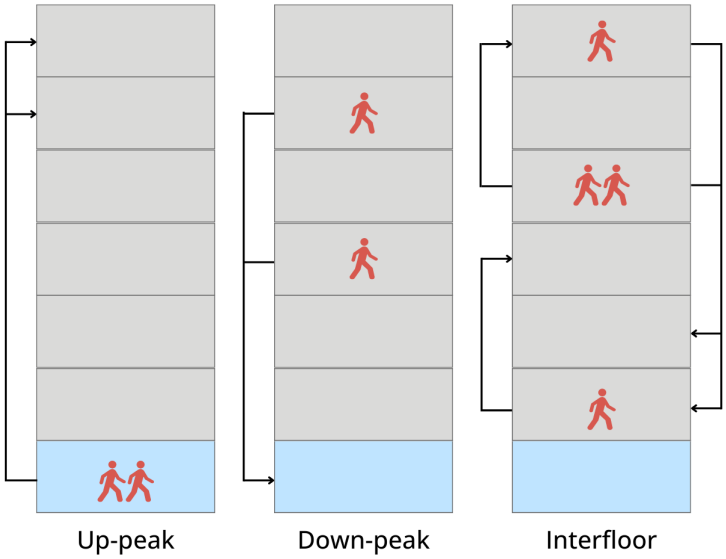
Figura 2.3 – Demanda típica de passageiros em um prédio de escritório



Fonte: FERNÁNDEZ *et al.* (2013)

Na Figura 2.3, podem ser observados três diferentes padrões de tráfego (Figura 2.4) ao longo do dia: o Up-peak, ocorrendo nas primeiras horas de trabalho, caracterizado por um grande fluxo de passageiros se deslocando para cima, a maioria deles provenientes do andar principal; o Down-peak, que ocorre no final do expediente e próximo ao meio-dia (intervalo de almoço), com um significativo fluxo de pessoas solicitando destino para o andar principal e outros andares diversos; e o Interfloor, que acontece nos demais horários do dia, apresentando um fluxo de pessoas se locomovendo entre andares de forma praticamente uniforme. (Amaral; Costa, 2015)

Figura 2.4 – Tipos de padrões de tráfego adotados durante o dia (adaptado de Amaral e Costa (2015)).



Fonte: Produzida pela autora

2.2 Técnicas de Modelagem

Ao desenvolver sistemas, é fundamental utilizar técnicas de modelagem para investigar os processos e eventos que podem ocorrer durante a simulação. Existem diversas técnicas adequadas para analisar cada tipo de aplicação, como, por exemplo, Redes de Petri e Unified Modelling Language (UML). (Amaral; Costa, 2015)

Os modelos de um sistema podem ser entendidos como simplificações do problema real e são construídos com o propósito de: (Amaral; Costa, 2015)

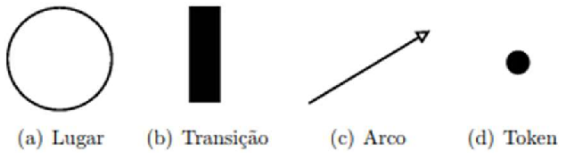
- Compreender melhor o sistema em questão;
- Auxiliar na visualização do sistema;
- Especificar a estrutura e/ou o comportamento do sistema;
- Documentar as decisões tomadas; e
- Proporcionar análises lógicas consistentes;

2.2.1 Rede de Petri

As Redes de Petri foram desenvolvidas por Carl Adam Petri como parte de sua tese de doutorado em 1962, que abordava as relações entre condições e eventos no estudo de protocolos de comunicação. No entanto, essa modelagem só ganhou reconhecimento na década de 80, quando sua teoria foi aplicada na implementação de práticas nas áreas de

informática e manufatura, impulsionada pela disponibilidade de novos recursos de hardware e software. (Montezano, 2009)

Figura 2.5 – Elementos de Rede de Petri



Fonte: Silva, Lima e Callou (2019)

A estrutura de uma rede de petri é formada por quatro estruturas básicas, conforme demostrado na Figura 2.5, sendo elas: lugares, que são os componentes passivos do sistema e são representados por círculos; transições, que são os eventos que levam o sistema de um estado a outro e são representando por retângulos; arcos orientados, as relações de troca entre lugares e transições, e é representado por setas; e *tokens* ou fichas, que é um conceito utilizado para informar se uma transição está ou não habilitada para disparar e são representados por ponto. (Amaral; Costa, 2015)

2.3 Algoritmos de Controle

Nos últimos anos, uma área amplamente estudada e desenvolvida foi o aumento da eficiência dos elevadores. Isso levou ao controle de conjuntos de elevadores, resultando em estudos sobre algoritmos de controle. Algumas das soluções desenvolvidas baseiam-se em técnicas de Inteligência Artificial, como redes neurais, lógica nebulosa e algoritmos genéticos. Essas abordagens buscam otimizar o funcionamento dos elevadores e proporcionar uma melhor experiência de transporte para os usuários.

2.3.1 Lógica Nebulosa

A "fuzzy logic"(lógica nebulosa ou lógica difusa) é uma técnica de Inteligência Artificial (IA) que surgiu a partir do princípio da incerteza, buscando se aproximar da forma como os seres humanos pensam em um sistema de controle, sem a necessidade de entender ou explicar o raciocínio dedutivo por trás dele (Forero, 2010). Essa lógica tem como objetivo atribuir graus de possibilidade para os elementos abordados, evitando assim apenas os valores de verdadeiro ou falso, uma vez que esses valores não são suficientes para lidar com a incerteza presente em muitas situações reais, ou seja, é fundamentada na teoria de conjuntos nebulosos. (Amaral; Costa, 2015)

Ao aplicar a lógica nebulosa, é possível considerar diversos objetivos que são avaliados simultaneamente por meio de regras de decisão. Assim, o sistema pode utilizar essa lógica

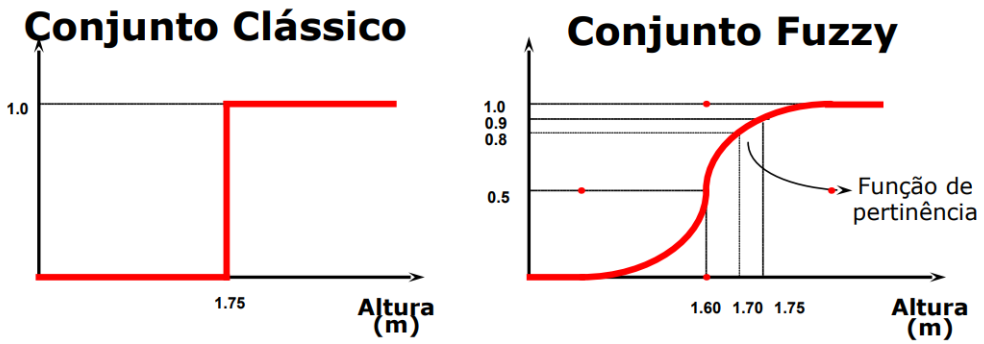
para avaliar cada elevador e atribuir um valor de prioridade para cada um deles, indicando aquele que se mostra mais adequado para atender às chamadas dos pavimentos de forma conveniente. (Forero, 2010)

Além disso, a aplicação das regras dessa lógica permite identificar o padrão de tráfego predominante em determinado momento e, assim, inferir qual controle deve ser implementado para otimizar o desempenho do sistema. Essa abordagem possibilita uma gestão mais eficiente dos elevadores, resultando em uma significativa melhoria na experiência dos usuários ao utilizar o transporte vertical. (Amaral; Costa, 2015)

2.3.1.1 Conjunto Nebuloso

Enquanto na teoria de conjuntos clássicos, que possui uma lógica booleana de grau de pertinência, ou seja, possui valores 0 (não pertence) ou 1 (pertence). Na teoria de conjuntos nebulosos, desenvolvida por Zadeh (1965), os elementos possuem uma função de pertinência, que determina um valor (grau) de pertinência entre 0 e 1. Na Figura 2.6 é possível verificar a diferença entre um conjunto clássico e um conjuto nebuloso de elementos. (Sandri; Correa, 1999)

Figura 2.6 – Diferença entre o conjunto clássico e o conjunto nebuloso.



Fonte: Fabro e Oliveira (2018)

O grau de pertinência($\mu_A(x)$), que indica o grau de compatibilidade entre o elemento (x) e o conceito do conjunto (A), podem expressar (Sandri; Correa, 1999):

- $\mu_A(x) = 1$**:** indica que x é completamente compatível com A
- $\mu_A(x) = 0$**:** indica que x é completamente incompatível com A
- $0 < \mu_A(x) < 1$**:** indica que x é parcialmente compatível com A, com grau $\mu_A(x)$.

2.3.1.2 Funções de pertinência

A função de pertinência ou número nebuloso reflete a intensidade com que um objeto pertence ao um conjunto fuzzy. Embora existam diversas funções válidas, existem algumas

funções parametrizáveis que são mais comumente utilizadas, alguns exemplos pode ser enumerados a seguir e exemplificados na Figura 2.7. (Simões; Shaw, 2007)

- **Funções Triangulares:** Parâmetros a, b e c

$$Triang_{a,b,c}(x) = max[min(\frac{x-a}{b-a}\frac{c-x}{c-b}),0] \tag{2.1}$$

- **Funções Trapezoidal:** Parâmetros a, b, c e d

$$Trap_{a,b,c,d}(x) = max[min(\frac{x-a}{b-a},1,\frac{d-x}{d-c}),0] \tag{2.2}$$

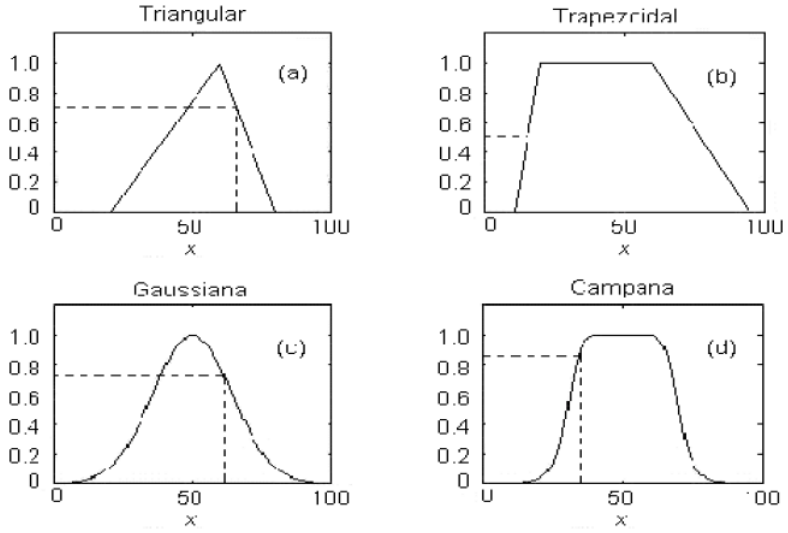
- **Funções Gaussianas:** Parâmetros σ e c

$$Gaussiana_{\sigma,c}(x) = e^{(\frac{x-c}{\sigma})^2} \tag{2.3}$$

- **Funções Campana:** Parâmetros a, b e c

$$Campana_{a,b,c}(x) = \frac{1}{1+|\frac{x-c}{a}|^{2b}} \tag{2.4}$$

Figura 2.7 – Funções de pertinência: (a) triangular (20,60,80); (b) trapezoidal (10,20,60,95); (c) gaussiana (20,50) e (d) campana (20,4,50).



Fonte: Forero (2010)

2.3.1.3 Variáveis Linguísticas

Definição 2.2. Para Zadeh (1965), uma variável linguística é dada por uma quíntupla:

$$< X, \tau(X), \chi, G, M >$$

Onde:

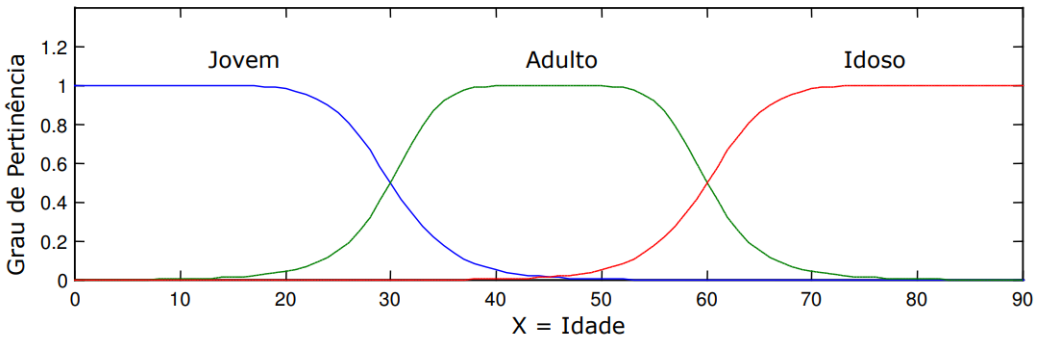
- **X**: Nome da variável linguística.
- $\tau(X)$**:** Conjunto de termos linguísticos.
- χ**:** Universo de discurso da variável linguística.
- **G:** Gramática para a geração dos termos ou rótulos.
- **M:** Regra que associa a cada rótulo l um conjunto fuzzy representando o seu significado.

Variáveis linguísticas são as variáveis cujos valores são definidos em um conjunto de termos, nomes ou rótulos, ou seja, os valores são palavras ou sentenças ao invés de números (Simões; Shaw, 2007). Assim as variáveis linguísticas admitem apenas valores linguísticos, por exemplo os termos **T(i)**, em que **i** é a idade de uma pessoa:

$$T(i) = \{Jovem, Adulto, Idoso\} \tag{2.5}$$

Na Figura 2.8, possui um exemplo de uma função de pertinência da variável linguística **idade**, formada pelos conjuntos nebulosos *"jovem"*, *"adulto"* e *"idoso"*.

Figura 2.8 – Exemplo da função de pertinência de idade.



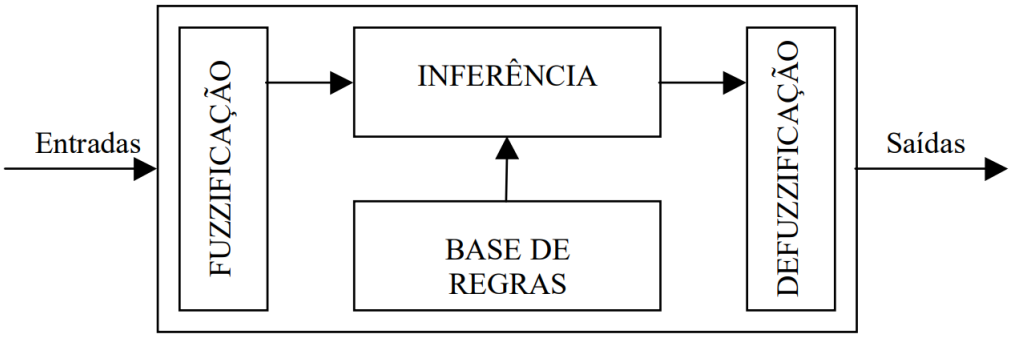
Fonte: Fabro e Oliveira (2018)

2.3.1.4 Sistema Nebuloso

Conforme Bojadziev e Bojadziev (2007), os sistemas nebulosos são, geralmente, compostos por três módulos principais: fuzzificação, inferência e defuzzificação, conforme pode

ser observado na Figura 2.9. Existem muitas variações propostas na literatura, dependendo do objetivo do projeto, porém este modelo é o suficiente para a identificação dos módulos que o compõem, que permite ter ideia do fluxo das informações.

Figura 2.9 – Diagrama do Sistema Nebuloso



Fonte: Bojadziev e Bojadziev (2007)

A estrutura da Figura 2.9 representa a transformação dos conceitos nebulosos, que foram pesquisados por diversos estudiosos, incluindo o Prof. Mandami, que projetou um modelo inicialmente denominado Fuzzy Logic Controller. Esse modelo inspirou muitos trabalhos e foi apresentado em várias obras literárias sobre o tema, demostrando a sua relevância tanto em sua forma original quanto em adaptações. (Sandri; Correa, 1999)

2.3.1.4.1 Base de Conhecimentos

A base de conhecimento é dividida em duas partes essenciais: a base de dados, responsável por armazenar informações sobre discretização e normalização dos universos de discurso, juntamente com as definições das funções de pertinência dos termos nebulosos; e a base de regras, composta por estruturas do tipo:

Se <premissa> **Então** <conclusão>

A base de conhecimento é responsável pelo procedimento de inferência, que interferem nas ações do controle. Dito isto, é importante que existam tantas regras quantas forem necessárias para mapear as combinações dos termos das variáveis linguísticas. (Sandri; Correa, 1999)

2.3.1.4.2 Fuzzificação

A interface de fuzzificação é responsável por identificar os valores das variáveis de entradas, que representam o estado do sistema. Esses valores são normalizados no universo de discurso padronizado do sistema nebuloso. Em outras palavras, os valores de entradas

são transformados por meio das funções de pertinência de forma que possam ser expressas como instâncias das variáveis linguísticas (qualitativos). (Sandri; Correa, 1999)

A Equação 2.6 é utilizada para determinar os valores de entrada dentro das funções de pertinência, a fim de determinar de qual instância da variável linguística a entrada pertence.

$$f(x) = f(x_0) + \frac{(f(x_1) - f(x_0)) \cdot (x - x_0)}{x_1 - x_0} \tag{2.6}$$

2.3.1.4.3 Inferência

O processo de inferência abrange várias etapas, incluindo a verificação do grau de compatibilidade entre os fatos e as premissas, o cálculo do grau de compatibilidade da premissa com cada regra, a determinação do valor da conclusão com base no grau de compatibilidade da regra com os dados e a ação de controle, e , por fim, a agregação dos valores obtidos. (Sandri; Correa, 1999)

Existem diferentes métodos para confeccionar a agregação dos valores obtidos, sendo os mais comuns o método de Mamdani (Mamdani e Assilian (1975) e Mamdani (1977)), o método de Tsukamoto (Tsukamoto (1979)) e o método de Takagi-Sugelo (Takagi e Sugeno (1985)). Esses modelos apresentam variações na representação dos tempos na premissa, na representação das ações de controle e nos operadores utilizados para implementação do controlador. (Forero, 2010)

Entre esses modelos, o método Mamdani é o mais reconhecido na literatura. A Figura 2.10 exemplifica um sistema com duas regras, cada uma contendo dois antecedentes (A e B) e está sendo utilizado o método Mamdani para agregação. Nesse contexto, a implicação é representada da seguite maneira:

Se A x B **Então** C

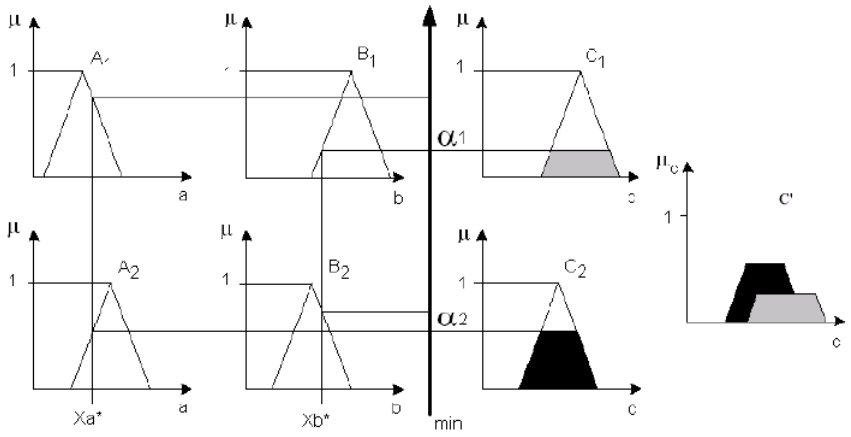
Considerando as seguintes regras:

Se (a é A1) e (B é B1) **Então** (c é C1) **Se** (a é A2) e (B é B2) **Então** (c é C2)

2.3.1.4.4 Defuzzificação

A interface de defuzzificação é utilizada para transformar o valor da variável linguística de saída em um valor discreto, com o objetivo de obter um único valor numérico, ou seja, uma única ação de controle. Existem diversos métodos conhecidos, porém os principais métodos são descritos abaixo: (Forero, 2010)

Figura 2.10 – Método Mamdani



Fonte: Forero (2010)

- **Método do Centro da Área (COA):** o valor de saída é o centro de gravidade da função de distribuição da ação de controle.

$$u = \frac{\sum_{i=1}^{N} u_i \cdot \mu_{out}(u_i)}{\sum_{i=1}^{N} \mu_{out}(u_i)} \tag{2.7}$$

- **Método da Média do Máximo (MOM):** encontra o ponto médio entre os valores que têm o maior grau de pertinência inferido pelas regras:

$$u = \sum_{m=1}^{M} \frac{u_m}{M} \tag{2.8}$$

- **Método do Primeiro Máximo (SOM):** encontra o valor de saída através do ponto em que o grau de pertinência atinge o primeiro valor máximo

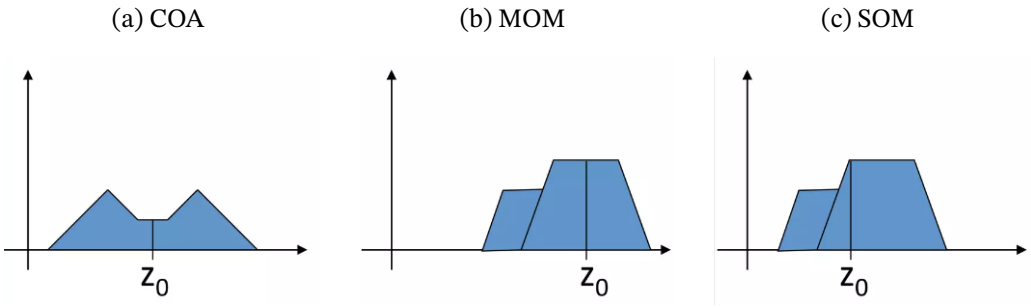
Na Figura 2.11 é possível verificar os métodos de defuzzificação, descritos acima, de forma gráfica.

