Universidade Federal de Minas Gerais (UFMG)

Departamento de Ciência da Computação

Isabelle Fernandes de Oliveira Sannier (2021432208)

Trabalho Prático 2: Sistema de Escalonamento Logístico

Belo Horizonte, 22 de junho de 2025

1. Introdução

Este trabalho tem como objetivo desenvolver um sistema para gerenciar e analisar a operação logística de uma rede de armazéns fictícia, denominada Armazéns Hanoi. O desafio principal consiste em simular o transporte e armazenamento de pacotes dentro dessa rede, que é representada por um grafo onde os nós correspondem aos armazéns e as arestas indicam conexões entre eles.

Um aspecto particular do problema é a manipulação dos pacotes dentro de cada armazém, organizada em seções que seguem a política LIFO (Last-In, First-Out). Além disso, o transporte entre armazéns deve respeitar uma regra de prioridade que determina que os pacotes mais antigos sejam transportados primeiro, o que requer um cuidado especial na manipulação das pilhas LIFO para garantir essa prioridade.

Além disso, faz-se necessário rastrear o ciclo de vida de cada pacote, desde sua chegada até a entrega final. Para isso, a solução adotada baseia-se na Simulação de Eventos Discretos (SED), que consiste em modelar o sistema por meio de eventos pontuais — como armazenamento, rearmazenamento, remoção, transporte e entrega de pacotes — processados em ordem cronológica. Esse método permite avançar o "relógio" da simulação de forma direta entre eventos, evitando a necessidade de simular cada unidade de tempo.

2. Método

A implementação envolve diversas classes para simular o funcionamento do sistema logístico dos Armazéns Hanoi. A execução é conduzida por um laço principal que está na main que gerencia a simulação por meio do processamento de eventos.

- **Classe Grafo:** Representa a topologia da rede de armazéns. Foi implementado como um grafo não direcionado, onde cada vértice corresponde a um armazém e cada aresta indica uma conexão bidirecional entre eles. Internamente, a estrutura utiliza uma lista de adjacência para armazenar as vizinhanças.
- **Classe Pacote:** Cada pacote possui atributos essenciais, tais como um identificador único (ID), o armazém de origem, o destino final, o armazém atual onde está armazenado, seu estado atual (como armazenado ou entregue), um ponteiro para a rota previamente calculada e um método que retorna qual o próximo destino do pacote, dado o armazém atual que ele está.
- **Classe Pilha:** Para simular as seções LIFO dentro de cada armazém, foi desenvolvido o classe Pilha. Essa pilha foi implementada usando uma lista encadeada, garantindo flexibilidade para crescimento dinâmico conforme a chegada de pacotes. Possui os métodos comuns de um TAD pilha de empilhar, desempilhar e verificar se a pilha está vazia.
- **Classe Armazém:** É responsável pelo gerenciamento dos pacotes armazenados em cada localidade. Cada armazém contém um array de Pilhas, onde cada pilha representa uma seção distinta do armazém. O número de seções é definido pelo grau do vértice no grafo, ou seja, pelo número de conexões do armazém com seus vizinhos. Para facilitar o acesso correto às pilhas, foi utilizado um mapa auxiliar (mapaDeDestinos) que associa o ID do armazém destino ao índice correspondente do array de pilhas.
- **Classe Evento:** A classe Evento encapsula ocorrências futuras na simulação, sendo os principais tipos de eventos definidos os de chegada de pacote (PACOTE) e transporte

(TRANSPORTE). Cada evento possui uma chave de prioridade composta que assegura a execução correta em ordem cronológica, resolvendo possíveis ambiguidades.

- **Classe Escalonador:** Foi implementado como uma fila de prioridade baseada em uma estrutura Min-Heap. Essa escolha permite que a operação de extrair o próximo evento a ser processado ocorra em tempo logarítmico, garantindo a eficiência da simulação mesmo com um grande número de eventos agendados. Ela possui métodos comuns a um TAD min-heap. Uma peculiaridade desse min-heap é o método implementado redimensionar(). Será melhor explicado na seção estratégia de robustez.

