Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

Estudo e Análise de Desempenho de Algoritmos para Elevadores Inteligentes

Autor:

_____

Thais de Oliveira Sobral

Orientador:

_____

Prof. Heraldo Luis Silveira de Almeida, D. Sc.

Examinador:

_____

Prof. Sergio Barbosa Villas-Boas, Ph.D.

Examinador:

_____

Prof. Sergio Palma da Justa Medeiros, D. Sc.

DEL

Maio de 2011

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica – Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro – RJ CEP 21949-900

DEDICATÓRIA

Ao meu irmão, meu pai e minha mãe, e a diretora do colégio Alfa Cem, pois sem o apoio deles eu não chegaria até aqui.

AGRADECIMENTO

Agradeço a todos que estiveram envolvidos na minha graduação, que me apoiaram, me incentivaram e tornaram este sonho possível. À minha família que teve papel fundamental na minha educação, aos meus amigos de classe que dividiram comigo os momentos difíceis e tornaram as aulas muito mais agradáveis, e a todos os meus professores e meu primeiro chefe de estágio, que se dedicaram e me ensinaram muito.

RESUMO

Este trabalho contém um estudo sobre algoritmos para grupos de elevadores inteligentes. O entendimento sobre o funcionamento dos elevadores convencionais e o que se espera do bom desempenho de um elevador eficiente são obtidos através da análise, observação e implementação de alguns desses algoritmos. A proposta de uma nova solução com o uso de algoritmos genéticos foi desenvolvida, testada, e seus resultados foram comparados com outros algoritmos.

Palavras-Chave: algoritmo genético, otimização, elevador inteligente, simulação.

ABSTRACT

This work presents a study on algorithms for groups of smart elevators. The understanding of the operation of conventional elevators and the expected performance of an efficient elevator are obtained through analysis, observation and implementation of some of these algorithms. The proposal for a new approach with the use of genetic algorithms has been developed, tested, and results were compared with other algorithms.

Key-words: genetic algorithm, optimization, smart elevator, simulation

SIGLAS

AJT – *Average Journey Time*

AWT – *Average Waiting Time*

DC – *Destination Control*

HI – *Heavy Incoming*

HL – *Heavy Lunch-time*

HO – *Heavy Outcoming*

HT – *Heavy Two-way*

MI – *Moderate Incoming*

ML – *Moderate Lunch-time*

MO – *Moderate Outcoming*

MT – *Moderate Two-way*

UFRJ – Universidade Federal do Rio de Janeiro

Sumário

1 Introdução 1

1.1 - Tema 1

1.2 - Delimitação 1

1.3 - Justificativa 2

1.4 - Objetivos 2

1.5 - Metodologia 3

1.6 - Descrição 3

2 Funcionamento dos Elevadores 5

2.1 - Panorama histórico 5

2.2 - Princípios de atendimento 6

2.3 - Controle de grupo de elevadores 8

2.4 - Algoritmos para chamadas de Sobe-Desce: Alocação contínua 9

2.5 - Algoritmos para chamadas com informação de desino: Alocação imediata 12

3 Motivação para Nova Abordagem 14

3.1 - O elevador como um transporte coletivo 14

3.2 - Problemas e soluções em transporte público 14

3.3 - Soluções para o futuro 16

4 Solução Proposta 18

4.1 - Visão geral sobre algoritmos genéticos 18

4.2 - Soluções em algoritmos genéticos aplicada ao problema 20
4.3 - Solução em algoritmo sobe-e-desce. 24

5 Simulador de Algoritmos 27

5.1 - A ferramenta desenvolvida 27
5.2 - Funcionalidades 28
5.3 - Premissas 29
5.4 - Controle de tempo 29
5.5 - Ciclos de execução 32
5.6 - Estruturas e funções 33
5.7 - Verificações e Soluções de Contorno 35

6 Testes 37

6.1 - Casos de teste 37
6.2 - Teste 1 - Algoritmo Genético x Sobe-e-desce 39
6.3 - Teste 2 - Algoritmo Genético x ETA 39
6.4 - Resultados encontrados 39

7 Conclusão 41

7.1 - Objetivos Alcançados 41
7.2 - Próximas Atividades 41

Bibliografia 43

A Código Fonte – Algoritmo Genético 44

B Código Fonte – Algoritmo Sobe-e-Desce 48

Lista de Figuras

2.1 – Ilustração de terminal externo com e sem informação de destino 6

2.2 – Ilustração dos modos coletivo e seletivo de atendimento 7

2.3 – Gráficos de fuzzificação das componentes de tráfego e fluxo 10

2.4 – Comparação entre o embarque de passageiros no sistema convencional de alocação contínua e o sistema de alocação imediata 12

2.5 – Algoritmos de customização e priorização de atendimento 13

3.1 – Semelhanças no atendimento de elevadores e ônibus 15

4.1 – Ilustração de uma rota permita na solução proposta. 20

4.2 – Algoritmo genético aplicado à solução 21

4.3 – Exemplo de um cromossomo 23

4.4 – Diferentes rotas adotadas por um elevador para atender uma chamada de andar 25

5.1 – Janela do simulador de algoritmos 27

5.2 – Esquema de tempos da rotina de um elevador. 31

5.3 – Matriz de tempos de deslocamento entre andares 31

5.4 – Cálculo de indicador de tempo médio de espera de um passageiro 32

5.5 – Pseudo-código do controle dos algoritmos. 34

5.6 – Escolha do elevador mais capaz. 35

7.1 – Ilustração para o caso de rotas de trânsito. 43

Lista de Tabelas

2.1 – Fragmento da tabela utilizada para reconhecimento de tráfego 10

4.1 – Correspondência entre os termos biológicos e computacionais 19

4.2 – Parâmetros configuráveis no código do algoritmo. 24

5.1 – Eventos e instantes de um passageiro. 30

5.2 – Principais variáveis do sistema simulador. 34

6.1 – Parâmetros dos prédios e as configurações dos elevadores 38

6.2 – Padrões de tráfego 38

6.3 – Resultado dos testes. 40

Capítulo 1

Introdução

1.1 – Tema

No ano 2011 não é novidade dizer que o avanço da tecnologia vem produzido grandes mudanças no cotidiano do homem. Com a democratização da internet no Brasil e o aumento do acesso por todo o mundo, um trabalhador moderno precisa estar constantemente conectado e ser capaz de se adaptar às constantes mudanças nas atividades do seu trabalho e da sua vida pessoal. Ou seja, precisa de cada vez mais tempo pra conseguir realizar tantas tarefas. Porém, o congestionamento das grandes cidades tem sido o principal obstáculo a esse avanço, fazendo com que em média os trabalhadores percam diariamente até 3 horas no trânsito [1].

É neste cenário que surgiu a idéia deste trabalho. Assim como o que acontece nas ruas, o trânsito também chega aos grandes edifícios e provocam longas esperas na fila por um elevador. Este trabalho irá falar sobre os algoritmos que controlam o movimento de um grupo de elevadores, fazendo uma analogia com os sistemas de ônibus urbano, e buscando uma solução alternativa para minimizar o tempo médio gasto diariamente nestes veículos. A solução para esse tipo de sistema tem recebido muita atenção devido a sua importância teórica e aplicação prática (Sutton & Barto 1998) apud [2].

1.2 – Delimitação

O estudo deste trabalho se concentra na otimização de algoritmo de controle de um conjunto de elevadores idênticos. Algoritmos desta natureza, também conhecidos como algoritmos inteligentes, são amplamente utilizados nos edifícios mais modernos em todo o mundo. Estes softwares controladores utilizam o poder e a agilidade computacional disponível atualmente, para tomar decisões em tempo real a partir do processamento do histórico recente de comandos, e da interpretação de sinais enviados por sensores de peso, movimento e posição, espalhados por todo o sistema.

Com o propósito de utilizar o que há de mais moderno em tecnologia de elevadores, este projeto terá como foco os elevadores onde a informação do andar de destino do passageiro deve ser informada

no piso do andar antes que ele ingresse na cabine. Esta mudança no sistema convencional de chamar um elevador tem vantagens e desvantagens, que serão estudadas mais à frente.

É importante observar que, apesar de implementar e testar estes algoritmos de otimização apenas no cenário de elevadores, devido às similaridades entre os problemas, a solução desenvolvida neste estudo também poderá ser utilizada como uma solução alternativa de controle de rotas de uma nova geração de transporte urbano.

1.3 – Justificativa

A melhoria na qualidade de vida dos moradores das grandes cidades é a principal justificativa para o estudo destes algoritmos. A redução no tempo médio de espera por um elevador proporciona aos seus usuários alguns minutos livres a mais por dia. Estes poucos segundos economizados, quando gastos na fila de um elevador, são integralmente sentidos pelo usuário e capazes de aumentar a sua irritação e impaciência com os congestionamentos dos atuais meios de transporte.

Eu, como moradora de uma grande e populosa cidade que é o Rio de Janeiro, compartilho todos os dias destes transtornos e inconvenientes diários. E, na tentativa de minimizar este desperdício de tempo ocioso, fui motivada a iniciar este estudo.

1.4 – Objetivos

O objetivo deste trabalho é desenvolver uma solução em algoritmo de controle de elevadores idênticos, utilizando técnicas de inteligência artificial, e comparar os resultados obtidos com a solução proposta com os demais algoritmos já conhecidos.

A solução desenvolvida neste projeto deve ter como objetivo minimizar o tempo de espera por um elevador, desde o inicio da realização do chamado até a chegada ao seu destino. E a comparação realizada no final dos testes deve ser capaz de enumerar e quantificar as vantagens de cada algoritmo estudado.

1.5 – Metodologia

Para desenvolver um algoritmo que atingisse o objetivo esperado, foi realizado um estudo prévio sobre inteligência artificial, mais especificamente sobre algoritmos genéticos. Este estudo foi fundamental para a modelagem e o desenvolvimento da solução de acordo com as regras e normas estabelecidas pela teoria.

Após a viabilização da idéia, conseguida através da implementação e dos bons resultados conseguidos com os primeiros testes, seguiu a etapa de pesquisas em artigos acadêmicos sobre o estado da arte. Através destas pesquisas, foi possível conceber o funcionamento padrão dos elevadores convencionais e inteligentes, para desenvolver algoritmos similares a serem utilizados nos testes de desempenho.

A estratégia não usual utilizada neste projeto – desenvolvimento seguido de pesquisa – foi utilizada para que, o conhecimento prévio das técnicas atuais utilizadas, não atrapalhasse o desenvolvimento criativo de uma nova solução. E esta decisão se mostrou acertada: após pesquisa em 20 principais artigos relacionados ao assunto, nenhuma solução se assemelha ao que foi desenvolvido neste projeto. Este aspecto não significa nada a priori, mas faz com que a comparação entre os algoritmos se torne muito mais interessante, e enriqueça a conclusão deste trabalho.

1.6 – Descrição

Os capítulos deste texto foram organizados de forma didática, com o objetivo de facilitar a compreensão do assunto pelo leitor. No próximo capítulo está um resumo sobre o funcionamento dos elevadores convencionais e inteligentes e seus algoritmos.

O capítulo 3 apresenta a motivação para o estudo de uma nova abordagem, diferente das que foram descritas no capítulo anterior.

No capítulo 4 está a descrição da solução proposta: a fundamentação da solução na teoria de algoritmos genéticos, fatores de sucesso para esta escolha e o desenho da solução.

O capítulo 5 descreve a ferramenta desenvolvida para a simulação do funcionmento dos elevadores e de seu algoritmo de controle .

Resultados experimentais obtidos na simulação de cenários de edifícios reais descritos na literatura são apresentados no capítulo 6, juntamente com quadros comparativos do desempenho de diferentes algoritmos.

Na conclusão são descritos os objetivos alcançados e sugestões de atividades para continuação deste trabalho de pesquisa.

Capítulo 2

Funcionamento dos Elevadores

2.1 – Panorama histórico

Os primeiros elevadores para o transporte de passageiros surgiram nos anos de 1890. Cada cabine era dirigida internamente por um atendente, responsável por decidir os andares de parada. Estas decisões foram se tornando mais difíceis com o aumento da altura dos edifícios e, na década de 1920, começaram a aparecer os primeiros controladores eletrônicos semi-automáticos, que reduziram o trabalho dos atendentes a apenas abrir e fechar a porta, e dar partida ao movimento da cabine. A partir de 1950, elevadores completamente automatizados eliminaram a necessidade de um atendente a bordo. [3]

No final dos anos 1980, com a utilização de microprocessadores para o controle de grupos de elevadores, tornou-se possível fazer cálculos estatísticos com o registro do tráfego de momentos anteriores, e conseguir prever os instantes de maior fluxo de entrada e saída de passageiros [4].

Para uma alocação ainda mais eficiente da cabine, mais importante do que prever os andares prováveis de ocorrência de chamados, é a informação de quantos passageiros estão aguardando para embarcar em cada um deles. O registro dos sensores de movimento na porta de cada andar, aliado às medidas de peso realizadas no interior de cada elevador antes e depois de abrir a porta, são utilizados para o cálculo de previsão estatística da quantidade de passageiros que devem embarcar nos próximos instantes. A partir de 1990, a utilização de avançada tecnologia em visão computadorizada foi proposta para detectar a quantidade de pessoas aguardando em um determinado andar. Apesar de demonstrar acurácia de 80-85%, esta tecnologia ainda é muito cara para ser utilizada regularmente [5].

