MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

MARCUS VINICIUS RODRIGUES LIMA

UMA ABORDAGEM AO ROTEAMENTO DE VEÍCULOS UTILIZANDO MÚLTIPLOS CRITÉRIOS

Rio de Janeiro
2008

INSTITUTO MILITAR DE ENGENHARIA

MARCUS VINICIUS RODRIGUES LIMA

UMA ABORDAGEM AO ROTEAMENTO DE VEÍCULOS UTILIZANDO MÚLTIPLOS CRITÉRIOS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Luiz Henrique da Costa Araújo - D.C.

Rio de Janeiro

2008

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro - RJ CEP: 22290-270

L732 Lima, Marcus Vinicius Rodrigues
Uma abordagem ao roteamento de veículos utilizando múltiplos critérios/ Marcus Vinicius Rodrigues Lima. - Rio de Janeiro: Instituto Militar de Engenharia, 2008.
107 p.: il.

Dissertação (mestrado) - Instituto Militar de Engenharia – Rio de Janeiro, 2008.

1. Algoritmos em Grafos. 2. Algoritmos de Roteamento. 3. Rodovias.

S11.S

INSTITUTO MILITAR DE ENGENHARIA

MARCUS VINICIUS RODRIGUES LIMA

UMA ABORDAGEM AO ROTEAMENTO DE VEÍCULOS UTILIZANDO MÚLTIPLOS CRITÉRIOS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Luiz Henrique da Costa Araújo, D.SC. UFRJ-RJ

Aprovada em 28 de Agosto de 2008 pela seguinte Banca Examinadora:

_____

Luiz Henrique da Costa Araújo, D.C. do IME - Presidente

_____

Lilian Markenzon, D.C. da UFRJ

_____

Paulo Renato da Costa Pereira, D.C. do IME

Rio de Janeiro

2008

A Deus, a minha mãe, esposa e filhos Gustavo e Gabriela.

AGRADECIMENTOS

A Deus pelo que sou e pelos meus filhos. À existência dos meus filhos e minha fé, que motivaram e me ajudaram a superar as barreiras para a conclusão deste trabalho.

Ao companheirismo de minha esposa Ana Luiza que me incentivou e me ajudou a obter tempo para a conclusão deste trabalho.

A minha mãe Sônia e avós maternos. A minha mãe, pelo exemplo de sucesso em retomar os estudos após o falecimento de meu pai. Aos meus avós maternos, por terem sido o alicerce do sucesso de seus filhos e netos.

Aos meus filhos Gustavo e Gabriela que compreenderam a minha ausência em prol deste trabalho.

Ao Senhor D.C. Major Luiz Henrique da Costa Araújo, que além de orientador foi um grande companheiro. Sempre compreensivo e disponível, promoveu o êxito do trabalho, ajudando a conciliar o tempo de estudo e trabalho.

Ao amigo de turma M.C. Cap. Sergio Cardoso, que durante a conclusão das cadeiras, auxiliou a consolidar a teoria com as suas revisões.

Aos senhores coordenadores e vice-coordenadores da pós-graduação, Ph.D Ronaldo Moreira Salles e Ph.D Paulo Fernando Ferreira Rosa, os quais atentos à alta qualidade de ensino e ao tratamento personalizado a este aluno, o auxiliaram e concederam-lhe tempo adicional para a conclusão deste trabalho.

Ao amigo M.C. Luiz Antônio Palmeira Monteiro, que além de aconselhar acadêmica e profissionalmente, foi o motivador de minha inscrição no curso.

Aos meus gestores e ex-gestores na empresa VIVO, em especial ao Javier Rodriguez, Antônio Augusto C Rocha e Rodrigo Grassi, que investiram parte de seu tempo em meu aperfeiçoamento e formação.

Por último, ao meu irmão Mauricio, Elias, familiares, professores e amigos que me incentivaram, apoiaram e possibilitaram esta oportunidade de ampliar meus horizontes.

“O poder é a habilidade de fazer boas coisas para os outros”.

BROOKE ASTOR

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

BDS	Bi-Directional Search
HAMCRA	Hybrid Auguring Multiple Constraints Routing Algorithm
LGS	Level Graph Search
MCP	Multi-Constrained Path
MCOP	Multi-Constrained Optimal Path
RSP	Restricted Shortest Path
SAMCRA	Self-Adaptive Multiple Constraints Routing Algorithm
SIG	Sistema de Informação Geográfica
TAMCRA	Tunable Accuracy Multiple Constraints Routing Algorithm

RESUMO

Este trabalho apresenta um estudo do problema do roteamento utilizando múltiplos critérios em um grafo direcionado, para isso, realiza a análise comparativa do algoritmo *TAMCRA* com a heurística “Level Graph Search- LGS”.

A análise é baseada na geração eficiente em tempo real de caminhos mínimos com múltiplos critérios entre dois pontos de um grafo direcionado e conexo.

Ao abordar o contexto rodoviário serão apresentadas análises comparativas de caminhos mínimos com múltiplos critérios, para possível aplicação em roteamento de veículos em malhas viárias urbanas, além da busca em mapas e problemas em rede.

ABSTRACT

This work presents a study of the multi-constrained path problem in the digraph, making the comparative analysis of algorithm *TAMCRA* with the heuristic “Level Graph Search- LGS”.

The analysis is based on the efficient generation in real time of multi-constrained paths between two edges in the connected digraph.

Approaching the context road comparative analyses of minimum paths with multiple criteria are presented, for possible application in urban road networks, search in maps and problems in networks.

1 APRESENTAÇÃO

1.1 INTRODUÇÃO

Uma quantidade crescente de veículos utiliza sistemas de planejamento de rotas. Usuários comuns e empresas possuem dispositivos adicionais utilizando sistemas em tempo real para determinar a melhor rota a ser realizada com seus veículos. Com estas potencialidades adicionais, veículos movem-se sem parar com sistemas isolados e sem integração com características das vias, do trânsito, de velocidade e das obras que acontecem em seu redor.

Em LEGNER(2002) é apresentada uma visão geral de serviços veiculares que podem ser oferecidos e transmitidos aos veículos. A utilização de um roteamento com tempo de execução que permita trocar informação direta com os veículos é uma das possíveis aplicações deste estudo. A troca de sentido dos veículos e sua localização são exemplos de métricas adicionais que poderão ser considerados para fornecer orientações mais precisas.

O contexto rodoviário nos traz alguns tópicos especiais. Dentre eles está o de ser informado sobre a rota a seguir dentro do veículo.

Especialmente a distribuição dos nós e a mobilidade são influenciadas pelo contexto veicular. Os veículos normalmente são limitados a rodovias e induzidos a uma distribuição muito especial de veículos no mapa. Por exemplo, a probabilidade de se encontrar veículos em regiões geográficas onde não existe estrada é quase nula. Conseqüentemente, a eficiência de se realizar um roteamento de veículos em diferentes regiões deve ser considerada.

Quando se trata de planejamento de rotas rodoviárias, a primeira coisa que nos vem à mente é a utilização de aplicações com mapas rodoviários, sem mesmo nos referir à infa-estrutura da rodovia. As aplicações com mapas, em sua maioria, são geo-referenciadas e nos remetem à utilização de distância geográfica entre um ponto e outro no mapa. Imagine que você esteja dentro de um veículo numa via expressa de mão dupla. Para utilizar o sentido contrário desta via, será necessário percorrer a via até o contorno mais próximo, contornar e percorrer a mesma distância em sentido contrário. Conforme ilustrado na Figura 1.1, uma suposição comum é que veículos fisicamente próximos também estejam próximos na topologia das rodovias, mas pode não ser verdade em alguns casos devido à topologia e ao sentido das rodovias.

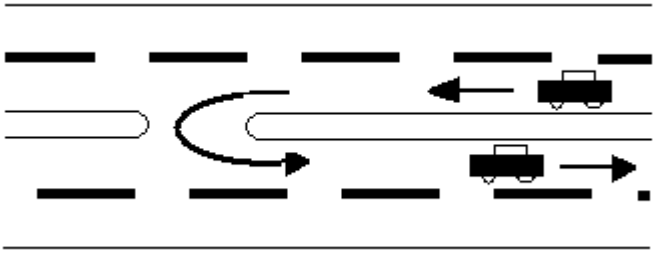


FIG.1.1: Veículos fisicamente próximos.

Adicionalmente, os veículos podem não estar igualmente distribuídos sobre a malha rodoviária, porque as estradas são geralmente organizadas em uma hierarquia. Os níveis superiores de rodovias possuem um número maior de pistas e um limite de velocidade elevado, visto que ambos fatores decrescem conforme a hierarquia. Geralmente, os maiores níveis consistem em rodovias interestaduais, avenidas etc. O nível mais baixo é composto de rodovias menores, como rodovias localizadas internamente aos bairros. Esta categorização conduz a uma distribuição não uniforme dos veículos na malha rodoviária, porque a densidade de veículos em uma marginal é geralmente maior do que em rodovias dos níveis mais baixos da hierarquia.

Uma malha rodoviária pode ser particionada por pesos. Muitos algoritmos de menor caminho não são adequados quando o interesse do motorista é conduzir seu veículo por ruas principais e queira adicionar outros critérios em sua busca do melhor caminho. Uma opção é o uso do algoritmo *LGS* que pode ser encontrado em SHAPIRO (1992), com o qual se pode minimizar o uso de ruas secundárias. Este

algoritmo apresenta restrições de otimalidade e orientação quando implementado em um sistema de roteamento para áreas urbanas. Em BRAZ (2006) o algoritmo é modificado e integrado a um SIG. A referida modificação nomeada de *LGS**, melhora sua otimalidade, elimina a restrição de orientação e não afeta a complexidade de pior caso original.

Da mesma forma que é minimizada a utilização de ruas secundárias, a proposta deste estudo contempla uma modelagem que possa utilizar múltiplos critérios. Como exemplo, além da hierarquia das ruas, um roteamento poderá utilizar métricas de segurança da rodovia e de tráfego em determinado dia da semana e horário. Com a contribuição deste trabalho, espera-se que a utilização de múltiplos critérios no contexto rodoviário, venha aperfeiçoar a busca de caminhos mínimos e será uma forma de garantir que os caminhos selecionados atenderão à expectativa do usuário.

Uma possível aplicação do estudo apresentado traduz-se em aplicações que utilizam planejamento de rotas em tempo real. Além de maximizar a utilização de ruas com maior hierarquia, poderemos garantir que o custo máximo fim a fim do caminho será atendido. A tarefa será encontrar uma rota que tenha recursos suficientes para satisfazer a múltiplas regras. O problema que estuda o roteamento com múltiplos critérios é conhecido como NP completo conforme descrito em WANG (1996) e por JAFFE (1984).

A idéia do estudo é reduzir o problema do caminho com múltiplos critérios a um problema mais simples. Através da minimização do problema e com o ajuste da métrica que limita a combinação de caminhos, será demonstrado que o problema pode ser resolvido em tempo polinomial.

Com objetivo de atender às expectativas dos seres humanos, descritas em estudos comportamentais de LIU (1996), serão analisados os caminhos produzidos pelos algoritmos *LGS* e *TAMCRA*.

Para a realização desta análise dos caminhos de forma homogênea, é utilizado um modelo conhecido como treliça estruturada descrita em SHAPIRO (1992). A escolha deste modelo é devida à facilidade em analisar níveis e critérios em grandes centros urbanos estruturados como a cidade de Manhattan.

Para alcançar estes objetivos, esta dissertação foi dividida em sete capítulos. Nesta apresentação é fornecida a base teórica, o posicionamento, a finalidade e a

organização do trabalho como um todo. A Seção 1.2 fornece conceitos básicos de grafos. Na Seção 1.3 é definido o conceito de métrica para a busca de caminhos. Por último, a Seção 1.4 fornece a definição dos tipos de métricas. A conceituação dos tipos de métricas, é fundamental para o entendimento da forma correta de aplicar os critérios na busca dos caminhos com múltiplos critérios.