- **Algoritmos Principais:**

i) Para o cálculo das rotas dos pacotes entre armazéns, foi adotado o algoritmo de Busca em Largura (BFS). Para isso, foram utilizadas as classes Fila e CalculaRota. A fila foi utilizada para fazer o caminhamento em nível no grafo, operado pela classe CalculaRota.

ii) Durante a simulação, o laço principal encontra-se na main e funciona retirando o evento de menor prioridade do escalonador, atualizando o relógio da simulação para o tempo do evento, processando-o por meio dos métodos definidos nas demais classes e, quando necessário, agendando novos eventos futuros para dar continuidade ao processo.

3. Análise de Complexidade

As operações utilizadas na main envolvem o cálculo de rota dos pacotes, as operações do escalonador de inserir e retirar eventos, operação de empilhar e desempilhar do armazém. Abaixo segue análise de complexidade para cada umas dessas operações.

Cálculo de Rota (BFS):

- Tempo: $O(V + A)$ O algoritmo percorre todos os vértices V e arestas A do grafo.
- Espaço: O(V) para as estruturas auxiliares (fila, vetores de visitado/predecessor).

Operações do Escalonador (Min-Heap):

- insereEvento: $O(log(N))$ Inserção de um novo evento no heap, onde N é o número de eventos atualmente na fila de prioridade.
- retiraProximoEvento: $O(1)$ para leitura do próximo evento e $O(log(N))$ para consertar o heap. Nessa conta, domina $O(log(N))$, onde N é o número de eventos na fila de prioridade.

Operações da Pilha:

- empilha e desempilha: $O(1)$.
- Remover 1 pacote de uma seção com k itens tem custo $O(k)$, pois precisa mover todos os pacotes, inclusive o último.

Complexidade Total da Simulação:

O programa processa um total de K eventos. A operação dominante dentro do laço de simulação são as operações do escalonador. Portanto, a **complexidade de tempo** do programa é $O(K \cdot log(N))$, onde K é o número total de eventos gerados e N é o tamanho máximo que o heap atinge. A **complexidade de espaço** é a soma do espaço para o grafo, os pacotes, os armazéns e o heap: $O(V + A + P + N)$, onde P é o número de pacotes.

4. Estratégias de Robustez

Gerenciamento de Memória Dinâmica

Todas as estruturas que fazem uso de alocação dinâmica de memória (como Grafo, Armazem, Pilha, Pacote e Escalonador) foram projetadas com destrutores, assegurando a liberação dos recursos alocados com new[ ].

Validação de Entrada

Antes de iniciar a simulação, é verificado se o nome do arquivo de entrada foi corretamente fornecido via linha de comando, utilizando a condicional if (argc < 2). Esta verificação impede que o programa tente acessar argumentos inexistentes.

Expansão de Arrays Dinâmicos

O Escalonador, implementado como um heap binário, possui um mecanismo interno de redimensionamento dinâmico através do método redimensionar(). Como não se sabe quantos eventos serão gerados inicialmente, o heap em algum momento poderia estourar se a sua capacidade fosse menor que a quantidade de eventos gerados. Então, esse método verifica, a cada insersão de evento no heap, se sua capaciadade comporta. Caso não comporte, ele copia os eventos do heap em um novo heap de capacidade dobrada e deleta o heap antigo. Da mesma forma, se a quantidade de eventos for a metade da capacidade do heap, quando ele retira um evento, ele copia os eventos do heap em um novo heap de capacidade pela metade e detela o heap antigo.

Uso da ferramente Valgrind

Durante a fase de testes, a ferramenta Valgrind foi utilizada para inspecionar o uso da memória e detectar erros como vazamentos (memory leaks), acessos inválidos (invalid reads/writes), dupla liberação (double free) e uso de memória liberada (use-after-free). A simulação foi ajustada iterativamente até que Valgrind reportasse execução limpa.