2.4 Virtual Basic for Applications (VBA)

VBA é uma linguagem de programação orientada a eventos que permite os usuários estender os aplicativos do Microsoft Office como Word, Excel e PowerPoint (Jerabek; Caputo; Brandl, 2023). Esta linguagem é baseada em Visual Basic, que é um aprimoramento da linguagem de programação Basic e do Visual, a mesma possui um ambiente de desenvolvimento integrado (IDE) totalmente gráfico. (Ávila, 2019)

Umas das grandes vantagens da programação em VBA é que elas são de fácil execução e tem a capacidade de criar scripts para acelerar as tarefas diárias. Na verdade, uns dos usos

Figura 2.11 – Métodos de Defuzzificação



Fonte: Adaptado de (Fabro; Oliveira, 2018)

mais comuns para a utilização do VBA no Office é a automatização de tarefas repetitivas, além de realizar integrações entre os diferentes aplicativos do Office. (Jerabek; Caputo; Brandl, 2023)

Neste trabalho, a principal vantagem da utilização dessa linguagem é que o Arena já possui suporte para a mesma, tendo a capacidade de interagir com o Excel no qual os dados da simulação são gravados.

2.5 Software Arena Simulator

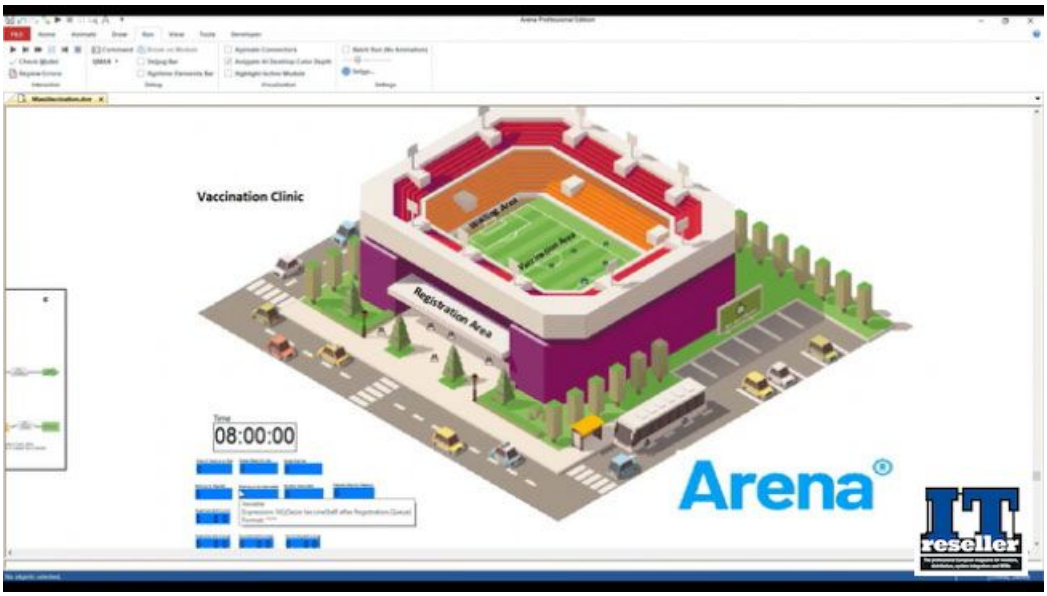
A escolha do software de simulação Arena é devido a facilidade em gerar ambientes de simulação a eventos discretos e a facilidade de verificar a construção dos modelos por possuir uma interface bastante gráfica. Outra vantagem é a possibilidade de adquirir uma licença estudantil, que apesar de limitada ainda permite a criação de sistemas mais complexos. Além disso, este é popular entre o meio acadêmico, o que facilita o acesso a recursos e dados relacionados a ele (Ávila, 2019).

Arena é um ambiente gráfico integrado de simulação distribuído pela Rockwell Automation. Este oferece recursos avançados para a modelagem de processos, desenho, animação, análise estatística e análise de resultados. Com ele, é possível simular diversos processos, como análises industriais, filas e linhas de produção, podendo ainda prever o comportamento dos ambientes modelados sem a necessidades da implementação no mundo físico. No geral, a simulação é uma ferramenta valiosa para a tomada de decisões, oferecendo a capacidade de realizar testes simulados e inferir o comportamento dos sistemas em diversos cenários sem a necessidade de implementação física.

O intuito para o desenvolvimento deste software é a de possibilitar a simulação de sistemas, e para isso proporciona uma interface gráfica baseada na linguagem SIMAM (Rockwell Automation Technologies, 2007). Sendo assim, uma ferramenta essencial para estudos de sistemas, por permitir a criação de modelos e protótipos que representam sistemas reais. Na Figura 2.12, é possível verificar um exemplo de aplicação do Software para uma clinica de vacinação, onde no centro da imagem pode ser vista a representação gráfica do

sistema.

Figura 2.12 – Exemplo de Aplicação no Software Arena



Fonte: Rockwell… (2021).

Neste, geralmente os modelos dos sistemas consistem em estações de trabalho interconectadas, que representam diferentes serviços para entidades, que represem, por exemplo, clientes ou matérias-primas. As entidades, são criadas, normalmente, com base em distribuições probabilísticas e se movem pelas estações a partir da lógica previamente determinada até sair do sistema. Atributos e outras características das entidades, como tamanho e posição, podem definir as ações a serem executadas pelas entidades no sistema. (Rodríguez, 2016)

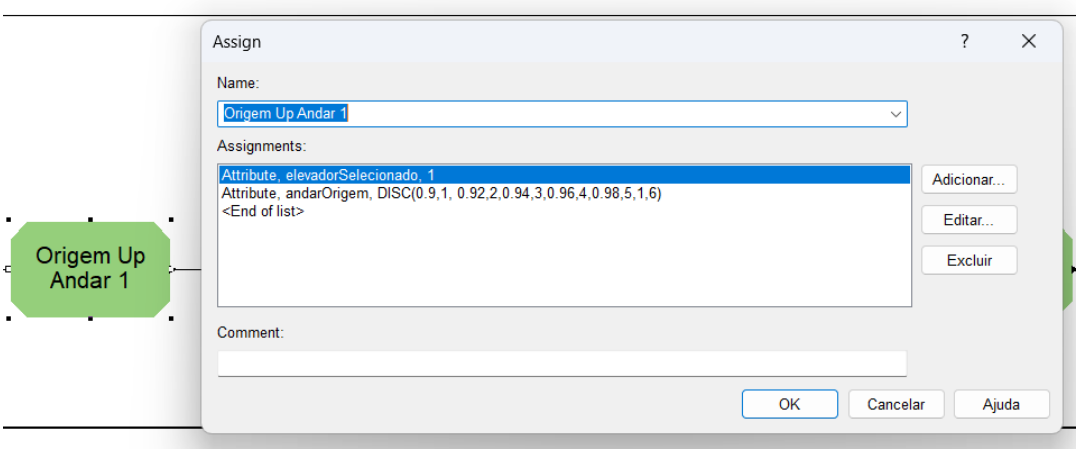
Além da capacidade de simulação, o Arena oferece ferramentas de acompanhamento e melhoria do modelo, permitindo a detecção de erros lógicos. A geração de animações facilita a compreensão dos elementos do sistema. O software também se destaca na parte estatística, coletando dados dos processos simulados e gerando relatórios detalhados, incluindo tempos médios de espera, tempos em filas e tempos de processamento, necessário para as análises dos sistemas simulados. (Altiok; Melamed, 2010)

2.5.1 Principais módulos

No simulador em questão, existe uma variedade de módulos, que desempenham funções específicas para diferentes tipos de sistemas e situações. Alguns módulos são apresentados abaixo, de forma individual (Rockwell Automation Technologies, 2007):

- **Módulo Assign** (Figura 2.13): Empregado para atribuir novos valores a variáveis, atributos de entidades, tipos de entidades, imagens de entidades ou outras variáveis do sistema.

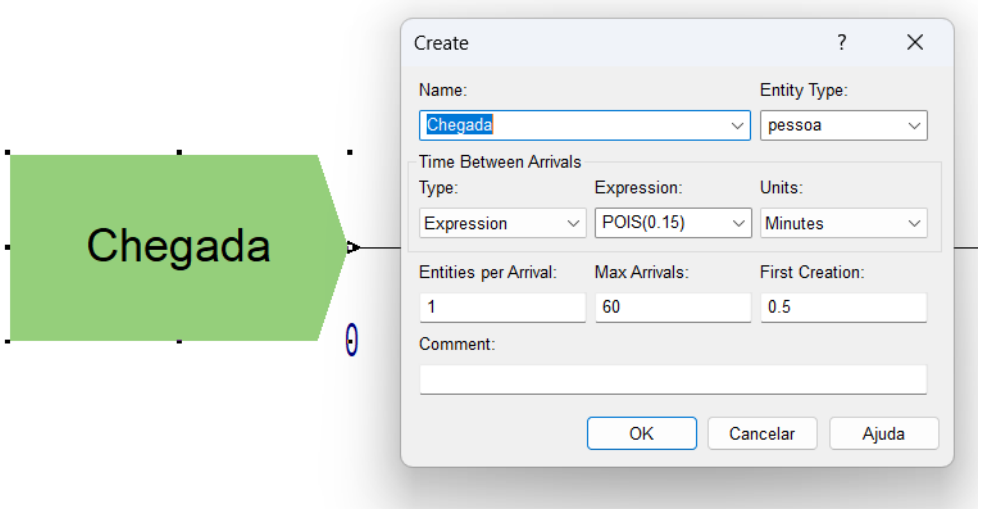
Figura 2.13 – Modulo de ajuste de elementos do Software Arena



Fonte: Produzida pela autora

- **Módulo Create** (Figura 2.14): Atua como ponto de partida para as entidades no modelo de simulação, ou seja, cria entidades com base em um cronograma ou intervalo de tempo entre chegadas, além de especificar os tipos de entidade.

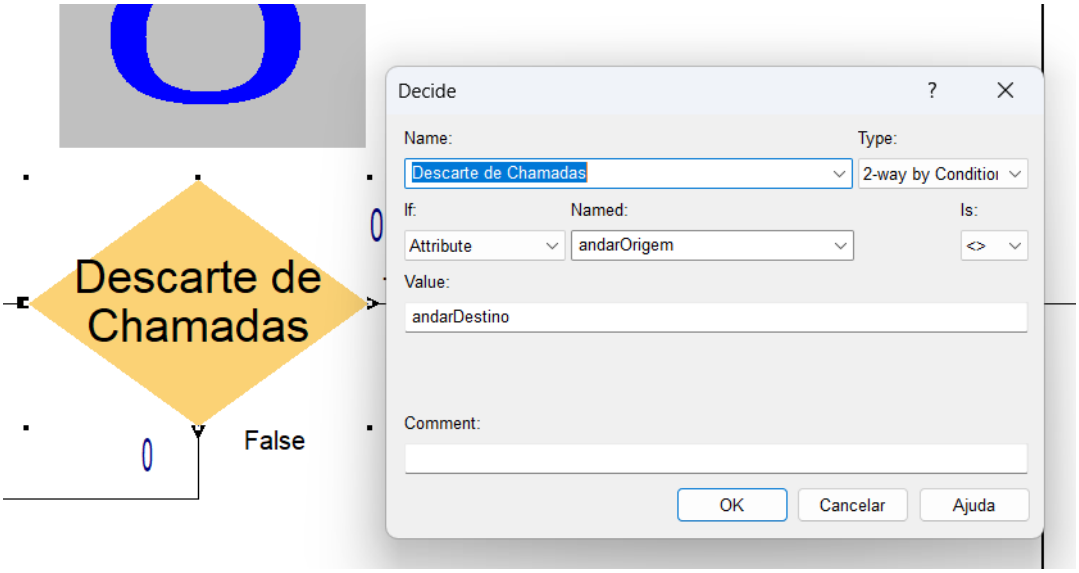
Figura 2.14 – Modulo de criação de entidade do Software Arena



Fonte: Produzida pela autora

- **Módulo Decide** (Figura 2.15): Permite tomadas de decisões com base em condições com valores atribuídos, valores de variáveis, tipo de entidade ou expressões. Este apresenta saídas distintas para cada condição informada e mais uma saída quando todas condições forem falsas.
- **Módulo Delay** (Figura 2.16): Responsável por atrasar uma entidade por um tempo

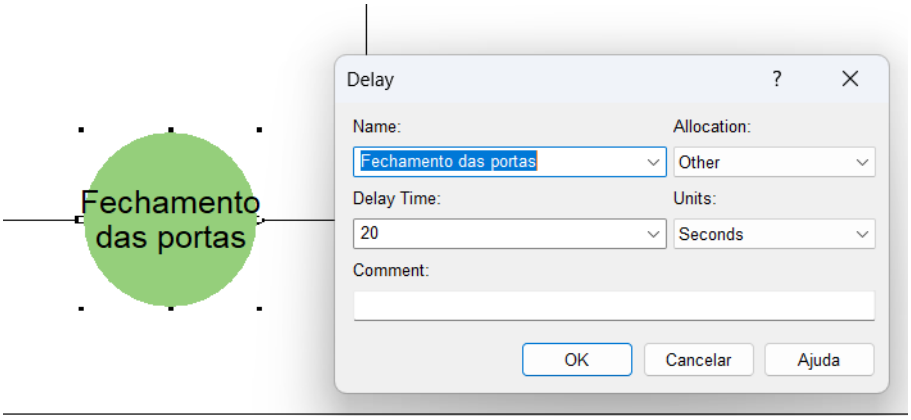
Figura 2.15 – Modulo de decisão de caminho do Software Arena



Fonte: Produzida pela autora

específico, ou seja, faz com que a entidade permaneça no mesmo até o término do tempo de atraso especificado.

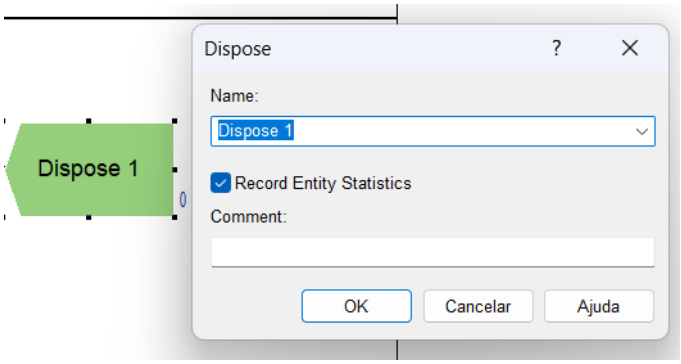
Figura 2.16 – Modulo de pausa do Software Arena



Fonte: Produzida pela autora

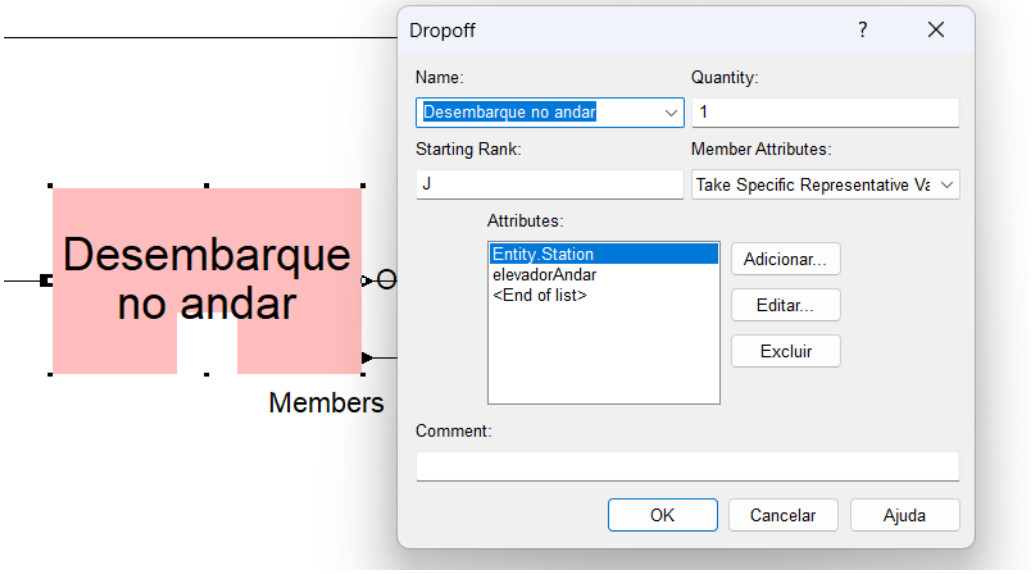
- **Módulo Dispose** (Figura 2.17): Configura o ponto final para as entidades em um modelo de simulação.
- **Módulo Dropoff** (Figura 2.18): Desempenha a função de transferência, ou seja, remove um número específico de entidades de um grupo e as envia para outro módulo, conforme as especificações da simulação.
- **Módulo Hold** (Figura 2.19):

Figura 2.17 – Modulo de retirada de entidade da simulação do Software Arena



Fonte: Produzida pela autora

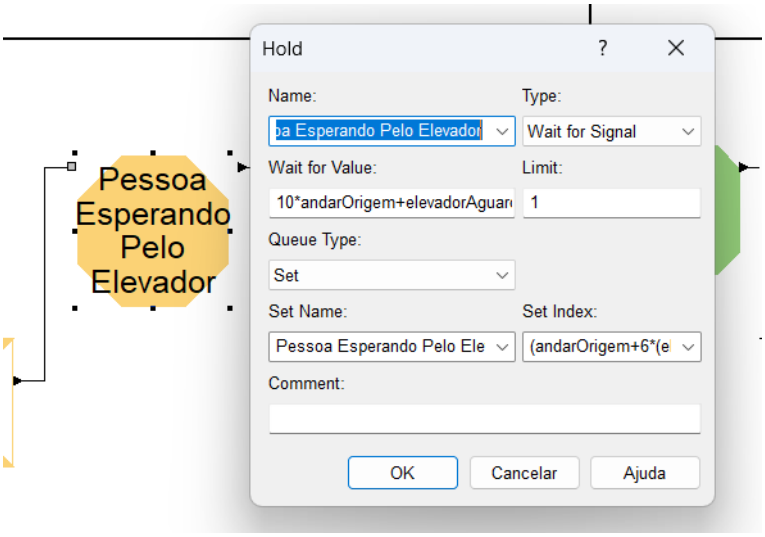
Figura 2.18 – Modulo de retirada de entidade do Software Arena



Fonte: Produzida pela autora

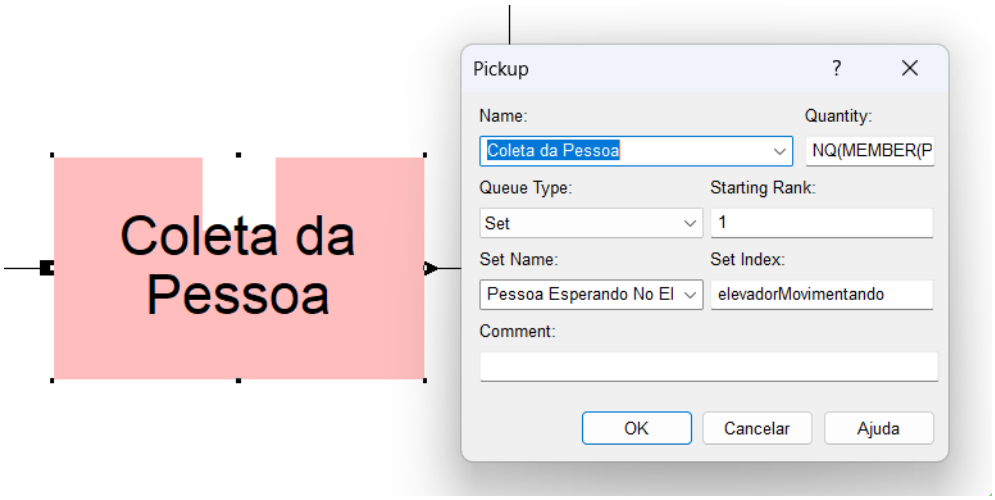
- **Módulo Pickup** (Figura 2.20): Remove entidades consecutivas de uma fila específica e adiciona-as ao final do grupo de entidades de entrada, desempenhando assim, um papel crucial na organização dinâmica dos fluxos de entidades.
- **Módulo Route** (Figura 2.21): Transfere uma entidade para uma estação específica ou para a próxima estação na sequência de visitação definida, ou seja, quando a entidade entra no módulo, seu atributo de Estação (Entity.Station) é configurado para a estação de destino e é então enviada para a estação de destino, utilizando o tempo de rota especificado.
- **Módulo Search** (Figura 2.22): Realiza pesquisas em uma fila, grupo (lote) ou ex-

