Trabalho Prático 2

Sistema de Escalonamento Logístico

Thiago Henrique Silva de Almeida

2024014180

Universidade Federal de Minas Gerais (UFMG)

Belo Horizonte - MG - Brasil

thiagohenriquesilva@ufmg.br

1. Introdução

A gestão logística, particularmente em sistemas de transporte e armazenamento, apresenta desafios complexos que impactam diretamente a eficiência e o custo operacional de uma empresa. Otimizar o fluxo de mercadorias, desde a postagem até a entrega final, requer um acompanhamento preciso e um roteamento inteligente, especialmente em redes com múltiplos centros de distribuição e restrições de movimentação. Nesse contexto, surge a necessidade de automatizar e simular o sistema logístico dos Armazéns Hanoi, uma empresa fictícia que opera com pacotes transportados e armazenados entre seus diversos armazéns.

Este trabalho consiste na implementação de um simulador de eventos discretos capaz de modelar e gerenciar a operação de uma rede logística, acompanhando o ciclo de vida de pacotes desde sua chegada inicial até a entrega final. O sistema deve ser capaz de rotear pacotes, organizar seu armazenamento em armazéns com lógica LIFO (Last-In First-Out) nas seções, e coordenar ciclos de transporte periódicos entre armazéns conectados.

Para isso, a solução proposta baseia-se na construção de Tipos Abstratos de Dados (TADs) essenciais, como Pacote, Armazem, Transporte e Escalonador. O Escalonador, em particular, é um componente central, implementado como uma fila de prioridade (min-heap) para garantir a ordem cronológica dos eventos na simulação, permitindo que o relógio "salte" entre os instantes de ocorrência, otimizando o processo.

A simulação envolverá a leitura de parâmetros como capacidade de transporte, latência entre armazéns, intervalo entre transportes e custo de remoção de pacotes, além da topologia da rede de armazéns e uma lista detalhada dos pacotes a serem processados. A saída do sistema deverá apresentar a sequência de eventos para cada pacote, registrando sua trajetória e os estados em que se encontra, como armazenamento, remoção, trânsito e entrega.

Este trabalho, portanto, não apenas implementa os componentes necessários para a simulação de eventos discretos, mas também busca garantir a robustez e a fidelidade da simulação aos requisitos do problema, contribuindo para uma análise precisa do comportamento logístico da rede dos Armazéns Hanoi.

2. Método

O sistema de simulação logística foi desenvolvido na linguagem C++, utilizando o compilador G++ do GNU Compiler Collection. A implementação foi organizada de forma modular, empregando o

paradigma de Tipos Abstratos de Dados (TADs) para representar os componentes essenciais do sistema de escalonamento logístico.

2.1 Estruturas

No contexto deste projeto, as estruturas de dados servem como base para a organização e manipulação das informações fundamentais.

2.1.1 Estrutura Evento

A estrutura Evento define um evento na simulação de eventos discretos. Ela contém o tempoOcorrencia (o instante em que o evento ocorre), o tipo do evento, um ponteiro para o Pacote envolvido (se houver), o armazemAlvo (o armazém afetado), o idSecao relevante (se aplicável), e uma chavePrioridade (string) usada para ordenar os eventos na fila de prioridade do Escalonador. O operador > é sobrecarregado para permitir a comparação lexicográfica das chaves de prioridade, garantindo a ordem correta dos eventos na fila.

2.2 Classes e TADs

As classes e TADs implementadas são o coração do sistema de simulação, cada uma com responsabilidades bem definidas.

2.2.1 Classe Pacote

A classe Pacote é responsável por encapsular todas as informações relativas a uma unidade de envio no sistema logístico. Além dos dados de entrada como identificador, armazém de origem e destino, e timestamp de postagem, o Pacote armazena a sua rota em termos de armazéns (representada como uma lista encadeada) e o seu estado atual no processo de transporte. É crucial que o Pacote também mantenha estatísticas de desempenho, como o tempo armazenado e o tempo em trânsito, que serão utilizadas para cálculos de métricas gerais do sistema. Seus estados possíveis incluem não postado, chegada escalonada, armazenado, removido e entregue.

2.2.2 Classe Armazem

O TAD Armazem gerencia as operações de armazenamento e organização de pacotes em suas instalações. Cada Armazem é composto por seções, onde cada seção é destinada a pacotes cujo próximo destino é um armazém vizinho específico. A peculiaridade dos Armazéns Hanoi é que cada seção opera com uma lógica LIFO (Last-In First-Out), similar a uma pilha, o que significa que o último pacote a ser colocado na seção é o primeiro a ser retirado. A classe Armazem implementa as operações de armazenaPacote e recuperaPacote , contabilizando o tempo necessário para essas operações.

2.2.3 Classe Transporte

O TAD Transporte orquestra o movimento de pacotes entre os armazéns. Ele gerencia os parâmetros relacionados ao transporte, como latência (tempo constante de trânsito) e capacidade (limite de pacotes por vez). Dada a natureza da simulação de eventos discretos, o transporte é implementado através do agendamento de eventos de chegada de pacotes no armazém de destino. Funções como CoordenarCicloDeTransporte, ProcessarRemocao, ProcessarTransporte, ProcessarRearmazenamento e ProcessarEntrega são responsáveis por gerenciar o fluxo de pacotes e atualizar seus estados e estatísticas de tempo. O módulo Transporte também é responsável por

calcular a rota dos pacotes usando uma busca em largura (BFS) em grafos não direcionados e não ponderados.

2.2.4 Classe Escalonador

A classe Escalonador é um elemento central na arquitetura da simulação de eventos discretos. Ela é implementada como uma fila de prioridade que garante que os eventos sejam processados em estrita ordem cronológica. Para isso, utiliza uma estrutura de dados de min-heap. As operações cruciais do Escalonador incluem InicializaSimulacao, AgendarEvento, ExecutaProximoEvento e RodaSimulacao. Ele mantém um controle do tempo atual da simulação (relogioSimulacao) e coordena a execução de eventos como a chegada inicial de pacotes e o início de ciclos de transporte.

2.3 Módulos Auxiliares

Além das classes principais, o sistema utiliza módulos auxiliares para gerenciar estruturas de dados específicas e utilitários.