O Capítulo 2 descreve como é possível utilizar o caminho em grafos para a busca de rotas rodoviárias. A Seção 2.1 apresenta a motivação do trabalho e descreve o problema de roteamento e de caminho com múltiplos critérios. A Seção 2.2 apresenta alguns algoritmos clássicos de menor caminho. A Seção 2.3 traz os algoritmos de múltiplos critérios analisados ao longo deste estudo. A Seção 2.4 apresenta o motivo da utilização de um algoritmo com múltiplos critérios e o porquê da escolha do algoritmo *TAMCRA*.

O Capítulo 3 define o algoritmo *TAMCRA* como nova proposta para o roteamento utilizando múltiplos critérios. A Seção 3.1 faz uma breve introdução exibindo o objetivo de utilização do *TAMCRA*. A Seção 3.2 descreve o funcionamento e os principais conceitos aplicados no algoritmo. A Seção 3.3 apresenta a complexidade do algoritmo.

O Capítulo 4 apresenta o algoritmo *LGS* e suas variações. A Seção 4.1 realiza uma introdução e explica como o algoritmo busca os caminhos no grafo em níveis. A Seção 4.2 traz as definições e procedimentos aplicados. A Seção 4.3 determina e explica a complexidade do algoritmo *LGS*.

O Capítulo 5 descreve o ambiente e as premissas utilizadas no experimento computacional. A Seção 5.1 apresenta a metodologia e motivo da realização da análise experimental. A Seção 5.2 exibe o modelo de referência, explica a treliça estruturada e sua utilização na análise. A Seção 5.3 descreve como serão obtidas as informações da análise experimental.

O Capítulo 6 traz os resultados obtidos com os dois algoritmos avaliados e suas aplicações em curta, média e longa distância.

E, finalmente, o Capítulo 7 apresenta as conclusões finais e sugestões para trabalhos futuros.

1.2 CONCEITOS BÁSICOS DE GRAFOS

Tendo em vista a grande variação nas definições encontradas na literatura, nesta seção será apresentada a terminologia, as definições e os conceitos que serão utilizados neste texto. Os mesmos podem ser encontrados em HARARY(1969), SZWARCFITER(1986), GOLUMBIC(1980).

Definição 1.1 - *Um grafo G consiste em um par (V,E) de conjuntos tais que V = {v_1,...,v_n} é finito não vazio e* $E \subseteq \{(x,y) \mid x,y \in V \; e \; x \neq y\}$. *Os elementos de V e E são chamados, respectivamente, vértices e arestas. O grafo G é representado por G=(V,E). A cardinalidade do conjunto de vértices é representada por n, enquanto a do conjunto de arestas é representada por m.*

Definição 1.2 – *Dois vértices, x e y, são adjacentes quando* $(x,y) \in E$. *O conjunto de adjacência de um vértice v, Adj(v), é o conjunto formado por todos os vértices do grafo que são adjacentes a v. O grau de um vértice x, grau(x), é a cardinalidade do seu conjunto adjacência.*

Definição 1.3 – *Seja S um conjunto e S' $\subseteq$ S. Diz-se que S' é maximal em relação a uma certa propriedade P, quando S' satisfaz a propriedade P e não existe subconjunto S'' $\supset$ S' que também satisfaz P.*

Definição 1.4 – *O grafo G' = (V', E') é um subgrafo de G = (V, E) quando V' $\subseteq$ V e E' $\subseteq$ E. Se, além disso, G' possuir toda aresta (u,v) de G tal que ambos u, v estejam em V, então G' é o subgrafo induzido pelo conjunto de vértices V'.*

Definição 1.5 – *Um grafo G = (V, E) é completo se para qualquer par de vértices u, v $\in$ V existe a aresta (u, v) $\in$ E.*

Definição 1.6 – *Seja G = (V, E) um grafo, e S $\subseteq$ V. Diz-se que S é uma clique se o subgrafo induzido por S é completo. Denota-se uma clique com n vértices por K_n.*

Definição 1.7 – *Uma seqüência de vértices $p_{v1,vk} = v_1,v_2,...,v_k$, tal que $(v_i, v_{i+1}) \in E$, $1 \leq i \leq k - 1$, é denominada um caminho de v_1 a v_k. Os vértices v_1 e v_k* são denominados extremos do caminho e os demais, vértices internos do caminho.

Definição 1.8 – *Dois caminhos $p_{u,v}$ e $p'_{u,v}$ entre o mesmo par de vértices u e v são considerados caminhos disjuntos quando todos os vértices internos de um caminho são diferentes dos vértices internos do outro.*

Definição 1.9 – *Um caminho $p_{u,v}$ é simples se não existem vértices repetidos na seqüência de vértices que define $p_{u,v}$.*

Definição 1.10 – *A distância d(u, v) entre dois vértices u e v de um grafo G é o comprimento (número de arestas) do caminho mais curto entre eles, se existir; caso contrário, $d(u, v) = \infty$.*

Definição 1.11 – *Um ciclo é um caminho $C = v_1,v_2,...,v_k,v_1$, $k \geq 3$. Se o caminho $v_1,v_2,...,v_k$ for simples então o ciclo C é denominado ciclo simples.*

Definição 1.12 – *Um grafo que possui um ciclo hamiltoniano é denominado grafo hamiltoniano. O ciclo simples que contém todos os vértices de G é o ciclo hamiltoniano de G.*

Definição 1.13 – *Um grafo G é conexo, se quaisquer dois vértices u e v são ligados por um caminho. Caso contrário, é desconexo.*

Definição 1.14 – *Em um grafo G desconexo, cada subgrafo maximal conexo é chamado de componente de G.*

Definição 1.15 – *Um vértice de articulação de G é qualquer vértice que, se retirado de um grafo, aumenta o seu número de componentes.*

Definição 1.16 – *Seja G = (V, E) um grafo. Uma aresta $e \in E$ é chamada ponte quando sua remoção de G o desconecta.*

Definição 1.17 – *Seja G = (V, E) um grafo e sejam u e v vértices não adjacentes. O conjunto S ⊂ V é um u-v separador se os vértices u e v pertencem a diferentes componentes do subgrafo G_{V-S} induzido por V – S. O conjunto S é um separador minimal se não existe S' ⊂ S tal que S' é um u-v separador.*

Definição 1.18 – *Um isomorfismo entre grafos é uma bijeção entre os seus conjuntos de vértices que preserva a relação de adjacência dos grafos. Dois grafos são isomorfos se existe um isomorfismo entre eles.*

Definição 1.19 – *A conectividade de um grafo conexo G, denotada por k(G), é o menor número de vértices que, ao serem removidos de G, o torna desconexo ou trivial. Um grafo é n-conexo se k(G) ≥ n. G é biconexo no caso particular de k(G) ≥ 2.*

Definição 1.20 – *Os blocos de um grafo são os seus subgrafos maximais biconexos ou isomorfos ao grafo completo com 2 vértices. Portanto, um grafo biconexo tem exatamente um bloco.*

1.3 MÉTRICAS PARA A BUSCA DOS CAMINHOS

Para conceituar os possíveis de tipos de métricas, nesta seção são descritas as definições e teoremas apresentadas em WANG (1996).

Para toda métrica escolhida, devem existir algoritmos eficientes para a computação do caminho, de modo que o roteamento seja capaz de ser utilizado em malhas rodoviárias. A complexidade dos algoritmos para a busca do caminho deve preferencialmente ser comparável aos atuais algoritmos de roteamento. Também é desejável que todos os algoritmos possam trabalhar juntos em um ambiente centralizado para a realização das análises experimentais.

As métricas aplicadas devem refletir as características básicas de uma malha rodoviária. A informação que elas contêm deve tornar possível suportar exigências básicas de roteamento de veículos. Note que todas as exigências de roteamento de veículos têm que ser mapeadas em critérios de um caminho expresso por métricas, assim as métricas determinam os tipos de roteamento que a malha rodoviária pode

suportar. Por exemplo, se a hierarquia da rua e o nível de segurança forem os critérios escolhidos, todas as exigências do roteamento têm que ser mapeadas nas métricas hierarquia da rua e nível de segurança respectivamente.

As métricas devem ser ortogonais entre si, de modo a não existir informação redundante entre elas. A informação redundante pode introduzir a interdependência entre as métricas. A avaliação recursiva entre duas métricas pode, substancialmente, complicar a busca do caminho.

1.3.1 ÚNICA MÉTRICA COMBINADA

Antes de se conceituar métricas para uma malha rodoviária, deve-se fazer uma pergunta natural: quais os únicos critérios que podem atender às necessidades do roteamento de veículos?

Uma possível aplicação, seria definir uma função e gerar um único critério com múltiplas métricas. A idéia é misturar várias partes da informação em uma única métrica e usá-la como base para decisões de roteamento. Por exemplo, a métrica combinada pode ser produzida com o número mínimo de pistas *B*, e distância *D* e probabilidade de tráfego *L* com a fórmula $f(p) = B(p)/(D(p)\ X\ L(p))$. Um caminho com maior valor é mais provável de ser escolhido em termos do número de pistas, distância e probabilidade de tráfego.

Outro problema será definir como tratar as métricas mistas com composições diferentes. Por exemplo, suponha que um caminho possua dois segmentos $\{a,b\}$ e $\{b,c\}$. Se o critério $f(p)$ for a distância, a regra de composição será $f((a,b)+(b,c)) = f(a,b) + f(b,c)$. Se o critério $f(p)$ for o número de pistas, a regra é $f((a,b)+(b,c)) = min[f(a,b),f(b,c)]$. Entretanto, se for considerada a expressão $B(p)/(D(p)\ X\ L(p))$, nenhuma das regras acima são válidas.

1.3.2 MÚLTIPLAS MÉTRICAS

Múltiplas métricas podem certamente modelar uma malha rodoviária mais precisa. Entretanto, o problema de encontrar um caminho com múltiplos critérios e os algoritmos de tempo polinomial para o problema podem não existir. Um problema simples com duas regras é mostrado em GAREY (1979) como um problema NP completo.

O problema de roteamento utilizando mais de uma métrica é mais complexo, desde que as necessidades de recurso especificadas pelas aplicações sejam diversas e dependentes da aplicação. A complexidade é determinada primeiramente pelas métricas de composição dos critérios. Há três regras de composição básicas que são analisadas:

Definição 1.21 - *Sendo $d(v_1,v_2)$ a métrica para a aresta (v_1,v_2). Para qualquer caminho $p_{v1,vk} = (v_1,v_2,...,v_k)$, é dito que a métrica pode ser :*

a) *aditivo se $d(p_{v1,vk})=d(v_1,v_2) + d(v_2,v_3) +... + d(v_{k-1},v_k)$.*
b) *multiplicativo se $d(p_{v1,vk})=d(v_1,v_2)$ x $d(v_2,v_3)$ x … x $d(v_{k-1},v_k)$.*
c) *côncavo se $d(p_{v1,vk})=min[d(v_1,v_2),d(v_2,v_3),...,d(v_{k-1},v_k)]$.*

Agora são observadas algumas métricas que são provavelmente consideradas como critérios de roteamento rodoviários citados anteriormente: distância, número de pistas e probabilidade de tráfego. A distância segue o critério de composição aditiva, o número de pistas segue o critério côncavo e a probabilidade de tráfego, a composição multiplicativa.

Abaixo são apresentados três teoremas NP completos gerais para métricas aditivas e multiplicativas. Estes darão a base para nossa métrica de seleção. As provas dos teoremas podem ser encontradas em WANG (1996).