Alternativamente, na década de 90, surgiram os primeiros terminais de controle de acesso, onde a informação de destino é registrada ainda no solo do andar, antes do embarque (Figura 2.1). Por se tratar de uma nova interface com o usuário, este tipo de terminal vem sendo preferencialmente utilizado em prédios que sejam freqüentados diariamente pelos mesmos funcionários, que se adaptam mais rapidamente com este novo recurso. A informação antecipada do destino permite o cálculo exato do tempo de atendimento de uma chamada, facilitando assim a escolha pelos menores tempos e a otimização do despache de elevadores.

Figura 2.1 – Ilustração de terminal externo com e sem informação de destino.
Fonte: Otis 2011 [9].

2.2 – Princípios de atendimento

Para a solução de atendimento de chamados, algumas premissas sobre o desempenho de sistema de elevadores são comumente assumidas:

- O elevador só abre a porta nos andares onde haja uma solicitação;

- As solicitações são atendidas de forma seqüencial obedecendo ao sentido do trajeto do elevador (subindo ou descendo);

- A quantidade de passageiros transportados é limitada pela capacidade da cabine;

- Quando o elevador possui passageiros embarcados, ele não pode mudar o sentido da viagem durante a execução do trajeto. [6]

Os algoritmos de atendimento mais conhecidos são baseados nos princípios coletivo e seletivo. No princípio coletivo, o elevador efetua suas paradas seqüencialmente, procurando a chamada do pavimento mais próximo que esteja no mesmo sentido do movimento, sem importar a ordem em que as chamadas foram feitas [7]. No modo seletivo, o elevador é direcionado para o andar mais alto ou mais baixo sem realizar qualquer parada intermediária. Este modo é utilizado para resolver problemas de pico de tráfego de entrada ou saída no edifício, agilizando o fluxo de pessoas. A questão lógica a ser resolvida é quando se deve optar pelo ativamento ou desligamento modo seletivo (Figura 2.2).

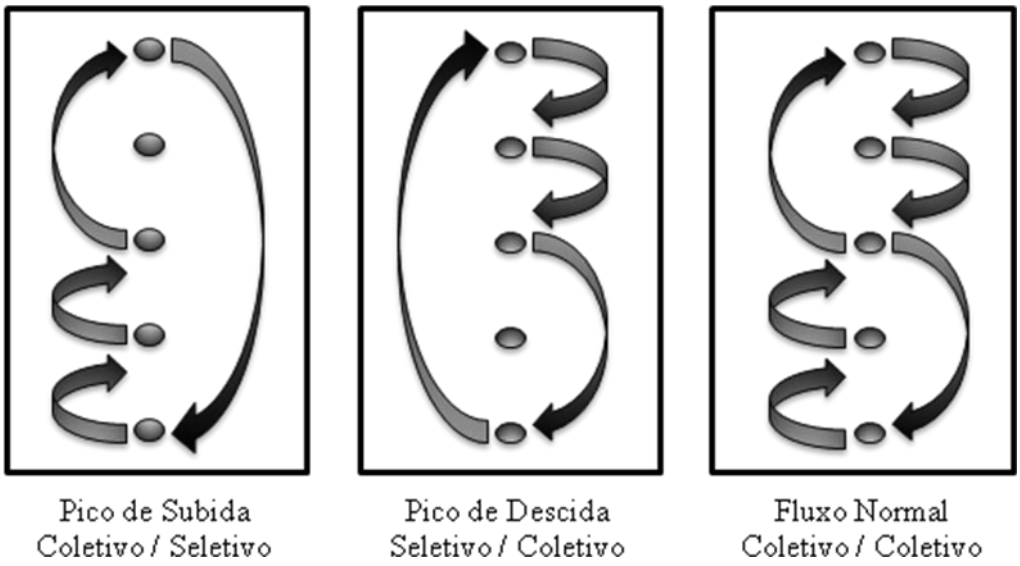

Figura 2.2 – Ilustração dos modos coletivo e seletivo de atendimento.

Entre os problemas enfrentados nos períodos de pico de tráfego, está a demora no atendimento dos andares mais altos de um edifício. Os passageiros que embarcam no térreo com destino ao último andar sofrem com as consecutivas paradas para embarque e desembarque nos andares intermediários. Uma solução freqüentemente utilizada é restringir os andares onde é permitida a parada de um elevador, e utilizar uma regra complementar a outro elevador do mesmo grupo como, por exemplo, andares ímpares e pares, ou inferiores e superiores.

A falta de inteligência no controle dos elevadores mais antigos pode ser percebida pelo fenômeno conhecido como *bunching*, onde vários carros se movimentam na mesma direção, disputando por um mesmo chamado. O resultado desta disputa geralmente são estes elevadores chegando ao mesmo tempo no andar, e apenas um embarcando os passageiros enquanto os demais elevadores da disputa apenas gastaram energia e atrasaram a viagem dos demais passageiros. [2]

2.3 – Controle de grupos de elevadores

A eficiência de um grupo de elevadores pode aumentar significantemente com um controle de grupo [4]. O objetivo deste controle é distribuir os elevadores entre as chamadas realizadas de forma a melhor atendê-las. O principal critério de otimização é a redução do tempo total de espera dos

passageiros, que inclui desde o tempo de espera pela cabine até o desembarque no destino escolhido. Outro critério bastante utilizado é a redução do consumo de energia. [6]

A otimização dos algoritmos de controle de um grupo de elevadores é um assunto muito estudado e possui diversas soluções em diferentes áreas: redes neurais, algoritmos genéticos, lógica fuzzy, entre outras. A aplicabilidade de cada um destes algoritmos depende: do tipo de terminal de acesso utilizado (botões de sobe/desce ou botões numéricos de andar de destino), da natureza comercial do prédio (se aloca uma única ou várias empresas), e do objetivo que se deseja alcançar com este controle (maior agilidade nas viagens, ou personalização e priorização de andares).

Existem dois métodos de alocação dos elevadores que são frequentemente utilizados no atendimento de chamadas: a alocação imediata e a alocação contínua. No método de alocação imediata, a cada solicitação recebida, um elevador é imediatamente associado e o passageiro é informado para qual elevador ele deve se encaminhar e aguardar para o embarque. A partir do momento em que o passageiro já recebeu esta informação, o elevador associado fica obrigado a cumprir este trajeto, independente de quais serão os próximos chamados.

No método de alocação contínua, o passageiro não sabe qual dos elevadores estará mais rapidamente disponível para atendê-lo, e deve aguardar até que se acenda um sinal luminoso no topo de alguma das cabines, lhe indicando onde deve embarcar. Neste método, cada elevador é associado a apenas um comando de cada vez, e sempre ao chamado com maior prioridade ou com menor custo de atendimento. Os chamados que ainda não estiverem associados a nenhum elevador são constantemente testados e sua prioridade é incrementada, até que ele seja associado a algum elevador. [2]

Enquanto o método de alocação imediata é muito utilizado pelos japoneses, os edifícios ocidentais utilizam tipicamente o método de alocação contínua.

2.4 – Algoritmos para chamadas de Sobe-Desce – Alocação Continua

Os algoritmos desenvolvidos para elevadores com botões de chamada do tipo sobe-desce possuem a dificuldade de não conhecerem a priori o destino dos passageiros que estão indo buscar, nem a quantidade de pessoas que irão embarcar em cada andar. Eles lidam com pelo menos três tipos de incerteza: tempo de chegada, andar de destino, e a quantidade de passageiros que possuem determinado destino. Estas variáveis são estocásticas, e acrescentam fator de dificuldade ao problema. [8]

Solução para isto foi implementada pela primeira vez em 1989 através dos controladores TMS9000 [4]. Através dos dados obtidos com a medição dos sensores e do cálculo de predição de

ocorrência de novos eventos, os algoritmos de controle passaram a ter informação suficiente para calcular e decidir quais as melhores rotas para os elevadores, no que diz respeito à redução do tempo de espera e economia de energia.

Para utilizar as informações obtidas com os sensores nos algoritmos de chamadas de sobe-e-desce, a Lógica Fuzzy tem sido amplamente utilizada em diferentes aplicações como, por exemplo, na estimativa do tempo de chegada até o destino, na definição de parâmetros de priorização e otimização das funções de fitness, ou no reconhecimento dos padrões de tráfego em curso.

A seguir, estão dois exemplos de soluções em algoritmos sobe-e-desce: o primeiro foi desenvolvido por Marja-Lissa em 1997, utilizando a tecnologia Lógica Fuzzy no reconhecimento de padrões de tráfego; o segundo foi desenvolvido por Henri Hakonen et al, em 2003 e atribui o despacho dos elevadores de acordo com a estimativa do tempo de chegada de cada um deles.

Solução 1: Elevator Group Control with Artificial Intelligence [4]

A lógica Fuzzy é muito utilizada no controle de grupo de elevadores pois evita que pequenas mudanças no tráfego de passageiros causem abruptas mudanças no padrão de tráfego(Zadeh, 1975) apud [4]. Nesta solução, o reconhecimento do padrão de tráfego foi feito utilizando como entrada a previsão de comandos calculada com as medições de todo o dia anterior e a intensidade relativa do fluxo de passageiros observada em uma janela com os últimos 5 minutos. A Figura 2.3 ilustra o processo de fuzzificação das componentes de tráfego e de intensidade de fluxo, e a Tabela 2.1 exibe um fragmento das regras utilizadas para o reconhecimento do padrão de tráfego atual.

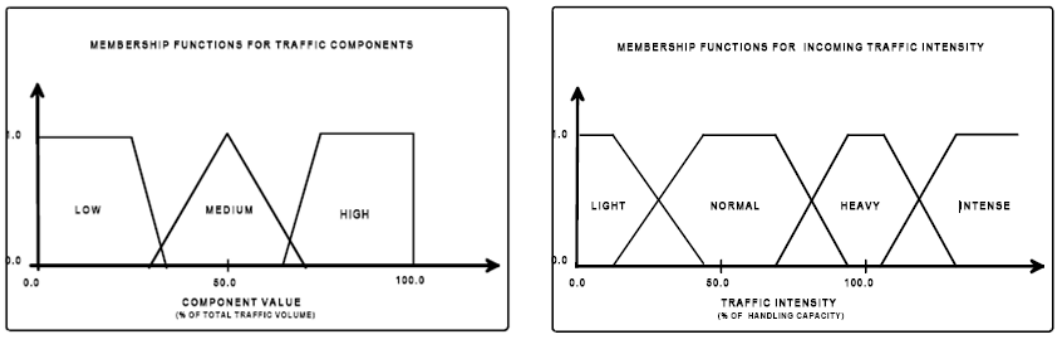

Figura 2.3 – Gráficos de fuzzificação das componentes de tráfego e fluxo.
Fonte: Siikonen, 1997 [4].

Intensity	Incoming	Outgoing	Interfloor	Traffic Pattern (Index)
intense	*high*	*low*	*low*	intense up-peak (1)
intense	*low*	*high*	*low*	intense down-peak (2)
intense	*low*	*low*	*high*	intense inter-floor (3)
intense	*medium*	*low*	*low*	intense incoming (4)
...	...	...	...	...
light	*low*	*high*	*low*	light outgoing (29)
light	*low*	*low*	*high*	light inter-floor (30)
light	*medium*	*low*	*low*	light incoming (31)
light	*low*	*medium*	*low*	light outgoing (32)
light	*low*	*low*	*medium*	light inter-floor (33)
light	*medium*	*medium*	*low*	light two-way (34)
light	*medium*	*low*	*medium*	light mixed (35)
light	*low*	*medium*	*medium*	light mixed (36)

Tabela 2.1 – Fragmento da tabela utilizada para reconhecimento de tráfego.
Fonte: Siikonen, 1997 [4].

Após a determinação do padrão de tráfego, a alocação dos elevadores é feita de acordo com a intensidade encontrada. Em períodos de fluxo normal, os chamados que possuem previsão de maior número de passageiro são priorizados em relação aos demais. Durante o período de tráfego pesado, carros extra são despachados para os andares mais ocupados. Já nos períodos de tráfego leve, alguns carros são estacionados nos andares com maior probabilidade de embarque de passageiros. [4]

Um edifício é dividido em setores; e a prioridade de cada setor é encontrada pela razão entre a quantidade de passageiros que embarcam nestes andares, pela quantidade de andares que compõe este setor. Os setores com maior prioridade são servidos primeiro.

Solução 2: Estimated Time of Arrival (ETA) Based Elevator Group Control Algorithm with More Accurate Estimation [2]

Seguindo o princípio de tempo de chegada, este algoritmo não estima apenas o tempo de atendimento, mas inclui também os atrasos que cada parada do elevador causa aos demais passageiros que ainda não foram atendidos. Este tempo total calculado para cada elevador é dado pela equação:

$$t_i^{total} = \sum_{j=1}^{n_i} t_{i,j}^{delay} + t_i^{attending} \quad (1),$$

onde a variável ‘i’ representa cada um dos elevadores do grupo e ‘j’ representa cada uma das chamadas associadas a este elevador.

O conceito de três passagens determina o tempo de atendimento de cada chamada: na primeira passagem (P1) são atendidos os chamados que possuem o mesmo sentido e que estão à frente do andar atual; na passagem (P2) estão os chamados que estão no sentido oposto ao sentido atual do elevador, e necessitam aguardar uma reversão de sentido; na terceira passagem (P3) estão os chamados que, apesar de estarem no mesmo sentido, estão em posição anterior à posição atual do elevador, e precisam aguardar duas reversões de sentido até serem atendidos. O tempo de atraso é incrementado a cada vez que um novo chamado é associado ao elevador.