2.3.1 Módulo ListaEncadeadaRota

Representa a rota de um pacote como uma lista encadeada de IDs de armazéns. Contém funcionalidades para adicionar armazéns à rota, avançar na rota (Avanca), verificar o próximo armazém (GetProximoArmazem), e determinar se o pacote atingiu o final da rota (EstaNoFinal).

2.3.2 Módulo MinHeapEventos

Este módulo implementa a estrutura de dados de min-heap, que é o coração da fila de prioridade do Escalonador. Ele garante que o evento com o menor tempo de ocorrência seja sempre o próximo a ser processado. A criação das chaves de prioridade para os eventos (chavePrioridade) é baseada no instante de ocorrência no tempo e no tipo do evento.

3. Análise de Complexidade

A análise de complexidade visa avaliar o desempenho assintótico (tempo e espaço) dos principais métodos do sistema de simulação logística, considerando o impacto de fatores como o número de armazéns (V, de Vértices) e o número total de pacotes (P).

3.1 InicializaSimulacao

O método InicializaSimulacao é responsável por configurar a simulação em seu início, agendando o primeiro evento para cada pacote e os primeiros eventos de transporte. Sua complexidade de tempo dominante é O(P * log E + (V + A) * log E), onde P é o número de pacotes, V o número de armazéns, A o número de arestas (seções), e E o número de eventos na fila de prioridade. Isso ocorre porque o método itera sobre todos os P pacotes de entrada, e para cada um, chama AgendarEvento, que tem custo O(log E) devido à inserção no min-heap. Em seguida, ele itera sobre os V armazéns e suas seções adjacentes (2 * A arestas), agendando eventos de transporte, cada um com custo O(log E). No contexto inicial, E é dominado por P e A, resultando em uma complexidade aproximada de O((P + V + A) * log(P + A)). Quanto ao espaço, a complexidade dominante é O(P * V_max_rota + (V + A)), onde V_max_rota é o comprimento máximo de uma rota (no pior caso, O(V)). Isso se deve à alocação de objetos Evento e cópias de Pacote para cada pacote inicial e para cada evento de transporte inicial, além das ListaEncadeadaRotas dentro de cada Pacote.

3.2 ExecutaProximoEvento

O método ExecutaProximoEvento é o coração do loop principal da simulação, extraindo e processando o próximo evento da fila de prioridade. A operação mais custosa aqui é filaEventos->ExtraiMin(), que possui complexidade de tempo de O(log E), onde E é o número atual de eventos no heap. O processamento do evento em si delega a chamada para métodos da classe Transporte. A complexidade total do método é, portanto, O(log E + C_evento), onde C_evento representa a complexidade da função de processamento específica do evento que foi extraído. Em termos de espaço, o método utiliza O(1) de espaço auxiliar para manipular o evento extraído, e o impacto no espaço do heap é de O(1) por evento processado, já que o evento é removido.

Para evento de chegada inicial, C_evento é O(V * A) em tempo e O(V) em espaço no pior caso, dominado pelo cálculo da rota via CalculaRotaBFS (que tem buscas lineares) e pelas operações de armazenamento no armazém. Para o evento inicia transporte, C_evento é O(K + C * log E + R) em tempo e O(K * V_max_rota) em espaço, sendo dominado pelas operações de remoção (K pacotes), agendamento de transportes (C pacotes) e rearmazenamento (R pacotes). Para o evento de armazenamento, C_evento é O(V) em tempo e O(1) em espaço, principalmente devido à busca pela seção de destino no armazém. Finalmente, para o evento de entrgea, C_evento é O(1) em tempo e O(V_max_rota) em espaço para a cópia do pacote entregue.

3.3 CalculaRotaBFS

A função CalculaRotaBFS é responsável por determinar o menor caminho entre um armazém de origem e um de destino, utilizando o algoritmo de Busca em Largura (BFS) em um grafo não direcionado e não ponderado. A BFS visita cada vértice e cada aresta do grafo uma vez. A complexidade de tempo para a BFS tradicional é O(V + A), onde V é o número de vértices (armazéns) e A é o número de arestas (conexões entre seções). No entanto, a implementação desta função utiliza uma ListaInfoBFS para armazenar as informações de visitação, distância e predecessor dos nós. Se a BuscaInfo dentro da ListaInfoBFS é implementada como uma busca linear em uma lista encadeada, cada chamada a BuscaInfo pode custar O(V) no pior caso. Como BuscaInfo é chamada para cada vértice visitado e para cada um de seus vizinhos durante a BFS (ou seja, V vezes no loop principal e A vezes dentro do loop de adjacência), a complexidade de tempo pode degradar para O(V^2 + A * V) ou simplificadamente O(V * A) no pior cenário devido a essas buscas lineares repetidas. A construção da ListaEncadeadaRota ao final, que reconstrói o caminho de volta, tem uma complexidade de O(V_rota * N_insercoes) onde V_rota é o comprimento da rota e N_insercoes são as inserções. No pior caso, O(V^2) para a rota. Em termos de espaço, a função utiliza O(V) para a ListaInfoBFS (ou seu equivalente de armazenamento de informações por vértice) e O(V) para a fila (FilaIDs), além do espaço para a ListaEncadeadaRota resultante (O(V) no pior caso). A complexidade de espaço total é, portanto, O(V).

4. Estratégias de Robustez

As funções implementadas no Sistema de Escalonamento Logístico demonstram diversas estratégias de robustez, essenciais para garantir a confiabilidade e a estabilidade da simulação.

Uma das principais estratégias de robustez é a manipulação cuidadosa do gerenciamento de memória, fundamental em um sistema que lida com alocações e desalocações dinâmicas de objetos como Pacote, Evento, ListaEncadeadaRota e RotaNo. O uso adequado de operadores new e delete em

pares, juntamente com a implementação correta de construtores de cópia e operadores de atribuição para classes que gerenciam recursos (como Pacote e ListaEncadeadaRota), é crucial para evitar vazamentos de memória, acessos inválidos (Invalid read/write) e liberações duplas (double free). A propriedade dos objetos (Pacote original vs. cópias para eventos) é gerenciada de forma que apenas um "dono" seja responsável por liberar a memória, prevenindo corrupções que poderiam levar a falhas de segmentação.

Outra estratégia importante é a validação e tratamento de condições de contorno e parâmetros inválidos. Em várias funções, verificações explícitas são realizadas para garantir que ponteiros não sejam nulos, que pilhas não estejam vazias antes de tentar desempilhar e que IDs de seções ou armazéns sejam válidos. Isso evita comportamentos indefinidos e crashes inesperados.