Teorema 1.1 - *Dado um grafo G = (V, E), x métricas aditivas $d_1(a)$, $d_2(a)$,..., $d_x(a)$ para cada a $\in$ E, dois vértices específicos v_1, v_2, e x inteiros positivos D_1, D_2, D_x, (x ≥ 2, $d_{v1}(a) \geq 0$, $D_{v1} \geq 0$ para $v_1 = 1,2,...,x$), o problema de decidir se existe um caminho simples $p_{v1,vk} = (v_1, v_2,..., v_k)$ que satisfaça as seguintes regras $d_{v1}(p_{v1,vk}) \leq D_{v1}$ onde $v_1 = 1,2,...,x$ (o problema de adição de x métricas) é NP completo.*

Teorema 1.2 - *Dado um grafo G = (V, E), y métricas multiplicativas $d_1(a)$, $d_2(a)$,..., $d_y(a)$ para cada a $\in$ E, dois vértices específicos v_1, v_2, e y inteiros positivos D_1, D_2...,D_y, (y ≥ 2, $d_{v1}(a) \geq 1$, $D_{v1} \geq 1$ para $v_1 = 1,2,.. .,y)$, o problema de decidir se existe um caminho simples $p_{v1,vk} = (v_1, v_2,..., v_k)$ que satisfaz as seguintes regras $d_{v1}(p_{v1,vk}) \leq D_{v1}$ onde $v_1 = 1,2,...,y$ (o problema da multiplicação de y métricas) é NP completo.*

Teorema 1.3 - *Dado um grafo G = (V, E), com x métricas aditivas e y multiplicativos, $d_1(a)$, $d_2(a)$..., $d_{x+y}(a)$ para cada a $\in$ E, dois vértices específicos v_1, v_2, e x+y inteiros positivos D_1, D_2... , D_{x+y}, (x ≥ 1, y ≥ 1, $d_{v1}(a) \geq 1$, $D_{v1} \geq 0$ para $v_1 = 1,2,...,x$, $D_{v1} \geq 1$ para $v_1 = x + 1, 2,...$, x+y), o problema de decidir se existe um caminho simples $p_{v1,vk} = (v_1, v_2,..., v_k)$ que satisfaz as seguintes regras $d_{v1}(p) \leq D_{v1}$ onde $v_1 = 1, 2,..., x+y$ (o problema de adição de x métricas e o problema de multiplicação de y métricas) é NP completo.*

Os três teoremas acima mostram que o problema de encontrar um caminho com dois ou mais métricas aditivas e multiplicativas em qualquer combinação possível é NP completo. Os resultados são aplicáveis a todos as métricas que seguirem métricas aditivas e multiplicativas de composição e a qualquer critério que puder ser transformado em métricas equivalentes que seguem a composição aditiva ou multiplicativa.

2 CAMINHOS EM GRAFOS E SUAS APLICAÇÕES EM ROTAS RODOVIÁRIAS

2.1 MOTIVAÇÃO

Conforme descrito em SZWARCFITER (1986), dentre as técnicas existentes para a solução de problemas algorítmicos em grafos, a busca de caminhos ocupa lugar de destaque devido ao grande número de problemas que podem ser resolvidos através da sua utilização. Provavelmente, a importância desta técnica é ainda maior quando o universo das aplicações for restrito aos algoritmos considerados eficientes.

A busca visa resolver um problema básico de exploração de um grafo. Isto é, dado um grafo deseja-se obter um processo sistemático de como caminhar pelos vértices e arestas do mesmo.

Para a utilização em malhas rodoviárias, além da necessidade de orientação e obtenção de caminhos mínimos, pesquisadores também procuram considerar resultados ligados a estudos comportamentais em seres humanos. Conforme verificado em SHAPIRO (1992), a introdução do conceito do caminho curto suficiente (*short enough*) procura considerar critérios associados a nossa natureza definidos em LIU (1996).

A busca baseada em critérios é indispensável para determinar a qualidade de rotas rodoviárias. Determinar antecipadamente a qualidade do caminho que se deseja percorrer em uma malha rodoviária, pode aumentar a satisfação do usuário e agregar valor ao serviço prestado. Para a tradução das expectativas em critérios para o roteamento rodoviário, serão utilizados os conceitos em MIEGHEN (2004), que abaixo define o problema de roteamento com múltiplos critérios.

Dado um grafo G, em cada aresta (u,v) é considerado um vetor de dimensão k com os pesos de cada critério, $\vec{w}(u,v) = [w_1(u,v), w_2(u,v),\ldots, w_z(u,v)]$ onde $w_i > 0$ são critérios como segurança, quantidade de pistas, etc. O algoritmo de roteamento computa o caminho $P_{v1,vk}$ que obedece a múltiplos critérios representados por $w_i(P_{v1,vk}) \leq L_i$ para todo $1 \leq i \leq z$.

Por exemplo, suponha a procura de um caminho rodoviário que seja percorrido da origem ao destino em até 10 minutos. Para alcançar o destino, é

esperado maximizar a utilização de caminhos com duas ou mais pistas. L_i representa a qualidade desejada do caminho, L_1 será o tempo de 10 minutos, L_2 será o menor número de pistas e L será o vetor que contém todos os limites desejados. Considerando o tempo como um critério aditivo, a soma dos tempos associadas as k-1 arestas de um caminho $P_{v1,vk} = v_1, v_2,..., v_k$, deverá ser menor ou igual ao critério L_1 .

$$\vec{w}(P_{\mathrm{v1,vk}}) = \sum_{i=1}^{k-1} w(\vec{v}_i, \vec{v}_{i+1}) \qquad (1)$$

O caminho com múltiplos critérios $P_{v1,vk}$ é um caminho entre o nó v_1 e o nó v_k que se satisfaz $w_i(P_{v1,vk}) \leq L_i$ para todo $1 \leq i \leq z$.

A seguir são apresentados os problemas de caminhos referentes ao problema de roteamento com múltiplos critérios definidos em MIEGHEN (2004). A Definição 2.1 conceitua o problema do caminho com múltiplos critérios e caminhos praticáveis. A Definição 2.2 apresenta o problema do caminho ótimo com múltiplos critérios.

Definição 2.1 – *Problema do caminho com múltiplos critérios: seja um grafo G(V,E), cada aresta (u,v)* $\in$ *E é associada a um vetor com z componentes. Cada um desses componentes é uma métrica aditiva segundo a Definição 1.21 encontrada na Subseção 1.3.2 do Capítulo 1. O problema é encontrar um caminho* $P_{v1,vk}$ *tal que*

$$w_i(P_{v1,vk}) = \sum_{(u,v)\in P_{v1,vk}} w_i(u,v) \leq L_i \qquad (2)$$

para todo $1 \leq i \leq z$.

Um ou mais caminhos no grafo G(V,E) que satisfaçam todos os z critérios são conhecidos como caminhos praticáveis. De acordo com a Definição 2.1, estes caminhos são conhecidos como uma solução para o MCP. Entretanto, é possível recuperar o caminho mínimo $l(P_{v1,vk})$ *dentre o conjunto de caminhos praticáveis. O problema que adicionalmente otimiza alguma função l() é chamado de Problema do caminho ótimo com múltiplos critérios conforme Definição 2.2.*

Definição 2.2 – *Problema do caminho ótimo com múltiplos critérios: seja um grafo G(V,E). Cada aresta (u,v) pertencente a E é representada com um vetor de pesos aditivos* $w_i(u,v) \geq 0$ *para todo* $1 \leq i \leq z$. *Dados z critérios* L_i, *onde* $1 \leq i \leq z$, *o problema de encontrar um caminho* $P_{v1,vk}$ *que satisfaz a expressão (2), e adicionalmente minimizando algum critério de tamanho, tal que* $l(P_{v1,vk}) \leq l(P'_{v1,vk})$ *para todos os caminhos* $P'_{v1,vk}$.

2.2 ALGORITMOS CLÁSSICOS DE MENOR CAMINHO

Nesta seção são apresentadas definições de problemas e algoritmos clássicos que tratam o problema de menor caminho.

Não é conveniente introduzir o conceito de menor caminho, sem antes citar este algoritmo de *Dijkstra*. Este algoritmo e outros podem retornar uma árvore de menor caminho a partir de um vértice origem. A descrição destes algoritmos clássicos e de suas execuções pode ser encontrada em GALLO (1988) e avaliações de desempenho em CHERKASSKY (1996). Muitas variações destes algoritmos são propostas na literatura, principalmente baseadas em diferentes propostas para uma fila de prioridade CHERKASSKY (1999).

Antes de explorar alguns algoritmos de menor caminho, uma propriedade importante primeiramente é apresentada. Esta propriedade trata do menor caminho em uma dimensão. Abaixo é apresentada a definição e a prova do caminho mais curto com uma dimensão encontrada em KUIPERS (2004).

Teorema 2.1 – Os subcaminhos dos caminhos mais curtos em uma dimensão também são menores caminhos. $p_{v1,vk} = (v_1, v_2, \ldots, v_k)$

Prova: Assumindo que *P* é o menor caminho de v_1 até v_k. Seja *Q* um subcaminho de *P* do nó $u \in P$ até $v \in P$. Uma vez que *Q* é o menor caminho de *u* até *v*, é possível obter um caminho $P' = v_1, \ldots, Q', \ldots v_k$ que seja tão curto quanto o caminho $P = v_1, \ldots, Q, \ldots, v_k$ de tamanho:

$$l(P) = \sum_{(u,v)\in P} w(u,v) = w(P_{v_1,u}) + w(Q) + w(P_{v,v_k}) > w(P_{v_1,u}) + w(Q') + w(P_{v,v_k}) = l(P'),$$

o que é uma contradição devido *P* ser o caminho mais curto de v_1 *até* v_k.

Segundo RODRIGUES (1999) quando se pretende resolver um problema

através de algoritmos de menor caminho, dois pontos devem ser considerados:

a) Tipo de grafo utilizado;
b) Número de vértices origem e o número de vértices destino.

Quanto ao tipo de grafo utilizado, deve ser observado se o grafo é valorado ou não, se possui arestas com pesos positivos e/ou negativos e se é cíclico ou acíclico.

Quanto ao número de vértices origem e número de vértices destino, o problema de menor caminho possui três classificações:

a) Um para um: dado um grafo *G*=(*V*,*E*) e os vértices *u* e *v* em *V*, encontrar o menor caminho entre *u* e *v*.
b) Um para todos: dado um vértice origem *u*, encontrar o menor caminho de *u* para todos os vértices que são alcançáveis por *u*. Também conhecido como problema do fecho transitivo parcial por SHENKHAR (2003).
c) Todos para todos: dado um grafo *G=(V,E)* ache o menor caminho entre todos os pares de vértices *u, v* em V.

Conforme definido por SHENKHAR (2003), os problemas de menor caminho acima classificados com as letras *b* e *c*, também são respectivamente conhecidos como problema do fecho transitivo parcial e problema do fecho transitivo.

2.2.1 DIJKSTRA

Edsger Wybe Dijkstra é muito considerado por ser um dos pioneiros no roteamento do menor caminho. Seu algoritmo guloso introduzido em 1959, conhecido simplesmente como o algoritmo de Dijkstra, teve um impacto enorme em diversos campos. Resolveu em tempo polinomial, o problema do menor caminho entre um vértice dado como origem, e todos os outros vértices como destino para os grafos direcionados que não fossem valorados negativamente.

O algoritmo de Dijkstra usa a atualização de distâncias apresentado no Teorema 2.1. Este relaxamento considera o fato de que os subcaminhos dos menores caminhos também são menores caminhos.

Segundo AHUJA (1993), o algoritmo de Dijkstra rotula cada vértice *i* pertencente a *V* com um valor de distância *d*(*i*), o qual é o limite superior da distância

do menor caminho para este vértice *i.*

Para cada passo intermediário, o algoritmo classifica os vértices em dois grupos: “vértices temporariamente rotulados” e “vértices permanentemente rotulados”. Abaixo é apresentada uma busca em um grafo, ilustrando este conceito e descrevendo o funcionamento do algoritmo.

O algoritmo de Dijkstra constrói uma árvore de menor caminho, cuja raiz é o vértice inicial *s* e cuja ramificação é o caminho mais curto de *s* para todos os outros vértices em *G*. O algoritmo admite que não existam pesos negativos no grafo.

Fundamentalmente, o algoritmo de Dijkstra funciona através da seleção repetida de um vértice, explorando as arestas incidentes nele, para determinar se o caminho mais curto para cada vértice pode ser melhorado. O algoritmo se parece com uma busca por amplitude porque explora todas as arestas incidentes em um vértice antes de se mover para um nível mais profundo no grafo. Para computar o menor caminho entre *s* e todos os outros vértices, o algoritmo de Dijkstra exige uma rotulação e, também, que a estimativa de menor caminho seja mantida para cada vértice. Normalmente as estimativas de menor caminho são representadas pela variável *d*.