Figura 2.19 – Modulo de espera do Software Arena



Fonte: Produzida pela autora

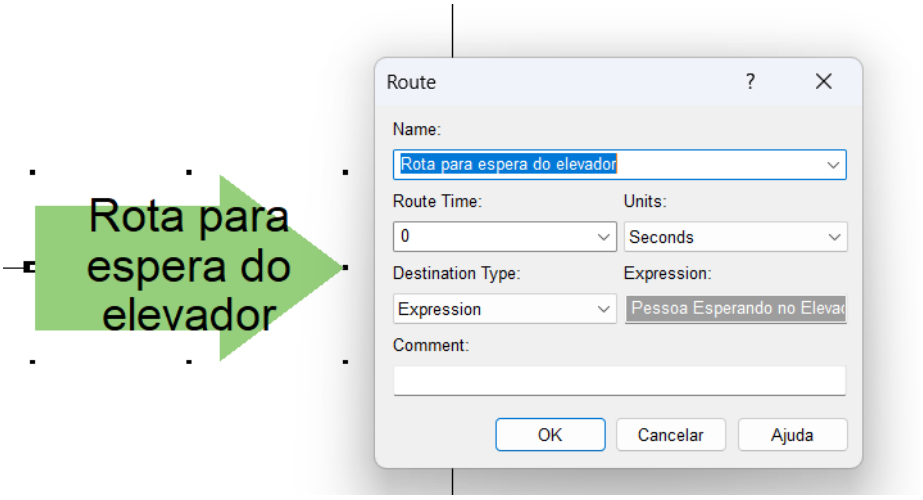
Figura 2.20 – Modulo de pegar entidade do Software Arena



Fonte: Produzida pela autora

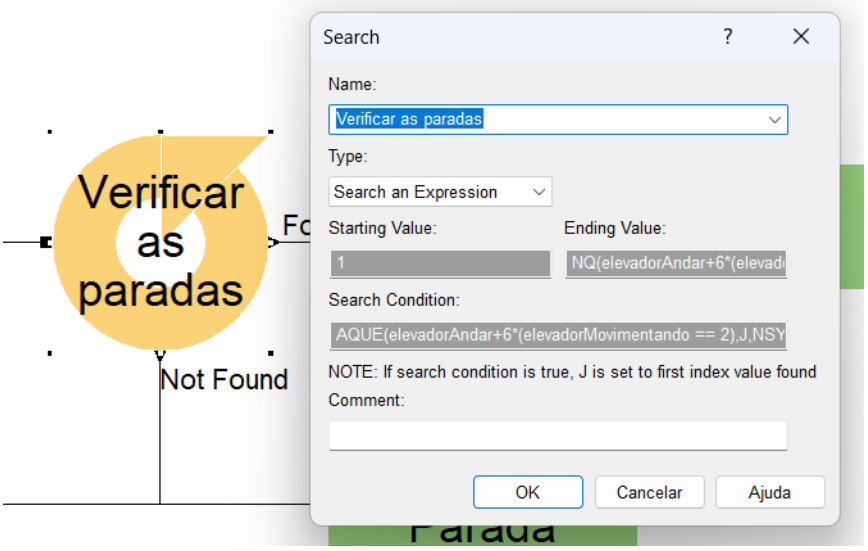
pressões para encontrar a posição da entidade ou o valor da variável J que satisfaz a condição especificada. Quando uma entidade chega a um módulo de Busca, o índice J é definido como o índice inicial e a condição de busca é verificada. Se a condição de busca for satisfeita, a busca termina e o valor atual de J é retido. Caso contrário, o valor de J é aumentado ou diminuído e a condição é verificada novamente. Esse processo se repete até que a condição de busca seja satisfeita ou o valor final seja alcançado. Se a condição não for atendida ou não houver entidades na fila ou no grupo, J é definido como igual a 0.

Figura 2.21 – Modulo de caminho entre estações do Software Arena



Fonte: Produzida pela autora

Figura 2.22 – Modulo de pesquisa do Software Arena



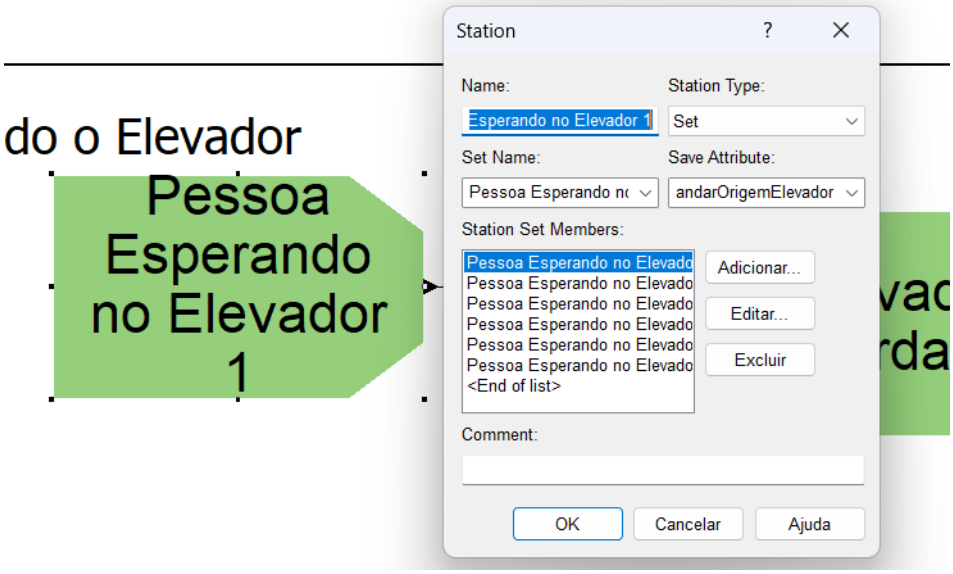
Fonte: Produzida pela autora

- **Módulo Signal** (Figura 2.23): Envia um valor de sinal para os módulos Hold definidos para aguardar um sinal e libera o número máximo especificado de entidades, ou seja, as entidades que estão aguardando o mesmo sinal são removidas de suas filas.
- **Módulo Station** (Figura 2.24): Define uma estação (ou um conjunto de estações) que corresponde a um local físico ou lógico onde ocorre o processamento. Se neste for definido um conjunto de estações, serão determinados múltiplos locais de processamento.

Figura 2.23 – Modulo de envio de sinal do Software Arena

Fonte: Produzida pela autora

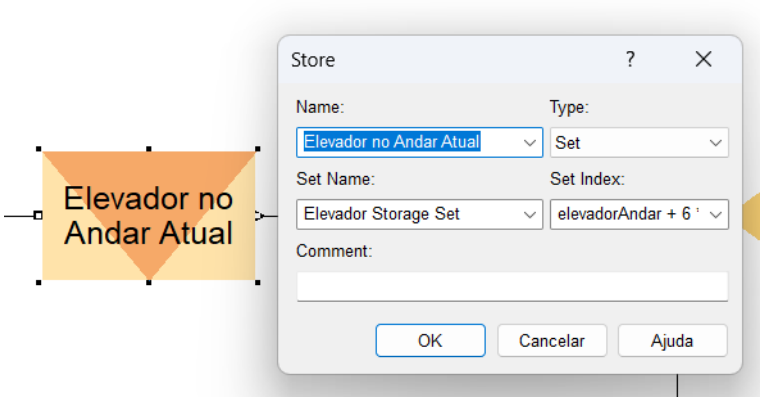
Figura 2.24 – Modulo de estação do Software Arena



Fonte: Produzida pela autora

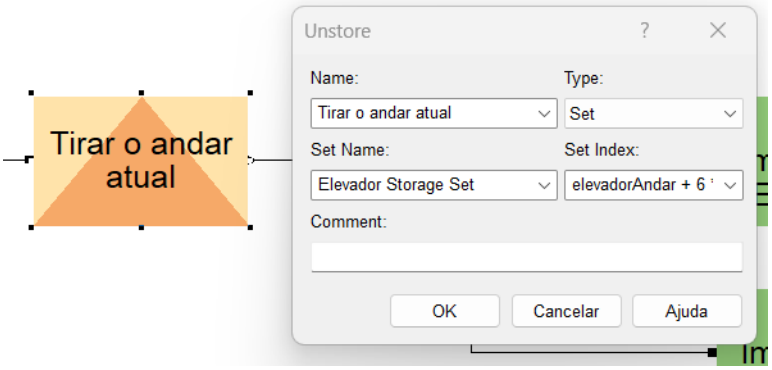
- **Módulo Store** (Figura 2.25): Adiciona uma entidade à reserva para exibir a entidade na animação enquanto passa por processamento em outros módulos, o que contribui para uma melhor visualização da entidade na animação.
- **Módulo Unstore** (Figura 2.26): Remove uma entidade do armazenamento, ou seja, quando uma entidade chega ao módulo Unstore, o armazenamento especificado é retirado e a entidade passa imediatamente para o próximo módulo no modelo.
- **Módulo VBA** (Figura 2.27): Utilizado para confeccionar a codificação em VBA do sistema, o que oferece uma abordagem flexível para personalização do comportamento

Figura 2.25 – Modulo Store do Software Arena



Fonte: Produzida pela autora

Figura 2.26 – Modulo Unstore do Software Arena



Fonte: Produzida pela autora

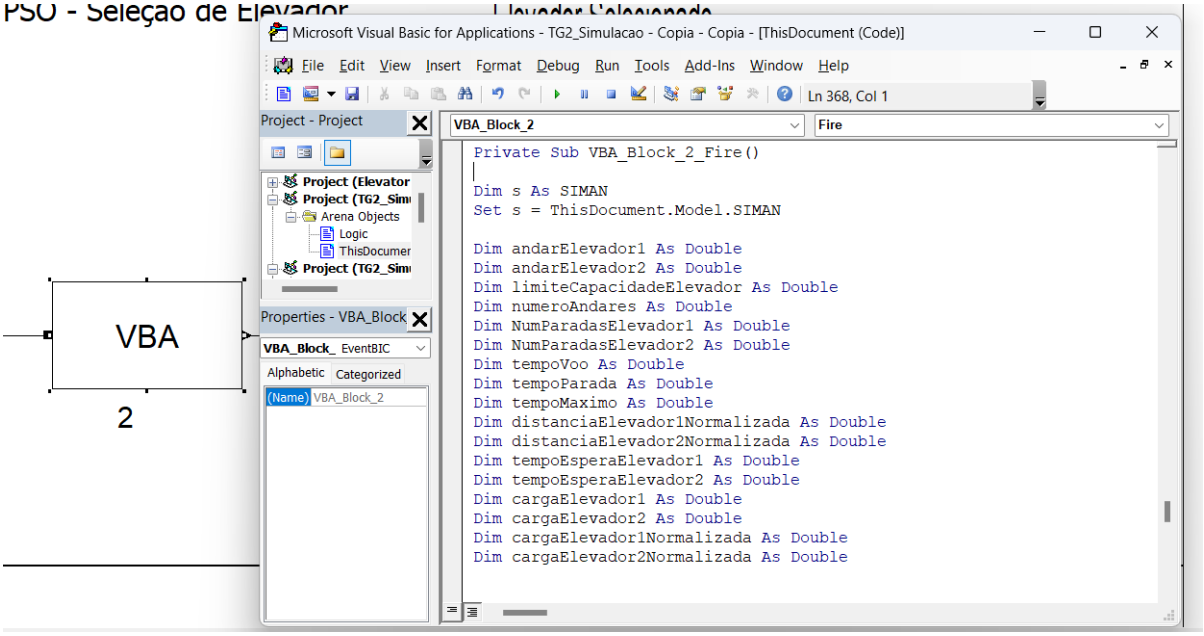
do modelo.

Os módulos apresentados desempenham papéis fundamentais na modelagem e simulação da dinâmica do sistema de grupo de elevadores empregado neste trabalho, pois possibilitam uma representação precisa e eficiente das operações analisadas.

2.6 Trabalhos Similares

Foi possível identificar, no banco de dados da UnB, duas teses de mestrado que são projetos similares ao que se deseja implementar neste. Os projetos identificados são: *"Estudo e simulação de técnicas de controle de tráfego de grupo de elevadores usando automação industrial"*, apresentado em 2010 pelo, até então, mestrando Alvaro Antonio Patinõ Forero; e *"Otimização de controle de tráfego de grupo de elevadores com algoritmos bioinspirados"*, apresentado em 2015 pelo, até o momento, mestrando Juan Pablo Diago Rodríguez. Estes

Figura 2.27 – Modulo de codificação em VBA do Software Arena



Fonte: Produzida pela autora

dois projetos foram orientados pelo Prof. Dr. Guilherme Caribé de Carvalho do Departamento de Mecânica (ENM) da Universidade de Brasília (UnB).

2.6.1 Estudo e simulação de técnicas de controle de tráfego de grupo de elevadores usando automação industrial (Forero, 2010)

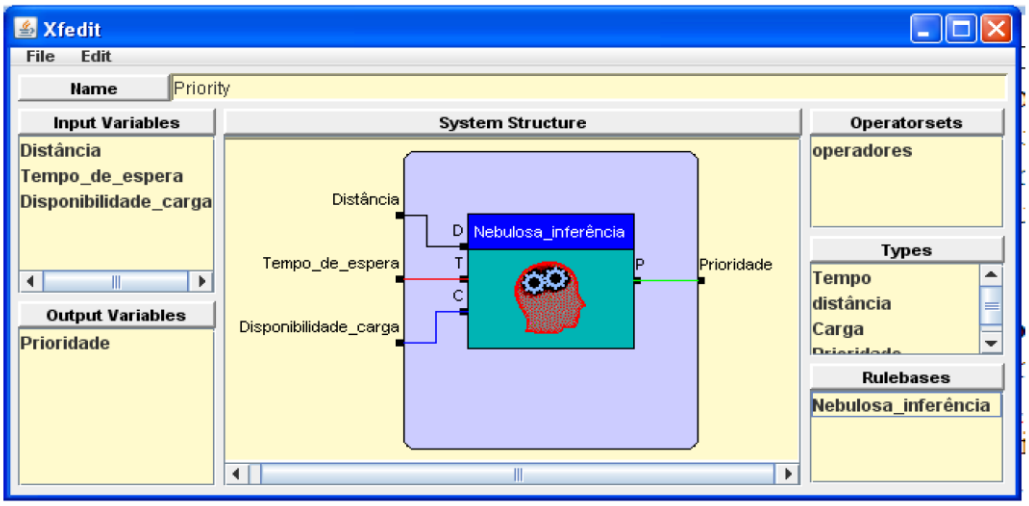
Na dissertação de mestrado, Forero (2010), o objetivo principal foi apresentar uma implementação de técnicas e controle de grupo de elevadores, visando a arquitetura moderna de DCS.

Nesse contexto, o estudo explorou a aplicação de técnicas de controle nebuloso na alocação de elevadores para atender chamadas de pavimento. Sendo assim, um controle nebuloso foi desenvolvido com três entradas (distância, disponibilidade de carga e tempo de espera dos passageiros) e uma saída relacionada à prioridade dos elevadores. Na Figura 2.28, pode ser observada a implementação confeccionada no projeto do sistema nebuloso utilizando uma ferramenta de CAD chamada de *Xfuzzy*.

Além disso, para realizar testes de desempenho do sistema de controle, foi criado um simulador de comportamento de elevadores, implementado por meio de um emulador de CLP (Controlador Lógico Programável), e um sistema de Gerador de chamadas de pavimento baseado em técnicas probabilísticas. Na Figura 2.29, pode-se observar a interface criada como gerador de chamadas, utilizando a linguagem JAVA e a IDE Eclipse.

Adicionalmente, desenvolveu-se um software cliente OPC com a finalidade de captu-

Figura 2.28 – Máquina de inferência nebuloda para cálculo de prioridades



Fonte: Forero (2010)

rar informações da dinâmica dos elevador e realimentá-las. Por fim, foram feitas diversas simulações com diferentes fluxos de passageiros e os resultados foram comparados com outros sistemas semelhantes na literatura.

Como resultado, essa dissertação demonstrou um desempenho consistente com os observados anteriormente na literatura, contribuindo para a compreensão e aprimoramento das técnicas de controle de grupo de elevadores em contextos modernos de automação industrial.

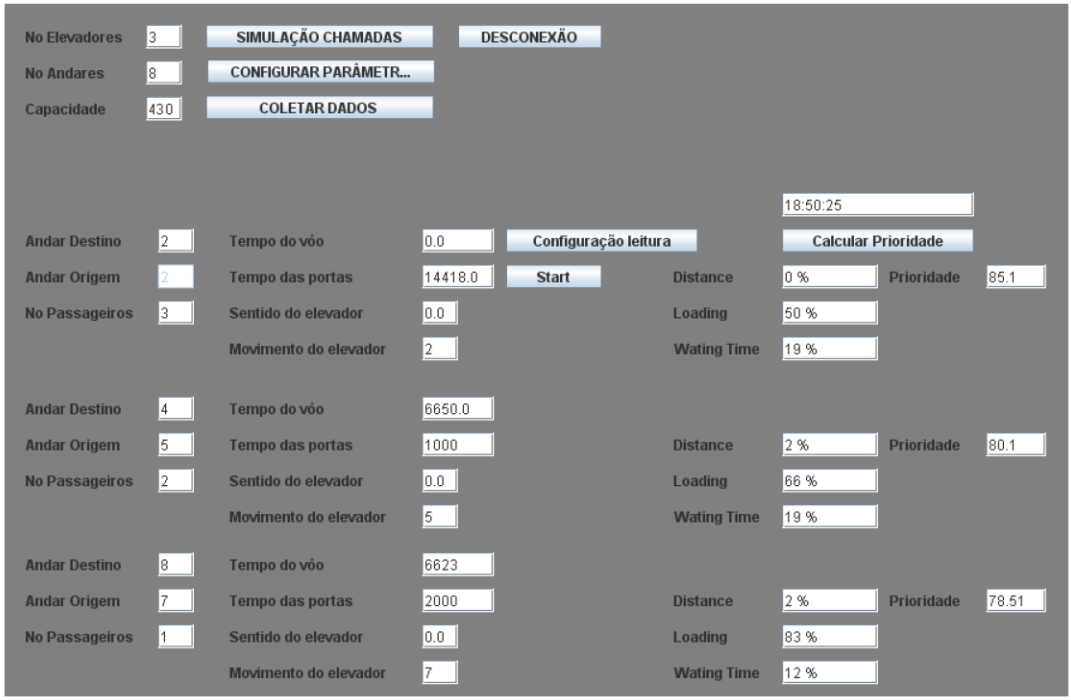
2.6.2 Otimização de controle de tráfego de grupo de elevadores com algoritmos bioinspirados (Rodríguez, 2016)

A dissertação de metrado de Rodríguez (2016) teve como objetivo apresentar a implementação de um técnica de otimização bioinspirada, um algoritmo de otimização por inteligência de enxame (PSO - Particle Swarm Optimization) do tipo binário, como solução ao problema de controle de tráfego em sistemas de grupos de elevadores.

Foram feitos testes com os algoritmos bioinspirados PSO, ABC (Artificial Bee Colony) e FA (Firefly Algorithm) e os resultados numéricos demonstraram a competitividade na solução do problema, sendo o PSO o mais eficaz na aproximação ao ótimo global. A pesquisa revelou, também, que o aumento do número de agentes no enxame melhora o desempenho dos algoritmos, embora possa resultar em maior consumo computacional.

O simulador de grupo de elevadores com o controlador proposto, baseado em inteligência de enxames BPSO (Binary Swarm Paricle Optimization), foi implementado no software Arena e testado em uma faixa crítica de fluxo de usuários em um prédio comer-

Figura 2.29 – Interface gráfica de simulação de chamadas



Fonte: Forero (2010)

cial (up-peak). A simulação demonstrou desempenho satisfatório no controle do tráfego no sistema, concordando com pesquisas similares e apresentando-se como uma solução promissora para problemas dessa natureza.