A lógica de agendamento de eventos no Escalonador e Transporte incorpora robustez ao tentar garantir o avanço correto do tempo da simulação. O agendamento do próximo ciclo de transporte é calculado com base no tempo de conclusão real das operações do ciclo atual, e mecanismos de segurança (como o avanço mínimo do tempo) são considerados para evitar estagnação do relógio da simulação e garantir a progressão. Isso previne que o sistema fique preso em estados inválidos e que a simulação possa completar sua execução.

Por fim, a conversão e formatação de dados também emprega estratégias de robustez. O uso de tipos de dados de maior capacidade, como long long, para o tempo na construção da chavePrioridade dos eventos, juntamente com std::setw e std::setfill('0'), garante que a ordenação no min-heap seja precisa e que não ocorram overflows de inteiro ou truncamentos de dados que poderiam corromper a ordem dos eventos e, consequentemente, o estado da simulação.

Em conjunto, essas estratégias de robustez garantem que o sistema de simulação de eventos logísticos opere de forma estável e confiável, mesmo sob condições de entrada desafiadoras ou cenários de simulação complexos.

5. Análise Experimental

Para avaliar o desempenho e a escalabilidade do Sistema de Escalonamento Logístico, foi conduzida uma análise experimental focada em três dimensões principais, o número de armazéns, a quantidade de pacotes (carga de trabalho) e a contenção por transporte (capacidade). A metodologia consistiu em isolar cada dimensão, variando seus parâmetros enquanto os demais eram mantidos em uma configuração base fixa.

O tempo de execução de cada simulação foi medido utilizando o utilitário time do sistema operacional Linux, sendo considerada a métrica user time, que reflete o tempo de CPU consumido pelo processo. Para garantir a reprodutibilidade e a criação de cenários complexos, foi utilizado um gerador de cargas de trabalho que cria topologias de rede aleatórias, porém conectadas, e distribui as origens e destinos dos pacotes de forma aleatória. Esta análise foi executada sobre a implementação simplificada do simulador, que não inclui todas as lógicas de gerenciamento de contenção e reroteamento presentes na seção de pontos extras.

5.1 Análise do Impacto do Número de Armazéns

Neste experimento, o número de armazéns foi variado de 10 a 500. Para manter a relevância da carga de trabalho em redes maiores, o número de pacotes foi ajustado de forma proporcional,

correspondendo a 10 vezes o número de armazéns. Os demais parâmetros, como capacidade de transporte, foram mantidos fixos.

Os resultados indicam uma tendência de crescimento do tempo de execução de forma não linear e acentuada com o aumento do número de armazéns. A curva de tendência ajustada aos dados sugere uma complexidade que se aproxima de uma função polinomial de grau superior (cúbica).

Este comportamento é consistente com a complexidade teórica do sistema. O fator predominante é o custo do cálculo de rotas, que deve ser executado para cada pacote. A implementação do algoritmo de busca em largura em uma rede maior (com mais vértices V e arestas A) é inerentemente mais custosa. Além disso, uma rede maior tende a gerar rotas mais longas, o que aumenta o número total de eventos de armazenamento e transporte que precisam ser gerenciados pela fila de prioridades do Escalonador, elevando o custo de cada operação no min-heap (que é logarítmico em relação ao número de eventos). A maior complexidade da rede também aumenta a probabilidade de cenários de contenção, que exigem processamento adicional.

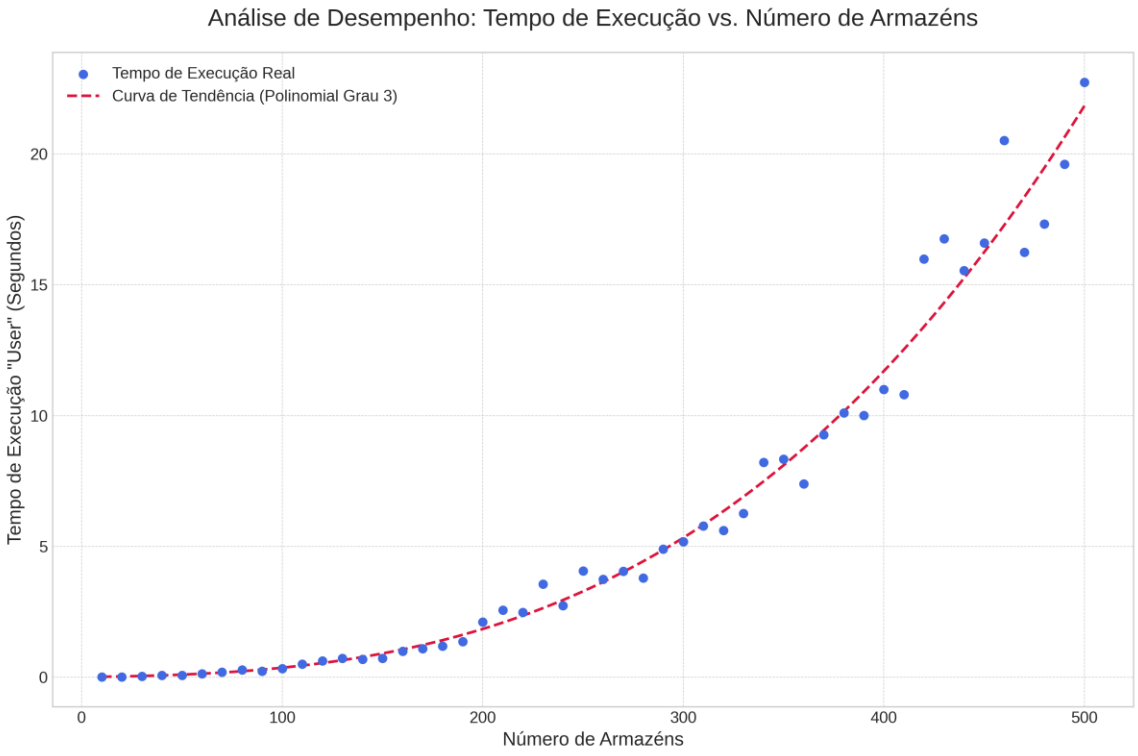

Figura 1 - Gráfico de Número de Armazéns x Tempo de Execução (Gerada pelo autor)

5.2 Análise do Impacto do Número de Pacotes

O segundo experimento visou avaliar como o sistema escala com o aumento da carga de trabalho. Para isolar este fator, a topologia da rede foi fixada em um tamanho relevante de 50 armazéns, enquanto o número de pacotes (P) foi variado em um amplo intervalo, de 100 a 5000.