A escolha entre qual a melhor opção de alocação do chamado é feita para o elevador com o menor tempo total de atendimento, obedecendo à seguinte equação:

$$\mathrm{e}_{\mathrm{candidate}} = \underset{i \in [1, M]}{\arg\min}\, t_i^{total} \quad (2).$$

Em uma segunda abordagem deste algoritmo, é proposta a contínua realocação dos chamados, com o objetivo de adaptar as alocações dos chamados, às mudanças que ocorreram no cenário. Desta forma, a vantagem do passageiro não saber a priori em qual cabine ele deverá embarcar é aproveitada, e o desempenho do algoritmo é maximizado.

2.5 – Algoritmos para chamadas com informação de destino – Alocação imediata

Para acabar com as incertezas nos sistemas de chamadas por botões de sobe-desce, surgiram os elevadores baseados na informação exata de destino do passageiro no momento em que o chamado é efetuado [8]. O primeiro sistema de controle de destino, o MICONIC 10, foi introduzido no mercado pela Schindler em 1996, e em 2002 já contava com mais de 1500 elevadores instalados em todo o mundo [5]. Além de permitir cálculos exatos e o agrupamento de passageiros que possuem o mesmo trajeto, esta participação com o usuário da informação antecipada do local de embarque, já produz um efeito psicológico de redução no tempo de espera para o passageiro[2].

A vantagem na organização do embarque proporcionada por este modelo, é enfatizada pela Empresa de Elevadores Otis, na propaganda dos seus equipamentos que funcionam neste modelo: "— Nos sistemas convencionais, os passageiros pressionam o
botão de chamada para subir ou descer e aguardam. Então, lotam a primeira cabine que aparece, empurram-se para selecionar seus destinos e vão parando em cada um dos pavimentos escolhidos até chegar ao seu. Com o CompassTM, os passageiros informam seus destinos antes de entrar na cabina,

(...) que imediatamente direciona cada passageiro ao elevador que mais rapidamente irá levá-lo ao andar desejado.” [9]

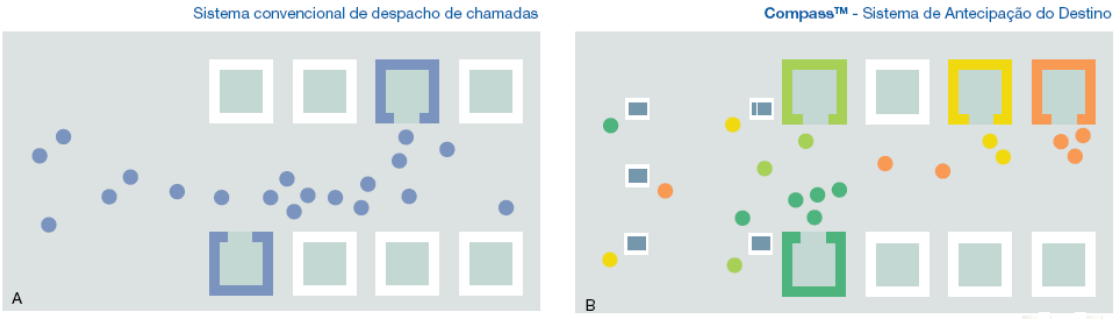

Figura 2.4 – Comparação entre o embarque de passageiros no sistema convencional de alocação contínua e o sistema de alocação imediata.
Fonte: Otis 2011 [9].

Porém, a alocação antecipada do elevador para o atendimento de uma chamada proíbe, em qualquer caso, o cancelamento ou a realocação deste chamado, pois o passageiro já estará aguardando a chegada da cabine no local informado. Esta impossibilidade, nos casos de mudança repentina no padrão de tráfego, pode causar a perda da garantia de otimização no atendimento [2] [8].

Para este tipo de elevador, existem menos algoritmos disponíveis na literatura acadêmica. E esse comportamento se justifica principalmente pela redução da complexidade que existia anteriormente com a dúvida sobre o destino de cada chamado. Com o fim da parte estocástica do problema, a solução para elevadores de alocação imediata se tornou essencialmente heurística, e resolvida através de alguns poucos cálculos de distância, para a escolha do melhor caso.

A solução proposta por Atsuya Fujino et al. em 1997, utiliza os algoritmos genéticos com o objetivo de otimização de parâmetros especiais configuráveis para cada andar. Sua implementação permite a prestação de serviços VIPs entre andares, com prioridades e tempos diferenciados. As decisões de despacho dos elevadores se torna muito mais complicada,e a adição destas restrições, de acordo com a heurística, impede que soluções ótimas sejam encontradas [5].

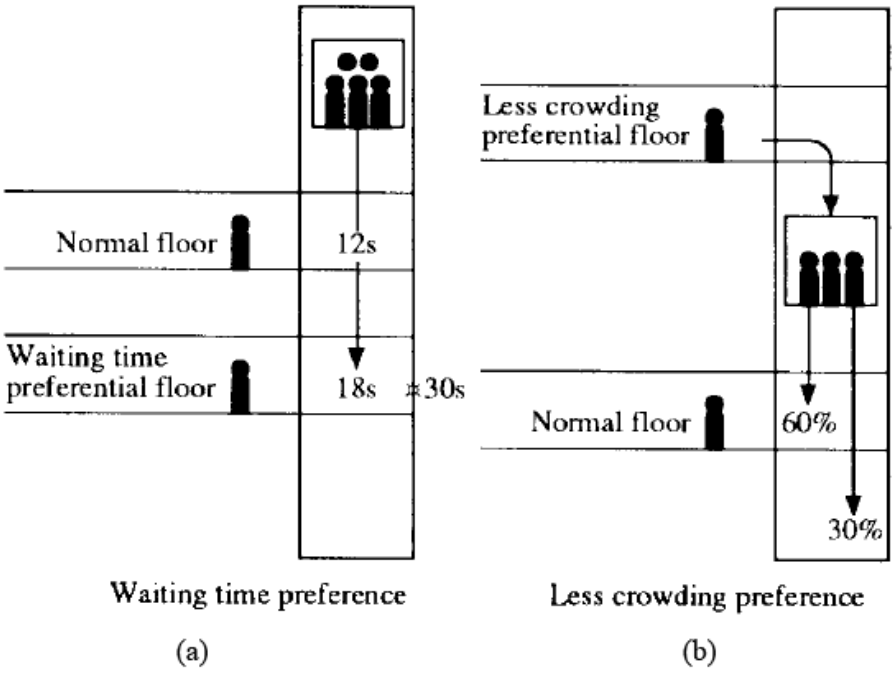

Figura 2.5 – Algoritmos de customização e priorização de atendimento.
Fonte: (Fujino, 1997) [10].

Capítulo 3

Motivação

3.1 – O elevador como um transporte coletivo

Em resumo, o funcionamento de um elevador pode ser descrito pelo repetido movimento de um carro que percorre uma rota de mesma direção, ora em um sentido, ora em seu sentido oposto, transportando passageiros que embarcam a qualquer momento de uma origem até um destino desejado.

Esta mesma descrição poderia ser utilizada para descrever a rotina de um trem, de um metrô, de um ônibus, ou até mesmo de um avião e suas escalas. A natureza do transporte público é semelhante em todas as modalidades, concentrando as principais diferenças nas distancias percorridas, na periodicidade do movimento e na capacidade de cada veículo.

Neste projeto, a busca por otimizar as rotas dos elevadores dentro do seu grupo, tem uma motivação especial: a aplicação deste resultado para o problema dos ônibus do transporte público. Apesar da grande diferença entre a dimensão dos dois problemas, o modelo de controle dos dois sistemas é logicamente muito parecido, e também o problema enfrentado por ambos provém da mesma natureza: o congestionamento de pessoas se movimentando no mesmo sentido em horários de entrada e saída do trabalho.

3.2 – Problemas e soluções em transporte público

– Qual o motivo que faz uma pessoa preferir ir dirigindo o seu carro para o trabalho, todos os dias, ao invés de pegar um ônibus que passa ao lado da sua casa e que percorre exatamente o mesmo trajeto até o seu local de trabalho?

A resposta mais comum é: conforto, pois ao pegar um ônibus não há a garantia de que ainda existe vaga para ir sentado, e o tempo de percurso – apesar de percorrerem o mesmo trajeto, as consecutivas paradas para embarque e desembarque de passageiros são extremamente desagradáveis para os que precisam percorrer longas distâncias, atrasando muito as viagens.

Da mesma forma como no caso dos elevadores convencionais, o funcionamento dos ônibus urbanos também é baseado nos métodos de atendimento coletivo e seletivo (Figura 3.1). Diariamente o método coletivo é o predominante. Os ônibus param em todos os pontos onde tenha sido feita uma solicitação, seja do interior ou no exterior do carro, de forma seqüencial. Apesar de possuírem maior mobilidade que uma cabine de um elevador, os ônibus possuem uma rota desenhada no mapa da cidade, do qual não podem se desviar. Não podem passar por um ponto onde haja um chamado e não parar para o embarque ou desembarque, não podem reverter o sentido a menos que tenham chegado à estação terminal, etc.

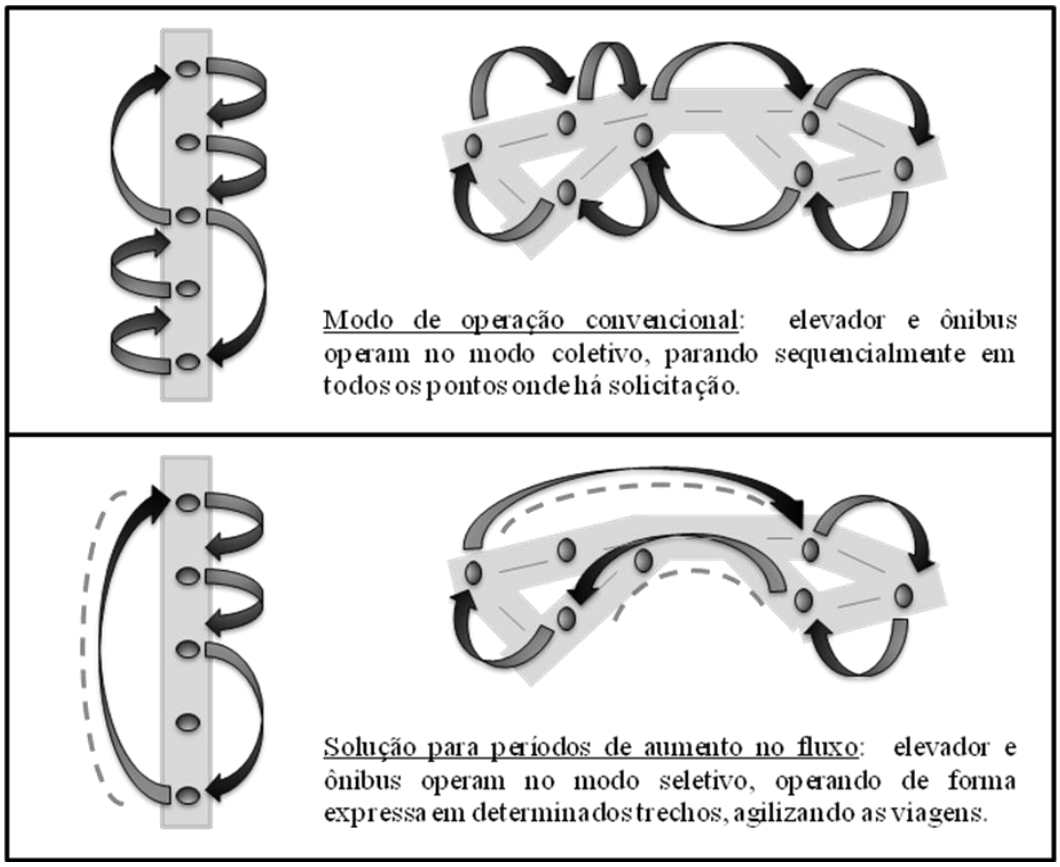

Figura 3.1 – Semelhanças no atendimento de elevadores e ônibus.

Nos momentos de tráfego leve, como domingos e feriados, alguns carro se mantém estacionados na garagem para evitar gasto desnecessário de energia e mão de obra. Já nos momentos de tráfego intenso, todos os carros são colocados em circulação. As linhas de ônibus que percorrem longos caminhos (da Baixada Fluminense até o Centro da Cidade do Rio de Janeiro, por exemplo) costumam reservar alguns de seus carros para o atendimento no modo seletivo, definindo alguns intervalos onde não é permitido nem embarque nem desembarque de passageiros. Estas linhas são chamadas de “linha

expressa" e percorrem longos intervalos sem paradas como ao longo da Av. Brasil, por exemplo, a fim de agilizar o trajeto dos passageiros já embarcados.

3.3 – Possíveis melhorias para o futuro

Com o passar dos anos, desde a invenção do elevador, a quantidade de prédios cresceu muito, os edifícios cada vez maiores, mais populosos, e a necessidade por um melhor controle no despache dos elevadores foi necessária. Vimos, no capítulo anterior, que novas tecnologias foram então absorvidas, e que novos algoritmos de otimização foram desenvolvidos para substituir o controle convencional.

Este crescimento populacional nas grandes cidades, não ocorreu somente dentro dos edifícios. Após descerem dos elevadores, estas mesmas pessoas também precisam se locomover até as suas casas, gerando o mesmo tráfego descrito no problema dos elevadores. Mas, no panorama de soluções em ônibus, não existe ainda nenhum algoritmo otimizado que se preocupe em reduzir significativamente o tempo de espera, o tempo de travessia, ou que se preocupe em manter a lotação dos ônibus abaixo da quantidade máxima permitida.

