TRABALHO PRÁTICO 0:
Simulador de um elevador

Arlei Lopes da Silva

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

arlei@dcc.ufmg.br

Resumo. *Este relatório descreve um controlador de elevador e um simulador orientado a eventos para a geração de requisições de usuários de elevador. Uma estratégia de controle simples, similar à utilizada pela maioria dos elevadores, foi empregada e avaliada em termos de diferentes parâmetros. Os resultados mostram que, além de variáveis mais simples como número de andares e capacidade do elevador, fatores mais complexos, como a popularidade dos trajetos, também afetam o tempo de simulação e o tempo de espera dos usuários. O trabalho cumpriu seus principais objetivos: (1) a prática da linguagem C e (2) a introdução ao estudo de problemas complexos.*

1. INTRODUÇÃO

Neste trabalho, nós descrevemos, implementamos e analisamos um controlador de elevador. Para fins de análise, foi implementado um simulador orientado a eventos para a geração de requisições de transporte. Os principais objetivos do trabalho são o exercício da linguagem C, a discussão sobre problemas e complexos e sua solução.

O problema específico a ser estudado neste trabalho é a implementação de um simulador para a geração de chamadas de passageiros de um elevador e a avaliação de uma estratégia de controle para o atendimento das chamadas. A estratégia de controle do elevador deve atender as requisições obedecendo à restrição de capacidade do elevador (lotação máxima). A contagem de tempo é realizada através de uma unidade básica, denominada Jepslon. Um tempo total de 1 Jepslon é necessário para que o elevador percorra a distância entre dois andares ou para que ele realize uma operação de carga ou descarga de passageiros.

A estratégia de controle proposta neste trabalho é similar à aquela utilizada pela maioria dos elevadores convencionais. Em cada instante de tempo, o elevador pode estar percorrendo os andares nos sentidos para cima ou para baixo. Ao passar por cada andar, o elevador verifica se existem requisições de descarga de passageiros para o andar corrente ou requisições de carga de passageiros que desejam seguir o sentido atual do elevador. No caso de requisições de carga, é verificada a restrição de capacidade do elevador.

O simulador desenvolvido, que gera requisições para o elevador, é simples. Dado um conjunto de requisições, que representam passageiros com um andar de origem e destino, além de um instante de geração da requisição (momento em que o passageiro aperta o botão chamando o elevador), o simulador gera cada uma das requisições no tempo correto. Essas requisições são carregadas na memória do elevador.

Uma análise experimental do tempo de execução do simulador e do tempo médio de espera dos usuários foi realizada. Para isso, foi necessária a criação de um gerador de

requisições. A modelagem do tempo de simulação em termos das características do problema não é uma tarefa simples, já que ela precisa considerar fatores como a distribuição dos trajetos e distribuição dos tempos entre as requisições. Para ilustrar o efeito desses fatores, nós avaliamos como a popularidade dos trajetos afeta o elevador.

O restante deste relatório é organizado da seguinte forma. A Seção 2 discute alguns temas relacionados ao trabalho. A Seção 3 descreve o simulador, o elevador e a estratégia de controle proposta. A Seção 4 trata de detalhes específicos da implementação do trabalho. A Seção 5 contém a avaliação experimental do controlador de elevador proposto. A Seção 6 conclui o trabalho.

2. REFERÊNCIAS RELACIONADAS

Podemos dividir as referências associadas ao problema estudado e à solução proposta dentre os seguintes grupos:

- **Escalonamento, análise de desempenho:** Um grande número de problemas em computação pode ser modelado como um conjunto de requisições à um dado serviço. Nesse caso, a estratégia de atendimento à essas requisições pode afetar o desempenho do serviço significativamente. Além disso, também é importante entender as características da entrada (carga), já que elas determinam diversas características do serviço. Mais informações sobre o assunto podem ser encontradas em [Jain 1991].
- **Linguagem C:** Todo o trabalho foi implementado em linguagem de programação C. A linguagem C é uma linguagem de propósito geral amplamente utilizada no desenvolvimento de sistemas de alto desempenho, de tempo real, sistemas operacionais, compiladores, dentre outros. O compilador mais utilizado para a linguagem C é o GCC (desenvolvido pelo projeto GNU). Mais informações sobre a linguagem C podem ser encontradas em [Kernighan et al. 1988].
- **Projeto e análise de algoritmos:** Algoritmos são procedimentos computacionais capazes de resolver diversos problemas do mundo real. O problema de controlar um elevador, estudado neste trabalho, é um dos exemplos de como algoritmos estão presentes no nosso dia-a-dia. O estudo de algoritmos é essencial para o desenvolvimento de técnicas mais eficientes e eficazes para a solução desses problemas. Mais informações sobre o projeto e análise de algoritmos podem ser encontradas em [Ziviani 2007, Cormen 2001, Sedgewick 2001]

3. SOLUÇÃO PROPOSTA

A solução proposta para a simulação discreta do funcionamento de um elevador utiliza um conjunto de estruturas de dados que contêm informações sobre o estado da simulação, do elevador, registros de requisições, dentre outros. Essas estruturas são manipuladas através de um conjunto de algoritmos que descrevem o funcionamento da solução.