3 Desenvolvimento do simulador

3.1 Solução Proposta

O objetivo principal desta solução consiste em desenvolver um simulador para o controle de um conjunto elevadores com as características expressas na Tabela 3.1, operando durante o horário com padrão de tráfego do tipo up-peak. Neste, a escolha dos elevadores para atender os chamados será realizada por meio de um algoritmo de lógica nebulosa.

Tabela 3.1 – Dados dos elevadores

Tipo de prédio	Comercial
População	240
Número de andares	6
Número de elevadores	2

Fonte: Produzido pela autora.

O número de elevadores e o número de andares se baseia no sistema físico apresentado no laboratório da Universidade de Brasília (UnB), pois o mesmo contém 2 elevadores e 6 andares. Já o tipo do prédio, foi selecionado pela grande quantidade de estudos existentes para o controle de tráfego do tipo de edifício comercial. Por fim, a população foi selecionada com base na quantidade de pessoas por andar do Forero (2010), que no caso, é de 40 pessoas por andar, sendo assim, como no caso são 6 andares no lugar de 10, a quantidade total de pessoas é de 240.

Antes de implementar as lógicas de controle utilizando o software Arena, é importante compreender o funcionamento de um sistema de grupo de elevadores. Para alcançar esse entendimento, elaborou-se uma modelagem simplificada do sistema por meio da técnica de Rede de Petri.

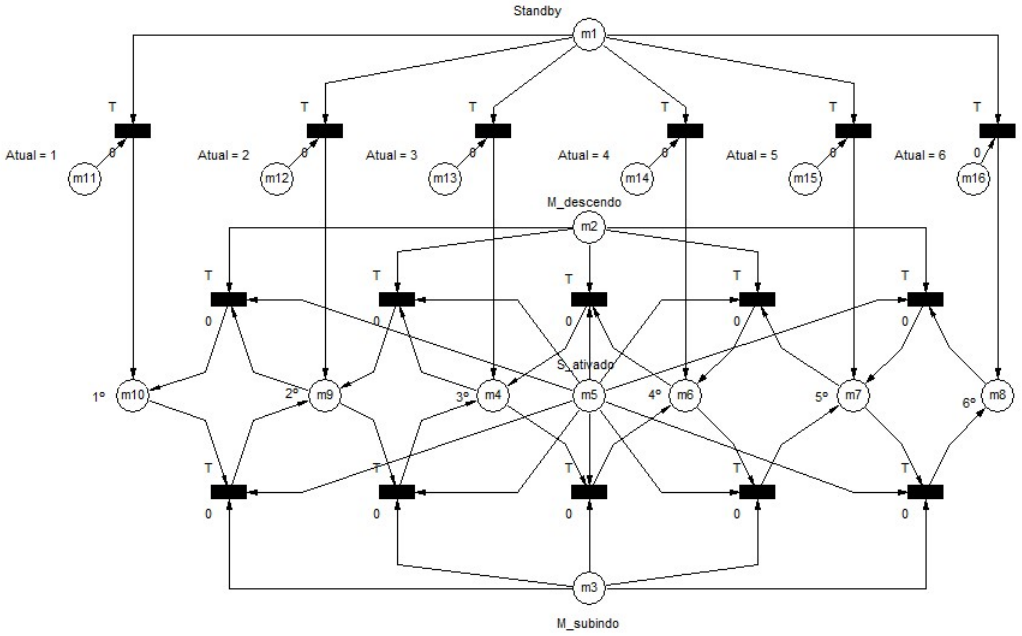
3.2 Modelagem de Rede de Petri

Foi utilizada a Técnica de Modelagem, Rede de Petri, para analisar o funcionamento de alguns módulos do sistema de controle de elevadores, nomeadamente: o módulo do andar(Figura 3.1), do motor(Figura 3.2), da botoeira (Figura 3.3) e da porta (Figura 3.4).

Na Rede de Petri do andar (Figura 3.2), os principais eventos representam os diferentes andares, variando do 1º ao 6º andar. Além dos andares, há um estado inicial "Standby"que indica o ponto onde o sistema é inicializado para verificar em qual andar se encontra ("Atual = x"), antes de transicionar para o estado correspondente a esse andar. Esse estado inicial só

é alcançado durante a inicialização do sistema. Outros estados incluem "M_descendo", que representa o motor ligado no sentido de descer; "M_subindo", que representa o motor ligado no sentido de subir; e "S_ativado", que indica a ativação do sensor indutivo, demonstrando que o elevador chegou a um novo andar. Essa rede demonstra a localização do sistema em relação aos andares e como ocorre a troca entre eles.

Figura 3.1 – Rede de Petri para o subsistema do Andar.

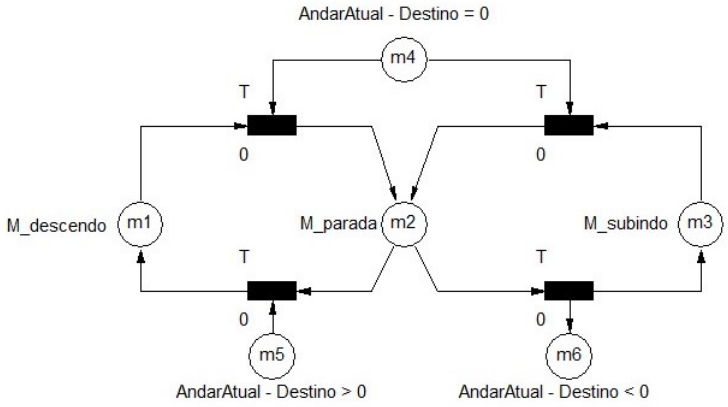


Fonte: Produzido pela autora.

Na Rede de Petri do motor (Figura 3.2), podem ser observados três principais estados em relação ao motor: "M_parado", quando o motor está parado; "M_descendo", que representa o motor ligado no sentido de descer; e "M_subindo", que representa o motor ligado no sentido de subir. Além disso, existem outros estados que verificam se é necessário descer (quando o andar de destino é menor que o atual), subir (quando o andar de destino é maior que o atual) ou parar (quando o andar de destino é igual ao atual). Essa rede demonstra como o sistema realiza o acionamento dos motores e a parada do elevador.

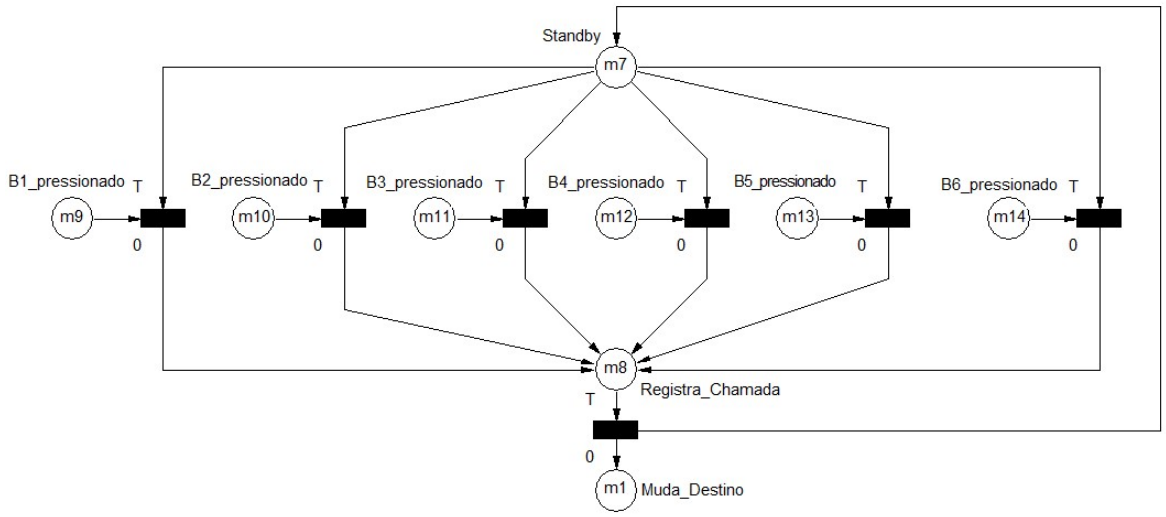
Na Rede de Petri da botoeira(Figura 3.3), é possível verificar os três principais estados: "Standby", quando o sistema aguarda o acionamento de algum botão; "Registra_chamada", que registra qual botão foi acionado para calcular o destino; e "Muda_destino", que atualiza o valor do destino para o próximo local em que o carro do elevador deve chegar. Além disso, existem estados que representam os botões sendo pressionados, enviando o sinal para registrar o sistema. Essa rede foi desenvolvida para apresentar a lógica aplicada no registro das botoeiras, sendo importante destacar que esse registro será feito para as botoeiras de cada andar.

Figura 3.2 – Rede de Petri para o subsistema do Motor.



Fonte: Produzido pela autora.

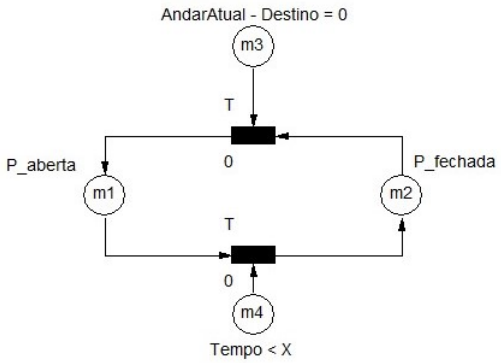
Figura 3.3 – Rede de Petri para o subsistema da Botoeira.



Fonte: Produzido pela autora.

Por fim, a Rede de Petri da porta (Figura 3.4) representa a forma como a porta será simulada por meio do sistema supervisório. Nela, podem ser observados os dois principais estados da porta: aberta e fechada. Além desses estados, existem outros que verificam o sistema, como "$AndarAtual - Destino = 0$", que verifica se o elevador chegou ao andar de destino para abrir a porta; e "$Tempo < X$", que é um estado que verifica o tempo que a porta permanece aberta antes de fechar. Essa rede proporciona uma visão da simulação da porta no sistema supervisório.

Figura 3.4 – Rede de Petri para o subsistema da Porta.



Fonte: Produzido pela autora.

3.3 Sistema Nebuloso

Para o início do desenvolvimento de um sistema nebuloso é necessário considerar as variáveis linguísticas. Sendo assim, foi determinado com base na abordagem de Forero (2010), as seguintes variáveis linguísticas, que caracterizam um sistema de controle de grupo de elevadores nebuloso (FEGCS – Fuzzy Elevator Group Control System):

- **Tempo de espera (T_w)**: medida do conforto dos usuários, ou seja, representa o tempo estimado desde a abertura do chamado até o momento que o usuário possa entrar no elevador. Essa medida pode ser determinada pela Equação 3.1.

$$T_w = (N_s * T_s) + (D * T_f) \tag{3.1}$$

  Onde:

  - N_s**:** Número de paradas
  - T_s**:** Tempo de paradas (3 segundos)
  - D**:** Distância percorrida em andares
  - T_f**:** Tempo vôo entre andarres (2 segundos)

- **Disponibilidade de carga:** representa o número de vags disponíveis dentro do elevador no momento em que uma chamada de pavimento é realizada. No caso do estudo, se estipulou que o máximo de pessoas em um elevador é de 6 pessoas.
- **Distância (D):** representa a distância percorrida pelo elevador desde do seu andar de origem até a posição do chamado.
- **Prioridade:** esta é a variável de saída da máquina de inferência, que representa o valor de conveniência de cada elevador de atender a chamada.

Uma observação importante a se fazer é que tempo de paradas usado é de 3 segundos, pois seria o tempo que a pessoa levaria para sair ou entrar do elevador, seguindo o tempo

utilizado por Rodríguez (2016). Já o tempo usado para o tempo de vôo entre os andares foi utilizado 2 segundos, um valor um pouco maior do que considerado pelo Rodríguez (2016), que foi de 1,1 segundos (com velocidade máxima de 3 m/s e com distância entre andares de 3,3 m).

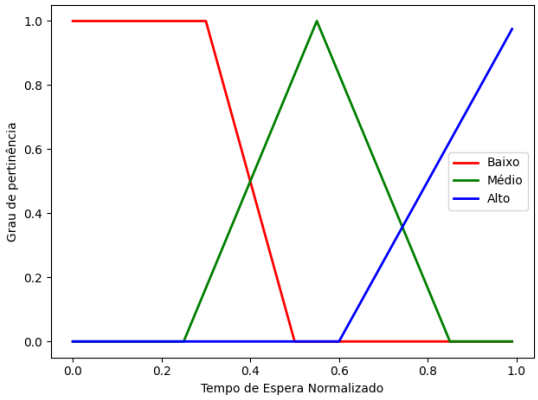
Após a definição das variáveis linguísticas, determinaram as funções de pertinência correspondente a cada uma delas. Para esse propósito todas as variáveis foram normalizadas, ou seja, pertencem a um intervalo de 0 a 1, e todas possuem como termos linguísticos *Baixo*, *Médio* e *Alto*. Os valores limites atribuídos a cada função de pertinência foram utilizadas os valores de referências anteriores, sendo elas Forero (2010) e Siikonen (1997). Os detalhes dessas funções poder ser visualizados na Figura 3.5 e descritas abaixo:

- **Tempo de espera:**
  - **Baixo:** função trapezoidal (Equação 2.2) com parâmetros a = 0, b = 0, c = 0,3 e d = 0,5;
  - **Médio:** função triangular (Equação 2.1) com parâmetros a = 0,25, b = 0,55 e c = 0,85;
  - **Alto:** função triangular (Equação 2.1) com parâmetros a = 0,6, b = 1 e c = 1;
- **Distância:**
  - **Baixo:** função trapezoidal (Equação 2.2) com parâmetros a = 0, b = 0, c = 0,3 e d = 0,5;
  - **Médio:** função triangular (Equação 2.1) com parâmetros a = 0,25, b = 0,55 e c = 0,85;
  - **Alto:** função triangular (Equação 2.1) com parâmetros a = 0,6, b = 1 e c = 1;
- **Capacidade:**
  - **Baixo:** função triangular (Equação 2.1) com parâmetros a = 0, b = 0 e c = 0,4;
  - **Médio:** função triangular (Equação 2.1) com parâmetros a = 0,2, b = 0,45 e c = 0,8;
  - **Alto:** função triangular (Equação 2.1) com parâmetros a = 0,7, b = 1 e c = 1;
- **Prioridade:**
  - **Baixo:** função triangular (Equação 2.1) com parâmetros a = 0, b = 0 e c = 0,4;
  - **Médio:** função triangular (Equação 2.1) com parâmetros a = 0,1, b = 0,5 e c = 0,9;
  - **Alto:** função triangular (Equação 2.1) com parâmetros a = 0,6, b = 1 e c = 1;

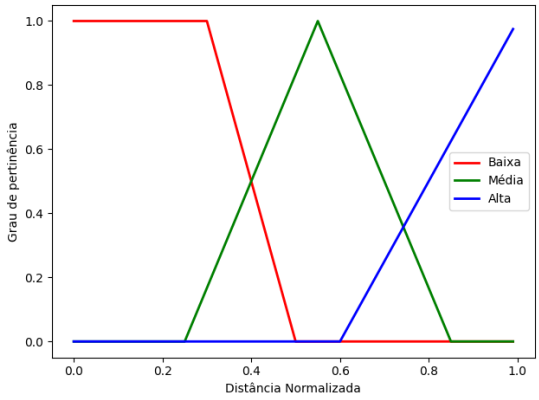
Finalmente, a implementação da interface de fuzzificação seguiu o método convencional, utilizando a função de interpolação como auxiliar. A etapa de inferência foi elaborada com base nas regras da Tabela 3.2 e adotando o método Mamdani. A fase de defuzzificação

Figura 3.5 – Funções de Pertinência

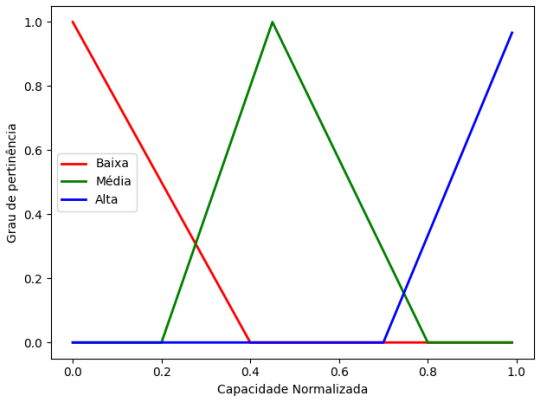
(a) Tempo de Espera



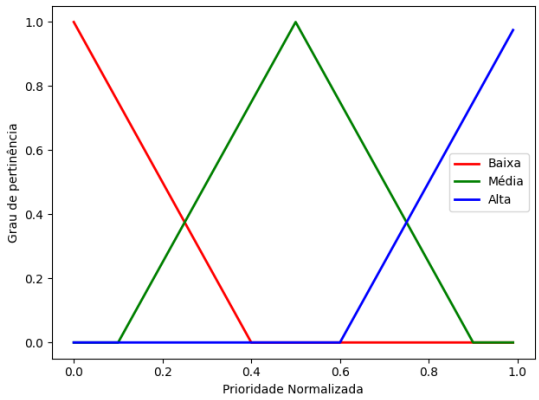
(b) Distância



(c) Capacidade



(d) Prioridade



Fonte: Produzida pela autora

foi desenvolvida com base no método COA (centroide). A visualização de todo o código relacionado ao sistema nebuloso de controle pode ser observado no Apêndice A.

3.4 Simulação no Arena

Na presente simulação, foram incorporados dois tipos de entidades. O primeiro tipo é a "pessoa", caracterizada pelos atributos do andar de origem, do andar de destino e do elevador ao qual foi designada. E o segundo tipo de entidade é o "elevador", que não possui atributos próprios, visto que seus valores são derivados a partir de variáveis do sistema.

As variáveis, importantes, implementadas na simulação são:

- **Paradas de Subida:** expressa por uma matriz de 2 colunas e 6 linhas, aonde as colunas representa o elevador e as linhas os andares. Podem ser atribuídos os valores 0 (não precisa parar) ou 1 (precisa parar) aos elementos dessa variável.
- **Paradas de Descidas:** que segue o modelo da variável de "Paradas de Subidas".
- **Estado do Elevador:** expressa por uma matriz de 2 colunas e 3 linhas, aonde as

Tabela 3.2 – Base de regras

Nº	Regras de inferência	Prioridade
R1	SE(Distância == Alto & Tempo de espera == Alto)	Baixa
R2	SE(Distância == Alto & Tempo de espera == Baixo)	Média
R3	SE(Distância == Alto & Tempo de espera == Médio)	Baixa
R4	SE(Distância == Baixo & Tempo de espera == Alto)	Média
R5	SE(Distância == Baixo & Tempo de espera == Baixo)	Alta
R6	SE(Distância == Baixo & Tempo de espera == Médio)	Média
R7	SE(Distância == Médio & Tempo de espera == Médio)	Média
R8	SE(Distância == Médio & Tempo de espera == Alto)	Baixa
R9	SE(Disponibilidade de Carga == Baixo & Tempo de espera == Alto)	Baixa
R10	SE(Distância == Baixo & Disponibilidade de Carga == Baixo)	Baixa
R11	SE(Disponibilidade de Carga == Médio & Tempo de espera == Baixo)	Alta

Fonte: Forero (2010)

colunas representa o elevador, a primeira linha representa se o elevador está subindo (1) ou não (0), a segunda linha representa o andar atual (de 1 a 6) e a terceira linha representa a quantidade de pessoas dentro do elevador (de 1 a 6).

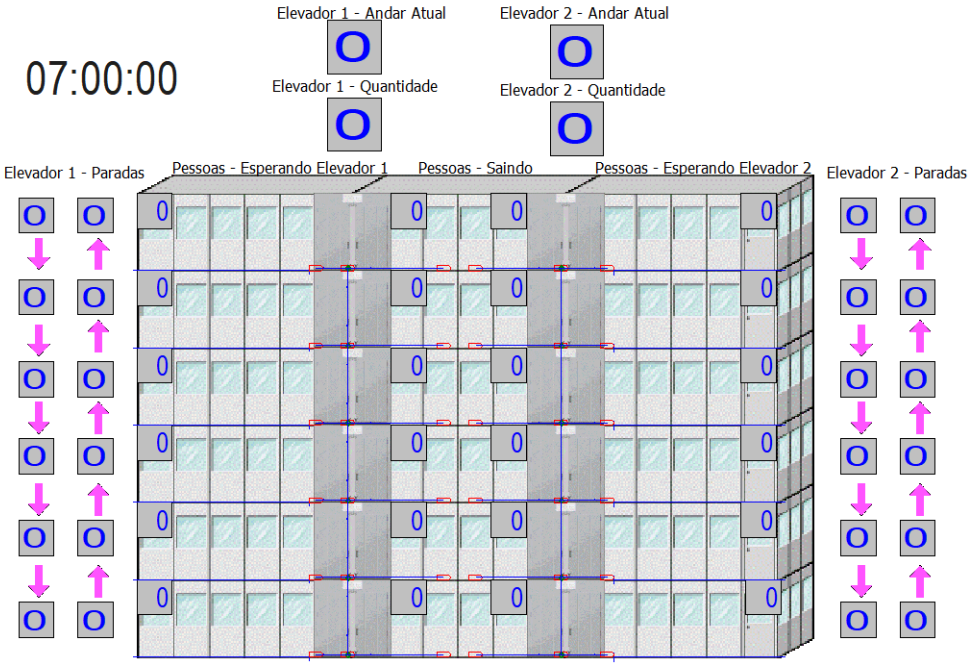
Além das variáveis importantes, foram implementadas outras variáveis auxiliares, filas de espera das pessoas no pavimento, estações de pavimentos, e outros mecanismos de forma a facilitar a implementação e funcionamento da simulação.