E qual seria uma solução de otimização para o transporte urbano? Com tantas semelhanças entre ônibus e elevadores, é razoável pensar que podemos buscar nos elevadores esta resposta. As perguntas abaixo fazem parte de um pequeno *brainstorming* que motivou a criação deste projeto, e que serão retomadas no final deste trabalho para a avaliação dos resultados obtidos:

- "É verdade que os elevadores não vão até o último andar se não houver chamados registrados por lá. Seria possível então avisar ao motorista onde estão localizados os seus passageiros para evitar percursos desnecessários?"

- "Se os passageiros pudessem informar com antecedência quais os seus destinos e origens (já que são geralmente idênticos no cotidiano de um trabalhador), esta informação permitiria uma melhor alocação dos ônibus e de suas rotas?"

- "E se as rotas de todos os ônibus fossem dinâmicas, e variassem de acordo com os chamados realizados. Seria possível agrupar passageiros com trajetos semelhantes para reduzir o número de paradas?"

- “E sendo as rotas dinâmicas, seria possível consultar em tempo real as informações de trânsito para excluir destas rotas as ruas mais congestionadas?”

- “Seria possível garantir aos passageiros que, ao embarcar no ônibus indicado, existiria um acento disponível para ele?”

- “Existe tecnologia disponível que viabilize todas essas mudanças?”

Capítulo 4

Solução Proposta

4.1 – Visão geral sobre algoritmos genéticos

Os Algoritmos Genéticos (GA) são algoritmos de busca baseado em mecanismos de seleção natural e genética natural. O primeiro algoritmo desta natureza foi inventado por John Holland em 1975, através do seu livro "Adaptation in natural and artificial systems". Holland propôs o GA como um método heurístico baseado na sobrevivência do individuo mais adaptado. [11]

Os algoritmos genéticos possuem algumas características fundamentais para se enquadrar como forte candidato na resolução do problema dos elevadores. Entre estas estão:

- Paralelismo: o processo evolutivo é altamente paralelizável. O desempenho de cada indivíduo (que costuma ser a etapa mais demorada do processo) pode ser calculado separado dos demais, sendo necessária a serialização somente no momento de seleção das próximas gerações. Esta capacidade de paralelização dos processos aumenta muito a velocidade de processamento do algoritmo, convergindo rapidamente ao resultado.

- Robustez a mudanças dinâmicas: por ser um método evolutivo, ele é capaz de se adaptar a mudanças nos parâmetros de seleção, sem que seja necessário reinicializar todo o sistema.

- Resolve problemas sem solução conhecida: partindo de um conjunto aleatório de possibilidades, os algoritmos genéticos chegam a resultados que nem mesmo o homem consegue raciocinar, dependendo da complexidade do problema. Por trabalhar com as populações mais diversas, abrange muitas combinações e possui alto desempenho em problemas estocásticos.

- Não garante a solução ideal: o que poderia ser ruim em uma primeira analise, é mais uma grande vantagem dos GAs. O algoritmo se satisfaz com soluções que apresentem boa performance, ou seja, mesmo quando não tem tempo hábil para buscar a melhor solução, ele responde com boas soluções logo em suas primeiras seleções, mantendo alta velocidade e acurácia nas suas respostas.

Como o funcionamento do algoritmo foi baseado nas teorias de Charles Darwin de evolução das espécies e na ciência genética, possui, portanto, uma correspondência direta entre os nomes biológicos e as estruturas computacionais (Tabela 4.1).

Evolução Natural	Algoritmo Genético
Cromossomo	String
Gen	Caractere
Alelo	Valor do recurso
Locus	Posição de string
Genótipo	Estrutura ou string codificada
Fenótipo	Conjunto de parâmetros, estrutura decodificada

Tabela 4.1 – Correspondência entre os termos biológicos e computacionais

Fonte: S.N. Sivanandam and S.N. Deepa [11].

O algoritmo genético é geralmente iniciado com uma população aleatória – pois, quanto maior a variedade de material genético disponível, uma maior gama de resultados ele será capaz de produzir e avaliar. Em seguida, o algoritmo executa de forma iterativa os seguintes processos:

- Seleção: Os indivíduos são selecionados randomicamente, com probabilidade relativa ao *fitness* associado. Os melhores indivíduos são escolhidos para a reprodução em detrimento dos demais.

- Reprodução: composta pelo processo de *crossover* (segmentação e recombinação dos gens a cada par) e mutação (modificação aleatória de um gen);

- Qualificação: o desempenho do indivíduo é avaliado e seu fitness é calculado;

- Substituição: de acordo com o fitness de cada indivíduo, uma nova população é selecionada, e a população anterior é substituída.

O processamento do algoritmo pode ser então encerrado de duas maneiras: ou ao final da execução de um número máximo de iterações definido previamente, ou até que a variação entre os resultados obtidos nas últimas iterações seja bem pequena, demonstrando já ter convergido para um resultado otimizado.

4.2 – Solução em algoritmos genéticos aplicada ao problema

O algoritmo desenvolvido neste projeto tem, como principal objetivo, a redução do tempo total de espera e viagem dos passageiros, através da otimização das rotas de cada um dos elevadores de um sistema. Ele realiza a alocação imediata de cada chamado, associando ao elevador que irá cumprir o trajeto de origem-destino da maneira mais eficiente para todos os passageiros.

Como a principal e única vantagem que os algoritmos de alocação contínua (botões sobe-desce) possuem em relação aos de alocação imediata (painel com teclado numérico) é a capacidade de realocação dos chamados entre os elevadores, permitindo que a solução se adapte constantemente às mudanças ocorridas no cenário, a solução aqui proposta permite o re-ordenamento dos próximos destinos de um elevador, a fim de compensar parcialmente a rigidez inerente a alocação imediata.

Isto significa que o elevador passa a estar livre para atender todos os chamados na ordem que melhor lhe convier, ao invés de manter sempre uma rota seqüencial de destinos ora crescentes, ora decrescentes.

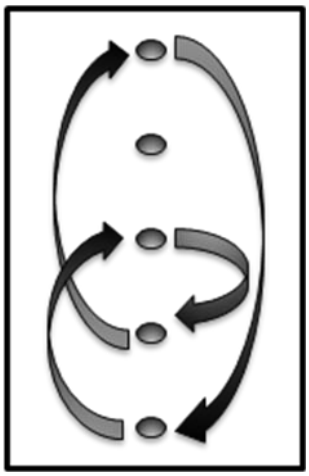

Figura 4.1 – Ilustração de uma rota permita na solução proposta.

Estas rotas são possíveis devido à natureza aleatória dos algoritmos genéticos. Outro fator que incentiva estas construções é a otimização do tempo de trajeto, introduzido na função *fitness* do algoritmo. Selecionando sempre as rotas que contenham o menor tempo médio (ou máximo) no atendimento de um chamado, a solução proposta possui maior justiça na distribuição dos chamados, fazendo com que todos demorem tempos parecidos de deslocamento no seu trajeto origem-destino, de maneira mais eficiente para todos os passageiros.

A solução foi desenhada de seguinte forma:

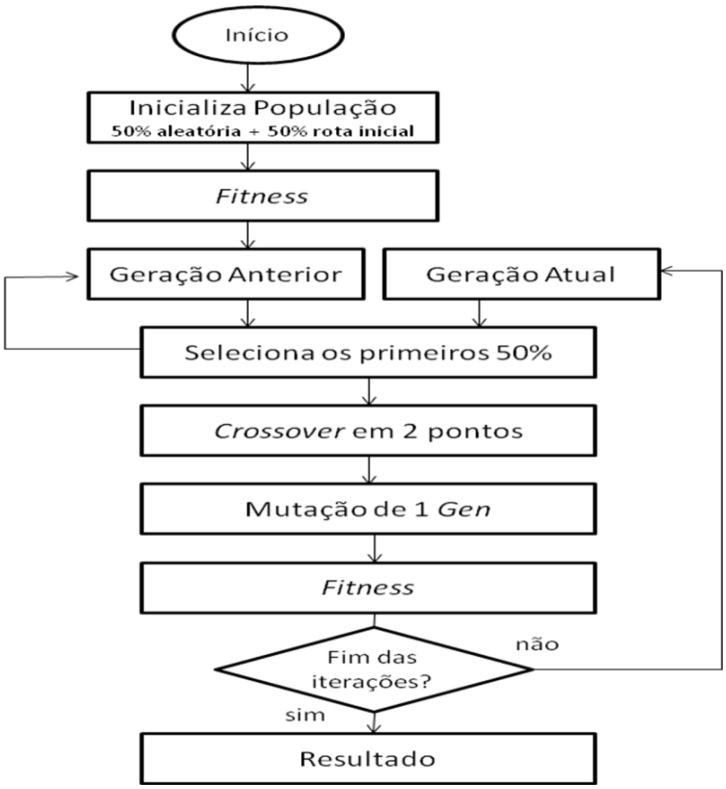

Figura 4.2 – Algoritmo genético aplicado à solução.

População: A população é formada por 50% de cromossomos aleatórios e 50% de cromossomos idênticos ao cromossomo da rota anterior. Cada cromossomo representa uma rota, e cada *gen* representa um andar do edifício onde existe solicitação de parada. O primeiro *gen* de cada cromossomo representa o próximo destino deste elevador, e se mantém fixo até que a cabine estacione neste andar. Quando isto acontece, a rota é circularmente deslocada em uma posição para esquerda; o segundo *gen* se torna então o próximo destino deste elevador; e todos os demais *gens* são livres para assumir o valor de qualquer andar.

Crossover: Os cromossomos são combinados em pares; os pais são escolhidos aleatoriamente em toda a população selecionada; e os filhos são formados por 3 segmentos, onde o primeiro e o terceiro segmento são os *gens* de um dos pais, e o segundo segmento são os *gens* do outro. Isto é, dois pontos são escolhidos aleatoriamente para seccionar os cromossomos dos pais, e os filhos são formandos pela recombinação dos segmentos encontrados.

Mutação: Todos os cromossomos sofrem a mutação de algum gen; isto é, em todo o cromossomo é sorteado uma posição aleatória para a modificação deste gen pelo valor de um andar que possua uma solicitação ativa. Esta aleatoriedade permite que ordenamentos que não existiam sejam testados e, por ser restrita aos andares válidos, diminui o universo de soluções agilizando a busca. Como a quantidade

de gens utilizados na rota varia de acordo com a quantidade de chamados, a probabilidade de mutação é maior nos casos de maior fluxo e menor nos casos de poucas solicitações ativas.

Fitness: A função *fitness* é responsável por atribuir um valor de qualidade e confiabilidade ao cromossomo, de acordo com a função objetiva de otimização Y:

$$Y = \alpha \, . \, Tmáx + (1 - \alpha) \, . \, Tméd \quad (3),$$

onde Tmáx e Tméd representam, respectivamente, o valor máximo e o valor médio dos tempos necessários para o atendimentos de cada um dos chamados ativos de acordo com a rota definida pelo cromossomo em análise.

Para o cálculo do tempo de atendimento, são levados em consideração os atrasos provocados pelas paradas estabelecidas nos gens anteriores ao gen de destino. Exemplo: para o cromossomo da Figura 4.5, vamos calcular o seu desempenho no atendimento das seguintes chamadas:

Ch1 – origem: 5º andar; destino: 1º andar
Ch2 – origem: 2º andar; destino: 4º andar
Ch3 – origem: 9º andar; destino: 2º andar.

O tempo total de atendimento de cada um dos chamados é dado pelo tempo de espera, desde a solicitação até o desembarque da cabine. O tempo de espera por solicitação efetuada e o tempo gasto na travessia de um andar para o outro, incluindo os tempos de abertura/fechamento de porta e o tempo de embarque/desembarque de apenas um passageiro, são pré-armazenados em duas matrizes distintas (T esp e T trav) de duas dimensões correspondentes a quantidade de andar do edifício, e são apenas consultados neste momento. Note que para o cálculo da matriz de travessia foi estimado o embarque e desembarque de apenas um passageiro, para diminuir o tempo de processamento do algoritmo. Como as rotas são variáveis, a quantidade de passageiros em cada instante da rota é diferente para cada um dos cromossomos, e este cálculo exato aumentaria muito o tempo de simulação.

Desta forma, o tempo total de atendimento de cada chamado Ch é dado pelas Equações (4).

$$\begin{aligned} T(Ch1) &= T\,esp(5,1) + T\,trav(9,5) + T\,trav(5,2) + T\,trav(2,4) + T\,trav(4,1) \\ T(Ch2) &= T\,esp(2,4) + T\,trav(9,5) + T\,trav(5,2) + T\,trav(2,4) \\ T(Ch3) &= T\,esp(9,2) + T\,trav(9,5) + T\,trav(5,2) \end{aligned} \quad (4).$$

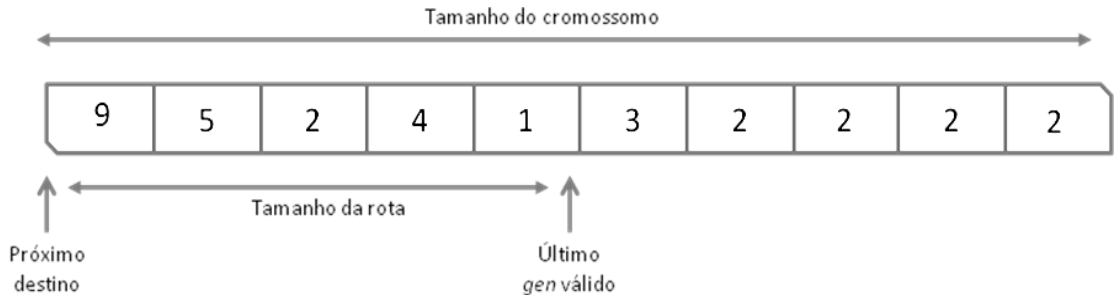

Figura 4.3 – Exemplo de um cromossomo.