Os resultados desta análise demonstraram uma tendência de crescimento fortemente linear entre o número de pacotes e o tempo de execução. Este é um indicativo muito positivo sobre a arquitetura do simulador, pois demonstra uma excelente e previsível escalabilidade em relação ao volume de itens a serem processados.

A linearidade pode ser explicada ao tratar cada pacote como uma unidade de trabalho relativamente independente dentro do sistema. Para uma topologia de rede fixa, o custo computacional médio para processar um único pacote, desde sua postagem até a entrega final, permanece praticamente constante. Este custo engloba um cálculo de rota inicial (cuja complexidade depende de V e A, que são fixos neste cenário) e o processamento de uma sequência de eventos. Como a topologia não muda, o comprimento médio das rotas também permanece constante, fazendo com que cada pacote gere, em média, um número similar de eventos.

Dessa forma, o custo total da simulação pode ser aproximado pela soma dos custos de processamento de cada pacote individual. Se o custo para processar um pacote é, em média, C, então o custo para processar P pacotes é aproximadamente $P*C$. Essa relação linear direta é o que o gráfico de resultados confirma.

As flutuações e o "ruído" observados nos pontos de dados, onde um aumento no número de pacotes pode ocasionalmente levar a uma redução marginal no tempo de execução, são atribuídos à natureza aleatória do gerador de carga. Uma determinada configuração de N pacotes pode, por acaso, resultar em um cenário com menos contenção e rotas mais simples do que uma configuração com N-100 pacotes, por exemplo. No entanto, a linha de tendência geral comprova de forma robusta a escalabilidade linear do sistema, uma característica fundamental para a confiabilidade e previsibilidade do simulador sob cargas de trabalho variáveis.

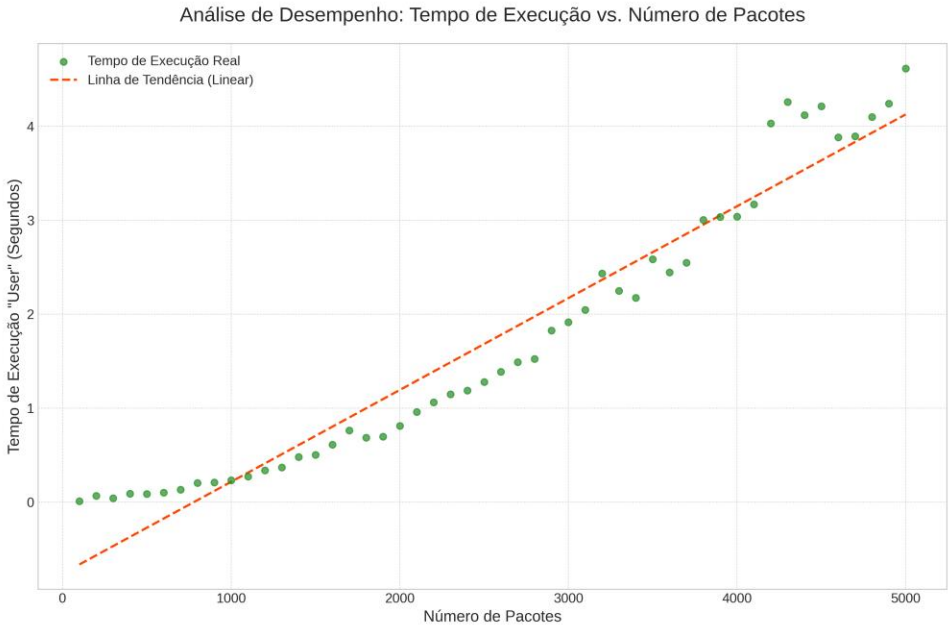

Figura 2 - Gráfico de Número de Pacotes x Tempo de Execução (Gerada pelo autor)

5.3 Análise do Impacto da Capacidade de Transporte

O último experimento foi projetado para avaliar o impacto direto da contenção por transporte no desempenho computacional da simulação. Para estressar a lógica do simulador, foi utilizada uma configuração de grande porte, com uma rede de 500 armazéns e uma carga de trabalho pesada de 5000 pacotes. Nesta configuração fixa, a capacidade de transporte, que define o limite de pacotes que podem ser movimentados por viagem entre dois armazéns conectados, foi sistematicamente variada em um intervalo de 1 a 30.

A análise dos dados coletados revelou um padrão clássico de desempenho de sistemas sujeitos a gargalos, ilustrando a lei dos rendimentos decrescentes de forma clara. O comportamento do tempo de execução pode ser dividido em três fases distintas, dependendo do nível de contenção imposto pela capacidade de transporte.

Na primeira fase, com capacidades de transporte muito baixas (entre 1 e 4), o tempo de execução da simulação é extremamente elevado. Com uma capacidade mínima, as rotas de transporte tornam-se rapidamente saturadas, criando um cenário de contenção severa em toda a rede. Este gargalo força o simulador a executar suas rotinas mais complexas e computacionalmente caras com alta frequência. Para cada ciclo de transporte, a função CoordenarCicloDeTransporte identifica que a maioria dos pacotes em uma seção não pode ser enviada, acionando a lógica de ProcessarRearmazenamento para um grande número de itens.

A segunda fase, em um intervalo de capacidade intermediário (aproximadamente entre 5 e 10), é caracterizada por uma queda drástica e não linear no tempo de execução. Cada incremento na capacidade de transporte alivia significativamente o gargalo do sistema. Com "caminhões" maiores, mais pacotes são escoados a cada ciclo, reduzindo o número de eventos de rearmazenamento e. É nesta fase que se observa o "joelho" da curva, onde o benefício de cada unidade adicional de capacidade começa a diminuir, indicando que o sistema está se aproximando de um ponto de equilíbrio.