5. Análise Experimental

Para fazer a análise experimental, foi necessário modificar o programa original: foi criada a classe Estatística, que ficou responsável de registrar a duração para cada pacote o tempo de duração para os estados rearmazenado, armazenado, em trânsito e o tempo da jornada. Com essas novas medidas, foi possível gerar experimentos que buscou variar apenas uma medida e fixando as demais para ver o impacto dessa medida no desempenho do sistema logístico. As medidas escolhidas para variar foram:

- Pacote: feito 200 sorteios sem reposição de 10 a 5000 de quantidade de pacotes.
- Armazém: feito 200 sorteios sem reposição de 5 a 300 quantidade de armazéns.

- Capacidade: feito 200 sorteios sem reposição de 1 a 200 para a capacidade de transporte.
- Intervalo: feito 200 sorteios sem reposição de 1 a 2000 para a intervalo de transporte.

Como dito, apenas uma medida foi variada por vez e as demais foram fixadas. Os valores fixados para as respectivas variáveis acima foram:

- Pacote: 500 pacotes.
- Armazém: 20 armazéns.
- Capacidade: 5 capacidade de transporte.
- Intervalo: 100 intervalo de transporte.

Ao todo, foram gerados 800 cenários diferentes, divididos como 200 experimentos para cada variável a ser variada. As variáveis escolhidas como medida de desempenho do sistema logístico foram:

- Tempo de rearmazenamento: consiste no tempo total de rearmazenamento para cada pacote.
- Tempo de armazenamento: consiste no tempo total armazenamento para cada pacote.
- Tempo de transporte: consiste no tempo total d transporte para cada pacote.
- Tempo total de jornada: consiste no tempo total de jornada (TempoEntregue - TempoPostado) para cada pacote.

Abaixo seguem os gráficos para cada medida de desempenho.