No caso de algum chamado não ser atendido pelo cromossomo, esta rota não pode ser escolhida como vencedora, ou este chamado estará fadado a não ser atendido pelo elevador ao qual for associado. Para que isto não aconteça, nestes casos de não-atendimento, o tempo de atendimento deste chamado recebe um valor constante alto suficiente para que o fator Y de qualidade deste cromossomo seja severamente penalizado.

O código escrito para a solução deste algoritmo permite a configuração de quatro parâmetros decisivos na obtenção do resultado, relacionados na Tabela 4.1. O fator α define o objetivo da função de otimização, decidindo a importância entre o valor máximo e o valor médio no tempo de espera de cada passageiro. Isto é, quando α =1 (Equação 4.3), os maiores tempos de espera são integralmente utilizados na comparação de desempenho; já para α = 0, espera-se encontrar a menor média de tempo de espera.

A quantidade de gens de um cromossomo está diretamente relacionada a capacidade de agendamento de uma rota. Em momentos de pico de tráfego, se ocorrerem muitas solicitações distintas, as rotas deverão suportar o armazenamento de uma grande sequência de destinos. Se o cromossomo for muito pequeno, algumas solicitações poderão ficar sem solução. Por outro lado, cromossomos muito grandes diminuem a velocidade do processamento dos dados, pois será necessário um maior número de iterações para que o cromossomo convirja para a rota otimizada.

Os parâmetros configuráveis do algoritmo são listados na Tabela 4.1 a seguir.

Nomes	**Valores**
Fator α de otimização	Valor entre 0 e 1
Quantidade de gens	2 * quantidade de andares
Quantidade de cromossomos	Valor > ou igual 2
Quantidade de iterações	Valor > ou = 1

Tabela 4.1 – Parâmetros configuráveis no código do algoritmo.

A quantidade de cromossomos e de iterações do algoritmo devem ser as maiores possíveis. Um alto número de cromossomos proporciona uma maior diversidade na população inicial. Este deve ser no mínimo igual a 2 para que possa conter cromossomos da rota atual e cromossomos aleatórios que permitam novas combinações. Mas estas quantidades não podem ser muito abusivas para não exceder o tempo limite de resposta que é esperado pelo usuário.

Na simulação, o algoritmo genético é utilizado em 2 modalidades diferentes: para localizar rotas ótimas no instante de alocação de um chamado, ou na obtenção de rotas ainda melhores quando na ausência de novos chamados. No primeiro modo, cada elevador disputa com uma rota otimizada, e o passageiro se associa ao elevador que executa o trajeto em menor tempo. No segundo modo, não existem novos passageiros para serem alocados, logo se uma rota apresentar melhores resultados que a anterior, ela será substituída. Esta possibilidade de realocação da rotas possibilita que o sistema gere respostas quase ótimas em um intervalo de tempo muito curto, e que depois incremente ainda mais a sua eficiência utilizando o tempo ocioso que houver nos próximos instantes.

4.3 – Solução em algoritmos sobe-e-desce

Como uma forma de comparar o desempenho do algoritmo genético com uma lógica convencional heurística, utilizando o mesmo sistema controlador, também foi implementado neste projeto o algoritmo sobe-e-desce. A sua lógica consiste no ordenamento dos chamados através das três possibilidades de trajeto de um ponto a outro mantendo a sequência e o sentido de trajeto do elevador. Este conjunto de possibilidades foi mencionado por Jamaludin et al. em 2009 [8], e pode ser visualizado pela Figura 4.4. O cálculo de pontuação e escolha do elevador mais capaz no algoritmo sobe-e-desce é feito como no algoritmo genético, utilizando as Equações (1) e (2).

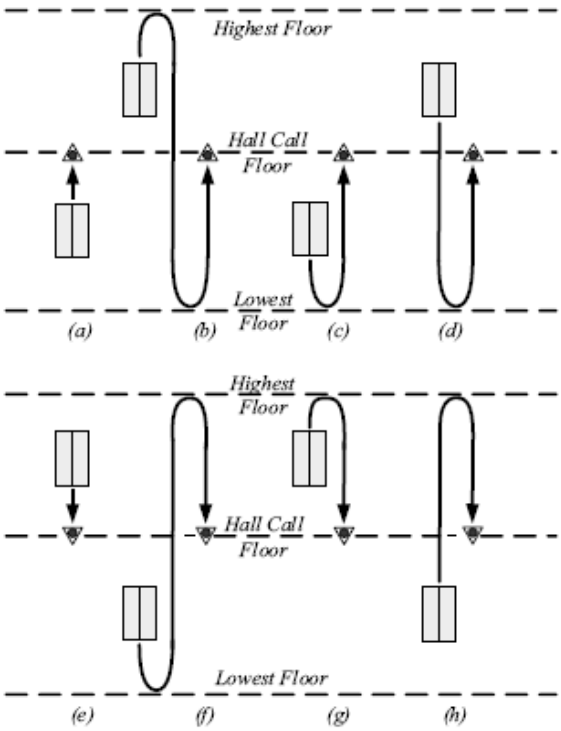

Figura 4.4 – Diferentes rotas adotadas por um elevador para atender uma chamada de andar.
Fonte: [8].

Apesar de diferentes abordagens, através da figura é mais fácil observar a seguinte sequência de ordenamento adotada. Se o elevador está subindo, a rota do elevador é composta por 3 segmentos:

1º - andar de destino de passageiros já embarcados; e andar de origem e andar de destino de passageiros que desejam subir e estão localizados em andares superiores Figura 4.4 (a);

2º - andar de origem e andar de destino de passageiros que desejam descer Figura 4.4 (g) e (h);

3º - andar de origem e andar de destino de passageiros que desejam subir, mas que possuem origem em andares inferiores ao atual Figura 4.4 (b);

Este conceito de três passagens também é utilizado no algoritmo ETA [2] que foi utilizado como parâmetro de comparação para os resultados deste projeto. Em ambos os algoritmos a solução para cada elevador é única, pois, de acordo com as premissas, ele deve sempre obedecer a sequência crescente ou decrescente dos números. As possibilidades variam durante a escolha de quais elevadores atenderão quais chamados. Feito isso, não existe outra possibilidade de rota que cumpra os mesmos casos.

Capítulo 5

Simulação dos Algoritmos

5.1 – A ferramenta desenvolvida

Para conseguir testar e simular o funcionamento do algoritmo proposto, o código foi escrito utilizando o Matlab 7.1. Nele foram escritos o código fonte dos algoritmos, a estrutura controladora do simulador de um grupo de elevadores, e a geração de cenários de teste.

O simulador foi desenvolvido através da interface gráfica GUIDE, disponível no Matlab. A janela principal foi dividida em setores: escolha do cenário, seleção de parâmetros configuráveis da ferramenta, escolha do algoritmo a ser testado, configuração dos parâmetros exclusivos do algoritmo genético e *link* direto para os resultados (Figura 5.1).

Figura 5.1 – Janela principal do simulador de algoritmos.

5.2 – Funcionalidades

A interface deste programa de simulação foi criada para auxiliar na execução de testes do algoritmo em diferentes cenários com diferentes combinações de parâmetros por repetidas vezes. Abaixo estão citadas as suas principais características:

- Permite o acesso direto aos arquivos de cenários disponíveis. Estes arquivos são do tipo texto (*.txt) e contém 3 colunas: andar de origem, andar de destino, e instante de tempo em que esta chamado foi feito. Esta escolha foi feita para que os cenários pudessem ter o seu funcionamento de tempo real, simulados por meio *off-line*, sem qualquer prejuízo.

- Após a seleção do arquivo desejado, é possível visualizar antecipadamente algumas de suas características: quantidade de andares, quantidade de chamados, e o tempo de gravação. Esta funcionalidade foi colocada para facilitar a identificação do cenário em análise.

- Permite a escolha alternativa entre os dois algoritmos a serem simulados: ou o algoritmo sobe-e-desce ou o algoritmo genético. Ambos utilizam a mesma estrutura de implementação do simulador, dada a independência lógica entre os algoritmos de otimização e a estrutura.

- Registra em um arquivo do tipo texto (*.txt) a resposta obtida após o processamento. Além das três colunas iniciais do cenário, ele acrescenta as informações: instante de embarque, instante de desembarque e número de identificação do elevador associado. A partir destes dados, é possível então calcular e reconstruir todo o trajeto realizado pelos elevadores. Uma lista contendo atalhos para estes resultados também foi colocado na interface, para facilitar o manejo da informação.

5.3 – Premissas

Algumas premissas foram adotadas para a implementação de um simulador simplificado. São estas:

- O elevador não freia nem acelera repentinamente para atender algum chamado recente que apareça durante o seu trajeto – o primeiro próximo destino é mantido fixo mesmo após as iterações do algoritmo. Esta consideração é bem razoável, pois uma mudança brusca só poderia beneficiar um

chamado recente, prejudicando um chamado mais antigo. Isto não seria justo seguindo a lógica do algoritmo.

- Cada chamado que está sendo lido do cenário corresponde a um único passageiro. Em ambientes reais, a quantidade de passageiros que realizaram o mesmo chamado dentro de um mesmo instante de tempo é obtida por métodos probabilísticos, que dependem de cenários anteriores. Como esta previsão trás incerteza para conjunto e não interfere no desempenho dos algoritmos de roteamento, a premissa de um passageiro para cada chamado foi adotada.

5.4 – Controle de tempo

A simulação do algoritmo é feita *offline*, isto é, os instantes de tempo registrados não correspondem ao tempo real em que os eventos ocorreram na simulação, e sim aos tempos correspondentes ao funcionamento real do sistema. A vantagem desta escolha é retirar do resultado de desempenho do algoritmo a dependência física de processamento do equipamento. As simulações são feitas considerando que o hardware utilizado nos controladores possui velocidade suficiente de processamento e resposta, o que não é nenhum exagero de aproximação.

A definição e compreensão do tempo gasto em cada etapa do deslocamento do elevador são fundamentais para uma correta correspondência dos tempos encontrados com os tempos percebidos nos edifícios reais. Analisando então a rotina de um passageiro, podemos definir as seguintes transições e intervalo de tempo associado como na Tabela 5.1.

	Evento	**Instante**	**Sigla**
1	Passageiro aperta botão	Instante inicial	T chamado
2	Passageiro recebe identificador do elevador	Tempo de resposta do sistema	T espera
3	Elevador pára no andar de origem	Tempo de espera no térreo	
4	Tempo de espera no térreo	Tempo de pré-abertura da porta	
5	Porta termina de abrir	Tempo de abertura de porta	
6	Passageiro entra	Tempo de embarque e desembarque	T embarque

7	Porta começa a fechar	Atraso de atuação do sensor fotoelétrico	T travessia
8	Porta termina de fechar	Tempo de fechamento de porta	
9	Elevador começa a se movimentar	Atraso de partida do elevador	
10	Elevador pára no andar destino	Tempo de viagem	
11	Porta começa a abrir	Tempo de pré-abertura da porta	
12	Porta termina de abrir	Tempo de abertura de porta	
13	Último passageiro sai	Tempo de embarque e desembarque	T desembarque

Tabela 5.1 – Eventos e instantes de um passageiro.

Na observação de um elevador, os instantes que fazem parte de sua rotina estão relacionados acima do item 4 ao item 10. Estes instantes se repetem ciclicamente e devem ser controlados pelo simulador. O instante 1 é um dado de entrada do sistema, sob o qual este não tem controle e não pode alterá-lo. Os instantes 2, 4 e 9 foram ignorados nesta simulação por serem muito rápidos e indesejáveis nos equipamentos, tendendo a se anularem com o avanço da tecnologia. Os instantes de 11, 12 a 13 são idênticos aos instantes 4, 5 e 6, demonstrando a periodicidade do movimento. Na figura 5.2 está o esquema de tempos da rotina de um elevador.

