Relatório de Simulação – Sistema Inteligente de Elevadores

BRUNO IBIAPINA SILVA MARQUES
Instituto de Ensino Superior - ICEV
Curso de Engenharia de Software
Email: bruno.marques@somosicev.com

PAULO HENRIQUE FERNANDES DA CRUZ
Instituto de Ensino Superior - ICEV
Curso de Engenharia de Software
Email: paulo$_h$enrique.cruz@somosicev.com

***Resumo*—Este relatório apresenta uma análise técnica e estatística de um sistema de simulação de elevadores inteligentes desenvolvido em Java. O objetivo principal é avaliar o desempenho de diferentes estratégias de controle em termos de tempo de espera, tempo de viagem e economia de energia. A simulação reflete o funcionamento de um edifício com múltiplos andares e fluxo variável de pessoas, utilizando uma arquitetura orientada a objetos. As conclusões apontam caminhos para melhorias em sistemas de transporte vertical em edificações inteligentes.**

***Index Terms*—Elevadores, Simulação, Java, Heurísticas, Otimização, Eficiência Energética, Sistema Inteligente, Transporte Vertical.**

I. Introdução

Com o crescimento vertical das cidades, edifícios altos tornaram-se cada vez mais comuns, exigindo sistemas eficientes de transporte interno. Elevadores desempenham papel essencial nesse contexto, impactando diretamente a qualidade de vida dos usuários. No entanto, o gerenciamento ineficaz desses sistemas pode gerar longos tempos de espera, desperdício energético e insatisfação geral dos ocupantes.

Este trabalho apresenta uma simulação computacional de um sistema inteligente de elevadores, projetado para testar diferentes estratégias de controle de despacho. O simulador foi desenvolvido em Java, com arquitetura orientada a objetos, permitindo flexibilidade na implementação e substituição de algoritmos de controle.

O objetivo é fornecer uma base sólida para estudos futuros, além de uma ferramenta prática de avaliação de desempenho, com foco em métricas como tempo médio de espera, tempo de viagem e consumo energético.

II. Descrição Técnica da Simulação

A simulação foi projetada com uma estrutura modular baseada em programação orientada a objetos. Os principais componentes do sistema são:

- **Pessoa**: classe que representa um usuário. Cada instância possui atributos como andar de origem, andar de destino, e tempo de espera acumulado.
- **Elevador**: gerencia o movimento entre andares, decide quando abrir ou fechar portas e trata chamadas internas (realizadas pelos passageiros) e externas (realizadas nos andares).
- **CentralDeControle**: componente central de decisão que monitora chamadas e distribui a carga de trabalho entre os elevadores disponíveis.
- **HeuristicaElevador**: interface para a implementação de diferentes políticas de decisão, permitindo testar múltiplas estratégias.
- **Simulador**: responsável por controlar o ciclo de tempo da simulação, gerar novas pessoas, atualizar o estado dos elevadores e registrar estatísticas.

Cada ciclo simulado representa um segundo de operação do sistema. Durante esse tempo, os elevadores avaliam sua situação, tomam decisões com base na heurística em uso e interagem com os passageiros. O fluxo de pessoas é gerado aleatoriamente, de forma que a simulação represente cenários realistas de uso.

III. Modelos de Heurística de Controle

A lógica de controle dos elevadores é definida por diferentes heurísticas, que influenciam diretamente a eficiência do sistema. Três abordagens foram implementadas:

A. Sem Heurística (FCFS)

A abordagem *First Come, First Served* trata as solicitações na ordem em que são recebidas. Os elevadores não têm conhecimento do estado geral do sistema, e nenhuma forma de otimização é aplicada. É um modelo útil como linha de base para comparação.

B. Otimização de Tempo de Espera

Essa heurística calcula os andares com maior acúmulo de chamadas e prioriza seu atendimento. O objetivo é reduzir o tempo médio que uma pessoa espera pelo elevador, mesmo que isso implique em maior deslocamento para os elevadores.

C. Otimização de Consumo de Energia

Foca na eficiência energética. O elevador mais próximo do andar chamado é selecionado, reduzindo o movimento total. Essa estratégia é ideal em edifícios que buscam menor consumo e manutenção dos elevadores, mesmo que o tempo de espera aumente.

Cada heurística é implementada na função `escolherElevador()` da classe `CentralDeControle`, permitindo alternância simples entre elas.

IV. Estrutura do Código e Papéis das Classes

A seguir, apresenta-se uma descrição detalhada dos arquivos que compõem o sistema:

- **CentralDeControle.java**: gerencia todas as chamadas externas e decide qual elevador deve responder. É o "cérebro" da aplicação.
- **Elevador.java**: lida com o estado interno de cada elevador, incluindo sua fila de destinos, direção, ocupação e tempo restante em cada andar.
- **Simulador.java**: implementa o ciclo de simulação, gerando novos usuários e controlando o avanço do tempo.
- **Pessoa.java**: classe simples que encapsula os dados do passageiro, como andares de origem/destino e tempo de entrada/saída.
- **HeuristicaElevador.java**: define o contrato para criação de novas estratégias de controle de forma independente.

V. Resultados Estatísticos

A Tabela I mostra os principais indicadores coletados durante a simulação, considerando 100 pessoas geradas por modelo.