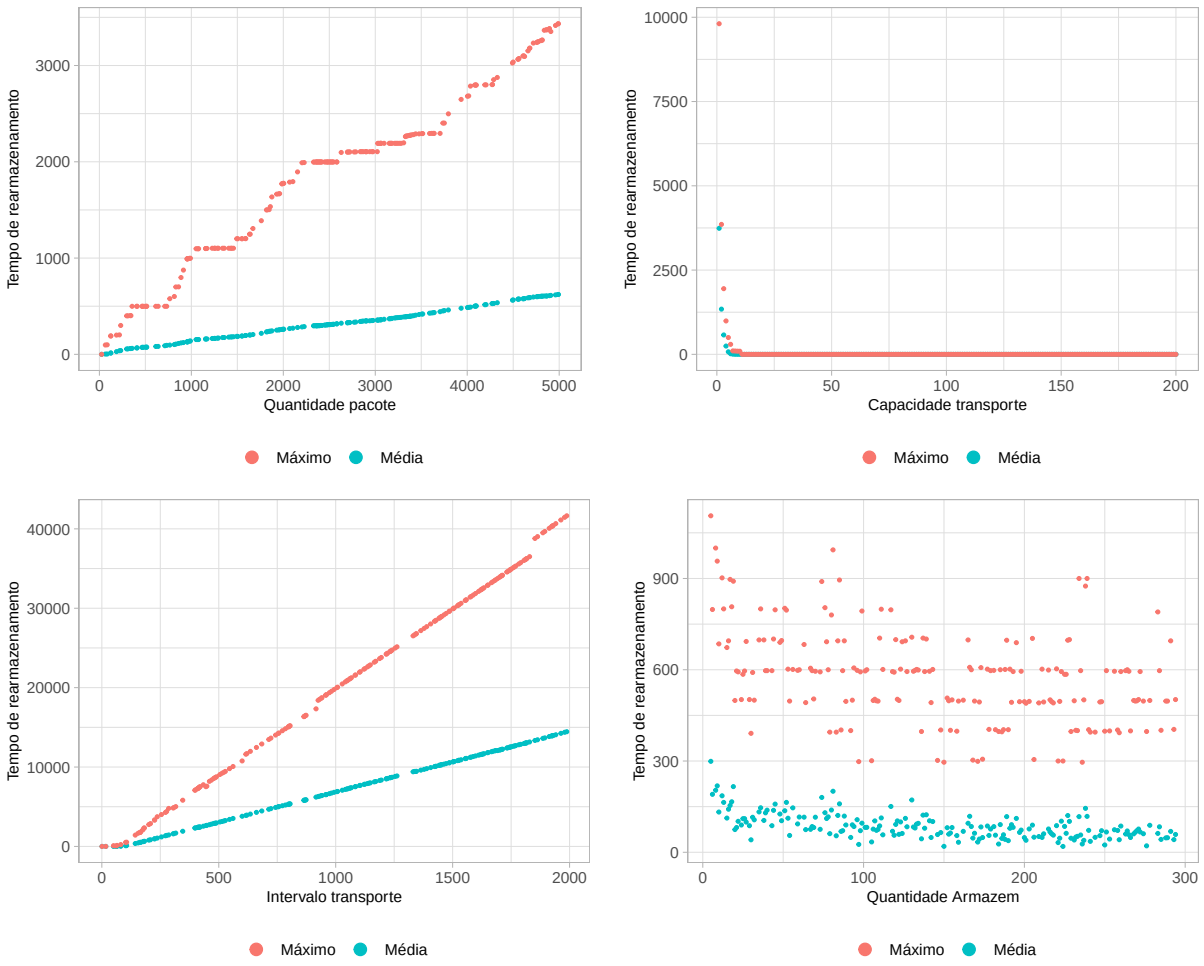

Figura 1: Tempo médio e tempo máximo de rearmazenamento por variável eixo x

Na Figura 1, percebe-se que o tempo (médio e máximo) de rearmazenamento é maior quando aumenta-se a quantidade de pacotes e intervalo de transporte. Por sua vez, o tempo (médio e máximo) de rearmazenamento é menor quando aumenta-se a capacidade de transporte. Para a variável quantidade de armazém, o tempo de rearmazenamento não sofre alteração.