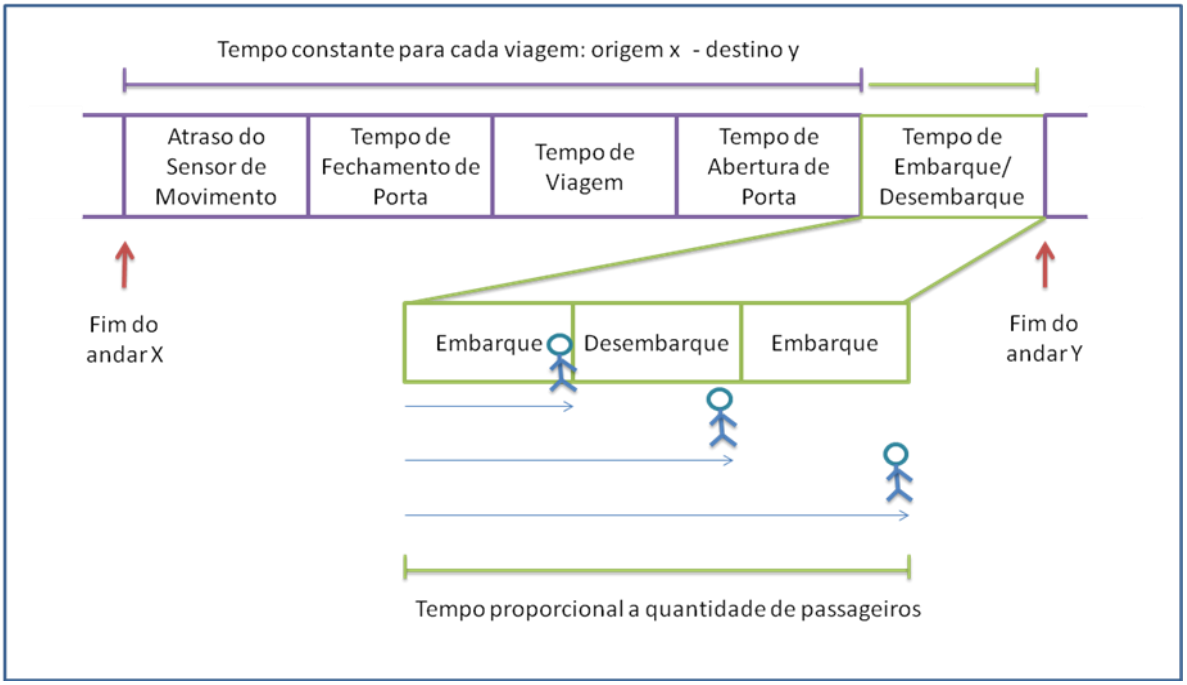

Figura 5.2 – Esquema de tempos da rotina de um elevador.

O ciclo é iniciado quando o sensor fotoelétrico percebe que não há mais trânsito de passageiros pelas portas da cabine. Isso significa que o elevador já cumpriu sua missão neste andar e já pode iniciar o movimento para o próximo. O tempo que decorre desde o completo embarque e desembarque de passageiros até o momento em que estará com as portas completamente abertas em seu próximo destino é constante para cada trecho de viagem. O tempo de fechamento e abertura de porta são intrínsecos ao equipamento mecânico, e por isso tendem a ser constantes. O tempo de deslocamento depende da aceleração dos motores do elevador, da velocidade máxima que ele consegue atingir, e também da distância entre os andares. Com posse desses dados e utilizando as equações físicas de cinemática, é possível então construir a matriz de tempos.

Nesta matriz de tempos (Figura 5.3), cada célula contém o tempo exato de deslocamento de uma andar de origem (linha) a um andar de destino (coluna). Esta matriz, somada aos valores constantes de porta, resulta na matriz de tempo de viagem, que contém todo o tempo de viagem entre andares. Esta nova matriz é utilizada no programa para obter a posição dos elevadores no tempo, e também para calcular a pontuação dos cromossomos nas funções de *fitness*. Ela agiliza o processamento do algoritmo, pois é necessário apenas uma consulta de valor na matriz para obter o valor desejado.

Origem \ Destino	1	2	3	4	5	6
1	0	5,85	9,65	13,45	17,25	21,05
2	5,85	0	5,05	8,85	12,65	16,45
3	9,65	5,05	0	5,05	8,85	12,65
4	13,45	8,85	5,05	0	5,05	8,85
5	17,25	12,65	8,85	5,05	0	5,05
6	21,05	16,45	12,65	8,85	5,05	0

Figura 5.3 – Matriz de tempos de deslocamento entre andares.

A parte não constante da jornada se refere ao período de embarque e desembarque de passageiros, que é proporcional à quantidade de chamados associados ao andar. Cada passageiro demora um tempo 'Ted' constante para embarcar ou desembarcar que é multiplicado pela quantidade de passageiros que estejam entrando ou saindo da cabine do elevador. Estes tempos são fundamentais para o cálculo de tempo médio de espera de cada passageiro utilizado como indicador de desempenho em [2].

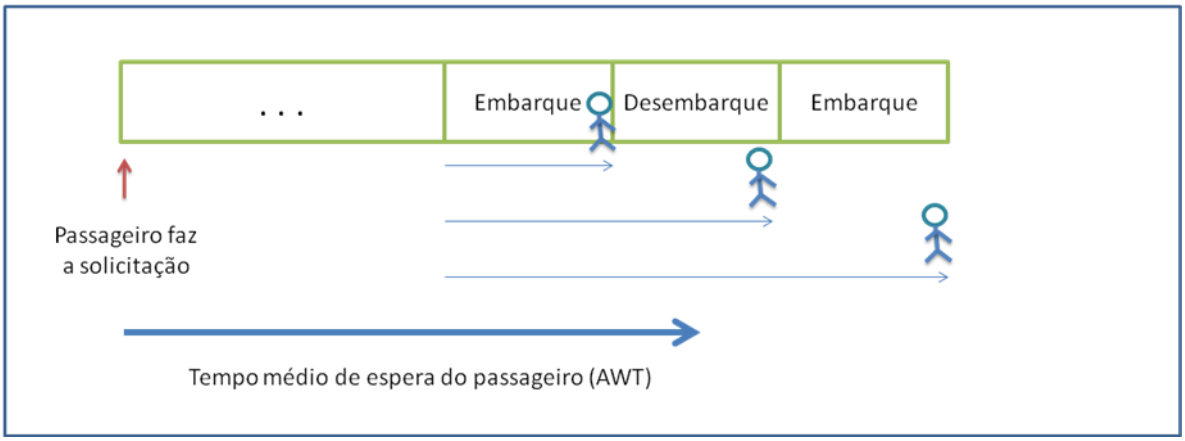

Figura 5.4 – Cálculo de indicador de tempo médio de espera de um passageiro.

Este mesmo cálculo do tempo médio de embarque e desembarque é também utilizado para o cálculo do tempo médio de jornada (AJT). Sendo assim, estes indicadores podem ser calculados pelas seguintes equações:

$$AWT(p) = T_{embarque}(p) - T_{chamado}(p) - ((Ned(a)-1)/2) * Ted \quad (5),$$

$$AJT(p) = T_{desembarque}(p) - T_{chamado}(p) - ((Ned(a)-1)/2) * Ted \quad (6),$$

onde ‘Ned(a)’ é o número de passageiros associados ao andar x, ‘Tchamado’ é o instante em que o passageiro faz a solicitação, e ‘T embarque’ e ‘T desembarque’ são, respectivamente, os instantes após o completo embarque e/ou desembarque de passageiros no andar de origem e no andar de destino, como definidos na Tabela 5.1..

5.4 – Ciclos de execução

O programa é executado a cada novo chamado, para garantir que o passageiro receba a resposta do sistema imediatamente após a sua solicitação. No entanto, podem demorar grandes intervalos de tempo até que um novo chamado seja realizado. Durante estes intervalos, o programa executa o código genético com a finalidade de procurar rotas melhores ainda que as obtidas anteriormente. Caso encontre, a nova rota é substituída pela anterior e os registros de tempo e passageiro são atualizados. Ao final da leitura do cenário, antes de encerrar a simulação, o programa continua executando o algoritmo genético para atualizar as rotas remanescentes, até que o último passageiro desembarque. Na Figura 5.5 este comportamento está descrito em um pseudo-código simplificado.

A função principal é proveniente da interface GUIDE do Matlab é responsável pelos comandos de tela, entrada de dados, e pelas demais funções implementadas. Neste caso, ela controla duas funções concorrentes: a que controla o algoritmo genético, e a que controla o algoritmo sobe-e-desce. Cada uma dessas funções controladoras possui a mesma organização de variáveis e compartilham da mesma função de cálculo de posição dos elevadores no tempo de execução. Na tabela 5.2 estão listadas as principais variáveis que viabilizaram a implementação dos algoritmos.

O código consiste basicamente em repetidas verificações e atualizações. A verificação que é crucial para o sistema se refere à lotação do elevador, que não pode exceder o limite determinado. Esta limitação obriga o sistema a tomar atitudes de contorno nos momentos de pico, que se afastam do modelo ideal.

```
inicializa variáveis;
fim_prog = 0;
N = 1;
while fim_prog == 0,
     if N <= qtd_chamados
          if chamado(n,t) <= tempo_atual + clock
               atualiza posição dos elevadores;
               executa genético – (inclusão de
          chamado);
               atualiza rota;
               atualiza embarque e desembarque;
               N = N + 1;
          else,
               atualiza posição dos elevadores;
               executa genético – (otimizacao de
          rotas);
               atualiza rota;
               atualiza embarque e desembarque;
          End
     elseif tempo_atual < tempo_limite
          atualiza posição dos elevadores;
          executa genético – (otimizacao de rotas);
          atualiza rota;
          atualiza embarque e desembarque;
     elseif
          fim_prog = 1;
     end
end
```

Figura 5.5 – Pseudo-código do controle dos algoritmos.

Nome da Matriz	Como é utilizada
Chamados	Armazena as informações do cenário, e os instantes medidos. Utilizada para análise de desempenho, controle de posição, e contagem de passageiros.
Tempo de Chamados	Registra e incrementa o tempo de espera do passageiro mais antigo de cada trajeto associado ao elevador. Utilizada para a criação de novas rotas.
Rota	Armazena a sequência de registros que serão executados por cada elevador (cromossomo). A cada andar atingido a rota é deslocada circularmente em uma posição.

Tempo de Rota	Registra o tempo que é necessário para o elevador atingir os andares que estão descritos na rota. Utilizada pelo controle de posição.
Embarque / Desembarque	Registram a quantidade de passageiros que desejam embarcar ou desembarcar em cada posição da rota. Utilizada para calcular se a lotação do elevador está abaixo da quantidade máxima permitida.
Posição Atual do Elevador	Registra o andar, o instante de tempo e o sentido do último atendimento de chamado realizado pelo elevador.
Tempo de Jornada	Matriz que contém o tempo gasto para o deslocamento da cabine entre os andares do edifício, utilizada para o cálculo da Matriz de Rota e na função *fitness*.

Tabela 5.2 – Principais variáveis do sistema simulador.

A matriz de embarque e desembarque é responsável pela contabilização de passageiros. Essa contagem se modifica a cada atualização da rota, para que se faça uma correta previsão da lotação do elevador nos instantes seguintes. Desta forma, o sistema evita que o passageiro aguarde por um elevador lotado. O único caso onde esta garantia falha, é quando todos os elevadores estão lotados durante o trajeto desejado. Para estes casos, a solução é uma escolha aleatória entre os elevadores, e o passageiro deverá aguardar a próxima viagem. Esta aleatoriedade foi incluída neste caso para que um único elevador lotado não absorva todos os chamados que desejam fazer o mesmo trajeto.

5.5 – Verificações e Soluções de Contorno

Nos momentos de pico de fluxo, as filas começam a aparecer. Isto porque a capacidade de transporte de peso dos elevadores é limitada a um número máximo de pessoas. Quando o número de passageiros supera este limite, a solução ótima não é mais garantida e soluções de contorno precisam ser aplicadas.

Quando um novo chamado é recebido, cada elevador executa o algoritmo genético para encontrar a melhor forma de posicionar o atendimento deste chamado dentre os demais compromissos já assumidos. O cromossomo vencedor representa uma nova rota e recebe a pontuação da função fitness

(Equação) para a disputa final entre os elevadores: o elevador que tiver a menor pontuação, ou seja, que possui o menor tempo médio de execução dos seus chamados será o vencedor (Figura 5.6).

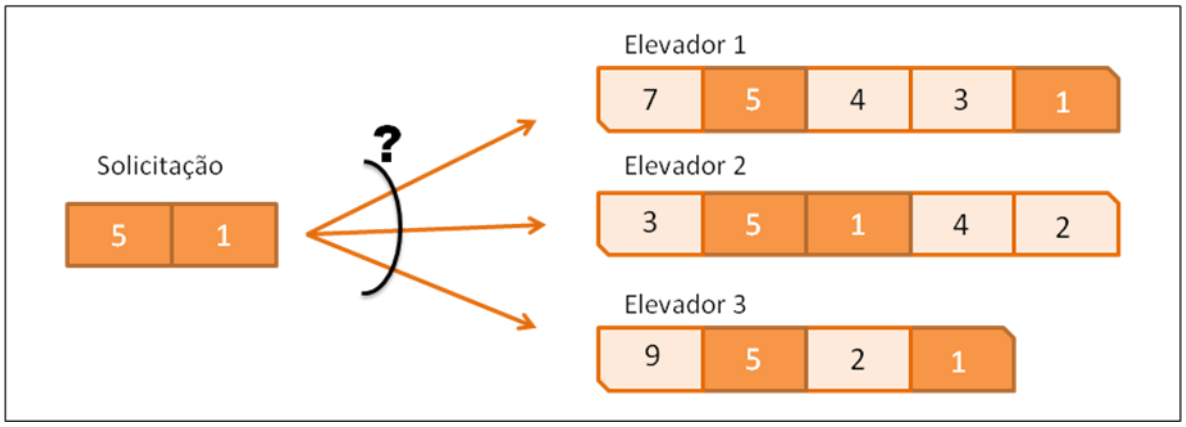

Figura 5.6 – Escolha do elevador mais capaz.

Porém, antes mesmo da disputa final, todas as novas rotas também são avaliadas quanto a sua lotação ao longo do trajeto. Se em algum momento, após a inclusão deste novo chamado, o elevador ficar superlotado, esta notificação é feita e este elevador não pode entrar na disputa.