Finalmente, na terceira fase, com capacidades de transporte elevadas (acima de 10), o gráfico estabiliza-se em um platô de desempenho. A partir deste ponto, o tempo de execução torna-se largamente independente de novos aumentos na capacidade. Isso ocorre porque o transporte deixou de ser o gargalo primário do sistema. A capacidade disponível já é suficiente para acomodar o fluxo de pacotes sem gerar contenção significativa, tornando a lógica de rearmazenamento raramente necessária. O desempenho da simulação passa a ser dominado por outros fatores de custo, como o processamento inicial de cálculo de rotas para todos os 5000 pacotes e o custo base de gerenciar a sequência de eventos (armazenamento, remoção, trânsito e entrega) para cada pacote no Escalonador.

Em suma, este experimento valida a robustez do simulador em modelar dinâmicas de contenção e demonstra sua utilidade para análises de otimização. Os resultados identificam um ponto de saturação claro, a partir do qual investir em maior capacidade de transporte traria um retorno marginal ou nulo para a eficiência do fluxo logístico, sendo uma informação valiosa para o planejamento de recursos em um cenário real.

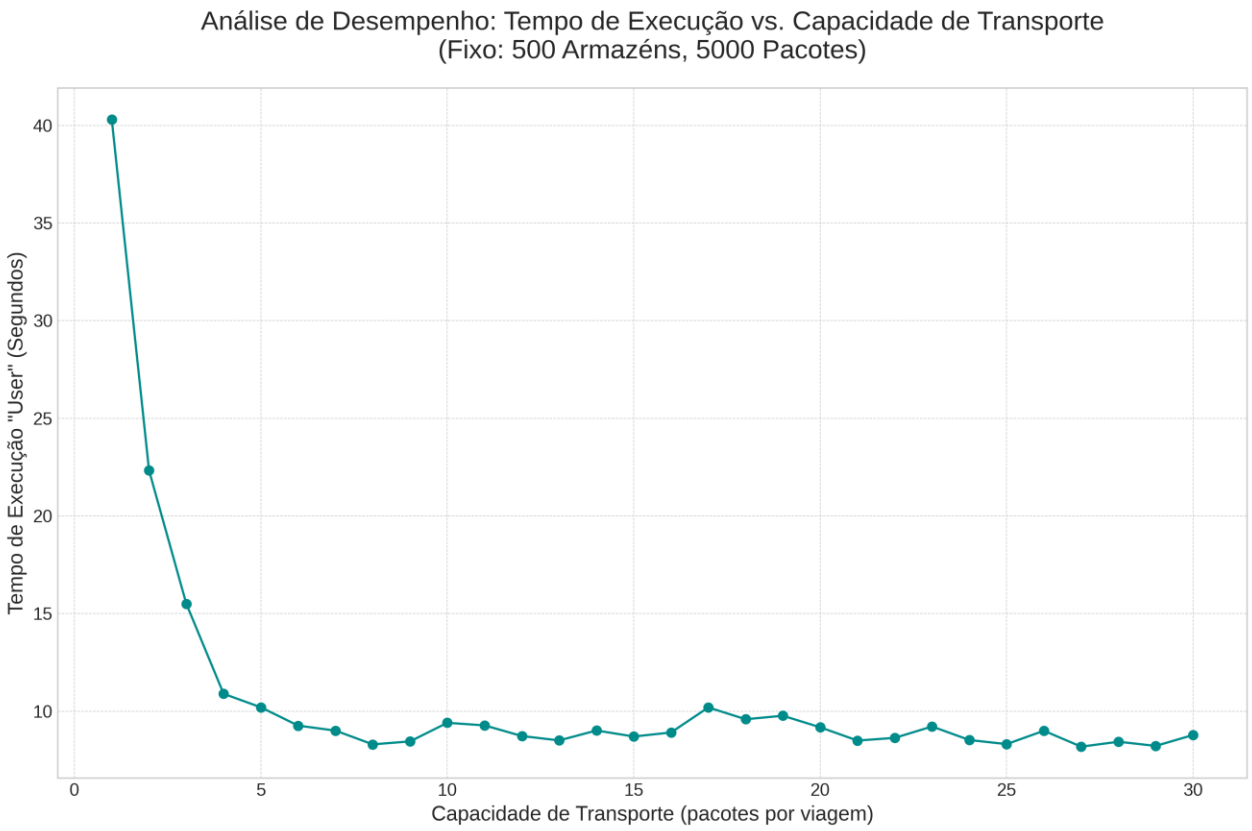

Figura 3 - Gráfico de Capacidade de Transporte x Tempo de Execução (Gerada pelo autor)

6. Conclusão

O presente trabalho dedicou-se ao desenvolvimento de um Sistema de Escalonamento Logístico, projetado para modelar e analisar a complexa rede de operações dos Armazéns Hanoi. O desafio central consistiu em criar um simulador de eventos discretos capaz de gerenciar o ciclo de vida de pacotes, desde a sua postagem até a entrega final, respeitando as restrições operacionais específicas do problema, como a organização de seções de armazenamento com lógica LIFO (Last-In First-Out) e a coordenação de ciclos de transporte com capacidade limitada. A solução foi implementada na linguagem C++, com uma arquitetura modular que abstrai os componentes do sistema.

A arquitetura foi fundamentada em um conjunto de Tipos Abstratos de Dados (TADs) que representam as entidades do domínio logístico. O TAD Pacote encapsula não apenas os dados de um envio, mas também o seu estado dinâmico e as estatísticas de sua trajetória. O TAD Armazem implementa a organização em seções e a manipulação de pacotes segundo a regra LIFO. O núcleo da simulação reside no TAD Escalonador, que, conforme as melhores práticas para simulação de eventos discretos, foi implementado com uma fila de prioridade baseada em um min-heap. Esta estrutura garante o processamento dos eventos em estrita ordem cronológica, permitindo que o relógio da simulação "salte" entre instantes de mudança de estado, uma característica que otimiza o desempenho computacional em comparação a uma simulação contínua.

A análise experimental, executada sobre a implementação completa, forneceu insights quantitativos sobre a escalabilidade e o comportamento do sistema. A avaliação do impacto do número de armazéns

revelou uma tendência de crescimento de desempenho não linear, confirmando a análise de complexidade teórica que aponta o cálculo de rotas e o gerenciamento de um volume crescente de eventos como os principais fatores de custo. Em contrapartida, a análise do número de pacotes demonstrou uma escalabilidade marcadamente linear, um resultado positivo que atesta a eficiência do sistema em lidar com o aumento da carga de trabalho sem degradação exponencial de desempenho. A análise mais reveladora foi a da capacidade de transporte, que ilustrou um claro padrão de rendimentos decrescentes. Os resultados mostraram que capacidades muito baixas levam a uma contenção severa, ativando rotinas computacionalmente caras de rearmazenamento e planejamento, enquanto capacidades elevadas levam a um platô de desempenho, onde o transporte deixa de ser o gargalo do sistema. Esta última análise valida a capacidade do simulador como uma ferramenta para otimização de recursos em cenários logísticos reais.