Para a melhor visualização, a simulação foi dividida em sete módulos distintos, sendo eles: animação, gerador de chamadas, seleção do elevador, pessoa aguardando o elevador, saída das pessoas, criação dos elevadores e movimentação dos elevadores.

3.4.1 Animação

O módulo de animação é apresentado na Figura 3.6, onde se pode visualizar um prédio de seis andares com dois elevadores. Na representação, acima de cada elevador são exibidos dois números, sendo o da esquerda referente ao andar atual e o outro referente a quantidade de pessoas no elevador. Nos andares são exibidos quatro números, sendo os dois a esquerda referentes ao primeiro elevador e os outros dois ao segundo elevador, sendo ainda os dois centrais referentes à quantidade de pessoas que sairão no andar e os outros, periféricos, referentes à quantidade de pessoas que irão entrar naquele andar. Adicionalmente, nas laterais dos prédios, são apresentados os valores que indicam as paradas que cada elevador precisa realizar por meio das variáveis “Paradas de Subida” e “Paradas de Descida”.

Figura 3.6 – Animação do Prédio de Estudo



Fonte: Produzido pela autora.

3.4.2 Gerador de Chamadas

O módulo de gerador de chamadas pode ser verificado na Figura 3.7. Neste é onde são criadas as entidades do tipo “pessoa”, que são geradas utilizando um distribuição de *Poisson*, conforme a Equação 3.2 e a Equação 3.3, sendo assim utilizou-se a densidade de probabilidade ($\lambda = 0.15$), que se refere a taxa de chegada. (Forero, 2010)

$$P = \frac{(\lambda T)^n e^{-\lambda T}}{n!} \quad , \quad n = 0,1,2,... \tag{3.2}$$

Onde:

- **P:** probabilidade de chegada
- **T:** tempo
- λ**:** taxa de chegada
- **n:** número de eventos

$$\lambda = \frac{0,01(TI)}{300} \sum_i POP_i \tag{3.3}$$

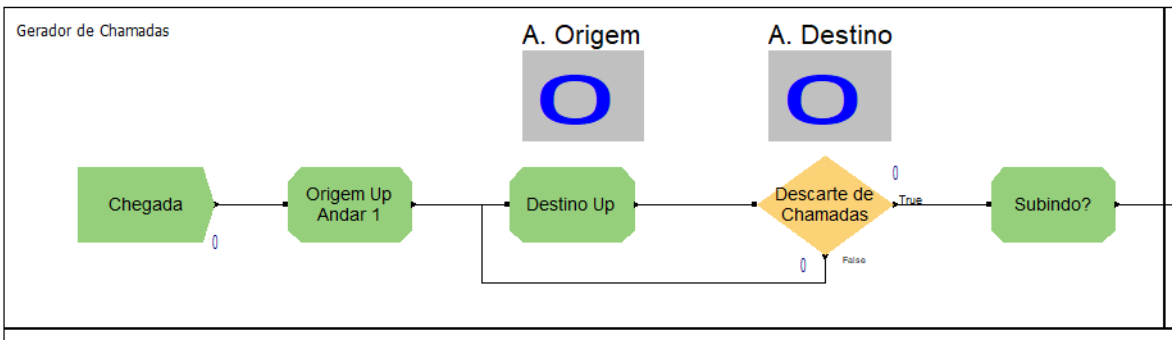
Onde:

- **TI:** intensidade do tráfego (em porcentagem)
- POP_i **:** população de cada andar

- **i:** número do andar

Como o intuito é desenvolver um simulador com a chegada de pessoas utilizando o padrão up-peak, utilizou-se para a seleção do andar de origem a probabilidade de 90% de ser o 1º andar e os outros 10% foram distribuídos em probabilidades iguais entre os outros andares.

Figura 3.7 – Gerador de Chamadas no Arena



Fonte: Produzido pela autora.

Na Figura 3.7, é possível verificar dois números, sendo o da esqueda a representação do andar de origem da chamada da pessoa e o outro, do andar de destino. Ainda neste módulo é necessário realizar uma verificação do chamado, pois não é necessário utilizar o elevador quando o andar de destino e o andar de origem são iguais.

3.4.3 Seleção do Elevador

O módulo de seleção do elevador, verificado na Figura 3.8, é responsável pela implementação do módulo de VBA, no qual foi codificado a lógica fuzzy para realizar a escolha do elevador que melhor atende o chamado, e, também, é responsável por direcionar a pessoa para esperar o elevador correto.

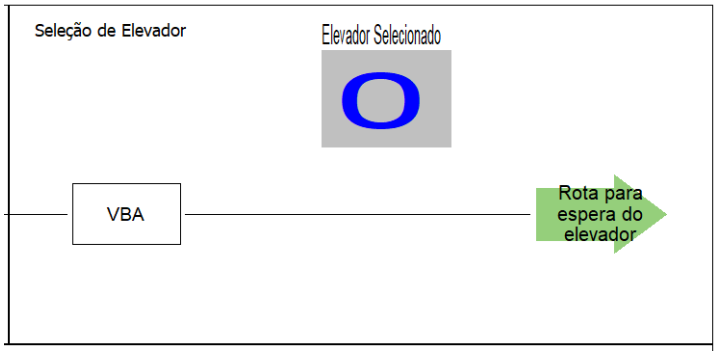
3.4.4 Pessoa esperando elevador

O módulo da “pessoa esperando o elevador no arena” (Figura 3.9) desempenha a função de ajustar as variáveis referentes à quantidade de pessoas que entrarão no andar e que sairão no andar. Este ajuste é feito com base nos atributos de andar de origem e andar de destino da pessoa. Além disso, o módulo é responsável por conduzir a movimentação das pessoas até a entrada no elevador.

3.4.5 Saída das Pessoas

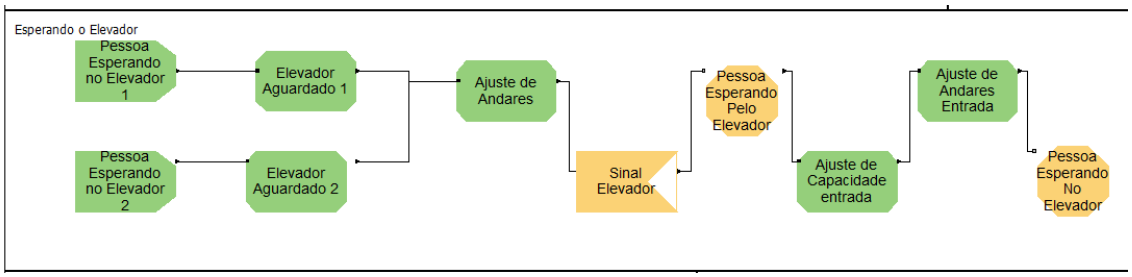
O módulo da “saída de pessoas” (Figura 3.10) desempenha a função de retirar a pessoas da quantidade de pessoas que sairão no andar, com base no atributo de andar de

Figura 3.8 – Seleção de Elevadores no Arena



Fonte: Produzido pela autora.

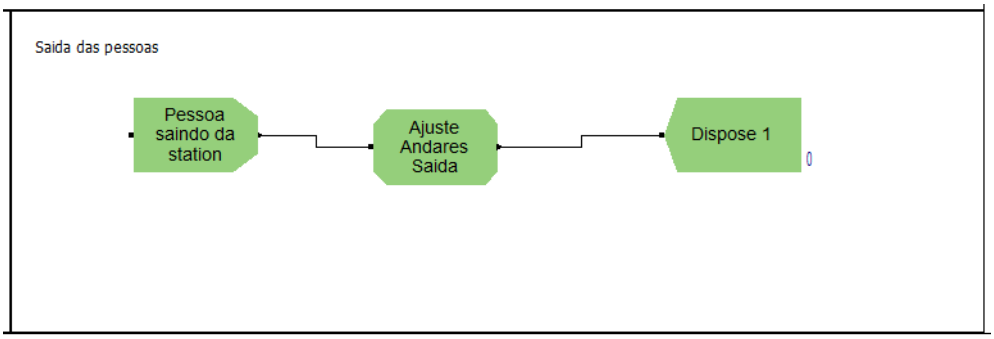
Figura 3.9 – Pessoa Esperando o Elevador no Arena



Fonte: Produzido pela autora.

destino da pessoa. Além disso, o módulo é responsável retirar a entidade do usuário da simulação.

Figura 3.10 – Saída de Pessoas no Arena

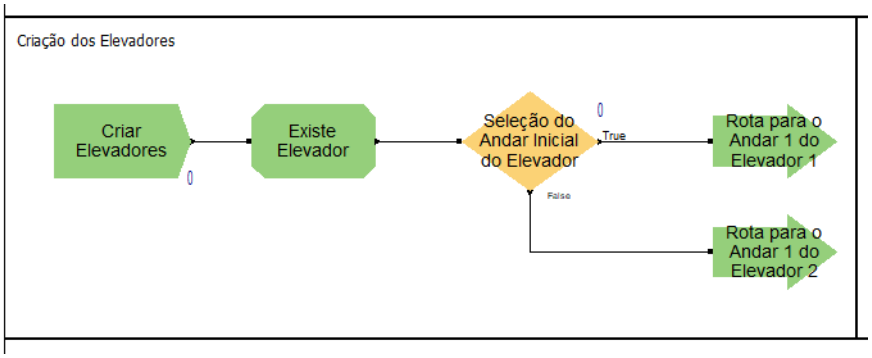


Fonte: Produzido pela autora.

3.4.6 Criação dos Elevadores

O módulo da “criação dos elevadores” pode ser visualizado na Figura 3.11. Este desempenha a função de criar as duas entidades do tipo “elevador” e as inserir nas estações de cada elevador no 1º andar.

Figura 3.11 – Criação das entidades dos Elevadores no Arena

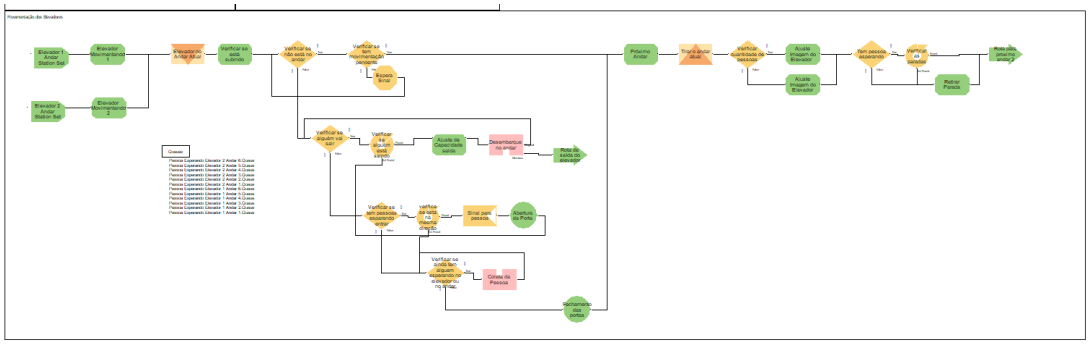


Fonte: Produzido pela autora.

3.4.7 Movimentação do Elevador

O último módulo da simulação é o de movimentação do elevador, que pode ser observado completamente na Figura 3.12 e desempenha a função de movimentar o elevadores, conforme o nome, de acordo com as chamadas.

Figura 3.12 – Movimentação dos Elevadores Completo no Arena



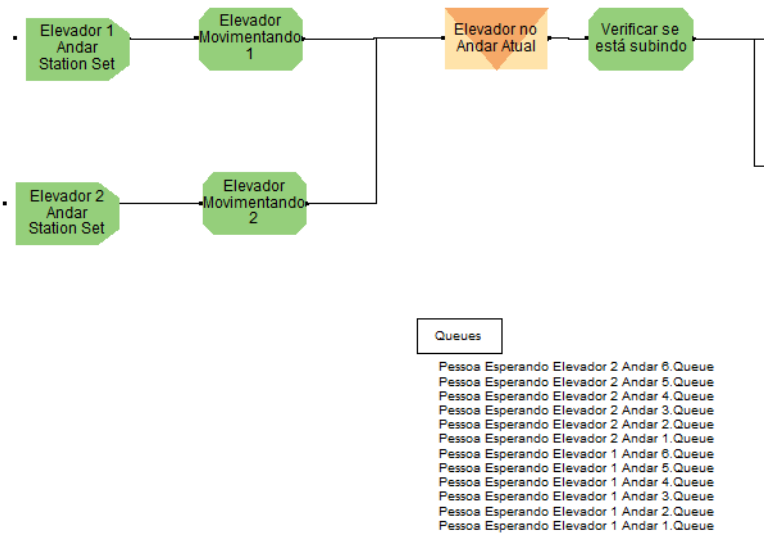
Fonte: Produzido pela autora.

Para facilitar a visualização deste módulo, separou-se a imagem em 5 partes, sendo elas:

- **Figura 3.13:** que é a parte inicial do processamento da estação do andar, onde se: verifica qual elevador está sendo movimentado, estabiliza-se a imagem do elevador na

simulação enquanto o mesmo é processado e verifica se o elevador está no momento subindo ou descendo.

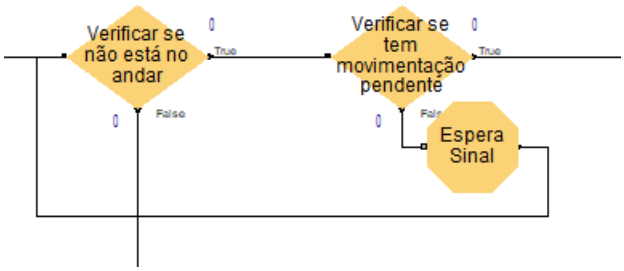
Figura 3.13 – Movimentação dos Elevadores da Parte de Chegada no Andar



Fonte: Produzido pela autora.

- **Figura 3.14:** que é parte de verificação se o elevador precisa sair do andar, se o elevador não tem nenhuma movimentação pendente e se tem pessoas que precisam sair ou entrar no elevador no andar que ele se encontra.

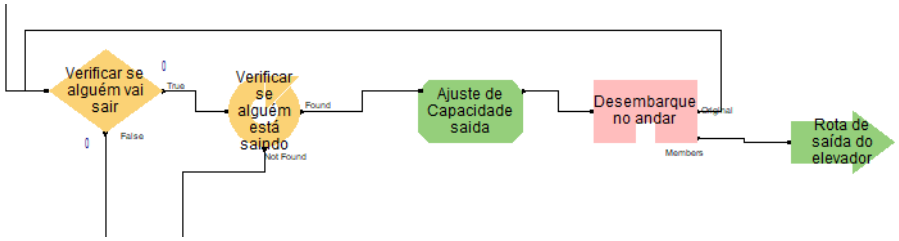
Figura 3.14 – Movimentação dos Elevadores da Parte de Verificação de Movimento



Fonte: Produzido pela autora.

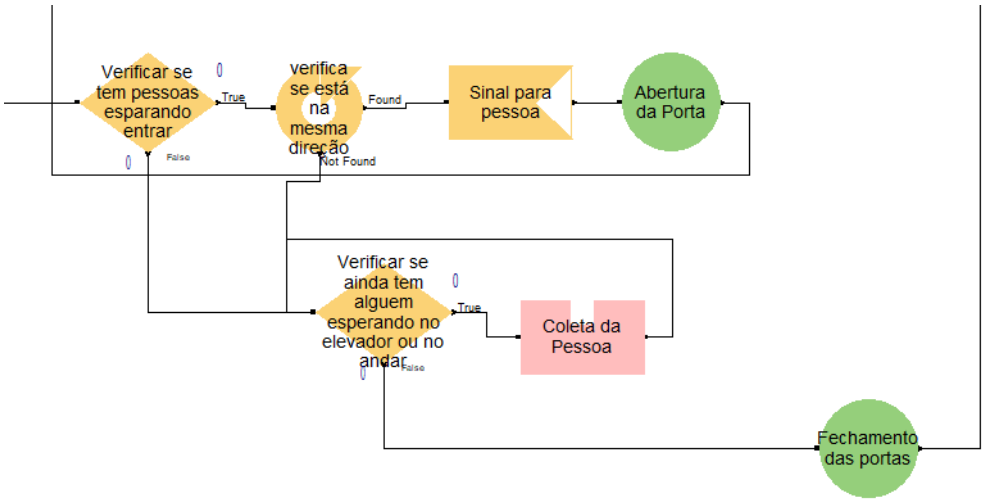
- **Figura 3.15:** parte onde é feita a verificação se tem uma pessoa que precisa desembarcar no andar em que o elevador se encontra.
- **Figura 3.16:** parte onde é feita a verificação se tem pessoas que entrarão naquele andar, de forma a não ultrapassar a capacidade máxima do elevador, ou seja, é nesse módulo que a entidade pessoa é retirada das filas de espera do elevador.

Figura 3.15 – Movimentação dos Elevadores da Parte de Verificação de Saída de Pessoas



Fonte: Produzido pela autora.

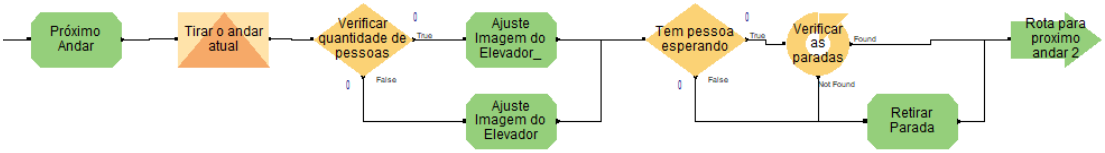
Figura 3.16 – Movimentação dos Elevadores da Parte de Verificação de Entrada de Pessoas



Fonte: Produzido pela autora.

- **Figura 3.17**, que: define o próximo andar, baseado nas paradas que o elevador precisa realizar; ajusta a imagem do elevador baseado na quantidade de pessoas que estão no elevador; e verifica a necessidade de retirar o andar dos locais em que o elevador precisa parar.

Figura 3.17 – Movimentação dos Elevadores da Parte de Chegada no Andar



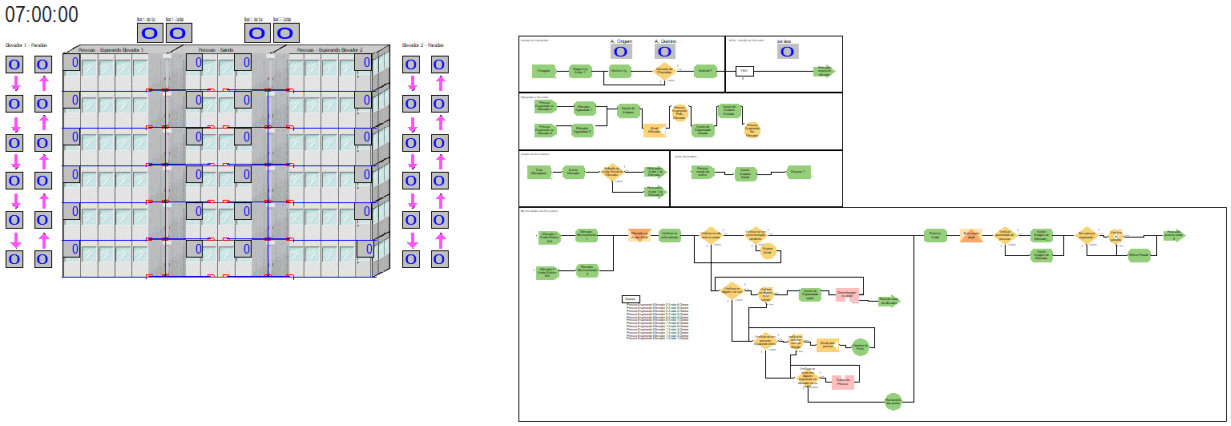
Fonte: Produzido pela autora.

4 Análises dos Resultados

Na Figura 4.1 é possível verificar a modelagem resultante da simulação, a qual foi utilizada para realizar os testes com as diferentes quantidades de pessoas. Por conta do gráfico do software, não é possível nessa figura observar todos os detalhes, porém eles foram todos especificados no Capítulo 3.