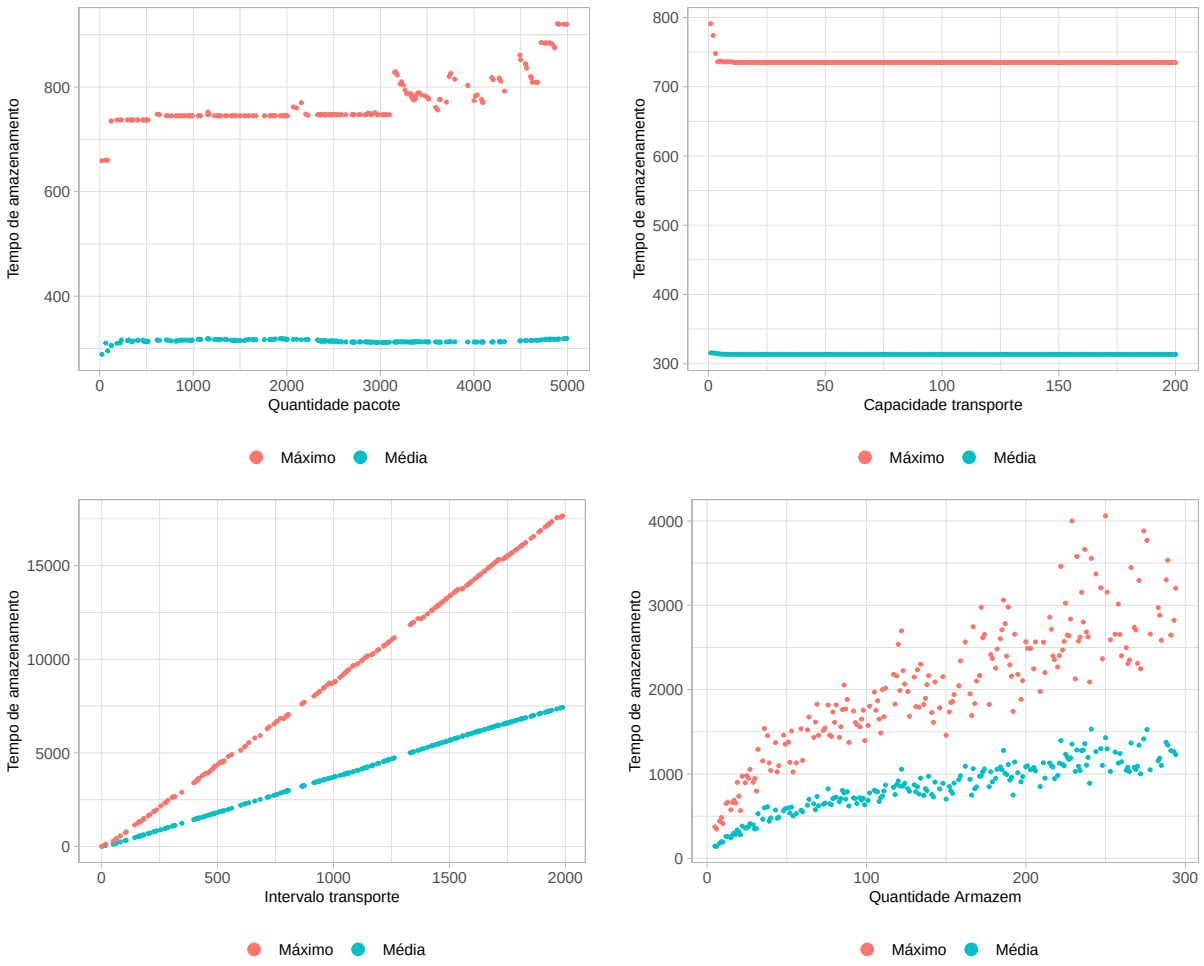

Figura 2: Tempo médio e tempo máximo de armazenamento por variável eixo x

Na Figura 2, percebe-se que o tempo (médio e máximo) de armazenamento é maior quando aumenta-se o intervalo de transporte e a quantidade de armazéns. Para as variáveis quantidade de pacote e capacidade de transporte o tempo de armazenamento não sofre muita alteração.