Tabela I
Resultados estatísticos por modelo de heurística

Modelo	Tempo Total (s)	Pessoas	Espera (s)	Viagem
FCFS (Sem Heurística)	605	100	95,83	1,69
Otimização de Tempo	468	100	97,76	2,08
Otimização de Energia	328	100	107,55	2,28

Observa-se que o modelo de otimização de tempo reduziu o tempo total da simulação, enquanto a estratégia de energia foi a mais eficiente em termos de deslocamentos, embora com maior tempo médio de espera.

VI. Discussão

A partir dos dados obtidos, observa-se um claro *trade-off* entre eficiência operacional e conforto do usuário. O modelo FCFS, embora simples, não se adapta bem ao aumento da demanda. O modelo de otimização de tempo demonstra bom desempenho geral, ideal para momentos de pico.

Já a heurística de energia mostra-se adequada para horários de menor fluxo, minimizando o desgaste dos equipamentos e reduzindo o consumo. No entanto, pode gerar insatisfação entre os usuários pela demora.

Essas conclusões indicam que a escolha da heurística deve ser adaptativa, considerando horários e padrões de uso.

VII. Considerações Adicionais

Durante a implementação, foram observadas boas práticas de desenvolvimento, como separação de responsabilidades, uso de interfaces e encapsulamento. Isso permite a fácil extensão do projeto. Como futuras melhorias, propõe-se:

- Implementação de heurísticas baseadas em aprendizado de máquina para previsão de demanda.
- Integração com sensores simulados (por exemplo, número de pessoas no andar).
- Coleta de métricas adicionais como número de paradas, consumo energético estimado e taxa de ocupação.
- Interface gráfica para visualização em tempo real da simulação.

VIII. Conclusão

O simulador de elevadores desenvolvido se mostrou eficiente para testar e comparar estratégias de controle. A arquitetura modular facilita alterações e expansões. Os resultados estatísticos destacam a importância da escolha da heurística de controle conforme os objetivos do sistema, seja desempenho, conforto ou economia energética. O trabalho também demonstra como simulações computacionais podem apoiar decisões em projetos reais de engenharia predial.

Anexos – Gráficos

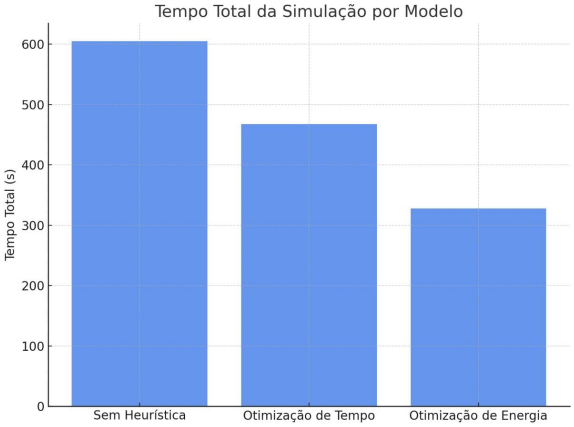

Figura 1. Tempo Total da Simulação por Modelo

Gráfico 2 – Tempo Médio de Espera por Modelo

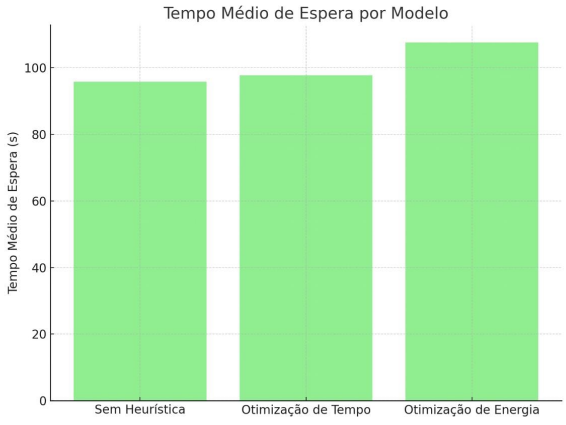

Figura 2. Tempo Médio de Espera por Modelo

Gráfico 3 – Tempo Médio de Viagem por Modelo

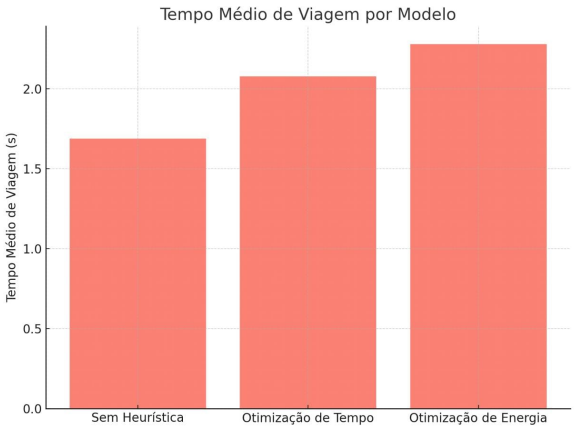

Figura 3. Tempo Médio de Viagem por Modelo

REFERÊNCIAS

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.

[2] R. Lafore, *Data Structures and Algorithms in Java*, 2nd ed. Indianapolis, IN: Sams Publishing, 2002.

[3] M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, *Data Structures and Algorithms in Java*, 6th ed. Hoboken, NJ: Wiley, 2014.

[4] E. Horowitz, S. Sahni, and D. Mehta, *Fundamentals of Data Structures in C++*, 2nd ed. Hyderabad, India: Universities Press, 2008.

[5] D. E. Knuth, *The Art of Computer Programming*, Vol. 1: Fundamental Algorithms, 3rd ed. Boston, MA: Addison-Wesley, 1997.