O simulador implementado tem como base um temporizador contínuo, ou seja, cada instante de simulação corresponde à uma iteração do simulador. Uma outra opção seria utilizar como base a contagem das requisições. Os critérios que levaram à escolha da iteração do simulador em cada instante são: (1) a simplicidade de implementação, (2) mapeamento direto entre tempo de execução e tempo de simulação e (3) o custo computacional reduzido de avaliar cada instante de simulação, supondo-se que as requisições sejam bem distribuídas ao longo do tempo.

O controle do elevador é baseado no que chamamos de *requisição corrente*, que é a requisição que define o sentido do elevador, e é sempre a requisição do andar mais alto quando o elevador está subindo ou a requisição do andar mais baixo quando o elevador está descendo. Através da identificação de uma requisição corrente é possível determinar o trajeto do elevador ao longo do tempo. Durante o percurso na direção de atender uma requisição corrente, outras requisições podem ser atendidas, sempre considerando as restrições de espaço do elevador.

A seguir, nós descrevemos as principais estruturas de dados e algoritmos propostos. Nós analisamos a complexidade dos algoritmos em termos do número de requisições (n), do número de andares (a) e da capacidade do elevador (c). A complexidade de espaço será determinada com base nas estruturas de dados e na análise do algoritmo principal, que determina a complexidade da solução.

3.1. Estruturas de dados

3.1.1. Requisição:

Armazena informações sobre uma requisição de transporte para o elevador.

1: Requisicao
identificador;
hora requisicao;
andar atual;
andar desejado;

3.1.2. Simulador

Armazena informações para a simulação, ou seja, uma fila de requisições de transporte a ser tratada pelo elevador. A complexidade de espaço do armazenamento das requisições é $O(n)$, onde n é o número de requisições.

2: Simulador
numero de requisicoes;
requisicoes;

3.1.3. Registro de requisição

Armazena um registro de requisição, que é uma unidade da memória de requisições a serem atendidas pelo elevador. Possui, além da requisição, um tipo, que a define se a operação necessária é de carga ou descarga de passageiro.

3.1.4. Elevador

Armazena informações sobre o elevador, como seu estado e algumas restrições definidas através do ambiente de execução. Contém uma lista de registros que representa a memória

3: Registro de requisicao

```
requisicao;
tipo;
```

de requisições do elevador. A complexidade de espaço do armazenamento dos registros é $O(n)$, onde n é o número de requisições.

4: Elevador

```
andar atual;
lotacao;
capacidade;
numero de andares;
sentido;
numero de registros;
registros;
```

3.2. Algoritmos

3.2.1. Atende requisição de carga

Realiza o atendimento de uma requisição de carga para o elevador. No caso da requisição atendida ser a corrente, o elevador altera o sentido de seu movimento. Cada requisição de carga está sujeita à limitação de espaço (lotação máxima) do elevador. Além disso, uma requisição de carga só é atendida quando o elevador se movimenta no sentido do andar de destino da requisição. Caso contrário, a requisição deve aguardar a passagem do elevador no futuro. A complexidade desse algoritmo em termos de tempo de execução é $O(n)$, onde n é o número de requisições, devido ao custo de identificação da requisição corrente, realizado através de uma busca linear.

5: atende-requisicao-carga(elevador, registro requisicao)

```
if requisicao corrente then
  | Inverte sentido elevador;
end
Atualiza lotacao elevador;
Gera requisicao de descarga;
```

3.2.2. Atende requisição de descarga

Similar ao atendimento de uma requisição de carga, apesar de não estar sujeita à restrições de capacidade do elevador. A complexidade desse algoritmo em termos de tempo de execução também é $O(n)$.