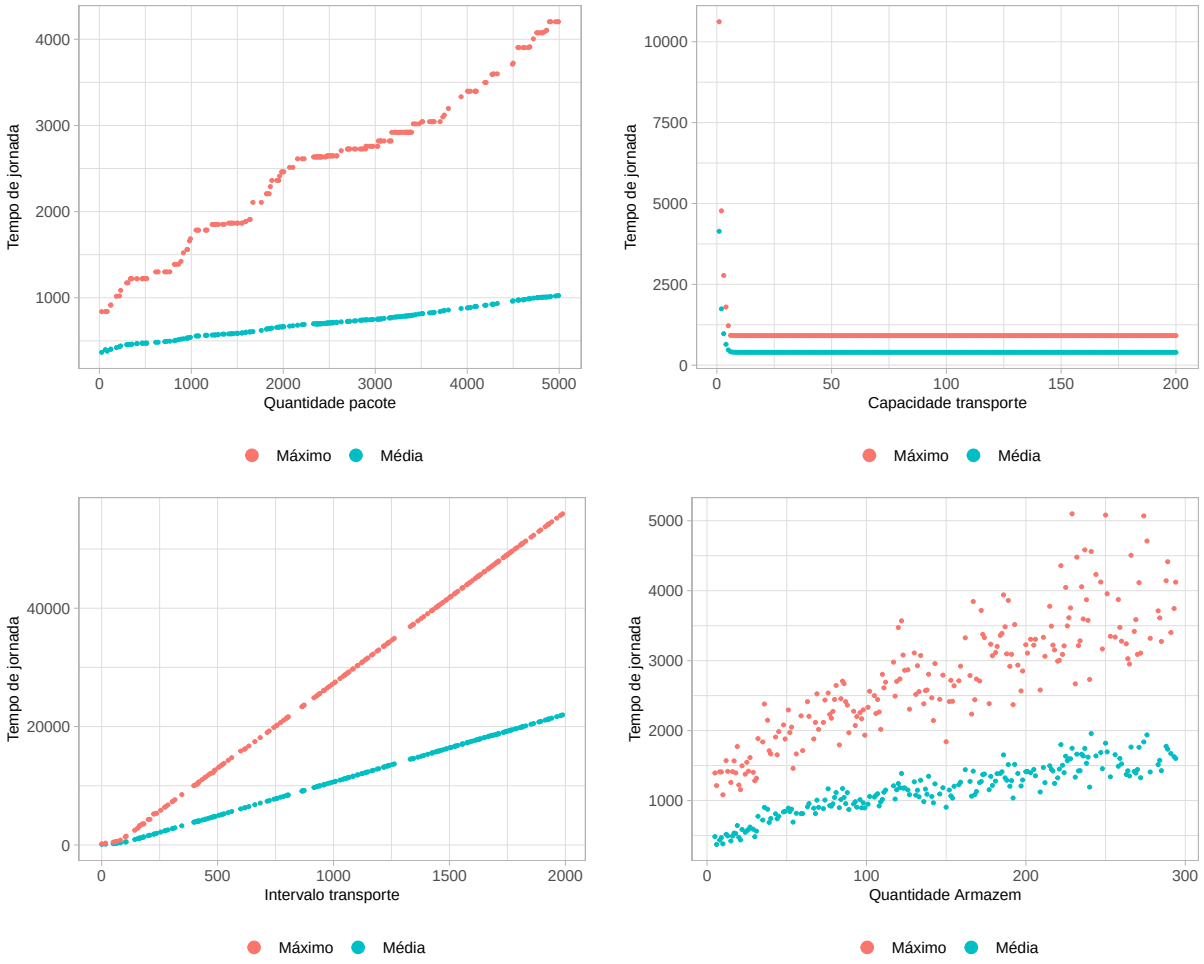

Figura 3: Tempo médio e tempo máximo de jornada por variável eixo x

Na Figura 3, percebe-se que o tempo (médio e máximo) de jornada é maior quando aumenta-se o intervalo de transporte, a quantidade de pacote e a quantidade de armazéns. Por sua vez, o tempo (médio e máximo) de jornada é menor quando aumenta-se a capacidade de transporte. Percebe-se que para essa medida de desempenho, as quatro variáveis impactam no sistema logístico.