Para ilustrar e facilitar o entendimento, da Figura 2.1 à Figura 2.6 são apresentadas as etapas da execução do algoritmo. Inicialmente todos os nós apresentados nas figuras não estão preenchidos por não estarem rotulados. Acima de cada nó é apresentado a distância e o vértice que permite chegar à raiz.

Todas as estimativas de menor caminho são definidas para ∞, que representa um valor arbitrariamente alto. A estimativa do menor caminho do vértice inicial recebe o valor 0. A Figura 2.1 apresenta o estado inicial da computação.

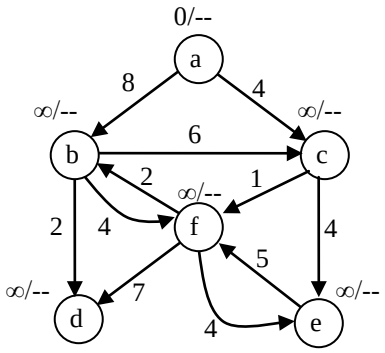


FIG.2.1: Estado inicial do grafo.

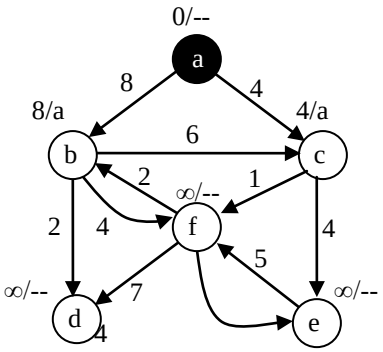


FIG.2.2: Estado do grafo após rotular o nó *a*.

À medida que o algoritmo progride, é definido o vértice de menor distância como pai para todos, exceto para o vértice inicial da árvore de menor caminho. O pai de um vértice pode mudar várias vezes antes que o algoritmo termine.

O algoritmo de Dijkstra continua da seguinte maneira: primeiro, dentre todos os vértices não rotulados do grafo, seleciona o vértice *u* com a menor estimativa de menor caminho. O vértice *a* será o inicial, uma vez que a sua estimativa de menor caminho é 0. Este vértice rotulado está pintado de preto na Figura 2.2. A seguir, para cada vértice não rotulado *v* adjacente ao nó *u*, é verificado o peso da aresta (*u*,*v*). Ao verificar o peso de uma aresta, é decidido se prosseguir por *u* melhora o menor caminho computado até agora para *v*. Para tomar tal decisão, é adicionado o peso de (*u*,*v*) para a estimativa de menor caminho para *u*. Se este valor for inferior ou igual à estimativa de menor caminho para *v*, este valor é designado para *v* como sua nova estimativa de menor caminho e o pai de *v* para *u* é definido. O processo é repetido, até que todos os vértices estejam rotulados. Uma vez computada a árvore de menor caminho do vértice inicial *s* ao final *t*, começando em *t* na árvore e seguindo seus respectivos pais sucessivamente, *s* é *alcançado*. O caminho inverso percorrido será o menor caminho de *s* para *t*.

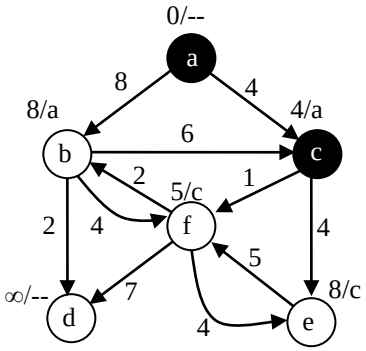


FIG.2.3: Estado do grafo após rotular o nó *c*.

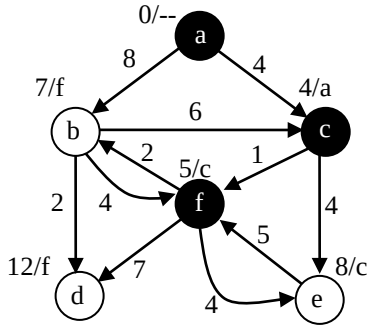


FIG.2.4: Estado do grafo após rotular o nó *f*.

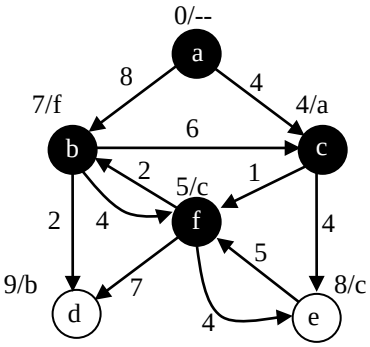


FIG.2.5: Estado do grafo após rotular o nó *b*.

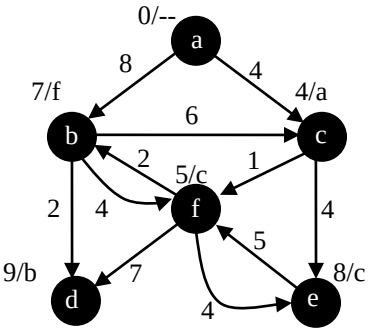


FIG.2.6: Estado do grafo após rotular todos os nós.

As Figuras 2.3 a 2.6 ilustram o restante da computação do menor caminho entre *a* e todos os outros vértices do grafo. Observe que o menor caminho de *a* até *b*, por exemplo, é (*a, c, f, b*) com distância 7.

Segundo AHUJA (1993), a análise da complexidade do algoritmo de Dijkstra está baseada nos procedimentos de seleção de vértices e atualização de distâncias. A complexidade para a estrutura de dados original do algoritmo de Dijkstra é $O(n^2+m)=O(n^2)$.

No esforço de reduzir a complexidade de tempo do algoritmo, são encontradas na literatura diversas variações na forma de implementação e na estrutura de dados. Conforme verificado em ZHAN(1996), abaixo são vistas algumas destas variações e sua complexidade.

- *Lista duplamente ligada circular* – A fila de prioridades é implementada através de uma lista ligada duplamente circular com cabeça de lista. Cada vértice utiliza quatro campos: *predecessor* (lembra o caminho de *s* a qualquer outro vértice do grafo); *dist* (guarda o custo do caminho de *s* a cada vértice do grafo; *direita* e *esquerda* (ponteiros para o vértice à direita e à esquerda na Lista duplamente ligada circular). A complexidade desta implementação original é $O(n^2)$.

- *d-Heap* – Conforme pode ser verificado em CORMEN (2002), um *heap* é uma fila de prioridades, que pode ser vista como uma árvore. Em um *d-Heap* cada nó tem, no máximo, *d* filhos. A complexidade desta implementação é $O(m.log_d(n))$, onde $d=m/n$.

- *Dial* - A idéia principal do algoritmo *Dial* é manter os vértices com as tentativas das distâncias em *buckets*. Então no *bucket*(*k*) vão estar todos os vértices com tentativas iguais a *k*. Agora, suponha que o *bucket(k)* é o primeiro *bucket* não vazio. Então, cada vértice nesse *bucket* possui a menor tentativa de distância, pois, as distâncias não decrescem. Assim, é possível visitar cada um dos vértices que pertencem a ele. Se for atualizar a tentativa de algum vértice *v*, de d_1 para d_2, move-se o vértice *v* do *bucket*(d_1), para o *bucket*(d_2). E da próxima vez que precisar escolher um

vértice com menor distância tentativa, a visita aos *buckets* é iniciada do *k*+1. A complexidade desta implementação é *O(m+nC), onde C = max(u,v), (u, v)* $\in$ *E.*

- *Radix Heap* - A implementação do *Radix Heap* utiliza a mesma idéia dos *buckets* mas, em lugar de guardar somente vértices com distância tentativa *k* no *k-ésimo bucket*, como no *Dial*, guarda-se no *bucket k*, todos os vértices com distância tentativa dentro de um determinado intervalo. A complexidade desta implementação é *O*(*m*+*n log*(*nC*)).

- *Fibonacci Heap* - Conforme pode ser verificado em CORMEN (2002), *Fibonacci Heap* é uma estrutura que mantém um conjunto de árvores, que mudam dinamicamente ao longo da computação. A complexidade desta implementação é *O*(*m*+*n.log*(*n*)).

2.2.2 FLOYD-WARSHALL

Abaixo é apresentado o algoritmo de Floyd-Warshall, que é um algoritmo da programação dinâmica que encontra os menores caminhos entre todos os nós em um grafo.

Richard Bellman em Bellman (1957) fornece os fundamentos para a programação dinâmica. A programação dinâmica foi projetada como uma ferramenta para solucionar de maneira ótima os problemas de decisão. Os problemas de programação dinâmica são caracterizados por sua estrutura hierárquica.

Conforme os algoritmos de dividir para conquistar, que podem ser encontrados em CORMEN (2002), a programação dinâmica divide em subproblemas e em partes específicas, que são resolvidas e usadas para chegar à solução final do problema. Ao contrário do dividir para conquistar, as partes específicas podem ser utilizadas por outros subproblemas pai. A programação dinâmica se baseia no princípio da otimalidade, que foi definida por Bellman como:

> *Qualquer que seja o estado e a decisão atual, as decisões restantes devem constituir uma política ótima no que se refere ao estado resultante da*

primeira decisão.

Isto significa que uma solução ótima dada a um problema, deve ser baseada em soluções ótimas aos subproblemas deste problema.

A melhor maneira de explicar a programação dinâmica requer a explanação de exemplos.

O algoritmo de Floyd-Warshall proposto em FLOYD (1962) é baseado no teorema de Warshall encontrado em WARSHALL (1962). O algoritmo considera subconjuntos $V(k) = (1, 2, ...,k)$ do conjunto V.

$d^{(k)}(u,v)$ representa o tamanho do menor caminho do nó u até o nó v, dado que os nós intermediários são obtidos do conjunto $V(k)$. Conseqüentemente $d^{(0)}(u, v)$ é o peso da aresta que interliga diretamente o nó u ao nó v, se a aresta existir.

Se os nós intermediários do menor caminho entre u e v estão no conjunto de $V(k-1)$, estes nós também existirão em $V(k)$. Conseqüentemente, se todos nós são usados, então todos os menores caminhos entre todos os pares de nós são encontrados.

Os requisitos da programação dinâmica de uma estrutura hierárquica pode ser representada através da fórmula:

$$d^{(k)}(u,v)=min(d^{(k-1)}(u,v), d^{(k)}(u,k)+d^{(k)}(k,v)).$$

Os dois termos separados por vírgula dentro do operador *min* significam respectivamente: que ou o nó k não é um nó intermediário no menor caminho de u até v, ou o nó k é um nó intermediário no menor caminho de u ao v.

O algoritmo de Floyd-Warshall tem complexidade $O(N^3)$. Conforme descrito em KUIPERS (2004), este é conhecido como o mais eficiente algoritmo para achar o menor caminho na classificação “todos para todos” definida na Seção 2.2.

2.3 ALGORITMOS DE CAMINHO COM MÚLTIPLOS CRITÉRIOS

Conforme verificado em MIEGHEM (2004), muitos algoritmos com objetivo de garantir a qualidade no problema de roteamento podem ser encontrados na literatura, mas somente alguns tratam o problema do caminho com múltiplos critérios

em tempo polinomial e apresentam boa performance. Abaixo são apresentados os principais algoritmos, conceitos e referências para o problema de roteamento com múltiplos critérios.

2.3.1 ALGORITMO DE JAFFE

Em JAFFE (1984), são apresentados dois algoritmos utilizando dois critérios. O primeiro é conhecido como o primeiro algoritmo de tempo pseudo-polinomial para o problema do caminho com múltiplos critérios.

A complexidade no pior caso do primeiro algoritmo apresenta $O\left(N^5 b \log Nb\right)$, onde b é o maior peso do grafo.

Devido à complexidade do primeiro algoritmo, somente no segundo algoritmo, conhecido como algoritmo de *Jaffe*, haverá a proposição de um algoritmo de menor caminho utilizando uma combinação linear dos pesos das arestas.

Determina-se, assim, dois multiplicadores positivos chamados de d_1 e d_2 e aplica-se estes multiplicadores nas arestas do caminho do nó *u* até o *v*. Multiplicando os dois pesos w_1 e w_2 compondo um novo peso $w(u,v)$ que é definida pela expressão $w(u,v) = d_1 \cdot w_1(u,v) + d_2 \cdot w_2(u,v)$.