3.2.3. Opera elevador

Controla a movimentação do elevador ao longo do tempo. Caso haja alguma requisição a ser atendida em um dado momento, ou seja requisições para o andar atual, aquele in-

6: atende-requisicao-descarga(elevador, registro requisicao)

```
if requisicao corrente then
    Inverte sentido elevador;
end
Atualiza lotacao do elevador;
gera relatorio de atendimento da requisicao;
apaga registro da memoria do elevador;
```

stante será utilizado para o atendimento dessas requisições. Caso contrário, é identificada a requisição corrente e o elevador é movimentado no sentido dessa requisição. A complexidade desse algoritmo em termos de tempo de execução é $O(n^2)$, onde n é o número de requisições, que corresponde ao custo de atender toda a fila de requisições, sendo que o atendimento de cada requisição tem custo $O(n)$.

7: opera-elevador(elevador, tempo)

```
if Ha requisicoes para o andar then
    Atende requisicoes descarga;
    Atende requisicoes carga;
else
    movimenta elevador no sentido da requisicao corrente;
end
```

3.2.4. Simula elevador

Determina o funcionamento da simulação. Para cada instante de tempo, em que haja uma requisição (na fila do simulador ou na memória do elevador), o elevador é operado (segundo a função opera-elevador), e as requisições de carga para aquele dado instante são armazenadas na memória do elevador, segundo a fila de requisições do simulador. Determinar a complexidade da simulação não é uma tarefa trivial, já que depende do tempo total de simulação (t). Devido ao custo de operação do elevador (função $opera - elevador$), sabemos que a complexidade de tempo do algoritmo é $O(n^2 \times t)$. Além disso, t é inversamente proporcional à capacidade do elevador (c) e diretamente proporcional ao número de andares (a) e ao número de requisições (n). No entanto, a relação entre t, c, a e n é afetada por:

- A popularidade dos trajetos no elevador: o que afeta o tempo de transporte, se caminhos mais longos tendem a ser mais frequentes o tempo de simulação aumenta.
- O tempo entre as requisições: se as requisições forem muito separadas no tempo, a simulação tende a ser longa, além disso, isso pode reduzir a taxa de utilização do elevador, através da frequência de trajetos em que elevador esteja quase vazio.

Acreditamos que a modelagem do problema considerando diferentes distribuições de popularidade e tempos entre requisições não está no escopo deste trabalho. A complexidade de espaço do algoritmo é $O(n)$, onde n é o número que requisições do elevador, já que é necessário é armazenamento de cada requisição na fila do simulador e na memória do elevador.

8: simula(sequencia requisicoes, ambiente)

```
tempo = 0;
while Ha requisicoes do
    opera-elevador(elevador,tempo);
    Registra requisicoes na memoria do elevador de acordo com o simulador;
    tempo++;
end
```

4. IMPLEMENTAÇÃO

4.1. Código

4.1.1. Arquivos .c

- **principal.c:** Arquivo principal do programa que implementa o simulador do elevador.
- **simulador.c:** Define as funções relacionadas à simulação do elevador.
- **lista.c:** Define funções relacionadas a manipulação de um TAD lista implementada através de apontadores.
- **fila.c:** Define as estruturas de dados e cabeçalhos de funções relacionadas a manipulação de um TAD fila implementada através de apontadores.

4.1.2. Arquivos .h

- **simulador.h:** Define as estruturas de dados e cabeçalhos de funções relacionadas a simulação do elevador
- **lista.h:** Define as estruturas de dados e cabeçalhos de funções relacionadas à manipulação de um TAD lista implementada através de apontadores.
- **fila.h:** Define as estruturas de dados e cabeçalhos de funções relacionadas a manipulação de um TAD fila implementada através de apontadores.

4.2. Compilação

O programa deve ser compilado através do compilador GCC através de um makefile ou do seguinte comando:

```
gcc principal.c simulador.c fila.c lista.c -o tp0
```

4.3. Execução

A execução do programa tem como parâmetros:

- Um arquivo de descrição do ambiente.
- Um arquivo de requisições.
- Um arquivo de saída.

O comando para a execução do programa é da forma:

```
./tp0 -a <arquivo de ambiente> -r <arquivo de requisicoes> -s <arquivo de saida>
```

4.3.1. Formato da entrada

O arquivo de descrição do ambiente contém apenas uma linha onde são definidos o número de andares do prédio atendido pelo elevador e a capacidade de elevador, separados por espaço. Já o arquivo de requisições contém o número de requisições na primeira linha seguido de cada uma das requisições.