A jornada de desenvolvimento do Sistema de Escalonamento Logístico para os Armazéns Hanoi representou uma experiência dual, marcada por desafios técnicos significativos e, ao mesmo tempo, por um profundo sentimento de satisfação e aprendizado. A transposição de um problema complexo do mundo real para um modelo de simulação funcional exigiu não apenas a aplicação de conhecimentos teóricos, mas também a superação de obstáculos práticos de implementação.

Em suma, este trabalho prático foi muito mais do que um exercício de programação; foi uma imersão completa no ciclo de vida de um projeto de software de complexidade moderada, desde a interpretação de requisitos e o desenho de uma arquitetura robusta até a implementação, depuração e, finalmente, a análise crítica dos resultados. A experiência foi fundamental para solidificar a compreensão sobre a aplicação prática de estruturas de dados avançadas e para desenvolver uma apreciação mais profunda dos desafios e recompensas da Ciência da Computação.

A seguir, será detalhada a implementação da versão otimizada do sistema, que incorpora funcionalidades avançadas descritas na seção de "Pontos Extras" do enunciado do trabalho. Com o objetivo de modelar cenários logísticos mais realistas e complexos, foram implementadas três melhorias centrais, o suporte a tempo de transporte variável entre os diferentes armazéns, a capacidade de transporte também variável por rota e, crucialmente, a introdução de capacidade de armazenamento limitada nas seções dos armazéns. Essas modificações alteraram fundamentalmente a dinâmica da simulação. A introdução de latências distintas transformou a rede em um grafo ponderado, mudando a forma como a menor rota é calculada. Mais significativamente, a restrição de espaço finito introduziu a possibilidade real de congestionamento, o que exigiu o desenvolvimento de uma lógica sofisticada de planejamento e reroteamento dinâmico de pacotes para evitar armazéns sobrecarregados e otimizar o fluxo de entregas em tempo real.

Bibliografia

Cunha, Ferreira, Lacerda e Meira Jr. (2025). Slides virtuais da disciplina de Estruturas de Dados. Disponibilizado via moodle. Departamento de Ciência da Computação. Universidade Federal de Minas Gerais. Belo Horizonte.

7. Pontos Extras: Sistema Logístico Adaptativo

Para estender a fidelidade da simulação a cenários mais realistas e dinâmicos, a versão base do sistema foi aprimorada com a implementação de funcionalidades avançadas, conforme sugerido na seção de "Pontos Extras" do enunciado. Especificamente, foram incorporados tempo e capacidade de transporte variáveis por rota e capacidade de armazenamento limitada. Essas melhorias transformaram o simulador de um modelo com regras fixas para um sistema adaptativo, capaz de lidar com heterogeneidade na infraestrutura e gerenciar ativamente o congestionamento.

7.1 Capacidade e Latência de Transporte Variáveis

A primeira grande modificação foi a substituição dos parâmetros globais de latência e capacidade de transporte por valores específicos para cada rota. Isso permite modelar uma rede logística mais realista, onde diferentes conexões entre armazéns possuem características distintas como por exemplo, uma rota mais curta pode ter um "caminhão" menor, enquanto uma mais longa pode ter maior capacidade.

7.1.1 Mudanças no Formato de Entrada

Para suportar essa funcionalidade, o formato do arquivo de entrada foi alterado. Agora a capacidade de armazenamento é recebida no input e a matriz de adjacência única foi substituída por duas matrizes distintas, sendo uma matriz de Latência e uma matriz de capaciade, duas matrizes V x V onde cada entrada (i, j) especifica o tempo de transporte (latência) e a capacidade de transporte entre o armazém i e o armazém j.

7.1.2 Alteração no Cálculo de Rotas: Algoritmo de Dijkstra

Com a introdução de latências variáveis, o grafo da rede de armazéns tornou-se ponderado. Consequentemente, o algoritmo de Busca em Largura (BFS), que é ótimo apenas para encontrar o caminho mais curto em grafos não ponderados, tornou-se inadequado. Para garantir o cálculo da rota de menor custo (menor tempo total de trânsito), foi necessária a implementação do Algoritmo de Dijkstra.

Este algoritmo encontra o caminho mais curto de um nó de origem para todos os outros nós em um grafo ponderado com arestas de peso não negativo. Uma funcionalidade crucial foi adicionada à implementação de Dijkstra, a capacidade de especificar um nó proibido, essa capacidade é fundamental para a lógica de reroteamento, pois permite ao planejador calcular uma rota alternativa que evita um armazém congestionado, como será detalhado a seguir.

7.2 Capacidade de Armazenamento Limitada e Planejamento de Ciclo

A introdução de uma capacidade de armazenamento finita nas seções dos armazéns foi, de longe, a adição mais desafiadora e que mais impactou a arquitetura do sistema. Com espaço limitado, um pacote não pode ser simplesmente armazenado ao chegar a um armazém se a seção de destino estiver cheia. Isso introduz a possibilidade real de congestionamento e a necessidade de uma lógica proativa de planejamento para evitar o bloqueio total do sistema.

Para gerenciar essa complexidade, foi introduzido um novo TAD, o PlanejadorDeCiclo, e uma nova função central, a PlanejarCicloDeTransporte, que é executada antes de cada ciclo de transporte para tomar decisões globais sobre o movimento de pacotes.

7.2.1 O TAD PlanejadorDeCiclo

Este TAD cria um "ambiente de simulação dentro da simulação". Antes de cada ciclo de transporte, ele constrói um modelo do estado futuro de todas as seções da rede. Cada seção real é representada por uma SecaoSimulada, que contém informações sobre sua capacidade, os pacotes que estão nela atualmente (pacotesAtuais) e os pacotes que estão previstos para chegar ou sair (pacotesPrevistos). Isso permite ao planejador prever o fluxo de pacotes e identificar potenciais gargalos antes que eles ocorram.