2.3.2 ALGORITMO IWATA'S FALLBACK

Em IWATA (1996), a heurística *Iwata's Fallback* é apresentada para resolver o problema do caminho com múltiplos critérios em tempo pseudo-polinomial. O algoritmo primeiramente computa o menor caminho baseado em um dos critérios selecionado pelo algoritmo. Posteriormente, a primeira busca verifica se todos os critérios foram obedecidos. Se existir pelo menos um critério não obedecido, o procedimento é repetido com outro critério até que um caminho praticável seja encontrado e todos os critérios sejam atendidos.

Utilizando o *Dijkstra* para realizar a busca e considerando a utilização de um menor caminho por critério, a complexidade no pior caso para o algoritmo é $O(mN.logN+mM)$. Este algoritmo não garante otimização na seleção das arestas para compor o caminho.

2.3.3 A HEURÍSTICA RANDÔMICA

Em KORKMAZ (2001) a heurística randômica é introduzida para o problema do caminho com múltiplos critérios.

Devido às múltiplas execuções ocasionarem o retorno de diferentes caminhos do mesmo nó origem ao nó destino, esta heurística costuma ser evitada. Em alguns cenários, sua utilização pode ser eficiente, tais como a de restringir os caminhos com o maior número de saltos. O algoritmo proposto consiste em duas fases: a primeira de inicialização e a segunda de busca randômica.

Durante a primeira fase, chamada de inicialização, é decidido se existem caminhos possíveis, além de ser computados os menores caminhos entre dois nós, considerando os critérios das arestas e a combinação linear de todas as métricas. Durante esta fase o algoritmo marca e remove caminhos que não atendam aos critérios. No fim desta fase, os limites mínimos de peso para os caminhos entre os nós são conhecidos.

Na segunda fase, chamada de busca randômica, baseado nos limites mínimos de peso obtidos pela primeira fase, o algoritmo deflagra uma busca em largura modificada a partir do nó inicial. Descobre cada nó de possível utilização para o caminho e escolhe aleatoriamente os nós que têm maior probabilidade de alcançar o destino até que uma solução para o problema seja encontrada.

A complexidade do pior caso deste algoritmo é *O(mN.logN+mM).*

2.3.4 A HEURÍSTICA H_MCOP

KORKMAZ (2001) apresenta uma heurística conhecida como H_MCOP. Esta heurística tenta encontrar um caminho inserido nos critérios estabelecidos utilizando uma função de tamanho não linear e o conceito *look-ahead*.

A Heurística utiliza duas execuções modificadas do algoritmo de Dijkstra em duas direções. O algoritmo pode ser interrompido ao deparar-se com um caminho, reduzindo, assim, a sua complexidade.

A complexidade do pior caso deste algoritmo é *O(mN.logN+mM).*

2.3.5 ALGORITMO DE CHEN

Em CHEN (1998) é dada uma proposta para um problema específico, chamado de *problema de roteamento dos critérios atraso e custo.* Conforme o próprio nome diz, este algoritmo utiliza as métricas atraso e custo, que são aditivos e aplicados a uma heurística de tempo polinomial.

Chen propõe reduzir o problema e posteriormente resolvê-lo. Utilizando o algoritmo *Dijkstra* estendido (*Extended Dijkstra's – EDSP*) ou o algoritmo *Bellman-Ford estendido* (*Extended Bellman-Ford – EBF*). Este mesmo algoritmo aplica uma função e as notações exibidas abaixo:

a) Função: *MCP*(*G, s, t, w_1, w_2, c1, c2*)
b) Notação:
   G = Grafo direcionado *G(V,E)*
   s = nó origem
   t = nó destino
   w_i = peso da métrica *i*
   p = caminho (vértice 0, vértice 1, ...,vértice *k*)
   c_i = limite de peso máximo para o critério *i* no caminho *p*
c) Restrição: Utilização de números reais para as funções e pesos.

Se existir um caminho, o algoritmo garante encontrar um caminho *p* no grafo *G(V,E)*, do vértice *s* ao *t*, tal que $w_1(p) \leq c_1$ e $w_2(p) \leq c_2$. Caso se deseje encontrar um caminho *p* que satisfaça ao $w_1(p) \leq c_1$ e $w_2(p) \leq c_2$.

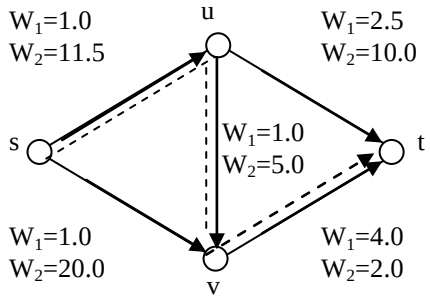


FIG.2.7: O problema original *MCP*(*G,s,t,w_1,w_2,*8.0,20.0) encontra caminhos praticáveis.

Para uma nova função w'_2 onde x é um número inteiro positivo obtemos:

$$w'_2(u,v) = \left\lceil \frac{w_2(u,v).x}{c_2} \right\rceil$$

Chegando a um problema mais simples com a função $MCP(G,s,t,w_1,w'_2,c_1,x)$.

Existindo um caminho p para a solução original, tendo l como tamanho do caminho p e se a expressão $w_2(p) \leq (1 - ((l - 1) / x)).c_2$ for verdadeira, encontra-se uma solução heurística de tempo polinomial.

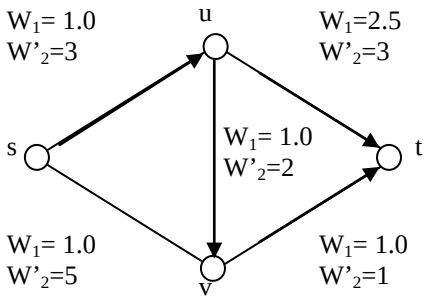


FIG.2.8: O problema reduzido $MCP(G,s,t,w_1,w'_2,8.0,5.0)$ não encontra caminhos praticáveis.

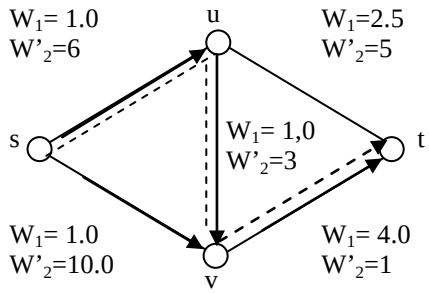


FIG.2.9: O problema reduzido $MCP(G,s,t,w_1,w'_2,8.0,10.0)$ encontra caminhos praticáveis.

Note para os exemplos acima, se o valor de x da função $MCP(G,s,t,w_1,w'_2,c_1,x)$ é aumentado, maior será a chance de se encontrar um caminho. A Figura 2.8 utiliza o x=5 e não encontra um caminho. Na Figura 2.9 com o x=10 o caminho é encontrado.

Associando ao problema original descrito acima e abordando o *problema de roteamento dos critérios atraso e custo*, Chen propõe a função $MCP(G,s,t,atraso,custo,D,C)$ onde as variáveis atraso e custo passam a ser as métricas e as variáveis D e C, os critérios.

Reduzindo o problema, Chen também propõe duas funções mais simples: *MCP*(*G, s, t, atraso, novo-custo, D, x*) e *MCP*(*G, s, t, novo-atraso, custo, x,C*), onde as funções novo-atraso e novo-custo são:

$$novo\text{-}atraso(u,v) = \left\lceil \frac{atraso(u,v).x}{D} \right\rceil \quad e$$

$$novo\text{-}custo(u,v) = \left\lceil \frac{custo(u,v).x}{C} \right\rceil .$$

Finalmente após as reduções mostradas acima, Chen propõe um algoritmo heurístico que pode ser aplicado duas vezes. Na primeira vez aplica *MCP*(*G,s,t,atraso,novo-custo,D,x*). Caso não encontre um caminho praticável, *MCP*(*G,s,t,novo-atraso,custo,x,C*) é aplicado.

Desta forma, quando houver uma solução, o algoritmo garante encontrar um caminho praticável se, pelo menos, uma das duas condições abaixo for satisfeita:

a) $atraso(p) \leq D \wedge custo(p) \leq (1 - ((l-1)/x))$

b) $atraso(p) \leq (1 - ((l-1)/x)).D \wedge custo(p) \leq C$

A complexidade no pior caso para o *Dijkstra estendido* é $O(x^2V^2)$ e para o *Bellman-Ford estendido* é $O(xVE)$.

2.3.6 A HEURÍSTICA *CAMINHO LIMITADO*

Em YUAN(2002) são apresentadas duas heurísticas. A primeira heurística é conhecida como *heurística de granularidade limitada* (*limited granularity*). De menor importância, esta heurística é semelhante ao algoritmo de Chen e possui, no pior caso, complexidade *O(NmM)*.

A segunda heurística é denominada de *heurística de caminho limitado* (*Limited Path Heuristic - LPH*), que possui, no pior caso, complexidade $O(k^2NM)$, onde k é o tamanho da fila em cada nó. Possui uma alta probabilidade de encontrar caminhos praticáveis para o problema do caminho com múltiplos critérios em tempo polinomial. Pelos estudos e recursos computacionais apresentados por Yuan, a *heurística de granularidade limitada* requer muito mais recursos computacionais do que a *heurística de caminho limitado*. A primeira heurística, mantém grandes tabelas em cada nó e consome muito tempo e recurso computacional. Já a *heurística de*

caminho limitado, possibilita a utilização de limites finitos para encontrar um caminho praticável em tempo polinomial.

Esta heurística utiliza o algoritmo de *Bellman-Ford*, o conceito de *k menores caminhos* com algumas alterações e o princípio do caminho não dominado introduzido por HENIG (1985).

2.3.7 O ALGORITMO *A*PRUNE*

Em LIU (2001), um algoritmo exato é proposto para o problema do caminho com múltiplos critérios. Este algoritmo foi baseado no algoritmo de *Jaffe,* introduzido em JAFFE(1984), utilizando sua forma de contabilizar os pesos das arestas do início ao fim de um caminho. Este algoritmo, também utiliza o conceito de *k menores caminhos*, onde é possível obter mais de um caminho que satisfaça os critérios estabelecidos para o algoritmo. Se o algoritmo não atinge a quantidade de *k* menores caminhos pré-estabelecidos, o algoritmo somente retorna os caminhos que atenderam aos critérios estabelecidos.

Durante a execução deste algoritmo é utilizada uma variação do algoritmo de *Dijkstra*, onde o nó escolhido para o menor caminho é extraído do *heap* e todos os nós vizinhos a este nó são verificados. Os nós vizinhos que conduzirem o algoritmo a um *loop* ou a uma violação dos critérios são cortados em tempo de execução.

Este algoritmo utiliza o conceito *look-ahead* e inicia o corte dos nós até que os *k menores caminhos* praticáveis sejam encontrados ou o *heap* se esgote.

A complexidade no pior caso para o algoritmo *A*PRUNE* é *O(N!(m + N + N log N)).*

Com objetivo de obter tempo polinomial é possível a utilização de uma heurística chamada de *Bounded A*Prune,* que permite a redução de complexidade do algoritmo porém, o algoritmo deixa de ser exato.

2.3.8 - ALGORITMOS TAMCRA, SAMCRA E HAMCRA

Os algoritmos *TAMCRA*, *SAMCRA* e *HAMCRA*, são conhecidos como algoritmos com um bom desempenho e apresentam três conceitos em comum, de interesse para o nosso estudo. Em análises experimentais obtidas em MIEGHEN

(2004), verifica-se que estes conceitos são os principais motivadores de análises e pesquisas devido a sua performance. Abaixo, serão apresentados brevemente os algoritmos e conceitos. No Capítulo 4, será descrito detalhadamente cada um destes conceitos.