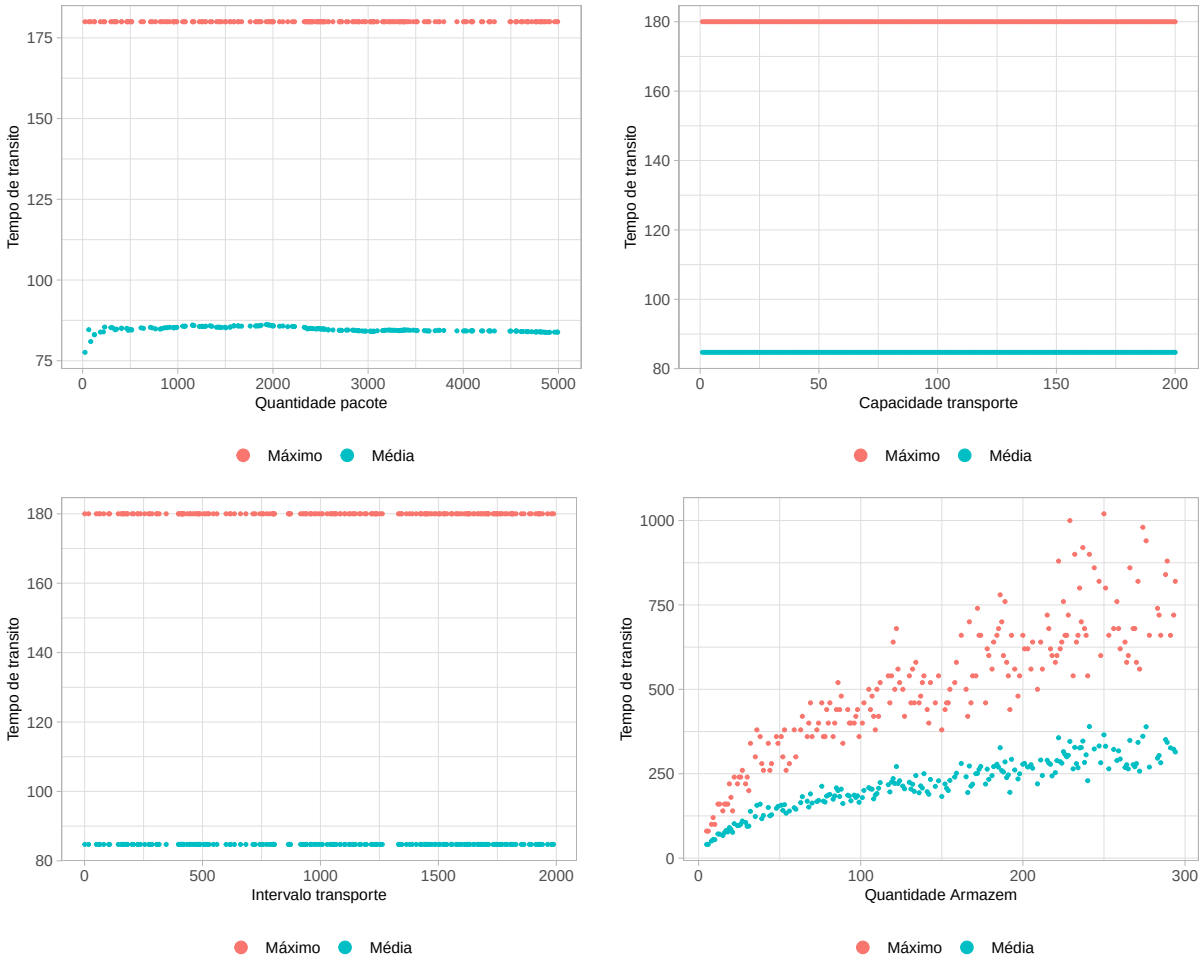

Figura 4: Tempo médio e tempo máximo de trânsito por variável eixo x

Por fim, na Figura 4, percebe-se que a única variável que impacta o tempo de transporte é a quantidade de armazéns. Quanto mais armazéns, maior é o tempo de transporte. Importante dizer que a latência de transporte também impactaria, mas ela não foi escolhida entre as 4 variáveis a serem modificadas.

6. Conclusão

O presente trabalho pretendeu desenvolver e analisar o sistema de simulação de eventos discretos projetado para modelar a operação logística dos Armazéns Hanoi, incorporando suas regras de armazenamento LIFO e prioridade de transporte. A solução implementada em C++ buscar ter o histórico detalhado do ciclo de vida de cada pacote e as estatísticas de desempenho do sistema sob diferentes cenários de carga e configuração.

A implementação do projeto proporcionou um aprendizado prático na aplicação de estruturas de dados para a resolução de um problema complexo do mundo real. A simulação foi garantida pelo uso de uma Fila de Prioridade baseada em um Min-Heap, que se mostrou ideal para gerenciar a linha do tempo de eventos de forma ordenada e com baixo custo computacional. A representação da rede de armazéns por meio de um Grafo com listas de adjacência e o uso do algoritmo de Busca em Largura (BFS) possibilitaram o cálculo otimizado das rotas. Adicionalmente, a implementação de uma Pilha com lista encadeada foi crucial para modelar a regra LIFO das seções dos armazéns sem se preocupar com o tamanho a priori de cada pilha.

Foi decidido centralizar toda a lógica da simulação de eventos discretos dentro da função main, onde nela retirava-se o próximo evento imediato no heap e atualizava o relógio do tempo atual para o tempo do evento.

Por fim, a Análise Experimental validou a utilidade do simulador como ferramenta de análise. Ao variar sistematicamente os parâmetros de entrada — a quantidade de pacotes, o número de armazéns, a capacidade de transporte e o intervalo de transporte —, foi possível observar quantitativamente seus impactos no desempenho do sistema, especialmente em métricas de contenção como o tempo de armazenamento e rearmazenamento.

7. Referência

LACERDA, Anisio; MEIRA JR., Wagner; CUNHA, Washington. Estruturas de Dados. Slides da disciplina DCC205 – Estruturas de Dados, Departamento de Ciência da Computação, ICEx/UFMG, 2025.

UNIVERSIDADE FEDERAL DE MINAS GERAIS. Departamento de Ciência da Computação. Especificação do Trabalho Prático 2 – Sistema de Escalonamento Logístico. Belo Horizonte: DCC/ICEx/UFMG, 2025. Documento em PDF.