Nos casos onde todos os elevadores estarão lotados com a inclusão deste chamado, ainda assim o passageiro precisa receber imediatamente o identificador do elevador do qual ele precisa esperar. A disputa é feita então de forma aleatória, para evitar que apenas um elevador se sobrecarregue nestes casos. Este passageiro deverá ser avisado que não poderá embarcar na primeira vez em que o elevador estacionar no seu andar de origem, para evitar que o limite máximo de carga seja excedido em algum momento do trajeto.

Note que, devido a natureza adaptativa do algoritmo, não há problemas em associar um passageiro a uma rota que ainda não esteja apta a lhe atender. Nas próximas iterações do algoritmo, o elevador terá conhecimento de que recebeu um novo compromisso e lhe incluirá nas rotas seguintes assim que for possível.

Capítulo 6

Testes

6.1 – Casos de teste

Os testes que foram aplicados obedeceram fielmente a todos os critérios e situações utilizadas nos testes dos algoritmos ETA [2]. Os cenários utilizados representam um grande número de situações práticas, visto que foram baseados em dados coletados de edifícios reais. A Tabela 6.1 mostra as características de cada edifício e as configurações de seus elevadores. O número de andares varia de 9 a 40 e a quantidade de elevadores variam de 3 a 8. O edifício B possui dois andares de entrada. O elevador do grupo C serve aos elevadores dos andares mais altos e não pára nos andares de 1 a 24. O subsolo é designado pelo número 0.

O tráfego de passageiros em um prédio pode ser descrito pela combinação de três componentes básicos: fluxo de entrada, fluxo de saída e fluxo entre andares.

- No fluxo de entrada os passageiros embarcam nos andares de entrada e viajam para cima até os andares populados do edifício;

- No fluxo de saída os passageiros embarcam nos andares populados e viajam para baixo, para os andares de entrada do edifício;

- No fluxo entre andares os passageiros embarcam em um andar populado e viajam até outro andar populado.

Os quatro padrões de tráfego utilizados testes de simulação são exibidos na Tabela 6.2. Para cada padrão de tráfego, foram gerados dois níveis de intensidade de chegada de passageiros: alta ou moderada. A intensidade de chegada é definida como o percentual da população em 5 minutos. Sendo assim, o total dos 8 cenários de tráfegos a serem investigados são: entrada densa (HI), entrada moderada (MI), saída densa (HO), saída moderada (MO), almoço denso (HL), almoço moderado (ML), entre-andares denso (HT) entre-andares moderado (MT). Para cada cenário, 10 exemplos randômicos com 1 hora de duração foram gerados. Portanto, são feitos 80 casos de teste para cada edifício e no total são 240 casos de teste para os 3 edifícios.

Na prática, a performance em situações de fluxo puramente de subida pode ser bem melhor se elevadores forem retornados para os andares de entrada depois que terminam de executar o seu atual

trajeto e nenhum novo chamado for associado a ele. Existem vários métodos para o retorno de elevadores, mas eles não estão incorporados a nenhum dos algoritmos, Por isso, o padrão de tráfego de entrada contém 5% do fluxo de passageiros de saída, para que os elevadores retornem ao andar de entrada.

Parameter	Building A	Building B	Building C
Total floors	9	16	40
Elevators	3	4	8
Capacity (persons)	13	20	21
Door opening time (s)	1.9	1.2	1.6
Door closing time (s)	2.8	2.5	2.6
Door pre-opening (s)	0	0	0
Photocell delay (s)	0.9	0.9	0.9
Loading time (s)	1.2	1.0	1.0
Unloading time (s)	1.2	1.0	1.0
Max velocity (m/s)	1.0	2.5	4.0
Acceleration (m/s^2)	0.8	1.0	1.0
Start Delay (s)	0	0	0
Floor height (m)	3.8	3.6	3.6
Exception heights (m)	Floor 0: 4.6 m	Floor 0: 3.9 m	Floor 0: 4 m
Lobby floor (entrance %)	Floor 0 (100%)	Floor 0 (20%) Floor 1 (80%)	Floor 0 (100%)
Population by floors	Floors 1,7,8: 30, other floors: 70	Floors 2-9: 20, floors 10-16: 16, floor 15: 2	Floors 25-39: 90
Total population	440	242	1350
Full arrival % of population / 5 min	15	40	13
Half arrival % of population / 5 min	7.5	20	6.5

Tabela 6.1 – Parâmetros dos prédios e as configurações dos elevadores.
Fonte: [2].

Traffic pattern	Incoming (%)	Outgoing (%)	Inter-floor (%)
Incoming	95	5	0
Outgoing	0	100	0
Lunch	40	40	20
Two-way	50	50	0

Tabela 6.2 – Padrões de tráfego.
Fonte: [2].

6.2 – Teste 1 - Algoritmo Genético x Sobe-e-desce

O primeiro teste comparou o desempenho de duas funções de distribuição de rotas, a utilizando a mesma estrutura controladora do sistema de elevadores implementada (Capítulo 5). Esta comparação entre a resposta gerada pelo algoritmo genético (item 4.2) e pelo algoritmo sobe-e-desce (item 4.3) foi feita para simples comprovação de que existem rotas mais ágeis para a solução do problema de elevadores do que as convencionais rotas do método coletivo de atendimento.

6.3 – Teste 2 - Algoritmo Genético x ETA

Para uma melhor comparação dos resultados do algoritmo genético com o mundo real, e também para analisar as vantagens e desvantagens obtidas com esta nova solução, foram realizados testes similares ao realizado por Henri Hakonen at al. [2] em 2003, no artigo em que descreve o seu algoritmo *Estimated Time of Arrival (ETA).*

Neste artigo, Hakonen compara os seus dois algorimos propostos ETA-U (sem realocação de rotas) e ETA-R (com realocação de rotas) com os resultados do algoritmo *Elevate (*ETA-E) comercialmente utilizado e desenvolvido pela empresa Peters Research Ltd. (2002).

O funcionamento básico do ETA-U foi resumidamente descrito anteriormente na solução 2 no item 2.4. Por se tratar de um algoritmo de alocação contínua, onde o passageiro não sabe antecipadamente qual elevador irá atendê-lo, ele tem a possibilidade de realocação continua dos elevadores, que foi implementada no ETA-R.

Com este teste, espera-se comprovar a boa performance do algoritmo genético, comparando seus resultados a algoritmos com eficiência mundialmente comprovada.

6.4 – Resultados encontrados

Os resultados do teste 1 e do teste 2 foram consolidados na Tabela 6.1. De acordo com os resultados, o código genético apresentou tempos de espera e de travessia bem inferiores aos encontrados pela lógica heurística de ordenamento seqüencial. Esta diferença já era esperada, devido a possibilidade de re-arrumação da rota de forma não seqüencial, e ficou comprovada a vantagem da realocação de rotas.

Prédio	Tráfego	ETA- R		Genético		Sobe-desce		Genetico (1s)	
		AWT	AJT	AWT	AJT	AWT	AJT	AWT	AJT
A	HI	23	76.6	35.73	84.30	37.98	89.61	31.10	73.81
A	MI	14.9	57.4	26.31	60.79	25.81	62.70	25.01	57.94
A	HL	26.3	69.7	41.98	87.19	49.97	102.38	36.07	71.61
A	ML	21.1	54.5	28.85	61.27	41.54	80.18	28.80	61.19
A	HO	31.6	73.6	43.92	91.82	72.60	120.90	36.26	71.26
A	MO	18.3	48.4	34.60	68.06	50.23	84.58	32.42	62.92
A	HT	29.4	71.9	38.08	81.79	48.81	98.71	30.94	68.36
A	MT	16	47.6	26.32	56.22	35.41	70.94	26.32	56.34
média		22.58	62.46	34.47	73.93	45.29	88.75		
B	HI	15.9	72.5	34.09	78.06	44.49	93.79		
B	MI	11.2	47.5	22.38	48.21	22.41	50.74		
B	HL	21.3	61.8	46.58	84.12	52.62	98.51		
B	ML	15.1	41.8	23.99	45.93	26.73	53.18		
B	HO	27.8	69.9	44.06	82.99	59.61	105.58		
B	MO	12.3	36.3	29.16	53.00	37.63	64.06		
B	HT	23.9	65	38.67	75.65	47.28	91.32		
B	MT	10.6	34.7	22.36	44.95	25.49	50.27		
média		17.26	53.69	32.66	64.11	39.53	75.93		
C	HI	31.6	129	46.81	110.13	65.68	140.81		
C	MI	14.9	104.5	38.98	90.05	52.96	108.25		
C	HL	26.9	96.7	45.16	105.65	72.51	144.27		
C	ML	22.2	85.2	29.30	73.33	46.29	94.01		
C	HO	24.8	99.9	27.25	115.98	66.98	144.41		
C	MO	14.5	65.7	31.85	87.20	59.39	112.73		
C	HT	22.6	100.7	35.39	98.88	67.95	139.44		
C	MT	13.3	69.2	29.73	77.13	45.75	95.30		
média		21.35	93.86	35.56	94.79	59.69	122.40		

Tabela 6.3 – Resultado dos testes.

O desempenho de código genético foi similar ao comportamento do ETA-R. Apesar do genético apresentar tempos maiores de espera, a diferença encontrada nos tempos de viagem foi menor. Essa diferença era esperada devido a função de otimização do código genético ser no tempo total da viagem, ao invés de dar preferência apenas tempo de espera. A coluna "Genético (1s)" especula sobre a possibilidade do artigo ETA-R só ter considerado o tempo de embarque e desembarque para apenas 1 passageiro em cada andar. O resultado encontrado demonstrou que este pode sim ter sido um fator de grande vantagem para o ETA-R, mas que não pode ser afirmado devido a ausência de mais informações no artigo.

Capítulo 7

Conclusão

7.1 – Objetivos alcançados

Os objetivos deste projeto foram alcançados de forma gratificante. A implementação de uma idéia, a descoberta de que se tratava de uma solução inovadora, e o alcance de resultados equivalentes a algoritmos recentes foram etapas vitoriosas na construção deste projeto.

Nos primeiros capítulos foi feito um resumo sobre a teoria e a prática que envolve o controle de rotas de elevadores, identificando as deficiências de uma algoritmo simplificado que funcione pelo método coletivo, e mostrando os desafios a serem solucionados, comentando algumas destas soluções já implementadas.

Em seguida, foi apresentada uma nova solução para elevadores com controle de destino utilizando algoritmos genéticos, que aproveita a vantagem do conhecimento exato dos destinos e tenta minimizar as desvantagens da alocação imediata através de re-alocações internas nas rotas de cada elevador.

Apesar do grande número de trabalhos publicados sobre elevadores inteligentes, a maioria destes artigos trata dos casos da alocação contínua, por este ter maior flexibilidade nas soluções. Foram poucos os artigos encontrados que tratavam do caso dos elevadores com controle de destino, sendo este um fator de dificuldade para a evolução deste projeto.

7.2 – Próximas atividades

Tendo sido comprovada a eficiência do algoritmo genético com reordenamento de rotas, a continuação deste trabalho consiste no melhoramento do código implementado para o controle dos elevadores através da implementação na linguagem C, e a otimização dos parâmetros do algoritmo genético. A tendência é que seja possível agilizar as simulações, e diminuir o tempo de resposta do sistema, otimizando ainda mais as rotas encontradas pelo algoritmo.

Feito isso, finalmente será possível a realização da idéia inicial da autora: a aplicação do algoritmo desenvolvido para as rotas de ônibus do transporte coletivo. E, para que isso seja feito, o esforço será apenas em ordem de grandeza - a lógica já está pronta.

Cada gen, que hoje representa um andar, passa a representar esquinas onde é possível a mudança de direção. Os cromossomos passam a ser formados por um conjunto de esquinas, e cada ponto de ônibus é associado à esquina mais próxima. Os passageiros passam a informar os seus destinos antes de sair de casa, informando qual o melhor ponto de origem que mais lhe convém. O sistema então responde ao passageiro o identificador do ônibus que irá executar este trajeto da melhor forma possível, com algumas opções de horários disponíveis. O passageiro seleciona a opção que mais lhe agrada, e pronto! Os ônibus passarão a ser um transporte confortável, confiável, pontual, e com soluções ótimas que evitarão engarrafamentos e superlotações.

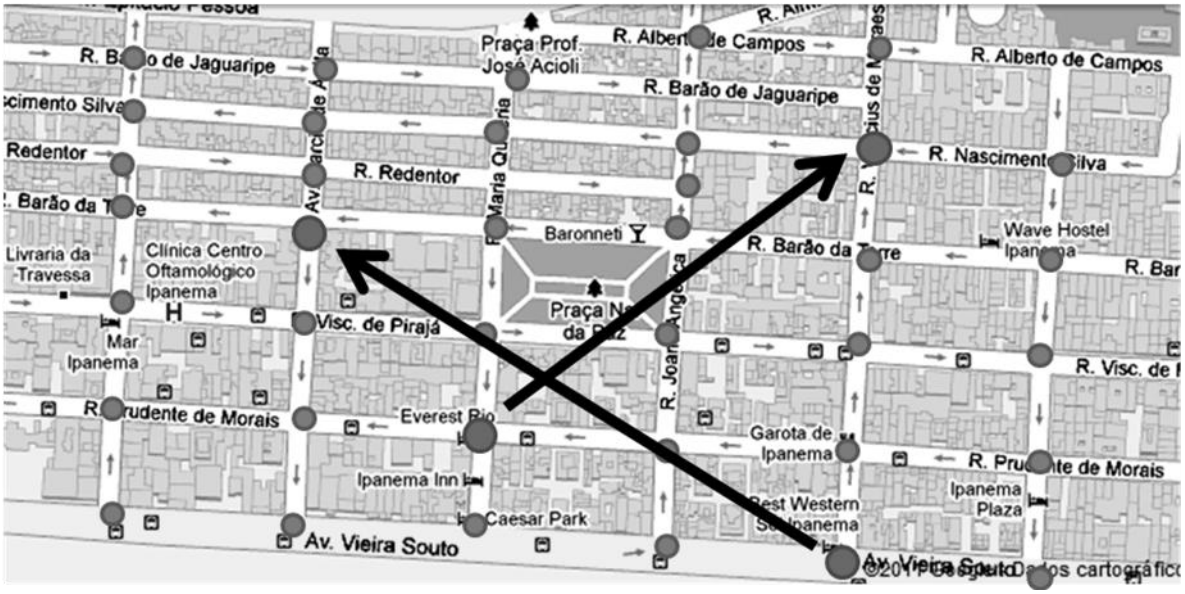

Figura 3.1 – Ilustração para o caso de rotas de trânsito.
Fonte: Google Maps [12].