A heurística *TAMCRA* (*Tunable Accuracy Multiple Constraints Routing Algorithm*) pode ser encontrada em NEVE(1998) e MIEGHEN(2001). Este algoritmo está baseado em três conceitos fundamentais descritos abaixo.

a) *Medida não linear para o tamanho do caminho* - Os menores caminhos até determinada parte do caminho, não necessariamente são os caminhos considerados para a solução final. O conceito considera o menor custo total do caminho.
b) *k menores caminhos* – O algoritmo não termina enquanto não encontrar *k* caminhos praticáveis. Cria uma hierarquia do menor ao maior caminho durante sua execução. A definição deste conceito pode ser encontrada em CHONG (1995).
c) *Caminho não dominado* - Para explicar este princípio, suponha dois caminhos *X* e *Z*, onde *X* é dominado por *Z* quando $w_i(Z) \leq w_i(X)$ para todo i =1...*m.* Este conceito foi introduzido em HENIG (1985).

As principais características do algoritmo *TAMCRA* são:

a) O algoritmo somente considera os caminhos não dominados com objetivo de reduzir a busca.
b) Para implementar a relação de tamanho do caminho, se $l(P) \leq 1$ então temos um caminho praticável, porém não é garantia de que o caminho seja ótimo.
c) Os algoritmos guardam múltiplos caminhos do nó origem até os nós intermediários.
d) Durante a escolha do caminho, todas as métricas aditivas têm a mesma importância e cada caminho está associado a um vetor de *m* dimensões que pode ser expresso por $w_j(P)=\{w_1(P),w_2(P),w_3(P)\ldots w_m(P)\}$.

O algoritmo *SAMCRA* (*Self-Adaptive Multiple Constraints Routing Algorithm*) é uma versão modificada do *TAMCRA*. Este algoritmo é exato e implementa uma versão melhorada de busca visando resolver o problema do caminho ótimo com múltiplos critérios. Se existir um caminho praticável, o algoritmo garante encontrá-lo. Ajusta os caminhos guardados em cada nó conforme a necessidade. Não predefine limite do número de caminhos a serem encontrados podendo levar a um crescimento exponencial.

O algoritmo *SAMCRA* inclui um quarto conceito chamado de *look-ahead*. Este conceito implementa uma pré-análise do caminho em tempo polinomial, antecipando os limites mínimos para cada nó do caminho até o nó que se deseja alcançar. Este algoritmo pode ser verificado em MIEGHEN(2004).

O algoritmo *HAMCRA* (*Hibrid Auguring Multiple Constraints Routing Algorithm*) é apresentado em KUIPERS(2003) e propõe um algoritmo híbrido e exato capaz de integrar a velocidade do *TAMCRA* com a exatidão do *SAMCRA*. Limitando o tamanho de sua fila e utilizando os mesmos três conceitos aplicados ao *TAMCRA*, realiza uma busca bi-direcional, do nó destino ao nó inicial e do nó inicial ao nó destino. O algoritmo *HAMCRA* inicialmente executa o *TAMCRA* permitindo somente uma alternativa de caminho em seus nós intermediários. Desta forma pode ser comparável pelo seu tempo de execução ao *Dijkstra*. Caso esta primeira busca encontre um caminho praticável, o algoritmo pode terminar e apresentar o caminho praticável. Caso esta busca inicial não seja suficiente para encontrar pelo menos um caminho praticável, o algoritmo *SAMCRA* é iniciado a partir do nó origem, fazendo uso de algumas informações já obtidas na primeira busca.

A complexidade no pior caso de *TAMCRA* e *SAMCRA* é $O(kV.log(kV)+k^2zE)$.

2.4 CONCLUSÃO

Conforme pode ser verificado em CHEN (1998), os algoritmos que estudam o problema do caminho com múltiplos critérios, podem ser classificados em dois tipos: não genéricos e genéricos. Abaixo estão descritas as características de cada tipo.

Os algoritmos denominados não genéricos possuem métricas específicas. Estes algoritmos têm seu número de critérios restritos para resolver um problema específico. Estes algoritmos não são capazes de resolver o problema de roteamento

com múltiplos critérios. Neste grupo estão inclusos os algoritmos que visam tratar o problema com critérios restritos (*RSP – Restricted Shortest Path*). O RSP é um sub-problema do caminho ótimo com múltiplos critérios (*MCOP – Multi-Constrained Optimal Path*). Como exemplo, é possível citar os algoritmos que têm o objetivo de obter o menor custo para o critério atraso.

Os algoritmos genéricos são capazes de computar um caminho da origem ao destino baseados em um número de critérios arbitrado.

Na busca de algoritmos genéricos, apesar da existência de algoritmos exatos como o *SAMCRA*, *HAMCRA* e *A*Prune*, é verificado que devido ao tratamento dinâmico dado à fila destes algoritmos para atender o problema MCOP, não é possível limitá-los para obtenção de tempo polinomial.

Em relação a tratar a hierarquia da rua como critério côncavo, como verificado em IWATA (1996), os algoritmos conhecidos para o MCP realizam dois procedimentos distintos. Primeiro filtra o critério côncavo antes da busca e, no segundo procedimento, trabalha com um único critério utilizando algoritmos clássicos.

Na categoria de algoritmo genérico, verifica-se que o antecessor do algoritmo *SAMCRA* poderia ser ajustado para obter o tempo polinomial utilizando critérios aditivos. Chamado de *TAMCRA*, este algoritmo pode ter sua fila e critérios ajustados para obter resultados de rotas rodoviárias em tempo polinomial atendendo às nossas expectativas.

Tendo em vista que as heurísticas *TAMCRA* e *LGS* podem tratar o contexto rodoviário, a proposta deste trabalho é realizar uma análise experimental de ambos os algoritmos, avaliando as vantagens e desvantagens dos conceitos aplicadas a rotas rodoviárias. Para conhecer melhor os algoritmos, estes são descritos nos Capítulos 3 e 4.

3 TAMCRA

3.1 INTRODUÇÃO

A heurística *TAMCRA* (*Tunable Accuracy Multiple Constraints Routing* Algorithm) pode ser encontrada em NEVE (1998) e MIEGHEN (2001). Este algoritmo foi escolhido para nossa análise por ser um algoritmo genérico e ajustável.

3.2 DESCRIÇÃO

O algoritmo *TAMCRA*, como seus sucessores, é baseado em três conceitos fundamentais. Nas Subseções 3.2.1, 3.2.2 e 3.2.3 são apresentados estes três conceitos. Na Subseção 3.2.4, a busca de um caminho praticável utilizando o *TAMCRA* é exemplificada.

3.2.1 MEDIDA NÃO LINEAR PARA O TAMANHO DO CAMINHO

O conceito de medida não linear, diferente dos aplicados em *Dijkstra*, considera que os menores caminhos até determinada parte do caminho, pode não ser necessariamente subcaminhos da solução final. O conceito de medida não linear considera o menor custo total do caminho.

A expressão (1) definida na Seção 2.1, denota a soma dos pesos ao longo de um caminho. A definição do tamanho do caminho $l(P_{v1,vk})$ é fornecida antecipadamente para poder comparar os caminhos, desde que todos os pesos das arestas reflitam diferentes regras com suas unidades específicas. Primeiramente é verificada a escolha direta de um tamanho de caminho linear conforme proposto por JAFFE (1984), onde para todo *j*, d_i são números reais positivos. Substituindo o vetor do peso $\vec{w}(u,v)$ de cada aresta (u,v) no grafo *G(V,E)* pela única métrica $\vec{d}.\vec{w}(u,v)$ de acordo com a expressão (3) a seguir. O problema de *z* critérios é transformado em um único problema permitindo o uso do *Dijkstra*. O algoritmo de *Dijkstra* aplicado ao grafo reduzido retornará um menor caminho *P* que minimiza *l(P)* definido em (3).

$$l(P) = \sum_{i=1}^{z} d_i w_i(P) = \vec{d}.\vec{w}(P) \qquad (3)$$

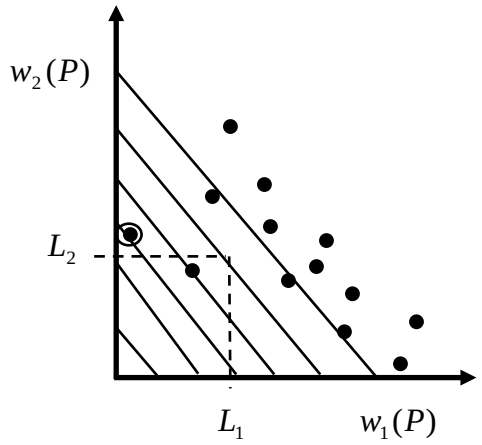


FIG. 3.1- Gráfico de 2 dimensões. Cada caminho *P* entre o nó origem e o nó destino é representado por um ponto (*w₁(P);w₂(P)*) no plano.
Fonte: MIEGHEM(2004)

Na Figura 3.1 é possível observar que as linhas paralelas exibem a equivalência dos nós $d_1w_1(P) + d_2w_2(P) = l$, os quais contêm soluções com comprimento igual. Todas as soluções que estão acima de uma determinada linha têm um comprimento maior do que as linhas abaixo. O menor caminho obtido por *Dijkstra* é aplicado a um grafo reduzido. A interseção das linhas paralelas é a primeira solução (circulada). Neste exemplo, o menor caminho (circulado) encontra-se fora da área dos critérios.

Ainda na Figura 3.1, observa-se a busca da solução com uma linha eqüidistante $l(P)=l$. A área fora da região marcada com as linhas pontilhadas será minimizada se a inclinação das linhas retas satisfizer $(d_1)/(d_2) = (L_2)/(L_1)$. Nas *duas* dimensões, o maior custo possível utilizado para busca da solução deverá obedecer à expressão $w_i(P) \leq L_i$ no plano, onde o somatório dos custos do caminho escolhido não poderá ultrapassar L_i , que é o limite máximo permitido para soma de todas as arestas. A equação desse plano é $\sum_{i=1}^{z} \left(w_i(P)\right)/(L_i) = 1$. Entretanto, a melhor escolha em (3) seria $d_i = (1)/(L_i)$ para todo $1 \leq i \leq z$. Nesse caso, a busca seria realizada com

a metade do volume, antes que a parte externa da solução *l*(*P*) > 1 possa ser selecionada. Apesar da vantagem da utilização de *Dijkstra*, o inconveniente de (3) é que o menor caminho não satisfaz necessariamente a todos os critérios.

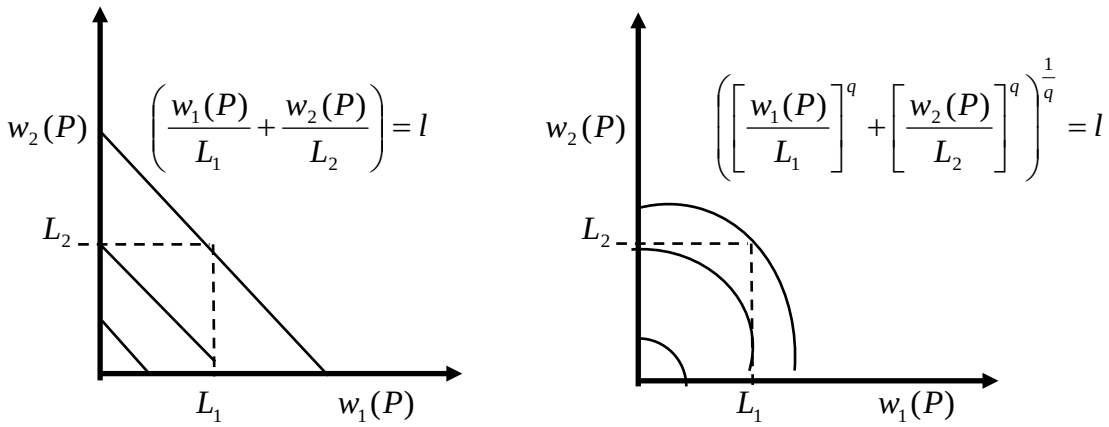
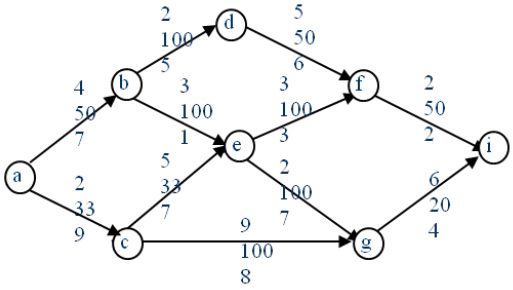


FIG. 3.2. Ilustração da curva das linhas paralelas.