A simulação e os seus resultados foram disponibilizados no Github a partir do link https://github.com/clarafortes12/tcc_simulacaoElevador.

Figura 4.1 – Resultado final da Simulação



Fonte: Produzido pela autora.

Os testes foram realizados com diferentes quantidades de pessoas no sistema, sendo 6,25%(15$pessoas$), 12,5%(30$pessoas$) e 25%(60$pessoas$) da população do prédio. Ao final dos testes, foram gerados relatórios pelo simulador Arena, que podem ser observados no Apêndice A, B e C.

A partir dos testes realizados foi possível coletar os tempos (Tabela 4.1): de espera, que é o tempo que o usuário permanece na fila antes de ser atendido pelo elevador; de viagem ou voo, que é o tempo que o passageiro ficou no elevador até chegar no seu destino; e de destino, que é o tempo total em que o passageiro sai do seu andar de origem e chega no seu andar de destino.

Uma análise que pode ser observada é que quando se tem menos usuários, pelo fato de os carros estarem inicialmente no térreo e 90% das chamadas terem este andar como origem, o atendimento das chamadas é mais rápido, ou seja, a espera é menor. Em contra partida, quando os elevadores já estão sendo usados, os tempos de espera aumentam, visto que o elevador não se encontra mais no térreo e precisa se locomover para lá antes de atender

Tabela 4.1 – Resultados dos tempos

Porcentagem da População	Quantidade de Passageiros	Tempo médio de espera (s)	Tempo médio de voo (s)	Tempo médio de destino (s)
6,25%	15	115,992	41,328	157,32
12,5%	30	152,676	41,328	194,004
25%	60	385,092	45,756	430,848

Fonte: Produzido pela autora.

a maioria das chamadas.

Outra análise importante a ser confeccionada é a verificação de quantas pessoas foram atendidas por cada elevador, o que pode ser visto na Tabela 4.2. Essa tabela nos faz perceber que inicialmente o código da lógica fuzzy tem uma tendência a selecionar o elevador 1, mas que com o aumento dos passageiros tende a selecionar o elevador 2.

Tabela 4.2 – Número de passageiros atendidos por cada elevador

Porcentagem da População	Quantidade de Passageiros	Elevador 1	Elevador 2
6,25%	15	11	4
12,5%	30	16	14
25%	60	18	43

Fonte: Produzido pela autora.

O teste com 12,5%(30$pessoas$) da população do prédio foi filmado e disponibilizado no Youtube a partir do link https://www.youtube.com/watch?v=iYAOfFz8jUM .

Além da análise dos resultados obtidos, realizou-se a comparação com outros trabalhos, que pode ser observado na Tabela 4.3. O que nos permite perceber, ao realizar uma análise qualitativa, pode se perceber que os resultados são bem priores que os dos demais autores.

Com relação as comparações, é possível que o grande aumento seja decorrente da forma como o sistema de movimentação do elevador foi feito, pois, na simulação, para permitir que o sistema realizasse a animação em conjunto com o sistema, se mostrou necessário adicionar tempos de intervalo para a confecção da animação, pois todo o sistema é feito de forma sequencial, sendo assim, o tempo de animação é contabilizado nos tempos do programa.

Outra possibilidade para a grande diferença dos tempos de espera e de destino, pode ter ocorrido por conta da necessidade da simulação de se ter um módulo de delay denominado de "abrir a porta do elevador", que é acionado toda vez que uma pessoa entra no elevador, o

Tabela 4.3 – Resultados dos outros autores

Trabalho	Lógica	Tempo médio de espera (s)	Tempo médio de voo (s)	Tempo médio de destino (s)
Forero (2010)	Nebulosa	47	20	67
Wu e Wu (2012)	Feedback Pseudo Diferencial (PDF)	35,85	15,83	51,68
Gu (2012)	Otimização de enxame de partículas em tempo real (RPSO)	44,87	38,34	83,21
Bolat, Altun e Cortés (2013)	Otimização de Enxame de Partículas (PSO)	x	x	33
Rodríguez (2016)	Otimização Binário de Enxame de Partículas (BPSO)	43,37	25,5	68,87
Este trabalho	Nebulosa	385,092	45,756	430,848

Fonte: Adaptado Rodríguez (2016)

que não ocorre no sistema real. Este problema de delay não foi possível ser solucionado por conta da limitação de módulos e variáveis existentes na licença estudantil do software, pois quando tentava realizar uma mudança no sistema, ultrapassava o limite e não era permitido realizar a animação.

Além disso, os valores discrepantes pode ser por conta da utilização de dois elevadores, em vez de o mínimo(três) exigido em um conjunto de elevadores, fazendo com que a quantidade de pessoas demandas para apenas 1 elevador seja maior, consequentemente, fazendo com que o tempo de espera seja maior. Por fim, outra possibilidade seria de o sistema está fazendo uma seleção de elevadores viciadas, ou seja, dando preferência a um elevador, quando na verdade deveria selecionar outro elevador, que pode ser por conta do ajuste atrasado das variáveis da simulação coletadas ou de alguma variável linguística está influenciando mais que outras.

5 Conclusões

O presente trabalho se propôs a aplicar a lógica nebulosa em um problema de controle de elevadores, utilizando o software Arena para a realização da modelagem e simulação de um prédio comercial funcionando no período do padrão up-peak. As simulações conseguiram realizar o transporte pelos diferentes andares com um tempo de espera de aproximadamente 6 minutos, considerando o tempo real, ou seja, a simulação cumpriu com o seu objetivo principal.

Entretanto, os resultados adquiridos nas simulações não foram satisfatórios em comparação com outras literaturas já existentes. Isso indica que ainda existem melhorias a serem realizadas na simulação, para que a mesma possa otimizar o tempo de espera e viagem de um elevador.

Desta forma, a maior contribuição deste trabalho foi o desenvolvimento de um simulador que reproduz, em tempo real, as trajetórias dos passageiros e carros, com cada um deles agindo de forma independente e que cada pessoa deve esperar pelos respectivos carros chegar na sua origem e no seu destino.

5.1 Sugestões de trabalhos futuros

Uma possibilidade para melhorar o sistema é analisar a modelagem da movimentação do elevador e fazer uma simplificação nos tempos de animação. Além disso, podem-se acrescentar outros variáveis linguísticas para melhor seleção dos elevadores ou ajustar as funções de pertinências de forma a se adequar melhor ao sistema de grupo de elevadores.

Outra possibilidade de trabalho futuro seria a troca do módulo de movimentação dos elevadores pelo sistema de grupo de elevadores que se encontra no GRACO, laboratório da UnB, de forma a ter um sistema físico mais próximo do real. Mas para tal implementação, seria necessário a confecção de um cliente OPC para que possam ser enviadas as informações para o CLP de controle do elevador.

Referências

ALTIOK, T.; MELAMED, B. **Simulation modeling and analysis with Arena**. Oxford, UK: Elsevier, 2010. Citado na p. 32.

AMARAL, G. C. d.; COSTA, H. J. d. A. Projeto, instalação e controle de um sistema de elevadores usando tecnologias de automação industrial. 2015. Citado nas pp. 8, 22, 23, 24 e 25.

ÁVILA, R. V. Controle ótimo para grupo de elevadores usando programação dinâmica. 2019. Citado nas pp. 15, 30 e 31.

BOJADZIEV, G.; BOJADZIEV, M. **Fuzzy logic for business, finance, and management**. Singapura: World Scientific, 2007. v. 23. Citado nas pp. 27 e 28.

BOLAT, B.; ALTUN, O.; CORTÉS, P. A particle swarm optimization algorithm for optimal car-call allocation in elevator group control systems. **Applied Soft Computing**, Elsevier, v. 13, n. 5, p. 2633–2642, 2013. Citado na p. 58.

CORTÉS, P.; ASTOLA, J. C. L.; ONIEVA, L.; MUÑUZURI, J.; CABO, I. Fernández de. Algoritmos de optimización en sistemas de transporte vertical. *In*: ASOCIACIÓN PARA EL DESARROLLO DE LA INGENIERÍA DE LA ORGANIZACIÓN. II Congreso de Ingeniería de la Organización, Vigo. p. 657–664. Citado na p. 14.

FABRO, J. A.; OLIVEIRA, A. S. de. **Lógica Fuzzy**. 2018. Accessed: 2023–11-10. Disponível em: https://pessoal.dainf.ct.utfpr.edu.br/andreoliveira/lib/exe/fetch.php?media=aula_logica_fuzzy_v2.pdf. Citado nas pp. 25, 27 e 31.

FERNÁNDEZ, J.; CORTÉS, P.; GUADIX, J.; NUZURI, J. M. Dynamic fuzzy logic elevator group control system for energy optimization. **International Journal of Information Technology & Decision Making**, v. 12, n. 03, p. 591–617, 2013. Disponível em: https://doi.org/10.1142/S0219622013500223. Citado na p. 22.

FORERO, A. A. P. **Estudo e simulação de técnicas de controle de tráfego de grupo de elevadores usando automação industrial**. Dissertação (Mestrado), 2010. Citado nas pp. 12, 14, 17, 18, 19, 21, 22, 24, 25, 26, 29, 30, 40, 41, 42, 43, 46, 47, 49, 50 e 58.

GU, Y. Multi-objective optimization of multi-agent elevator group control system based on real-time particle swarm optimization algorithm. Scientific Research Publishing, v. 4, n. 3, p. 368–378, 2012. Citado na p. 58.

GUSTIN, B.; DEIFAN, G. **Aplicação de redes de Petri interpretadas na modelagem de sistemas de elevadores em edifícios inteligentes.** Tese (Doutorado) — Universidade de São Paulo, 1999. Citado nas pp. 18 e 19.

HUMMET, G. T.; MOSER, T. D.; POWELL, B. A. **Real time simulation of elevators**. Winter Simulation Conference, Miami Beach, 1978. 393–402 p. Citado nas pp. 10 e 21.

JERABEK, A.; CAPUTO, L.; BRANDL, K. **Referência do VBA do Office**. 2023. Accessed: 2023-12-12. Disponível em: https://learn.microsoft.com/pt-br/office/vba/api/overview/. Citado nas pp. 30 e 31.

MAMDANI. Application of fuzzy logic to approximate reasoning using linguistic synthesis. **IEEE transactions on computers**, IEEE, v. 100, n. 12, p. 1182–1191, 1977. Citado na p. 29.

MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. **International journal of man-machine studies**, Elsevier, v. 7, n. 1, p. 1–13, 1975. Citado na p. 29.

MONTEZANO, A. F. M. Modelo em rede de petri de um sistema de automaçao de elevador de passageiros. Universidade Federal do Rio de Janeiro, 2009. Citado nas pp. 19, 20, 21 e 24.

PAULA, P. M. d. **Protótipo de elevador didático utilizando controlador programável com PIC16F877A**. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2014. Citado na p. 18.

ROCKWELL Automation donates Arena Simulation Software to support COVID-19 vaccination efforts. 2021. Accessed: 2023-12-12. Disponível em: https://www.itrportal.com/articles/2021/04/13/rockwell-automation-donates-arena-simulation-software-to-support-covid-19-vaccination-effor Citado na p. 32.

ROCKWELL AUTOMATION TECHNOLOGIES. **Arena User's Guide**. United States of America, 2007. Disponível em: https://studerende.au.dk/fileadmin/www.medarbejdere.au.dk/it/BSS_Analysevaerktoejer/Arena/Arena_User_s_Guide_EN.pdf. Citado nas pp. 31 e 32.

RODRÍGUEZ, J. P. D. **Otimização de controle de tráfego em grupo de elevadores com algoritmos bioinspirados**. Dissertação (Mestrado), 2016. Citado nas pp. 12, 14, 15, 17, 32, 41, 47 e 58.

SANDRI, S.; CORREA, C. Lógica nebulosa. **Escola de redes neurais: conselho nacional de redes neurais**, ITA São José dos Campos, v. 5, p. 73–90, 1999. Citado nas pp. 25, 28 e 29.

SIIKONEN, M.-L. **Elevator group control with artificial intelligence**. Helsinki, Finland: Citeseer, 1997. Citado na p. 47.

SILVA, A. V. G. da; LIMA, C. J. de; CALLOU, G. R. de A. Análise de desempenho do ambiente virtual de aprendizagem na nuvem privada apache cloudstack. **GESTÃO. Org**,

Universidade Federal de Pernambuco (UFPE), v. 17, n. 8, p. 120–133, 2019. Citado na p. 24.

SIMÕES, M. G.; SHAW, I. S. **Controle e modelagem fuzzy**. Brasil: Editora Blucher, 2007. Citado nas pp. 26 e 27.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. **IEEE transactions on systems, man, and cybernetics**, IEEE, n. 1, p. 116–132, 1985. Citado na p. 29.

THYSSENKRUPP DSC). Accessed: 2023–12-20. Disponível em: https://elevation.fandom.com/wiki/Thyssenkrupp_DSC. Citado na p. 14.

TSUKAMOTO, Y. An approach to fuzzy reasoning method. **Advances in fuzzy set theory and applications**, North Holland, 1979. Citado na p. 29.

WU, S.; WU, G. A novel elevator group control scheduling algorithm based on pseudo differential feedback. *In*: IEEE. **2012 IEEE International Conference on Automation and Logistics**. Zhengzhou, China, 2012. p. 111–115. Citado na p. 58.

ZADEH, L. A. Fuzzy sets. **Information and control**, Elsevier, v. 8, n. 3, p. 338–353, 1965. Citado nas pp. 25 e 27.

Apêndices

Apêndice A – Código de Seleção

A.1 Código da função trimp

Código desenvolvido para a implementação da função de pertinência desenvolvida por meio de funções triangulares, conforme Equação 2.1.

Código A.1 – Código trimp

```
Function trimf(x As Variant, a As Double, b As Double, c As Double) As
    Variant
    Dim y(100) As Double

    For intI = 0 To UBound(y)
        If x(intI) > a And x(intI) < b Then
            y(intI) = (x(intI) - a) / (b - a)
        ElseIf x(intI) > b And x(intI) < c Then
            y(intI) = (c - x(intI)) / (c - b)
        ElseIf x(intI) = b Then
            y(intI) = 1
        Else
            y(intI) = 0
        End If

    Next

    trimf = y
End Function
```

A.2 Código da Função trapmf

Código desenvolvido para a implementação da função de pertinência desenvolvida por meio de funções trapezoidais, conforme Equação 2.2.

Código A.2 – Código trapmf

```
Function trapmf(x As Variant, a As Double, b As Double, c As Double, d
    As Double) As Variant
    Dim y(100) As Double

    For intI = 0 To UBound(y)
        If x(intI) <= b Then
            If x(intI) > a And x(intI) < b Then
                y(intI) = (x(intI) - a) / (b - a)
            ElseIf x(intI) = b Then
                y(intI) = 1
            Else
```

```
                y(intI) = 0
            End If
        ElseIf x(intI) >= c Then
            If x(intI) > c And x(intI) < d Then
                y(intI) = (d - x(intI)) / (d - c)
            ElseIf x(intI) = c Then
                y(intI) = 1
            Else
                y(intI) = 0
            End If
        ElseIf x(intI) < a Then
            y(intI) = 0
        ElseIf x(intI) > d Then
            y(intI) = 0
        Else
            y(intI) = 1
        End If

    Next

    trapmf = y
End Function
```

A.3 Código da Função interpolação

Código desenvolvido para a implementação da função de interpolação, conforme Equação 2.2.

Código A.3 – Código interpolacao

```
Function interpolacao(x As Variant, y As Variant, ponto As Double) As
   Double
'f(x)=f(x0) + (((f(x1)-f(x0))*(x-x0))/(x1 - x0))
'f(x) interpolacao para o ponto x
'x0 e x1 pontos conhecidos próximos a x

    Dim x0 As Double
    Dim x1 As Double
    Dim y0 As Double
    Dim y1 As Double
    Dim resultadoInterp As Double

'Encontrar os pontos conhecidos mais próximos a x
    For i = 0 To UBound(x)
        If x(i) <= ponto Then
            x0 = x(i)
            y0 = y(i)
        End If
        If x(i) >= ponto Then
            x1 = x(i)
            y1 = y(i)
            Exit For
```

```
22          End If
23      Next i
24
25      If x0 = x1 Then
26          resultadoInterp = y0
27      Else
28          'Calcular o valor interpolado usando a fórmula de interpolação
               linear
29          resultadoInterp = y0 + (((y1 - y0) * (ponto - x0)) / (x1 - x0))
30      End If
31
32      'Retornar o resultado interpolado
33      interpolacao = resultadoInterp
34
35  End Function
```

A.4 Código da Função minimoEntreDoisNumeros

Função desenvolvida para verificar qual o valor mínimo entre dois números.

Código A.4 – Código minimoEntreDoisNumeros

```
1  Function minimoEntreDoisNumeros(numero1 As Double, numero2 As Double) As
       Double
2      If numero1 < numero2 Then
3          minimoEntreDoisNumeros = numero1
4      Else
5          minimoEntreDoisNumeros = numero2
6      End If
7  End Function
```

A.5 Código da Função maximoEntreDoisNumeros

Função desenvolvida para verificar qual o valor máximo entre dois números.

Código A.5 – Código maximoEntreDoisNumeros

```
1  Function maximoEntreDoisNumeros(numero1 As Double, numero2 As Double) As
       Double
2      If numero1 > numero2 Then
3          maximoEntreDoisNumeros = numero1
4      Else
5          maximoEntreDoisNumeros = numero2
6      End If
7  End Function
```

A.6 Código da Função minimoDoVetor

Função desenvolvida para verificar qual o valor mínimo entre cada elemento de um vetor e um número de referência.

Código A.6 – Código minimoDoVetor

```
Function minimoDoVetor(minValor As Double, vetor As Variant) As Variant
    For i = 0 To UBound(vetor)
        If vetor(i) > minValor Then
            vetor(i) = minValor
        End If
    Next i

    minimoDoVetor = vetor
End Function
```

A.7 Código da Função somaVetor

Função desenvolvida para realizar a soma dos valores de todos os elementos de um vetor.

Código A.7 – Código somaVetor

```
Function somaVetor(vetor As Variant) As Double
    Dim soma As Double
    soma = 0

    For i = 0 To UBound(vetor)
        soma = soma + vetor(i)
    Next i

    somaVetor = soma
End Function
```

A.8 Código da Função maximoDosVetores

Função desenvolvida pare verificar o máximo valor entre os elementos de vários vetores, gerando um vetor com os elementos máximos.

Código A.8 – Código maximoDosVetores

```
Function maximoDosVetores(vetor As Variant) As Variant
    Dim numeroVetores As Double
    Dim numeroElementos As Double
    Dim valorMaximo As Double
    Dim valorResultante(100) As Double

    numeroVetores = UBound(vetor)
```

```
8       numeroElementos = UBound(vetor(0))
9
10      For i = 0 To numeroElementos
11          valorResultante(i) = 0
12          For j = 0 To numeroVetores
13              If vetor(j)(i) > valorResultante(i) Then
14                  valorResultante(i) = vetor(j)(i)
15              End If
16          Next j
17      Next i
18
19      maximoDosVetores = valorResultante
20
21  End Function
```

A.9 Código da Função defuzzificacao

Função desenvolvida para a confecção da defuzzificação do sistemas estudado, utilizando método Centro de Área (centróide), conforme a Equação 2.7.

Código A.9 – Código defuzzificacao

```
1   Function defuzzificacao(normalizacao As Variant, saida As Variant) As
        Double
2
3       Dim valorDefuzzificacao As Double
4       Dim somaMomentoArea As Double
5       Dim somaArea As Double
6
7       Dim x1 As Double
8       Dim x2 As Double
9       Dim y1 As Double
10      Dim y2 As Double
11
12      Dim momentoArea As Double
13      Dim area As Double
14
15      Dim epsilon As Double
16      epsilon = EPS
17
18
19      If somaVetor(saida) = 0 Then
20          valorDefuzzificacao = 0
21      Else
22          somaMomentoArea = 0
23          somaArea = 0
24
25          If UBound(normalizacao) = 1 Then
26              somaMomentoArea = normalizacao(0) * saida(0)
27              somaArea = saida(0)
28          Else
29              'centroide
```

```
30              For i = 1 To UBound(normalizacao)
31                  x1 = normalizacao(i - 1)
32                  x2 = normalizacao(i)
33                  y1 = saida(i - 1)
34                  y2 = saida(i)
35
36                  If ((y1 <> 0 And y2 <> 0) Or x1 <> x2) Then
37                      If y1 = y2 Then
38                          momentoArea = 0.5 * (x1 + x2)
39                          area = (x2 - x1) * y1
40                      ElseIf y1 = 0 And y2 <> 0 Then
41                          momentoArea = ((2 / 3) * (x2 - x1)) + x1
42                          area = 0.5 * (x2 - x1) * y2
43                      ElseIf y2 = 0 And y1 <> 0 Then
44                          momentoArea = ((1 / 3) * (x2 - x1)) + x1
45                          area = 0.5 * (x2 - x1) * y1
46                      Else
47                          momentoArea = (((2 / 3) * (x2 - x1) * (y2 + 0.5
                                * y1)) / (y1 + y2)) + x1
48                          area = 0.5 * (x2 - x1) * (y1 + y2)
49                      End If
50                      somaMomentoArea = somaMomentoArea + (momentoArea *
                            area)
51                      somaArea = somaArea + area
52                  End If
53              Next i
54          End If
55
56          valorDefuzzificacao = somaMomentoArea /
                maximoEntreDoisNumeros(somaArea, epsilon)
57      End If
58
59      defuzzificacao = valorDefuzzificacao
60
61  End Function
```

A.10 Código da Função resultadoFuzzi

Função desenvolvida para a confecção da análise de todo o sistema nebuloso desde a fuzzificação até a defuzzificação para um elevador.