Cada requisição contém um identificador, o instante de chamada do elevador (que tem como base um temporizador sequencial), o andar de origem do passageiro e o andar desejado. Um exemplo de arquivo de requisições é dado a seguir:

```
3
0 0 0 10
1 3 0 3
2 5 13 15
```

4.3.2. Formato da saída

A saída do programa, armazenada em um arquivo de saída, contém informações sobre todas as requisições atendidas. Para cada requisição as informações de interesse são o identificador da requisição, o tempo de espera de carga (até que a pessoa entre no elevador) e o tempo de espera dentro do elevador. A seguir, a saída correspondente à entrada apresentada na seção anterior.

```
id :0
espera carga :1
espera elevador :11

id :1
espera carga :27
espera elevador :4

id :2
espera carga :11
espera elevador :31
```

5. AVALIAÇÃO EXPERIMENTAL

Nesta seção nós avaliamos a estratégia de controle de elevador proposta e o simulador implementado em termos do tempo de execução (em segundos) e do tempo total de espera dos usuários (em Jepslons) para diferentes números de andares, capacidades do elevador, números de requisições e popularidades de trajetos. O tempo total de espera é a soma dos tempos de espera para entrar no elevador e para chegar ao andar desejado. Os experimentos foram executados em um computador pessoal com sistema operacional Suse Linux, processador AMD Athlon 3500+ 64 bits e 1GB de memória principal.

Como destacado anteriormente, o tempo de execução provê, além do tempo para que toda a simulação seja realizada, uma aproximação sobre o quão demorado é o atendimento de todas as requisições do elevador (já que cada instante de tempo é simulado em uma iteração). Já o tempo de espera pelo elevador é uma maneira simples de avaliar a qualidade do serviço prestado. Outros critérios como utilização do elevador, distância total percorrida pelo elevador, dentre outros, poderiam ser avaliados. Além disso, também

seria interessante considerar como o tempo entre as requisições afeta o elevador, no entanto, devido à restrições de tempo e espaço, não avaliaremos todos os parâmetros da simulação.

As entradas para a simulação do elevador foram geradas a partir de um gerador sintético de requisições. Esse gerador recebe como parâmetros: (1) o número de andares, (2) a capacidade do elevador, (3) um atributo que determina a popularidade dos trajetos e (4) o número de requisições. Como saída, o gerador cria um arquivo de ambiente e um arquivo de requisições. O atributo que determina a popularidade dos trajetos pode ser: (1) escolha aleatória, ou seja, trajetos escolhidos aleatoriamente, (2) probabilidade do trajeto proporcional ao tamanho, ou seja, trajetos mais longos são mais frequentes, e (3) probabilidade do trajeto inversamente proporcional ao tamanho do trajeto, o que resulta em maior frequência de trajetos curtos.

Em todos os experimentos nós variamos um parâmetro e mantivemos os outros constantes. Cada experimento foi executado 5 vezes e foram tomados os valores médios. Nosso objetivo é reduzir o impacto de propriedades de uma entrada específica produzida pelo gerador de entradas. Além do tempo de execução, foi medido também o tempo de usuário, que é o tempo gasto pelo processo do programa. O tempo de sistema, utilizado na realização de tarefas no nível do kernel, também foi medido, mas não é mostrado nos resultados pois é muito menor que os tempos de usuário e execução. O tempo médio entre as requisições do elevador foi mantido constante, igual a 3 Jepslons.

A Tabela 1 mostra como o aumento do número de andares afeta os tempos de execução, usuário e espera dos usuários. O número de andares varia de 25 a 40. O número de requisições é 50000, a capacidade do elevador é de 10 usuários e a escolha dos trajetos é aleatória. É importante notarmos que, ao aumentarmos o número de andares, o tamanho médio dos trajetos tende a aumentar. O número de andares afeta significativamente os tempos de execução, usuário e espera. Experimentos com número de andares maior que 40 requerem tempos de execução muito longos. Nossa hipótese de que esse impacto se deve ao aumento dos trajetos pode ser explicada pelo grande aumento do tempo de espera, ou seja, os usuários estão permanecendo no elevador por um longo período de tempo.

#andares	Tempo execução(seg.)	Tempo usuário(seg.)	Tempo de espera (Jepslons)
25	2,03	1,85	56,19
30	3,00	2,77	76,13
35	6,96	6,69	140,37
40	2068,28	1952,83	1947,29

Tabela 1. Tempo de simulação e de espera para diferentes números de andares

A Tabela 2 mostra impacto da capacidade do elevador sobre o tempo de simulação e o tempo de espera. A capacidade varia de 10 a 13 lugares. O número de andares é 40, o número de requisições é 50000 e a escolha dos trajetos é aleatória. Como era esperado, o tempo de simulação e o tempo de espera são inversamente proporcionais à capacidade do elevador.