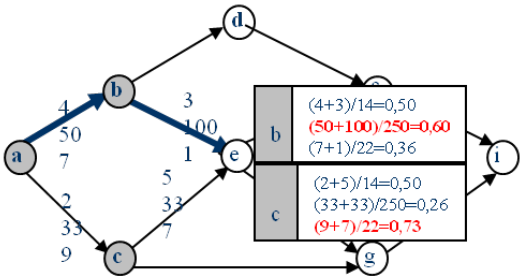
Fonte: MIEGHEM(2004)

Conforme ilustrado na Figura 3.2, as linhas em curva representam melhor a solução. A definição não-linear pode ser encontrada em KORKMAZ (2003). É claro, que o melhor resultado obtido será o limite onde $q \rightarrow \infty$, desde que as linhas eqüidistantes sejam retângulos precisos como os limites. Nesse caso, a definição (4) reduz para o maior vetor dividido pelo critério correspondente.

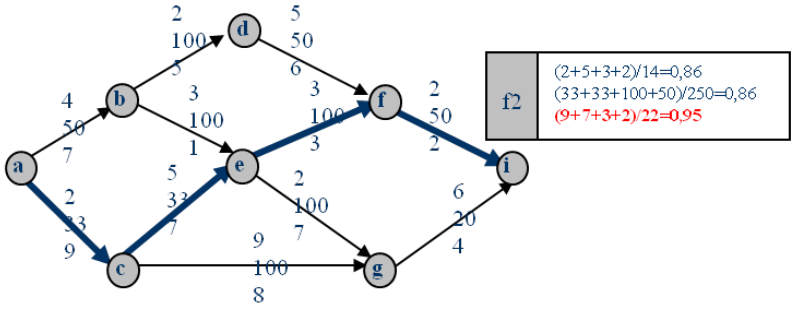
$$L_q(P)=\left(\sum_{i=1}^{z}\left[\frac{w_i(P)}{L_i}\right]^q\right)^{\frac{1}{q}} \qquad (4)$$



(a) Grafo com 3 métricas e 3 critérios (14,250 e 22)



(b) menor caminho do nó *a* até *e*



(c) menor caminho do nó *a* até *i*

FIG.3.3. Exemplo de tamanho não linear.

Observe na Figura 3.3 que os subcaminhos dos menores caminhos não necessariamente são menores caminhos. Em (b), o tamanho *l*(*a,b,e*) = max((4 + 3/14); (50 + 100/250); (7 + 1/22)) = 0,60 é menor que *l(a,c,e)* = max((2+ 5/14); (33 + 33/250); (9 + 7/22)) ≈ 0,73. Entretanto em (c), se for considerado que o nó destino é *i*, o subcaminho (*a ,c, e*) é um subcaminho do menor caminho final.

$$L_{\infty}(P) = \max_{1 \le i \le m} \left[\frac{w_i(P)}{L_i} \right] \quad (5)$$

Se o menor caminho computado com a definição de tamanho (5) for maior do que 1 e violar um dos critérios, nenhum outro caminho satisfará os critérios. Assim, encontrar o menor caminho com a definição (5) resolve o problema de múltiplos critérios. O algoritmo de Dijkstra não garante encontrar o menor caminho com múltiplos critérios, porque confia na propriedade de uma definição linear. Devido às múltiplas dimensões e à utilização de uma definição não linear de tamanho, os subcaminhos dos menores caminhos não necessariamente são os menores caminhos, conforme mostrados na Figura 3.3. A prova desta propriedade e a função não-linear de comprimento podem ser encontradas em MIEGHEM (2001). Possivelmente, será necessário determinar mais de um caminho para cada nó, que nos conduza ao algoritmo de menor caminho (com $k \geq 1$).

Embora a definição (5) seja a escolhida para o *SAMCRA*, a estrutura apresentada aplica-se a qualquer definição de tamanho *l*() o qual obedece aos critérios do vetor com os pesos:

1) $l(\vec{p}) > 0$ para todos os vetores não vazios $\vec{p}$ e $l(\vec{p}) = 0$ somente se $p = 0$ e

2) para todos os vetores, $\vec{p}$ e $\vec{u}$ com $l(\vec{p} + \vec{u}) \le l(\vec{p}) + l(\vec{u})$. Se $\vec{p}$ e $\vec{u}$ não forem negativos (isto é, todos os itens do vetor não sejam negativos), nós temos $l(\vec{p} + \vec{u}) \ge l(\vec{p})$ porque o tamanho de um vetor não pode decrescer no caso de uma adição.

3.2.2 K MENORES CAMINHOS

O algoritmo *k menores caminhos* definido em CHONG (1995) é similar ao algoritmo de Dijkstra. Em vez de armazenar em cada nó intermediário somente o valor do nó precedente e o tamanho do menor caminho da origem até o nó intermediário, este pode armazenar o primeiro menor caminho, o segundo, o terceiro etc. Assim é possível armazenar menos do que *k* caminhos em um nó. No caso em que o valor de *k* não seja restrito a uma quantidade, o algoritmo retorna todos os caminhos possíveis ordenados pelo tamanho, entre a origem até o destino. O valor

de *k* pode ser limitado no *TAMCRA* conforme mostrado em NEVE(2000). Para este algoritmo, há sempre a possibilidade que o menor caminho possa não ser encontrado. Para aumentar a probabilidade do algoritmo de TAMCRA em encontrar os caminhos, é possível ajustar o *k* ou fazer uso do seu sucessor *SAMCRA*.

O algoritmo *SAMCRA* escolhe os caminhos de forma auto-ajustável. Conforme o k_n definido é alocado, automaticamente aumenta o tamanho da fila para cada nó. Com objetivo de definir um limite para a complexidade do algoritmo, para o *SAMCRA* é definido o valor $k_{SAMCRA} = max_{n \epsilon G}(k_n)$ como um valor limite.

O fato de k_n não ser restrito no *SAMCRA*, implica que todos os caminhos possíveis entre a origem e o destino podem ser computados, obtendo, assim, a complexidade NP Completo conforme seu número de entradas que é característica do problema MCP.

3.2.3 CAMINHO NÃO DOMINADO

O terceiro conceito aplicado ao *TAMCRA* utiliza uma terminologia definida em HENIG (1985). Este conceito trata essencialmente de uma técnica de redução de complexidade na busca de caminhos visando aumentar a eficiência computacional.

Para exemplificar a definição de dominância do caminho, a seguir são comparadas as dimensões de dois caminhos (de uma origem a um nó intermediário), onde cada aresta possui um vetor com pesos. Nas Figuras 3.4 e 3.5 são apresentados dois cenários possíveis para a definição de caminhos dominados.

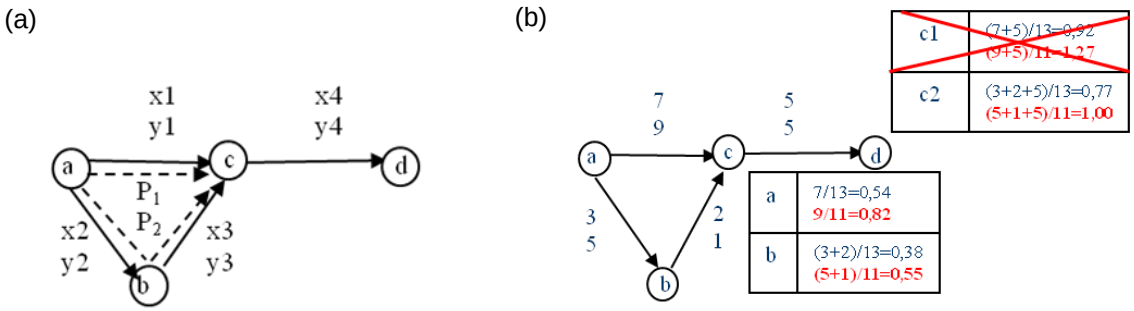


FIG. 3.4. Exemplo de caminho dominado no grafo.

O grafos (a) e (b) da Figura 3.4 apresentam uma associação das variáveis aos pesos em um grafo com 2 critérios (de valores 13 e 11). O grafo (a) apresenta os 2 caminhos, P_1 e P_2. O grafo (b) apresenta o cálculo do caminho em 2 momentos

distintos, quando alcança o vértice *c* e posteriormente o *d*.

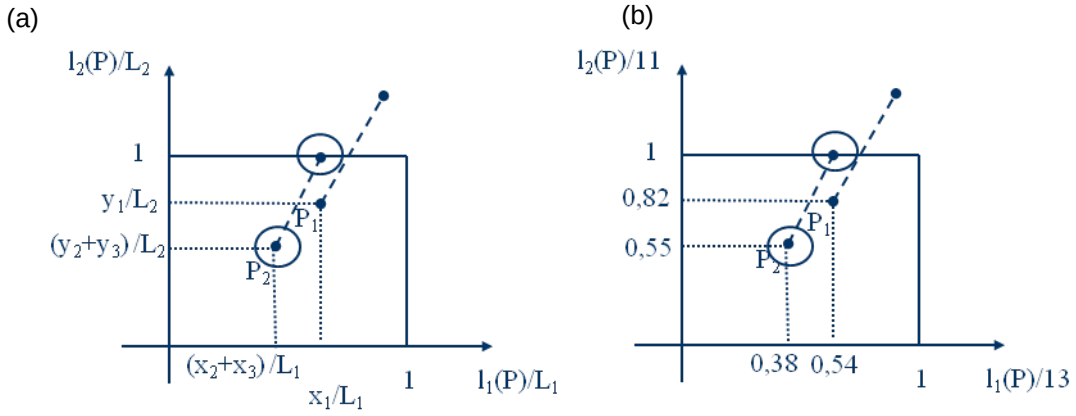


FIG. 3.5. Exemplo de caminho dominado representado no gráfico.

Na Figura 3.5, os quatro pontos em cada gráfico mostram os 2 caminhos P_1 e P_2 em 2 momentos distintos. Ao alcançar o vértice *c* e, posteriormente, o vértice *d*. Observamos que o menor caminho tem seus pontos circulados em dois momentos distintos onde o caminho P_2 é selecionado ambas as vezes.

Na figura 3.5, é possível observar que P_2 é menor do que P_1 e $w_i(P_2) < w_i(P_1)$ para todo $1 \leq i \leq z$. Neste caso, todo caminho (da origem ao nó final) que utiliza o caminho P_2 será menor do que qualquer outro caminho (desta origem a esse destino) que utilize o caminho P_1. Certamente, se para todo *i*, $w_i(P_2) \leq w_i(P_1)$ então, $w_i(P_2) + u_i \leq w_i(P_1) + u_i$ para qualquer u_i. Para todas as definições de tamanho *l()* satisfaz os critérios da definição do vetor (5) apresentada na Subseção 3.2.1. Assim, $l(\vec{w}(P_2)+\vec{u}) \leq 1(\vec{w}(P_1)+\vec{u})$. Conclui-se que P_1 nunca será o caminho mais curto e conseqüentemente P_1 não deve ser incluso na fila, ou seja, P_1 é dito dominado por P_2 se para todo *i*, $w_i(P_2) < w_i(P_1)$.

Por ser um caminho dominado, este caminho seria desconsiderado pelo algoritmo *TAMCRA*. O *TAMCRA* somente considera os caminhos não dominados, que são ilustrados pelas Figuras 3.6 e 3.7.

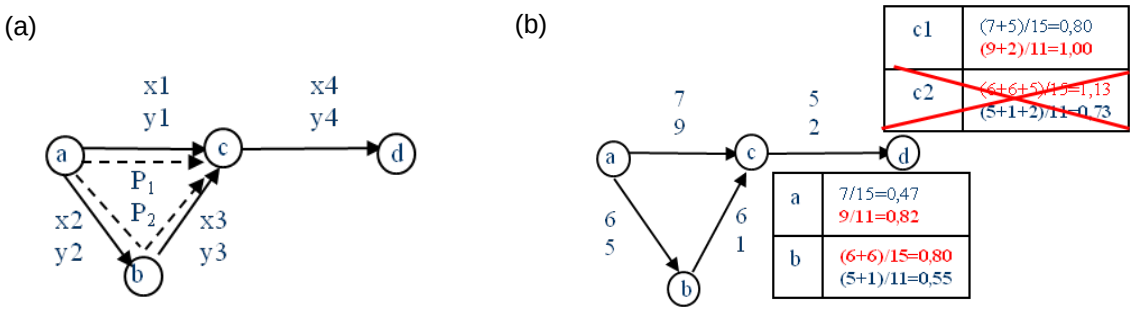


FIG. 3.6. Exemplo de caminho não dominado no grafo.