Código A.10 – Código resultadoFuzzi

```
1   Function resultadoFuzzi(distanciaElevador1Normalizada As Double,
        tempoEsperaElevador1 As Double, cargaElevador1Normalizada As Double)
        As Double
2       Const tamanho As Integer = 100
3       Dim normalizacao(tamanho) As Double
4
5       Dim tempoEsperaBaixo As Variant
6       Dim tempoEsperaMedio As Variant
7       Dim tempoEsperaAlto As Variant
```

```
8       Dim disponibilidadeCargaBaixo As Variant
9       Dim disponibilidadeCargaMedio As Variant
10      Dim disponibilidadeCargaAlto As Variant
11      Dim distanciaBaixo As Variant
12      Dim distanciaMedio As Variant
13      Dim distanciaAlto As Variant
14      Dim prioridadeBaixo As Variant
15      Dim prioridadeMedio As Variant
16      Dim prioridadeAlto As Variant
17
18      Dim tempoEsperaBaixoElevador1 As Double
19      Dim tempoEsperaMedioElevador1 As Double
20      Dim tempoEsperaAltoElevador1 As Double
21      Dim cargaBaixoElevador1 As Double
22      Dim cargaMedioElevador1 As Double
23      Dim cargaAltoElevador1 As Double
24      Dim distanciaBaixoElevador1 As Double
25      Dim distanciaMedioElevador1 As Double
26      Dim distanciaAltoElevador1 As Double
27
28      'Regras Elevador 1
29      Dim rule1Elevador1 As Variant
30      Dim rule2Elevador1 As Variant
31      Dim rule3Elevador1 As Variant
32      Dim rule4Elevador1 As Variant
33      Dim rule5Elevador1 As Variant
34      Dim rule6Elevador1 As Variant
35      Dim rule7Elevador1 As Variant
36      Dim rule8Elevador1 As Variant
37      Dim rule9Elevador1 As Variant
38      Dim rule10Elevador1 As Variant
39      Dim rule11Elevador1 As Variant
40
41      Dim saidaBaixaElevador1 As Variant
42      Dim saidaMediaElevador1 As Variant
43      Dim saidaAltaElevador1 As Variant
44
45      Dim saidaFinalElevador1 As Variant
46      Dim defuzzificacaoElevador1 As Double
47
48
49      For intI = 0 To UBound(normalizacao)
50          normalizacao(intI) = intI / UBound(normalizacao)
51      Next
52
53      tempoEsperaBaixo = trapmf(normalizacao, 0, 0, 0.3, 0.5)
54      tempoEsperaMedio = trimf(normalizacao, 0.25, 0.55, 0.85)
55      tempoEsperaAlto = trimf(normalizacao, 0.6, 1, 1)
56
57      disponibilidadeCargaBaixo = trapmf(normalizacao, 0, 0, 0.3, 0.5)
58      disponibilidadeCargaMedio = trimf(normalizacao, 0.25, 0.55, 0.85)
59      disponibilidadeCargaAlto = trimf(normalizacao, 0.6, 1, 1)
60
```

```
61      distanciaBaixo = trimf(normalizacao, 0, 0, 0.4)
62      distanciaMedio = trimf(normalizacao, 0.2, 0.45, 0.8)
63      distanciaAlto = trimf(normalizacao, 0.7, 1, 1)
64
65      prioridadeBaixo = trimf(normalizacao, 0, 0, 0.4)
66      prioridadeMedio = trimf(normalizacao, 0.1, 0.5, 0.9)
67      prioridadeAlto = trimf(normalizacao, 0.6, 1, 1)
68
69      tempoEsperaBaixoElevador1 = interpolacao(normalizacao,
            tempoEsperaBaixo, tempoEsperaElevador1)
70      tempoEsperaMedioElevador1 = interpolacao(normalizacao,
            tempoEsperaMedio, tempoEsperaElevador1)
71      tempoEsperaAltoElevador1 = interpolacao(normalizacao,
            tempoEsperaAlto, tempoEsperaElevador1)
72
73      cargaBaixoElevador1 = interpolacao(normalizacao,
            disponibilidadeCargaBaixo, cargaElevador1Normalizada)
74      cargaMedioElevador1 = interpolacao(normalizacao,
            disponibilidadeCargaMedio, cargaElevador1Normalizada)
75      cargaAltoElevador1 = interpolacao(normalizacao,
            disponibilidadeCargaAlto, cargaElevador1Normalizada)
76
77      distanciaBaixoElevador1 = interpolacao(normalizacao, distanciaBaixo,
            distanciaElevador1Normalizada)
78      distanciaMedioElevador1 = interpolacao(normalizacao, distanciaMedio,
            distanciaElevador1Normalizada)
79      distanciaAltoElevador1 = interpolacao(normalizacao, distanciaAlto,
            distanciaElevador1Normalizada)
80
81      rule1Elevador1 =
            minimoDoVetor(minimoEntreDoisNumeros(distanciaAltoElevador1,
            tempoEsperaAltoElevador1), prioridadeBaixo)
82      rule2Elevador1 =
            minimoDoVetor(minimoEntreDoisNumeros(distanciaAltoElevador1,
            tempoEsperaBaixoElevador1), prioridadeMedio)
83      rule3Elevador1 =
            minimoDoVetor(minimoEntreDoisNumeros(distanciaAltoElevador1,
            tempoEsperaMedioElevador1), prioridadeBaixo)
84      rule4Elevador1 =
            minimoDoVetor(minimoEntreDoisNumeros(distanciaBaixoElevador1,
            tempoEsperaAltoElevador1), prioridadeMedio)
85      rule5Elevador1 =
            minimoDoVetor(minimoEntreDoisNumeros(distanciaBaixoElevador1,
            tempoEsperaBaixoElevador1), prioridadeAlto)
86      rule6Elevador1 =
            minimoDoVetor(minimoEntreDoisNumeros(distanciaBaixoElevador1,
            tempoEsperaMedioElevador1), prioridadeMedio)
87      rule7Elevador1 =
            minimoDoVetor(minimoEntreDoisNumeros(distanciaMedioElevador1,
            tempoEsperaMedioElevador1), prioridadeMedio)
88      rule8Elevador1 =
            minimoDoVetor(minimoEntreDoisNumeros(distanciaMedioElevador1,
            tempoEsperaAltoElevador1), prioridadeBaixo)
```

```
89      rule9Elevador1 =
           minimoDoVetor(minimoEntreDoisNumeros(cargaBaixoElevador1,
           tempoEsperaAltoElevador1), prioridadeBaixo)
90      rule10Elevador1 =
           minimoDoVetor(minimoEntreDoisNumeros(distanciaBaixoElevador1,
           cargaBaixoElevador1), prioridadeBaixo)
91      rule11Elevador1 =
           minimoDoVetor(minimoEntreDoisNumeros(cargaMedioElevador1,
           tempoEsperaBaixoElevador1), prioridadeAlto)
92
93      saidaBaixaElevador1 = maximoDosVetores(Array(rule1Elevador1,
           rule3Elevador1, rule8Elevador1, rule9Elevador1, rule10Elevador1))
94      saidaMediaElevador1 = maximoDosVetores(Array(rule2Elevador1,
           rule4Elevador1, rule6Elevador1, rule7Elevador1))
95      saidaAltaElevador1 = maximoDosVetores(Array(rule5Elevador1,
           rule11Elevador1))
96
97      saidaFinalElevador1 = maximoDosVetores(Array(saidaBaixaElevador1,
           saidaMediaElevador1, saidaAltaElevador1))
98      defuzzificacaoElevador1 = defuzzificacao(normalizacao,
           saidaFinalElevador1)
99
100     resultadoFuzzi = defuzzificacaoElevador1
101
102 End Function
```

A.11 Código da Função selecaoElevador

Função desenvolvida para a confecção da análise e seleção do elevador do conjunto.

Código A.11 – Código selecaoElevador

```
1  Function selecaoElevador(distanciaElevador1Normalizada As Double,
      distanciaElevador2Normalizada As Double, tempoEsperaElevador1 As
      Double, tempoEsperaElevador2 As Double, cargaElevador1Normalizada As
      Double, cargaElevador2Normalizada As Double) As Double
2       Dim elevador1 As Double
3       Dim elevador2 As Double
4       Dim elevadorSelecao As Integer
5       Dim prioridadeMaior As Double
6
7       If (cargaElevador1Normalizada >= 1 And cargaElevador2Normalizada <
           1) Then
8           elevadorSelecao = 2
9       ElseIf (cargaElevador1Normalizada < 1 And cargaElevador2Normalizada
           >= 1) Then
10          elevadorSelecao = 1
11      Else
12          If cargaElevador1Normalizada > 1 Then
13              cargaElevador1Normalizada = 1
14          End If
15
```

```
16          If cargaElevador2Normalizada > 1 Then
17              cargaElevador2Normalizada = 1
18          End If
19
20          If tempoEsperaElevador1 > 1 Then
21              tempoEsperaElevador1 = 1
22          End If
23
24          If tempoEsperaElevador2 > 1 Then
25              tempoEsperaElevador2 = 1
26          End If
27
28          If distanciaElevador1Normalizada > 1 Then
29              distanciaElevador1Normalizada = 1
30          End If
31
32          If distanciaElevador2Normalizada > 1 Then
33              distanciaElevador2Normalizada = 1
34          End If
35
36          elevador1 = resultadoFuzzi(distanciaElevador1Normalizada,
                tempoEsperaElevador1, cargaElevador1Normalizada)
37          elevador2 = resultadoFuzzi(distanciaElevador2Normalizada,
                tempoEsperaElevador2, cargaElevador2Normalizada)
38
39          prioridadeMaior = maximoEntreDoisNumeros(elevador1, elevador2)
40
41          If prioridadeMaior = defuzzificacaoElevador1 Then
42              elevadorSelecao = 1
43          Else
44              elevadorSelecao = 2
45          End If
46      End If
47
48      selecaoElevador = elevadorSelecao
49
50  End Function
```

A.12 Código da Main do Módulo VBA

Função principal que realiza a coleta das informações do modelo simulado e atribui os valores necessários as variáveis para a confecção dos testes.

Código A.12 – Código modulo VBA

```
1  Private Sub VBA_Block_2_Fire()
2
3  Dim s As SIMAN
4  Set s = ThisDocument.Model.SIMAN
5
6  Dim andarElevador1 As Double
7  Dim andarElevador2 As Double
```

```
8  Dim limiteCapacidadeElevador As Double
9  Dim numeroAndares As Double
10 Dim NumParadasElevador1 As Double
11 Dim NumParadasElevador2 As Double
12 Dim tempoVoo As Double
13 Dim tempoParada As Double
14 Dim tempoMaximo As Double
15 Dim distanciaElevador1Normalizada As Double
16 Dim distanciaElevador2Normalizada As Double
17 Dim tempoEsperaElevador1 As Double
18 Dim tempoEsperaElevador2 As Double
19 Dim cargaElevador1 As Double
20 Dim cargaElevador2 As Double
21 Dim cargaElevador1Normalizada As Double
22 Dim cargaElevador2Normalizada As Double
23
24 Dim andarOrigemValor As Integer
25 Dim elevadorSelecionadoValor As Integer
26 Dim andarDestinoValor As Integer
27 Dim isPessoaSubindoValor As Integer
28
29 limiteCapacidadeElevador = 6
30 numeroAndares = 6
31 tempoVoo = 2
32 tempoParada = 3
33 tempoMaximo = ((numeroAndares - 1) * tempoVoo) + (2 * numeroAndares *
       tempoParada)
34
35 andarOrigemValor = s.EntityAttribute(s.ActiveEntity,
       s.SymbolNumber("andarOrigem"))
36 andarDestinoValor = s.EntityAttribute(s.ActiveEntity,
       s.SymbolNumber("andarDestino"))
37 isPessoaSubindoValor = s.EntityAttribute(s.ActiveEntity,
       s.SymbolNumber("isPessoaSubindo"))
38
39 andarElevador1 = s.VariableArrayValue(s.SymbolNumber("estadoElevadores",
       1, 4))
40 andarElevador2 = s.VariableArrayValue(s.SymbolNumber("estadoElevadores",
       2, 4))
41
42 NumParadasElevador1 = 0
43 NumParadasElevador2 = 0
44 For i = 1 To numeroAndares
45     If s.VariableArrayValue(s.SymbolNumber("paradasElevadoresSubindo",
           1, i)) = 1 Then
46         NumParadasElevador1 = NumParadasElevador1 + 1
47     End If
48     If s.VariableArrayValue(s.SymbolNumber("paradasElevadoresSubindo",
           2, i)) = 1 Then
49         NumParadasElevador2 = NumParadasElevador2 + 1
50     End If
51     If s.VariableArrayValue(s.SymbolNumber("paradasElevadoresDescendo",
           1, i)) = 1 Then
```

```
52          NumParadasElevador1 = NumParadasElevador1 + 1
53      End If
54      If s.VariableArrayValue(s.SymbolNumber("paradasElevadoresDescendo",
            2, i)) = 1 Then
55          NumParadasElevador2 = NumParadasElevador2 + 1
56      End If
57  Next i
58
59  cargaElevador1 = s.VariableArrayValue(s.SymbolNumber("estadoElevadores",
        1, 3))
60  cargaElevador2 = s.VariableArrayValue(s.SymbolNumber("estadoElevadores",
        2, 3))
61
62  distanciaElevador1Normalizada = Abs(andarOrigemValor - andarElevador1) /
        (numeroAndares - 1)
63  distanciaElevador2Normalizada = Abs(andarOrigemValor - andarElevador2) /
        (numeroAndares - 1)
64
65  tempoEsperaElevador1 = ((Abs(andarOrigemValor - andarElevador1) *
        tempoVoo) + (2 * NumParadasElevador1 * tempoParada)) / tempoMaximo
66  tempoEsperaElevador2 = ((Abs(andarOrigemValor - andarElevador2) *
        tempoVoo) + (2 * NumParadasElevador2 * tempoEntra)) / tempoMaximo
67
68  cargaElevador1Normalizada = cargaElevador1 / limiteCapacidadeElevador
69  cargaElevador2Normalizada = cargaElevador2 / limiteCapacidadeElevador
70
71  elevadorSelecionadoValor =
        selecaoElevador(distanciaElevador1Normalizada,
        distanciaElevador2Normalizada, tempoEsperaElevador1,
        tempoEsperaElevador2, cargaElevador1Normalizada,
        cargaElevador2Normalizada)
72
73  s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("elevadorSelecionado"))
        = elevadorSelecionadoValor
74
75  If isPessoaSubindoValor = 1 Then
76      s.VariableArrayValue(s.SymbolNumber("paradasElevadoresSubindo",
            elevadorSelecionadoValor, andarOrigemValor)) = 1
77      s.VariableArrayValue(s.SymbolNumber("paradasElevadoresSubindo",
            elevadorSelecionadoValor, andarDestinoValor)) = 1
78  Else
79      s.VariableArrayValue(s.SymbolNumber("paradasElevadoresDescendo",
            elevadorSelecionadoValor, andarOrigemValor)) = 1
80      s.VariableArrayValue(s.SymbolNumber("paradasElevadoresDescendo",
            elevadorSelecionadoValor, andarDestinoValor)) = 1
81  End If
82
83  End Sub
```

Apêndice B – Geração de Gráficos de Funções de Pertinência

Código B.1 – Código de Python para a geração dos gráficos de funções de pertinência

```
import numpy as np
import matplotlib.pyplot as plt
import skfuzzy.membership as mf

normalizacao = np.arange(0, 1, 10**(-2))

tempoEsperaBaixo = mf.trapmf(normalizacao, [0, 0, 0.3, 0.5])
tempoEsperaMedio = mf.trimf(normalizacao, [0.25, 0.55, 0.85])
tempoEsperaAlto = mf.trimf(normalizacao, [0.6, 1, 1])

distanciaBaixo = mf.trapmf(normalizacao, [0, 0, 0.3, 0.5])
distanciaMedio = mf.trimf(normalizacao, [0.25, 0.55, 0.85])
distanciaAlto = mf.trimf(normalizacao, [0.6, 1, 1])

capacidadeBaixo = mf.trimf(normalizacao,[0, 0, 0.4])
capacidadeMedio = mf.trimf(normalizacao,[0.2, 0.45, 0.8])
capacidadeAlto = mf.trimf(normalizacao,[0.7, 1, 1])

prioridadeBaixo = mf.trimf(normalizacao,[0, 0, 0.4])
prioridadeMedio = mf.trimf(normalizacao,[0.1, 0.5, 0.9])
prioridadeAlto = mf.trimf(normalizacao,[0.6, 1, 1])

plt.plot(normalizacao, tempoEsperaBaixo, 'r', linewidth = 2, label =
    'Baixo')
plt.plot(normalizacao, tempoEsperaMedio, 'g', linewidth = 2, label =
    'Médio')
plt.plot(normalizacao, tempoEsperaAlto, 'b', linewidth = 2, label =
    'Alto')
plt.xlabel("Tempo de Espera Normalizado")
plt.ylabel("Grau de pertin ncia")
plt.legend()

plt.tight_layout()
plt.show()


plt.plot(normalizacao, distanciaBaixo, 'r', linewidth = 2, label =
    'Baixa')
plt.plot(normalizacao, distanciaMedio, 'g', linewidth = 2, label =
    'Média')
plt.plot(normalizacao, distanciaAlto, 'b', linewidth = 2, label = 'Alta')
plt.xlabel("Distância Normalizada")
plt.ylabel("Grau de pertin ncia")
```

```
39 plt.legend()
40
41 plt.tight_layout()
42 plt.show()
43
44 plt.plot(normalizacao, capacidadeBaixo, 'r', linewidth = 2, label =
       'Baixa')
45 plt.plot(normalizacao, capacidadeMedio, 'g', linewidth = 2, label =
       'Média')
46 plt.plot(normalizacao, capacidadeAlto, 'b', linewidth = 2, label =
       'Alta')
47 plt.xlabel("Capacidade Normalizada")
48 plt.ylabel("Grau de pertin ncia")
49 plt.legend()
50
51 plt.tight_layout()
52 plt.show()
53
54 plt.plot(normalizacao, prioridadeBaixo, 'r', linewidth = 2, label =
       'Baixa')
55 plt.plot(normalizacao, prioridadeMedio, 'g', linewidth = 2, label =
       'Média')
56 plt.plot(normalizacao, prioridadeAlto, 'b', linewidth = 2, label =
       'Alta')
57 plt.xlabel("Prioridade Normalizada")
58 plt.ylabel("Grau de pertin ncia")
59 plt.legend()
60
61 plt.tight_layout()
62 plt.show()
```

Anexos

Anexo A – Relatório Arena de 15 pessoas

ARENA Simulation Results
maria.clara.of@hotmail.com - License: STUDENT

Sumário para Replicação 1 de 1

Projeto: Unnamed Project Data da execução :12/11/2023
Analista: maria.clara.of@hotmail.com Data de revisão do modelo:12/11/2023