7.2.2 A Função PlanejarCicloDeTransporte

Esta função orquestra toda a lógica de decisão e opera em seis passos principais:

Passo A: Inicialização do Ambiente de Simulação

A função começa criando uma instância do PlanejadorDeCiclo e populando-o com o estado atual de todas as seções de todos os armazéns da rede.

Passo B: Coleta e Priorização de Movimentos

O planejador analisa todos os pacotes em todas as seções e calcula o tempo de chegada previsto de cada um em seu próximo destino, caso fosse transportado neste ciclo. Esses pacotes, junto com suas previsões, são inseridos em um min-heap (pacotesOrdenadosPorChegada) para serem analisados em ordem de chegada prevista.

Passo B.2: Criação do Manifesto de Partidas

Após a coleta e priorização inicial de todos os movimentos de pacotes potenciais (Passo B), o planejador executa um passo de pré-cálculo fundamental, a criação de um Manifesto de Partidas. O objetivo desta etapa é construir uma visão global e sumária de todas as intenções de saída para o ciclo de transporte que está prestes a começar, antes de simular as complexas interações de chegada e congestionamento, sendo a base para as decisões inteligentes que serão tomadas no passo seguinte.

Passo C: Simulação do Fluxo e Detecção de Congestionamento

O planejador processa o min-heap, um pacote por vez. Para cada pacote que chegaria a um armazém, ele verifica se haverá espaço na próxima seção de saída, considerando a capacidade finita da seção e o fluxo de pacotes que já foram aprovados para chegar e sair dela. Se a previsão indicar que a seção ficará superlotada, o pacote é marcado como congestionado e colocado em uma fila de espera. Caso contrário, ele é aprovado para transporte.

No entanto, o número de pacotes que efetivamente podem sair não é apenas o número de pacotes que querem sair, mas é limitado pela capacidade física do transporte naquela rota. Portanto, o planejador realiza o cálculo final numSaindoEfetivo = min(contagemSaidas_do_manifesto, capacidadeDaRota_real) que é a previsão mais precisa possível do número de vagas que serão liberadas na seção B -> C.

Passo D: Análise de Pacotes em Espera e Reroteamento

Este é o passo mais crítico. Para cada pacote marcado como congestionado, o sistema tenta encontrar uma solução proativa para evitar que ele fique parado:

Cálculo de Rota Alternativa: O sistema invoca o algoritmo de Dijkstra, utilizando o recurso de nó proibido, para calcular uma nova rota do armazém atual do pacote até seu destino final, evitando o próximo armazém da rota original (que está congestionado).

Análise de Viabilidade: Uma nova rota só é considerada viável se atender a um conjunto de critérios rigorosos, para evitar que o reroteamento piore a situação. O pacote não é reroteado se:

Nenhuma rota alternativa existe.

A nova rota é muito mais longa: A latência da nova rota não pode exceder a latência restante da rota antiga mais o intervalo de um ciclo de transporte. Isso evita desvios que atrasariam demais a entrega.

A próxima seção da nova rota também está congestionada: O planejador verifica se o primeiro "salto" da nova rota levaria o pacote a outra seção que também não tem espaço previsto.

Remanejamento: Se, e somente se, todos os critérios forem atendidos, o pacote é efetivamente reroteado. Sua rota interna é atualizada, e ele é fisicamente movido da seção original congestionada para a nova seção de partida dentro do armazém.

Passo E: Execução do Plano

O Passo E é a fase de execução do plano, onde as decisões tomadas nas etapas anteriores são convertidas em ações concretas na simulação. Para organizar esse processo, utiliza-se a estrutura temporária GrupoDespacho para agregar os pacotes que foram aprovados para viajar.

Primeiramente, o sistema identifica todos os pacotes que não foram marcados para esperar e os agrupa por sua rota de partida (origem e seção de destino), criando um GrupoDespacho para cada "viagem". Em seguida, o sistema percorre esses grupos e, para cada um, agenda um único evento EVENTO_INICIA_TRANSPORTE no Escalonador.

Dessa forma, o GrupoDespacho atua como um agregador que garante que todos os pacotes aprovados para a mesma rota sejam despachados juntos em uma operação coordenada, executando o plano de forma eficiente e organizada.

Passo F: Reagendamento

Por fim, o próximo ciclo de planejamento global é agendado para ocorrer no próximo intervalo de transporte.

Essa arquitetura de planejamento proativo transforma o simulador em um sistema dinâmico, capaz não apenas de seguir rotas pré-calculadas, mas de se adaptar a gargalos em tempo real, refletindo a complexidade das operações logísticas modernas.

8. Prova de Valor

Para demonstrar a superioridade e o valor prático da versão otimizada do sistema, foi conduzido um experimento comparativo. O objetivo não era comparar o tempo de execução bruto, mas sim a eficiência logística de cada sistema ao enfrentar um cenário de rede com um gargalo controlado. Este teste foi projetado para expor as limitações do sistema base e, em contrapartida, destacar a capacidade de adaptação do sistema otimizado, que implementa as funcionalidades de pontos extras.

8.1 Metodologia do Experimento

O experimento foi construído sobre uma topologia estratégica de 4 armazéns, com duas rotas distintas conectando um armazém de origem (0) a um de destino (3), sendo a Rota A (Rápida e Estreita) um caminho 0 -> 1 -> 3 com baixa latência, mas também com capacidade de transporte e armazenamento extremamente limitadas e a Rota B (Lenta e Larga) um caminho 0 -> 2 -> 3 com latência significativamente maior, mas com alta capacidade de transporte e armazenamento.