Bibliografia

[1] DECICINO, R., "Trânsito – Geografia – UOL Educação" http://educacao.uol.com.br/geografia/transito-congestionamentos.jhtm (Acesso em 22 Abril 2011).

[2] RONG A., HAKONEN H. and LAHDELMA R.: Estimated Time of Arrival (ETA) Based Elevator Group Control Algorithm with More Accurate Estimation. *Technical Report 584, TUCS - Turku Centre for Computer Science*, Turku, Finlândia, 2003.

[3] PEPYNE, D.L. & CASSANDRAS, C.G. (1997). Optimal dispatching control for elevator systems during uppeak traffic. *IEEE Transactions on Control Systems Technology*, 5, 629–643.

[4] SIIKONEN, M-L.. Elevator group control with artificial intelligence, Helsinki *University of Technology, Systems Analysis Laboratory, Research Reports A67,* 1997.

[5] KOEHLER, J., and OTTIGER, D. An AI-based approach to destination control in elevators. AI Magazine 23(3):59–79., 2002

[6] JIANCHANG Liu, YIVANG Liu, *Ant Colony Algorithm and Fuzzy Neural Networkbased Intelligent Dispatching Algorithm of An Elevator Group Control* System, Conferência Internacional IEEE de Controle e Automação FrB3-2, Guangzhou, China, 2007.

[7] ARBOLEDA, D., "Implementação e Simulação de Algoritmos de Escalonamento para Sistemas de Elevadores usando Arquiteturas Reconfiguráveis" M. Sc. Sissertação, Universidade de Brasília, Novembro 2006.

[8] CORTÉS, P. et al., "Algoritmos de Optimización en Sistemas de Transporte Vertical", *II Conferencia de Ingeniería de Organización*, Vigo, Setembro 2002.

[9] OTIS Compass, "A forma mais rápida de chegar ao destino", http://www.otis.com/site/br/OT_DL_Documents/OT_DL_DocumentLibrary/Compass/Compass_Catalogo.pdf (Acesso em 28 Abril 2011).

[10] FUJINO A., TOBITA T.,SEGAWA K., YONEDA K., TOGAWA A., An elevator group control system with-attribute control method and system optimization using genetic algorithms. *IEEE Transactions on Industrial Electronics*, 44(4):546–552, 1997.

[11] SN Sivanandam and SN Deepa, Introduction to genetic algorithms. Springer, 2008

[12] GOOGLE, "Google Maps", http://maps.google.com (Acessado em Abril 2011).

Apêndice A

Código Fonte - Algoritmo Genético

```
function [pontu_atual, nova_pontuacao, nova_rota, len_rota] = genetico (alfa,
rota_ini, tmp_ch, distancia, param_gen, atual, origem, destino)

% ---  Inicializa Variaveis  ---------------------
[dummy, qtd_gens] = size(rota_ini);
qtd_cromoss = ceil(param_gen(1)/2)*2;
metade = qtd_cromoss/2;
qtd_repet = param_gen(2);
infinito = qtd_gens*distancia(1,end);
[chamado(:,1) chamado(:,2)] = find(tmp_ch);
unico = unique(chamado);
len_unico = length(unico);
[len_ch dummy] = size(chamado);
fitness = zeros(qtd_cromoss,1);
metrica = zeros(qtd_cromoss,len_ch);

% ---  Geracao inicial  ----------------------------------------------------
for c=1:qtd_cromoss,

    if c<=metade,
        ger_ini(c,:)= rota_ini;
    else
        for g=1:qtd_gens,
            ger_ini(c,g)= unico(ceil(len_unico*rand(1)));
        end
    end

    % Congelamento do primeiro gen
    ger_ini(c,1) = rota_ini(1,1);

    % Fitness
    for ch=1:len_ch,
        [dummy pos_ini] = find(ger_ini(c,:)==chamado(ch,1));
        if length(pos_ini)==0,
            metrica(c,ch) = infinito;
        else
            [dummy pos_fim] = find(ger_ini(c,pos_ini(1):end)==chamado(ch,2));
            if length(pos_fim)==0,
            metrica(c,ch) = infinito;
            else
                 metrica(c,ch) = (tmp_ch(chamado(ch,1),chamado(ch,2))/10);
                for h=1:(pos_ini(1)+pos_fim(1)-2),
                    metrica(c,ch) = metrica(c,ch) +
                    distancia(ger_ini(c,h),ger_ini(c,h+1));
                end
            end
        end
    end

    tmp_med = sum(metrica(c,:),2)/ len_ch;
```

```
    tmp_max = max(metrica(c,:)');
    fitness(c) = ((1-alfa)* tmp_med) + (alfa*tmp_max);
end
ger_ante = ger_ini;
fitness_ante = fitness;

 % ---  Geracoes Seguintes  ------------------------------------

for r=1:qtd_repet,
    fitness = zeros(qtd_cromoss,1);
    metrica = zeros(qtd_cromoss,len_ch);
    nova_ger = ger_ante;
    pont_gen=0;
    c=1;
    while (c <= qtd_cromoss),

% -- Crossover ---------------
    if c <= metade,
        cross1 = ceil(qtd_cromoss*rand(1)); %todos participam
        cross2 = ceil(qtd_cromoss*rand(1));

        corte1 = ceil(ceil(qtd_gens)* rand(1)); % comprimento da fatia
        corte2 = ceil(ceil(qtd_gens)* rand(1)); % inicio da fatia
        corte = sort([corte1; corte2] ,1);

        nova_ger(c,:) = ger_ante(cross1,:);
        nova_ger(metade+c,:) = ger_ante(cross2,:);
        nova_ger(c,corte(1):corte(2)) = ger_ante(cross2, corte(1):corte(2));
        nova_ger(metade+c,corte(1):corte(2)) = ger_ante(cross1,corte(1):corte(2));
    end
        % Mutacao
        nova_ger(c,ceil(qtd_gens*rand(1)))= chamado( ceil(len_ch*rand(1)),
                ceil(2*rand(1)) );

        % Congelamento do primeiro gen
        nova_ger(c,1) = rota_ini(1,1);

% -- Fitness -----------------
        for ch=1:len_ch,
            [dummy pos_ini] = find(nova_ger(c,:)==chamado(ch,1));
            if length(pos_ini)==0,
                metrica(c,ch) = infinito;
            else
                 [dummy pos_fim] = find(nova_ger(c,pos_ini(1):end)==chamado(ch,2));
                if length(pos_fim)==0,
                    metrica(c,ch) = infinito;
                else
                    metrica(c,ch) = tmp_ch(chamado(ch,1),chamado(ch,2))/10;
                    for h=1:(pos_ini(1)+pos_fim(1)-2),
                        metrica(c,ch) = metrica(c,ch) +
                        distancia(nova_ger(c,h),nova_ger(c,h+1));
                    end
                end
            end
        end

    end %fim do while

% -- Selecao -----------------
```

```
    geracoes = [ger_ante; nova_ger];
    fitness_geracoes = [fitness_ante; fitness];
    [fit_cromoss, id_cromoss] = sort(fitness_geracoes,1,'ascend');
    ger_ante = geracoes(id_cromoss(1:qtd_cromoss),:);
    fitness_ante = fit_cromoss((1:qtd_cromoss),:);
end
nova_rota = ger_ante(1,:);

% ---------------------------------------------------------------------

nova_pontuacao=0;
% Atualizando NOVA PONTUACAO
for ch=1:len_ch,
    [dummy pos_ini] = find(nova_rota(1,:)==origem);
    if length(pos_ini)==0,
        nova_pontuacao = infinito;
    else
        [dummy pos_fim] = find(nova_rota(1,pos_ini(1):end)==destino);
        if length(pos_fim)==0,
            nova_pontuacao = infinito;
        else
            metrica(1,ch) = tmp_ch(origem, destino)/10;
            for h=1:(pos_ini(1)+pos_fim(1)-2),
            nova_pontuacao = nova_pontuacao +
                 distancia(nova_rota(1,h),nova_rota(1,h+1));
            end
        end
    end
end


% Atualizando Len_rota
for ch=1:len_ch,
    [dummy pos_ini] = find(nova_rota==chamado(ch,1));
    if length(pos_ini)==0,
        comprimento(ch)=0;
    else
         [dummy pos_fim] = find(nova_rota(1,pos_ini(1):end)==chamado(ch,2));
        if length(pos_fim)==0,
            comprimento(ch)=0;
        else
            comprimento(ch) = pos_ini(1) + pos_fim(1)-1;
        end
    end
end
len_rota = max(comprimento);




% Atualizando Pontuacao Atual
for ch=1:len_ch,
    [dummy pos_ini] = find(rota_ini(1,:)==chamado(ch,1));
    if length(pos_ini)==0,
        metrica(1,ch) = infinito;
    else
        [dummy pos_fim] = find(rota_ini(1,pos_ini(1):end)==chamado(ch,2));
```

```
        if length(pos_fim)==0,
            metrica(1,ch) = infinito;
        else
            metrica(1,ch) = tmp_ch(chamado(ch,1),chamado(ch,2));
            metrica(1,ch) = metrica(1,ch) + distancia(atual,rota_ini(1,1));
            for h=1:(pos_ini(1)+pos_fim(1)-2),
                metrica(1,ch) = metrica(1,ch) +
distancia(rota_ini(1,h),rota_ini(1,h+1));
            end
        end
    end
end
tmp_med = sum(metrica(1,:),2)/ len_ch;
tmp_max = max(metrica(1,:)');
pontu_atual = ((1-alfa)* tmp_med) + (alfa*tmp_max);
```

Apêndice B

Código Fonte - Algoritmo Sobe-e-Desce

```
function [pontu_atual, nova_pontu, nova_rota, len_rota] = sobe_e_desce (rota_ini,
tmp_ch, distancia, atual, direcao, alfa,origem,destino)

[orig dest] = find(tmp_ch);
nova_rota = rota_ini; %para igualar as dimensoes
[qtd_andar dummy] = size(distancia);
infinito = qtd_andar*distancia(1,end);

if direcao==0, %sobe
    in = find(orig >= rota_ini(1,1));
    ex = find(orig < rota_ini(1,1));
    s_in = find(orig(in) <= dest(in));
    d_in = find(orig(in) > dest(in));
    s_ex = find(orig(ex) <= dest(ex));

    andares_1 = sort(unique([orig(in(s_in)); dest(in(s_in))]),'ascend');
    andares_2 = sort(unique([orig(in(d_in)); dest(in(d_in))]),'descend');
    andares_3 = sort(unique([orig(ex(s_ex)); dest(ex(s_ex))]),'ascend');

else % desce
    in = find(orig <= rota_ini(1,1));
    ex = find(orig > rota_ini(1,1));
    s_in = find(orig(in) >= dest(in));
    d_in = find(orig(in) < dest(in));
    s_ex = find(orig(ex) >= dest(ex));

    andares_1 = sort(unique([orig(in(s_in)); dest(in(s_in))]),'descend');
    andares_2 = sort(unique([orig(in(d_in)); dest(in(d_in))]),'ascend');
    andares_3 = sort(unique([orig(ex(s_ex)); dest(ex(s_ex))]),'descend');
end

if length(andares_1)~=0,
if length(andares_2)~=0,
if (andares_1(end)==andares_2(1)),
    if length(andares_2)>(1),
        andares_2 = andares_2(2:end);
    else
        andares_2 = [];
    end
end
end
end

if length(andares_2)~=0,
if length(andares_3)~=0,
if (andares_2(end)==andares_3(1)),
    if length(andares_3)>(1),
        andares_3 = andares_3(2:end);
    else
        andares_3 = [];
```

```
    end
end
end
end

andares = [andares_1; andares_2; andares_3];
len_rota = length(andares);
nova_rota = rota_ini;
nova_rota(1,1:len_rota)= andares';
nova_rota(len_rota+1:length(rota_ini))= 1;


% NOVA PONTUACAO
[dummy pos_ini] = find(nova_rota(1,:)==origem);
nova_pontu =0;
if length(pos_ini)==0,
    nova_pontu = infinito;
else
    [dummy pos_fim] = find(nova_rota(1,pos_ini(1):end)==destino);
    if length(pos_fim)==0,
        nova_pontu = infinito;
    else
        for h=1:(pos_ini(1)+pos_fim(1)-2),
           nova_pontu = nova_pontu + distancia(nova_rota(1,h),nova_rota(1,h+1));
        end
    end
end
```