Replicação terminada no tempo de : 0.11944444 Horas
Unidade de Tempo Base: Horas

VARI VEIS DE REGISTRO

Identificador	Média	Meia Largura	Mínimo	Máximo	Observações
Elevador.VATime	--	--	--	--	0
Elevador.NVATime	--	--	--	--	0
Elevador.WaitTime	--	--	--	--	0
Elevador.TranTime	--	--	--	--	0
Elevador.OtherTime	--	--	--	--	0
Elevador.TotalTime	--	--	--	--	0
pessoa.VATime	.00000	(Insuf)	.00000	.00000	15
pessoa.NVATime	.00000	(Insuf)	.00000	.00000	15
pessoa.WaitTime	.03222	(Insuf)	.00556	.08333	15
pessoa.TranTime	.00000	(Insuf)	.00000	.00000	15
pessoa.OtherTime	.01148	(Insuf)	.00556	.02222	15
pessoa.TotalTime	.04370	(Insuf)	.01111	.10556	15
Pessoa Esperando Elevador 1 Andar 1.Queue.	.01970	(Insuf)	.00000	.05556	11
Pessoa Esperando Elevador 1 Andar 2.Queue.	--	--	--	--	0
Pessoa Esperando Elevador 1 Andar 3.Queue.	--	--	--	--	0

```
Pessoa Esperando Elevador 1 Andar 4.Queue.      --         --         --         --
   0
Pessoa Esperando Elevador 1 Andar 5.Queue.      --         --         --         --
   0
Pessoa Esperando Elevador 1 Andar 6.Queue.      --         --         --         --
   0
Pessoa Esperando Elevador 2 Andar 1.Queue.  .01296     (Insuf)    .00000     .02222
   3
Pessoa Esperando Elevador 2 Andar 2.Queue.      --         --         --         --
   0
Pessoa Esperando Elevador 2 Andar 3.Queue.      --         --         --         --
   0
Pessoa Esperando Elevador 2 Andar 4.Queue.      --         --         --         --
   0
Pessoa Esperando Elevador 2 Andar 5.Queue.  .00000     (Insuf)    .00000     .00000
   1
Pessoa Esperando Elevador 2 Andar 6.Queue.      --         --         --         --
   0
No Elevador 1.WaitingTime                   .01818     (Insuf)    .00556     .03333
   11
No Elevador 2.WaitingTime                   .00694     (Insuf)    .00556     .01111
   4
Elevador 1 Andar 6 Parado.WaitingTime           --         --         --         --
   0
Elevador 1 Andar 5 Parado.WaitingTime           --         --         --         --
   0
Elevador 1 Andar 4 Parado.WaitingTime           --         --         --         --
   0
Elevador 1 Andar 3 Parado.WaitingTime           --         --         --         --
   0
Elevador 1 Andar 2 Parado.WaitingTime           --         --         --         --
   0
Elevador 1 Andar 1 Parado.WaitingTime       .00833     (Insuf)    .00833     .00833
   1
Elevador 2 Andar 6 Parado.WaitingTime           --         --         --         --
   0
Elevador 2 Andar 5 Parado.WaitingTime           --         --         --         --
   0
Elevador 2 Andar 4 Parado.WaitingTime       .01667     (Insuf)    .01667     .01667
   1
Elevador 2 Andar 3 Parado.WaitingTime           --         --         --         --
   0
Elevador 2 Andar 2 Parado.WaitingTime           --         --         --         --
   0
Elevador 2 Andar 1 Parado.WaitingTime       .02472     (Insuf)    .02472     .02472
   1
```

VARI VEIS DISCRETAS

```
Identificador                                Média   Meia Largura  Mínimo    Máximo   Valor
   Final
-------------------------------------------------------------------------------------------
```

Elevador.WIP	1.9976	(Insuf)	.00000	2.0000	2.0000
pessoa.WIP	5.4883	(Insuf)	.00000	11.000	.00000
Pessoa Esperando Elevador 1 Andar 1.Queue.	1.8139	(Insuf)	.00000	6.0000	.00000
Pessoa Esperando Elevador 1 Andar 2.Queue.	.00000	(Insuf)	.00000	.00000	.00000
Pessoa Esperando Elevador 1 Andar 3.Queue.	.00000	(Insuf)	.00000	.00000	.00000
Pessoa Esperando Elevador 1 Andar 4.Queue.	.00000	(Insuf)	.00000	.00000	.00000
Pessoa Esperando Elevador 1 Andar 5.Queue.	.00000	(Insuf)	.00000	.00000	.00000
Pessoa Esperando Elevador 1 Andar 6.Queue.	.00000	(Insuf)	.00000	.00000	.00000
Pessoa Esperando Elevador 2 Andar 1.Queue.	.32558	(Insuf)	.00000	2.0000	.00000
Pessoa Esperando Elevador 2 Andar 2.Queue.	.00000	(Insuf)	.00000	.00000	.00000
Pessoa Esperando Elevador 2 Andar 3.Queue.	.00000	(Insuf)	.00000	.00000	.00000
Pessoa Esperando Elevador 2 Andar 4.Queue.	.00000	(Insuf)	.00000	.00000	.00000
Pessoa Esperando Elevador 2 Andar 5.Queue.	.00000	(Insuf)	.00000	1.0000	.00000
Pessoa Esperando Elevador 2 Andar 6.Queue.	.00000	(Insuf)	.00000	.00000	.00000
No Elevador 1.NumberInQueue	1.6744	(Insuf)	.00000	6.0000	.00000
No Elevador 2.NumberInQueue	.23256	(Insuf)	.00000	2.0000	.00000
Elevador 1 Andar 6 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000	.00000
Elevador 1 Andar 5 Parado.NumberInQueue	.00000	(Insuf)	.00000	1.0000	1.0000
Elevador 1 Andar 4 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000	.00000
Elevador 1 Andar 3 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000	.00000
Elevador 1 Andar 2 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000	.00000
Elevador 1 Andar 1 Parado.NumberInQueue	.06977	(Insuf)	.00000	1.0000	.00000
Elevador 2 Andar 6 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000	.00000
Elevador 2 Andar 5 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000	.00000
Elevador 2 Andar 4 Parado.NumberInQueue	.13953	(Insuf)	.00000	1.0000	.00000
Elevador 2 Andar 3 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000	.00000

```
Elevador 2 Andar 2 Parado.NumberInQueue     .13953     (Insuf)     .00000     1.0000
    1.0000
Elevador 2 Andar 1 Parado.NumberInQueue     .20698     (Insuf)     .00000     1.0000
    .00000

                                            RESULTADOS

Identificador                                  Valor
-----------------------------------------------------------------

Elevador.NumberIn                             2.0000
Elevador.NumberOut                            .00000
pessoa.NumberIn                               15.000
pessoa.NumberOut                              15.000
System.NumberOut                              15.000


Tempo executando a simulação: 1.58 minutos.
Simulação completada.
```

Anexo B – Relatório Arena de 30 pessoas

ARENA Simulation Results
maria.clara.of@hotmail.com - License: STUDENT

Sumário para Replicação 1 de 1

Projeto: Unnamed Project Data da execução :12/11/2023
Analista: maria.clara.of@hotmail.com Data de revisão do modelo:12/11/2023

Replicação terminada no tempo de : 0.20833333 Horas
Unidade de Tempo Base: Horas

VARI VEIS DE REGISTRO

Identificador	Média	Meia Largura	Mínimo	Máximo	Observações
Elevador.VATime	--	--	--	--	0
Elevador.NVATime	--	--	--	--	0
Elevador.WaitTime	--	--	--	--	0
Elevador.TranTime	--	--	--	--	0
Elevador.OtherTime	--	--	--	--	0
Elevador.TotalTime	--	--	--	--	0
pessoa.VATime	.00000	(Insuf)	.00000	.00000	30
pessoa.NVATime	.00000	(Insuf)	.00000	.00000	30
pessoa.WaitTime	.04241	(Insuf)	.00556	.08889	30
pessoa.TranTime	.00000	(Insuf)	.00000	.00000	30
pessoa.OtherTime	.01148	(Insuf)	.00556	.02222	30
pessoa.TotalTime	.05389	(Insuf)	.01111	.11111	30
Pessoa Esperando Elevador 1 Andar 1.Queue.	.02188	(Insuf)	.00000	.05556	16
Pessoa Esperando Elevador 1 Andar 2.Queue.	--	--	--	--	0
Pessoa Esperando Elevador 1 Andar 3.Queue.	--	--	--	--	0

```
Pessoa Esperando Elevador 1 Andar 4.Queue.      --         --         --         --
    0
Pessoa Esperando Elevador 1 Andar 5.Queue.      --         --         --         --
    0
Pessoa Esperando Elevador 1 Andar 6.Queue.      --         --         --         --
    0
Pessoa Esperando Elevador 2 Andar 1.Queue.  .03667     (Insuf)    .00000     .06667
    10
Pessoa Esperando Elevador 2 Andar 2.Queue.  .02778     (Insuf)    .02778     .02778
    1
Pessoa Esperando Elevador 2 Andar 3.Queue.  .03889     (Insuf)    .03889     .03889
    1
Pessoa Esperando Elevador 2 Andar 4.Queue.      --         --         --         --
    0
Pessoa Esperando Elevador 2 Andar 5.Queue.  .00000     (Insuf)    .00000     .00000
    1
Pessoa Esperando Elevador 2 Andar 6.Queue.  .02222     (Insuf)    .02222     .02222
    1
No Elevador 1.WaitingTime                   .01806     (Insuf)    .00556     .03333
    16
No Elevador 2.WaitingTime                   .01270     (Insuf)    .00556     .03333
    14
Elevador 1 Andar 6 Parado.WaitingTime           --         --         --         --
    0
Elevador 1 Andar 5 Parado.WaitingTime           --         --         --         --
    0
Elevador 1 Andar 4 Parado.WaitingTime           --         --         --         --
    0
Elevador 1 Andar 3 Parado.WaitingTime           --         --         --         --
    0
Elevador 1 Andar 2 Parado.WaitingTime           --         --         --         --
    0
Elevador 1 Andar 1 Parado.WaitingTime       .00833     (Insuf)    .00833     .00833
    1
Elevador 2 Andar 6 Parado.WaitingTime           --         --         --         --
    0
Elevador 2 Andar 5 Parado.WaitingTime           --         --         --         --
    0
Elevador 2 Andar 4 Parado.WaitingTime       .01667     (Insuf)    .01667     .01667
    1
Elevador 2 Andar 3 Parado.WaitingTime           --         --         --         --
    0
Elevador 2 Andar 2 Parado.WaitingTime           --         --         --         --
    0
Elevador 2 Andar 1 Parado.WaitingTime       .02472     (Insuf)    .02472     .02472
    1
```

VARI VEIS DISCRETAS

```
Identificador                                Média   Meia Largura  Mínimo    Máximo   Valor
   Final
-------------------------------------------------------------------------------------------
```

Elevador.WIP	1.9986	(Insuf)	.00000	2.0000
2.0000				
pessoa.WIP	7.7600	(Insuf)	.00000	17.000
.00000				
Pessoa Esperando Elevador 1 Andar 1.Queue.	1.6800	(Insuf)	.00000	6.0000
.00000				
Pessoa Esperando Elevador 1 Andar 2.Queue.	.00000	(Insuf)	.00000	.00000
.00000				
Pessoa Esperando Elevador 1 Andar 3.Queue.	.00000	(Insuf)	.00000	.00000
.00000				
Pessoa Esperando Elevador 1 Andar 4.Queue.	.00000	(Insuf)	.00000	.00000
.00000				
Pessoa Esperando Elevador 1 Andar 5.Queue.	.00000	(Insuf)	.00000	.00000
.00000				
Pessoa Esperando Elevador 1 Andar 6.Queue.	.00000	(Insuf)	.00000	.00000
.00000				
Pessoa Esperando Elevador 2 Andar 1.Queue.	1.7600	(Insuf)	.00000	6.0000
.00000				
Pessoa Esperando Elevador 2 Andar 2.Queue.	.13333	(Insuf)	.00000	1.0000
.00000				
Pessoa Esperando Elevador 2 Andar 3.Queue.	.18667	(Insuf)	.00000	1.0000
.00000				
Pessoa Esperando Elevador 2 Andar 4.Queue.	.00000	(Insuf)	.00000	.00000
.00000				
Pessoa Esperando Elevador 2 Andar 5.Queue.	.00000	(Insuf)	.00000	1.0000
.00000				
Pessoa Esperando Elevador 2 Andar 6.Queue.	.10667	(Insuf)	.00000	1.0000
.00000				
No Elevador 1.NumberInQueue	1.3866	(Insuf)	.00000	6.0000
.00000				
No Elevador 2.NumberInQueue	.85333	(Insuf)	.00000	6.0000
.00000				
Elevador 1 Andar 6 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000
.00000				
Elevador 1 Andar 5 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000
.00000				
Elevador 1 Andar 4 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000
.00000				
Elevador 1 Andar 3 Parado.NumberInQueue	.21333	(Insuf)	.00000	1.0000
1.0000				
Elevador 1 Andar 2 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000
.00000				
Elevador 1 Andar 1 Parado.NumberInQueue	.04000	(Insuf)	.00000	1.0000
.00000				
Elevador 2 Andar 6 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000
.00000				
Elevador 2 Andar 5 Parado.NumberInQueue	.00000	(Insuf)	.00000	1.0000
1.0000				
Elevador 2 Andar 4 Parado.NumberInQueue	.08000	(Insuf)	.00000	1.0000
.00000				
Elevador 2 Andar 3 Parado.NumberInQueue	.00000	(Insuf)	.00000	.00000
.00000				

```
Elevador 2 Andar 2 Parado.NumberInQueue     .00000     (Insuf)     .00000      .00000
    .00000
Elevador 2 Andar 1 Parado.NumberInQueue     .11867     (Insuf)     .00000     1.0000
    .00000

                                            RESULTADOS

Identificador                                  Valor
-------------------------------------------------------------

Elevador.NumberIn                             2.0000
Elevador.NumberOut                            .00000
pessoa.NumberIn                               30.000
pessoa.NumberOut                              30.000
System.NumberOut                              30.000


Tempo executando a simulação: 0.98 minutos.
Simulação completada.
```

Anexo C – Relatório Arena de 60 pessoas

ARENA Simulation Results
maria.clara.of@hotmail.com - License: STUDENT

Sumário para Replicação 1 de 1

Projeto: Unnamed Project Data da execução :12/11/2023
Analista: maria.clara.of@hotmail.com Data de revisão do modelo:12/11/2023

Replicação terminada no tempo de : 0.475 Horas
Unidade de Tempo Base: Horas

VARI VEIS DE REGISTRO

Identificador	Média	Meia Largura	Mínimo	Máximo	Observações
Elevador.VATime	--	--	--	--	0
Elevador.NVATime	--	--	--	--	0
Elevador.WaitTime	--	--	--	--	0
Elevador.TranTime	--	--	--	--	0
Elevador.OtherTime	--	--	--	--	0
Elevador.TotalTime	--	--	--	--	0
pessoa.VATime	.00000	(Insuf)	.00000	.00000	59
pessoa.NVATime	.00000	(Insuf)	.00000	.00000	59
pessoa.WaitTime	.10697	(Insuf)	.00556	.28889	59
pessoa.TranTime	.00000	(Insuf)	.00000	.00000	59
pessoa.OtherTime	.01271	(Insuf)	.00556	.04444	59
pessoa.TotalTime	.11968	(Insuf)	.01111	.31111	59
Pessoa Esperando Elevador 1 Andar 1.Queue.	.02188	(Insuf)	.00000	.05556	16
Pessoa Esperando Elevador 1 Andar 2.Queue.	--	--	--	--	0
Pessoa Esperando Elevador 1 Andar 3.Queue.	.00556	(Insuf)	.00556	.00556	1

```
Pessoa Esperando Elevador 1 Andar 4.Queue.      --         --         --         --
   0
Pessoa Esperando Elevador 1 Andar 5.Queue.      --         --         --         --
   0
Pessoa Esperando Elevador 1 Andar 6.Queue.      --         --         --         --
   0
Pessoa Esperando Elevador 2 Andar 1.Queue.  .13824     (Insuf)    .00000     .28333
   34
Pessoa Esperando Elevador 2 Andar 2.Queue.  .02778     (Insuf)    .02778     .02778
   2
Pessoa Esperando Elevador 2 Andar 3.Queue.  .03889     (Insuf)    .03889     .03889
   1
Pessoa Esperando Elevador 2 Andar 4.Queue.      --         --         --         --
   0
Pessoa Esperando Elevador 2 Andar 5.Queue.  .03519     (Insuf)    .00000     .05556
   3
Pessoa Esperando Elevador 2 Andar 6.Queue.  .04444     (Insuf)    .02222     .06667
   2
No Elevador 1.WaitingTime                   .01732     (Insuf)    .00556     .03333
   17
No Elevador 2.WaitingTime                   .01601     (Insuf)    .00556     .03333
   42
Elevador 1 Andar 6 Parado.WaitingTime           --         --         --         --
   0
Elevador 1 Andar 5 Parado.WaitingTime           --         --         --         --
   0
Elevador 1 Andar 4 Parado.WaitingTime           --         --         --         --
   0
Elevador 1 Andar 3 Parado.WaitingTime           --         --         --         --
   0
Elevador 1 Andar 2 Parado.WaitingTime           --         --         --         --
   0
Elevador 1 Andar 1 Parado.WaitingTime       .00833     (Insuf)    .00833     .00833
   1
Elevador 2 Andar 6 Parado.WaitingTime           --         --         --         --
   0
Elevador 2 Andar 5 Parado.WaitingTime           --         --         --         --
   0
Elevador 2 Andar 4 Parado.WaitingTime       .01667     (Insuf)    .01667     .01667
   1
Elevador 2 Andar 3 Parado.WaitingTime           --         --         --         --
   0
Elevador 2 Andar 2 Parado.WaitingTime           --         --         --         --
   0
Elevador 2 Andar 1 Parado.WaitingTime       .02472     (Insuf)    .02472     .02472
   1
```

VARI VEIS DISCRETAS

```
Identificador                                Média   Meia Largura  Mínimo    Máximo   Valor
   Final
--------------------------------------------------------------------------------------------
```

```
Elevador.WIP                                    1.9994     (Insuf)     .00000     2.0000
    2.0000
pessoa.WIP                                      15.532     (Insuf)     .00000     39.000
    1.0000
Pessoa Esperando Elevador 1 Andar 1.Queue.      .73684     (Insuf)     .00000     6.0000
    .00000
Pessoa Esperando Elevador 1 Andar 2.Queue.      .00000     (Insuf)     .00000     .00000
    .00000
Pessoa Esperando Elevador 1 Andar 3.Queue.      .67836     (Insuf)     .00000     2.0000
    1.0000
Pessoa Esperando Elevador 1 Andar 4.Queue.      .00000     (Insuf)     .00000     .00000
    .00000
Pessoa Esperando Elevador 1 Andar 5.Queue.      .00000     (Insuf)     .00000     .00000
    .00000
Pessoa Esperando Elevador 1 Andar 6.Queue.      .00000     (Insuf)     .00000     .00000
    .00000
Pessoa Esperando Elevador 2 Andar 1.Queue.      9.8947     (Insuf)     .00000     28.000
    .00000
Pessoa Esperando Elevador 2 Andar 2.Queue.      .11696     (Insuf)     .00000     1.0000
    .00000
Pessoa Esperando Elevador 2 Andar 3.Queue.      .08187     (Insuf)     .00000     1.0000
    .00000
Pessoa Esperando Elevador 2 Andar 4.Queue.      .00000     (Insuf)     .00000     .00000
    .00000
Pessoa Esperando Elevador 2 Andar 5.Queue.      .22222     (Insuf)     .00000     2.0000
    .00000
Pessoa Esperando Elevador 2 Andar 6.Queue.      .18713     (Insuf)     .00000     1.0000
    .00000
No Elevador 1.NumberInQueue                     .61988     (Insuf)     .00000     6.0000
    .00000
No Elevador 2.NumberInQueue                     1.4152     (Insuf)     .00000     6.0000
    .00000
Elevador 1 Andar 6 Parado.NumberInQueue         .00000     (Insuf)     .00000     .00000
    .00000
Elevador 1 Andar 5 Parado.NumberInQueue         .00000     (Insuf)     .00000     .00000
    .00000
Elevador 1 Andar 4 Parado.NumberInQueue         .00000     (Insuf)     .00000     .00000
    .00000
Elevador 1 Andar 3 Parado.NumberInQueue         .00000     (Insuf)     .00000     .00000
    .00000
Elevador 1 Andar 2 Parado.NumberInQueue         .61988     (Insuf)     .00000     1.0000
    1.0000
Elevador 1 Andar 1 Parado.NumberInQueue         .01754     (Insuf)     .00000     1.0000
    .00000
Elevador 2 Andar 6 Parado.NumberInQueue         .00000     (Insuf)     .00000     .00000
    .00000
Elevador 2 Andar 5 Parado.NumberInQueue         .00000     (Insuf)     .00000     1.0000
    1.0000
Elevador 2 Andar 4 Parado.NumberInQueue         .03509     (Insuf)     .00000     1.0000
    .00000
Elevador 2 Andar 3 Parado.NumberInQueue         .00000     (Insuf)     .00000     .00000
    .00000
```

```
Elevador 2 Andar 2 Parado.NumberInQueue      .00000     (Insuf)     .00000      .00000
    .00000
Elevador 2 Andar 1 Parado.NumberInQueue      .05205     (Insuf)     .00000     1.0000
    .00000

                                              RESULTADOS

Identificador                                    Valor
-----------------------------------------------------------------

Elevador.NumberIn                               2.0000
Elevador.NumberOut                              .00000
pessoa.NumberIn                                 60.000
pessoa.NumberOut                                59.000
System.NumberOut                                59.000


Tempo executando a simulação: 0.65 minutos.
Simulação completada.
```