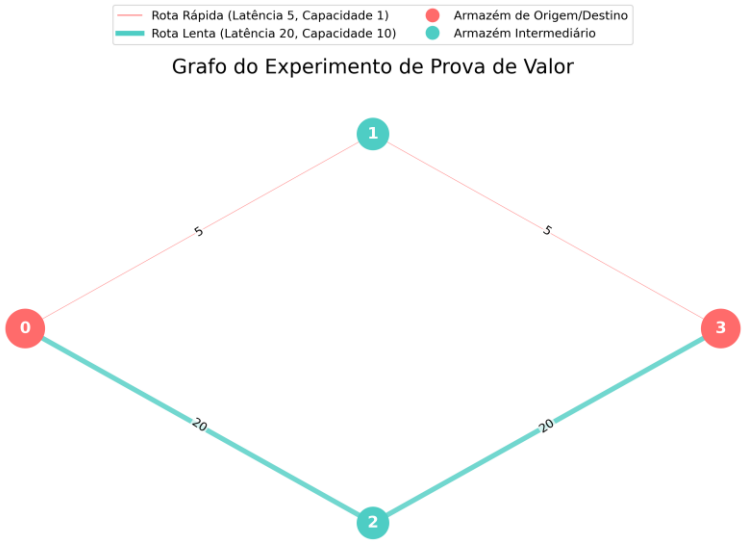

Figura 4 – Grafo do Experimento Prova de Valor

A carga de trabalho consistiu em 50 pacotes, todos postados no mesmo instante no armazém 0 com destino ao armazém 3, criando uma demanda massiva e simultânea.

No Sistema Base, esperava-se que o algoritmo BFS direcionasse todos os 50 pacotes para a Rota A, por ter o mesmo número de saltos que a Rota B. Devido à capacidade de transporte global de apenas 1 e ao armazenamento infinito, previa-se a formação de uma longa fila no armazém 1, com pacotes sendo entregues um a um a cada ciclo de transporte, resultando em tempos de entrega extremamente altos.

No Sistema Otimizado, esperava-se que o algoritmo Dijkstra também escolhesse inicialmente a Rota A por sua baixa latência. Contudo, a capacidade de armazenamento limitada a 5 na seção 1 -> 3 rapidamente criaria um gargalo. Previa-se que o PlanejadorDeCiclo detectaria esse congestionamento e utilizaria sua lógica de reroteamento para desviar o fluxo de pacotes subsequentes para a Rota B. Esse balanceamento de carga dinâmico permitiria o escoamento paralelo dos pacotes, reduzindo significativamente o tempo total para a conclusão de todas as entregas.

8.2 Análise Comparativa dos Resultados

A execução das duas simulações confirmou a hipótese com resultados contundentes, que demonstram o valor prático das otimizações implementadas. As métricas de eficiência logística, extraídas dos logs de simulação, revelam a superioridade do sistema otimizado:

Métrica	Sistema Base	Sistema Otimizado	Melhoria
Tempo Médio de Entrega	2245.98	569.88	74.6%
Tempo Médio Armazenado	2235.98	542.48	75.7%
Tempo Médio em Trânsito	10.00	36.40	-

Tabela 1 - Métricas de eficiência dos TADs (Gerada pelo Autor)

O sistema otimizado entregou os pacotes, em média, quase 4 vezes mais rápido que o sistema base. O tempo que os pacotes passaram parados em armazéns foi drasticamente reduzido em mais de 75%, evidenciando um fluxo logístico muito mais ágil. O aumento no tempo médio em trânsito no sistema otimizado é, neste caso, um indicador positivo, pois demonstra que os pacotes passaram mais tempo efetivamente se movendo (pela rota mais longa, porém livre) em vez de ficarem parados no gargalo, como ocorreu no sistema base.

Nota: Devido à sua extensão, os arquivos de entrada (input_base.txt, input_otimizado.txt) e os arquivos de saída completos (output_simples.txt, output_otimizado.txt) estão disponibilizados como materiais suplementares a este relatório.

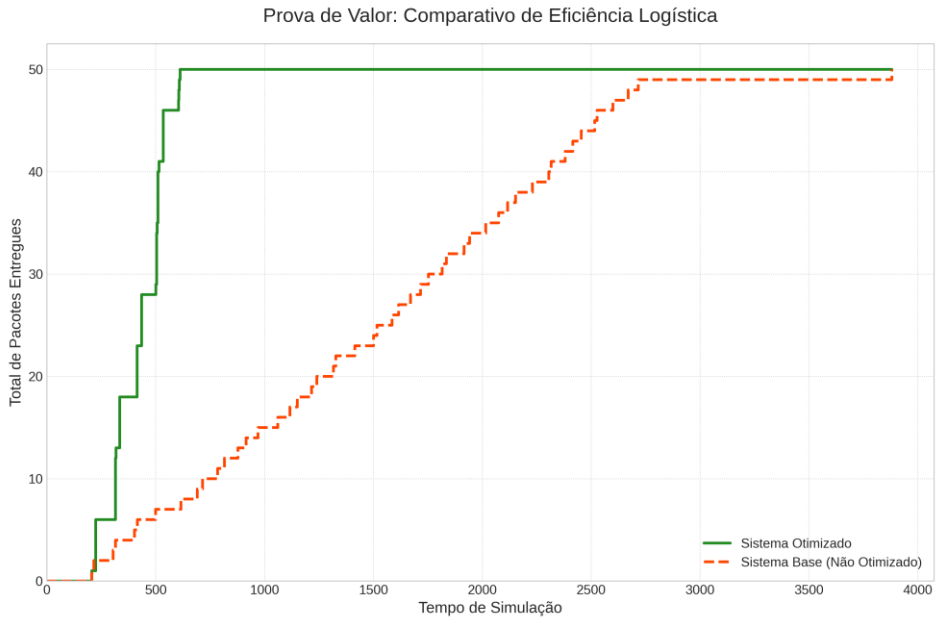

Figura 5 – Pacotes Entregues x Tempo da Simulação (Gerada pelo autor)

8.3 Conclusão da Prova de Valor

O experimento comparativo demonstra de forma inequívoca o valor agregado pelas funcionalidades avançadas implementadas no sistema otimizado. Enquanto a versão base, embora funcional em cenários simples, se mostrou incapaz de lidar eficientemente com um gargalo de rede, a versão otimizada utilizou suas capacidades de planejamento proativo e reroteamento dinâmico para se adaptar às condições da rede em tempo real.

O resultado foi uma melhoria drástica na eficiência logística, com uma redução de quase 75% no tempo médio de entrega. A capacidade de modelar e reagir a restrições heterogêneas de latência, transporte e armazenamento não é apenas um ponto extra, mas uma transformação fundamental que eleva o simulador de um modelo mecânico para uma ferramenta de análise estratégica robusta. Fica provado, portanto, que o investimento na complexidade arquitetural do planejador e do roteamento com Dijkstra é amplamente justificado pelos ganhos massivos em desempenho e pela capacidade de modelar com maior fidelidade os desafios de um sistema logístico real.