A Figura 1 avalia o tempos de simulação e de espera em termos do número de requisições. O número de requisições varia de 30000 a 50000. O número de andares é 40, a capacidade do elevador é 10 e a escolha dos trajetos é aleatória. Como indicado na análise de complexidade, o custo computacional do algoritmo é polinomial em relação ao

capacidade	Tempo execução(seg.)	Tempo usuário(seg.)	Tempo de espera (Jepslons)
10	2068,28	1952,83	1947,29
11	17,63	17,16	234,37
12	5,43	5,20	121,51
13	4,31	4,07	101,04

Tabela 2. Tempo de simulação e de espera para diferentes capacidades do elevador

número de requisições. O crescimento do tempo de execução se aproxima de uma função quadrática, também de acordo com a análise de complexidade. Já o tempo de espera aumenta linearmente com o número de requisições, uma hipótese para esse resultado é que quanto maior o número de requisições, mais tempo é necessário para que o simulador gere todas as requisições para o elevador. A partir do momento que todas as requisições foram geradas, a tendência é que a fila de requisições se reduza gradativamente. No entanto, um longo período com geração constante de requisições pelo simulador congestiona a fila de registros de requisição do elevador, aumentando o tempo médio de espera.

(a)(b)

Figura 1. Tempo de simulação e de espera para diferentes números de requisições

Como última parte de nossa avaliação experimental, nós analisamos como a popularidade dos trajetos afeta os tempos de execução e espera. Três relações entre as frequências dos trajetos e seus tamanhos foram consideradas: (1) trajetos escolhidos aleatoriamente, (2) probabilidade do trajeto diretamente proporcional à sua distância e (3) probabilidade do trajeto inversamente proporcional à sua distância. O número de andares é 40, a capacidade do elevador é 10 e o número de requisições é 15000. A Tabela 3 mostra os resultados obtidos. Podemos notar que a distribuição das requisições afeta significativamente os tempos de execução e espera, principalmente o tempo de execução. Quando trajetos longos se tornam mais frequentes, o tempo de execução cresce, não somente devido ao aumento do número de iterações de simulador, mas também do custo da manipulação de um grande número de requisições. Já o tempo de espera, menos afetado, aumenta devido à uma maior carga de trabalho para o elevador.

Frequência tamanho dos trajetos	Tempo de execução(seg.)	Tempo de espera(seg.)
Aleatória	73,52	778,48
Proporcional à distância	9.011,42	7.011,46
Inv. Proporcional à distância	2,24	100,61

Tabela 3. Tempos de simulação e de espera para diferentes frequências de tamanhos de trajeto

6. CONCLUSÃO

Neste trabalho nós descrevemos um simulador discreto de eventos e uma estratégia de controle de um elevador. O simulador gera requisições a serem atendidas ao longo do

tempo de simulação e o elevador atende à essas requisições de acordo com uma estratégia similar à utilizada pelos elevadores convencionais.

O trabalho atingiu seus principais objetivos: a prática da linguagem de programação C e o estudo de problemas complexos. O autor do trabalho já havia implementado alguns programas nessa linguagem e, portanto, não teve grandes problemas na implementação do trabalho. Já a solução do problema demandou a modelagem de diversos cenários, de acordo com diferentes estados do elevador. Uma importante decisão foi a simulação de cada Jepslon e não de instantes específicos associados a eventos, o que facilitou o projeto da solução.

A análise de complexidade da solução proposta não foi uma tarefa simples, já que o custo computacional da solução está sujeito à diversas características da entrada. Por outro lado, o impacto de diferentes fatores associados à entrada pôde ser avaliado através de diversos experimentos realizados.

Algumas melhorias que poderiam ser consideradas neste trabalho são:

- Uma modelagem mais complexa do tempo de simulação em termos da popularidade dos trajetos, do tempo entre as requisições, dentre outros;
- A análise do comportamento do elevador considerando características como a distância percorrida e a taxa de utilização do elevador.

Referências

Cormen, T. (2001). *Introduction to algorithms*. MIT press.

Jain, R. (1991). *The art of computer systems performance analysis*. John Wiley & Sons New York.

Kernighan, B., Ritchie, D., and Ejeklint, P. (1988). *The C programming language*. Prentice-Hall Englewood Cliffs, NJ.

Sedgewick, R. (2001). Algorithms in C.

Ziviani, N. (2007). *Projeto de Algoritmos com implementações em Pascal e C*. Pioneira Thomson Learning.