Da mesma forma, observando os grafos (a) e (b) da Figura 3.6, é apresentada uma associação das variáveis aos pesos em um grafo com 2 critérios (agora com valores 15 e 11). O grafo (a) apresenta os 2 caminhos, P_1 e P_2. O grafo (b) apresenta o cálculo do caminho em 2 momentos distintos, quando alcança o nó *c* e, posteriormente, o nó *d*.

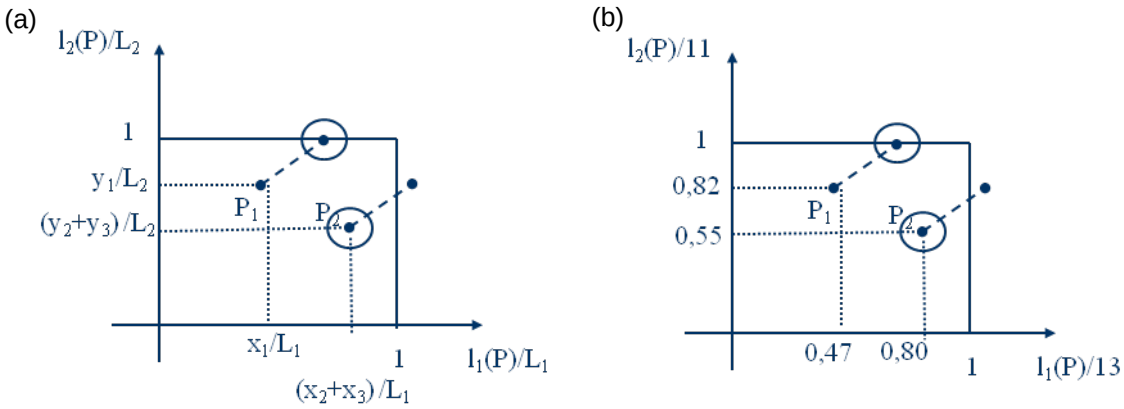


FIG. 3.7. Exemplo de caminho não dominado representado no gráfico.

Na Figura 3.7, os quatro pontos em cada gráfico mostram os 2 caminhos P_1 e P_2 em 2 momentos distintos. Ao alcançar o nó *c* e, posteriormente, o nó *d*. Em cada instante o menor caminho tem seus pontos circulados. Observe que ao alcançar o nó *c,* o menor caminho é P_2; já ao alcançar o nó *d*, o menor caminho se transforma e passa a ser P_1. Outra característica observada para caminhos não dominados é o cruzamento da representação do eixo das ordenadas ou das abscissas quando os dois nós são representados no gráfico.

Na Figura 3.7, os caminhos P_1 e P_2 têm pontos comuns de cruzamento: $w_i(P_2)$

< $w_i(P_1)$ para alguns índices *i*, mas $w_i(P_2) > w_i(P_1)$ pelo menos para um índice *j*. Em tais cenários, o menor caminho (P_2) entre uma origem e o nó intermediário não necessariamente faz parte do menor caminho da origem ao destino da solução final. Observe na Figura 3.7, que adicionando o vetor (x_3,y_3) o caminho é terminado até o destino. A figura ilustra que P_1 é o caminho mais curto no segundo momento. Se dois subcaminhos tiverem valores de cruzamento comuns, todos os caminhos deverão ser colocados na fila.

Alternativamente, dois caminhos não são dominados se $\vec{w}(P_1) - \vec{w}(P_2)$, ou seja, se pelo menos um tiver valor negativo ou zero.

Para facilitar o entendimento dos conceitos acima, a seguir é apresentado um exemplo da busca de caminhos praticáveis para o problema do caminho com múltiplos critérios, abordando o contexto rodoviário.

3.2.4 BUSCA DE CAMINHOS PRATICÁVEIS

O exemplo, nesta seção, apresenta o algoritmo de *TAMCRA* utilizando $k = 2$. Isto quer dizer que esta versão do algoritmo mantém em seus nós intermediários, até dois menores caminhos aplicando o conceito dos *k* menores caminhos já definido neste capítulo. Na busca de caminhos praticáveis, utiliza-se o nó *a* como início da busca até ao nó *i* utilizando 3 critérios.

Para a busca do caminho, este exemplo possui três critérios aditivos: tamanho da rodovia, hierarquia da via e nível de segurança. Os respectivos critérios possuem os valores 14, 250 e 22. A seguir, é explicado cada um dos critérios.

O primeiro critério atribui um peso conforme o comprimento da Rodovia. Os critérios estão expressos em kilômetros e para este exemplo o limite do caminho terá um tamanho máximo de 14 Km.

Para o segundo critério, denominado hierarquia da via, é utilizada uma função que divide 100 pelo número de vias existentes na rodovia. Tendo em vista a utilização de um critério aditivo em lugar de um critério côncavo, utiliza-se a parte inteira do valor dado pela expressão (100/p), onde p é o número de vias da rodovia. Exemplo: (100/1) = 100, (100/2) = 50 , (100/3) = 33 etc.

Em nosso exemplo, o limite para a hierarquia da via será 250, ou seja, o veículo deverá priorizar as rodovias com o menor valor da métrica hierarquia da via.

O menor valor da métrica hierarquia da via, significa que a rodovia possui maior número de pistas. Este veículo deverá trafegar em rodovias praticáveis, de forma que o resultado final da soma desta métrica não ultrapasse 250.

Tipos de Rodovia	(p) Número de Pistas na rodovia	(100/p) Hierarquia da via
Rua	1	100
Avenida	2	50
Via expressa	3	33
BR	4	25

TAB 3.1: Tabela que demonstra o efeito do peso conforme o número de pistas.

O terceiro critério é conhecido como nível de segurança. Utiliza um peso pré-determinado conforme o nível de segurança da rodovia. Para este critério, os pesos são expressos de 1 a 9, onde 1 é uma rodovia totalmente segura e 9 é uma rodovia sem nenhuma segurança. Para aplicação no mundo real, as ruas poderiam receber a classificação da tabela abaixo, conforme a quantidade e natureza das ocorrências policiais registradas nas delegacias. O caminho final gerado deverá considerar que o valor máximo a ser acumulado para este critério será 22.

Descrição do Nível de Segurança	**Nível de segurança**
Inexistente	9
Insuficiente	8
Fraca	7
Irregular	6
Regular	5
Boa	4
Muito Boa	3
Excelente	2
Totalmente Segura	1

Tabela 3.2: Faixa utilizada para determinar o Nível de segurança da rodovia.

Tendo em vista que já foram definidos as métricas e os critérios para a busca de uma rota rodoviária com múltiplos critérios, verifica-se que a escolha do caminho

pode variar em função da região e horários distintos. O algoritmo poderá permitir que um usuário opte por um caminho com maior segurança do que um caminho mais curto. Neste caso, rodovias com menor número de pistas e grande incidência de roubos podem ser evitadas. A utilização destes critérios é um exemplo de que é possível considerar aspectos comportamentais dos seres humanos, encontrando um caminho praticável na malha rodoviária.

As Figuras 3.8 a 3.19 ilustram a busca de um caminho praticável. Cada figura apresentará dois grafos (a) e (b).

No grafo (a) ao lado esquerdo de cada figura, as linhas pontilhadas representam o estado atual de busca. Os retângulos ao lado de cada nó destino, apresentam o nó origem (à esquerda) e os cálculos dos 3 critérios (à direita). Durante a execução do algoritmo, conforme é realizada a busca dos vizinhos, a parte interior esquerda do retângulo é pintada de cinza. Na parte interior direita do retângulo, o maior resultado dentre os 3 critérios calculados é marcado em vermelho e será utilizado pelo algoritmo.

O grafo (b) ao lado direito de cada figura, fornece a visão de todo grafo. Este grafo mostra as arestas que estão sendo analisadas e aumenta a espessura das arestas do menor caminho praticável. Conforme o término da busca dos nós vizinhos, os nós origem também são pintados de cinza.

Conforme mostrado na Figura 3.8, a partir do nó origem *a* é iniciada a busca para cada nó vizinho. Para cada nó vizinho, as 3 métricas são calculadas, onde o maior valor das 3 métricas será utilizado para comparação com outros nós vizinhos. O maior valor de cada nó vizinho não pode ultrapassar 1. Caso o valor ultrapasse 1, significaria informar que aquele caminho ultrapassa, também o critério pré-definido do algoritmo antes de alcançar o nó destino. Caso seja um caminho não dominado e não seja maior do que 1, o nó vizinho armazena o nó precedente e as *z* métricas (neste caso *z* = 3) para futura utilização.

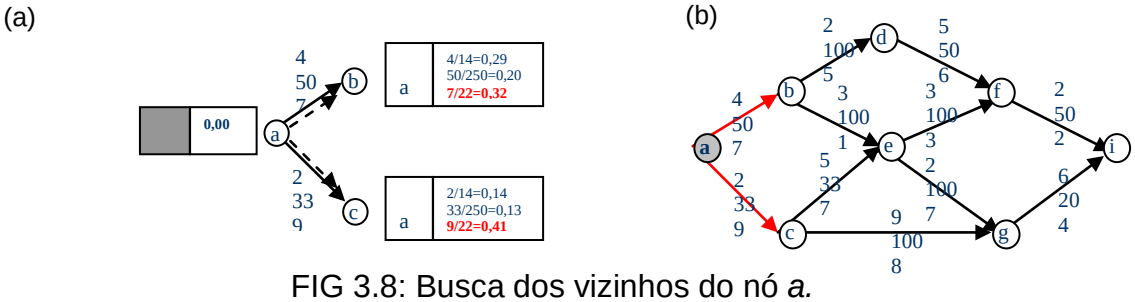


FIG 3.8: Busca dos vizinhos do nó *a.*

Semelhante ao algoritmo de Dijkstra, dentre os nós *b* e *c* da Figura 3.9, o *TAMCRA* escolhe o menor caminho na fila (nó *b*=0,32) e verifica seus vizinhos. Vale a pena lembrar que os algoritmos tradicionais que buscam o caminho mais curto utilizam uma única métrica, ou seja, se fosse do *Dijkstra*, somente poderia armazenar o único maior valor dentre os três critérios calculados.

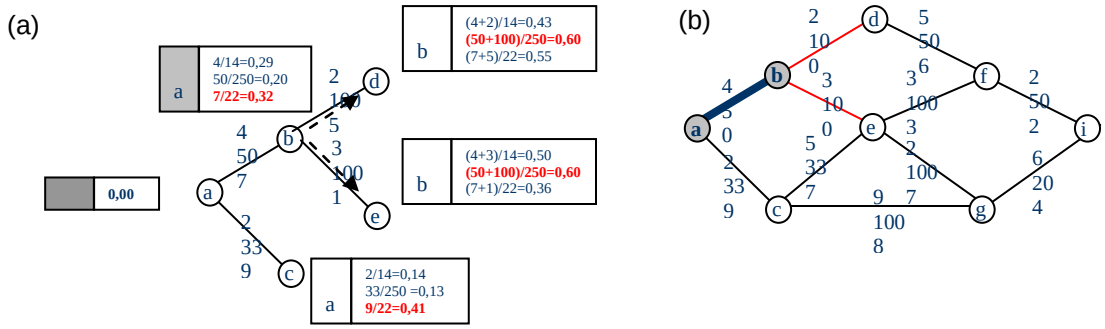


FIG. 3.9 : Busca dos vizinhos do nó *b*.

O próximo menor caminho na fila é o nó *c* com o valor de 0,41. Observe na Figura 3.10, que o nó *e* guarda 2 caminhos praticáveis diferentes. Trata-se da aplicação do conceito *k* menores caminhos. (Neste caso é permitido armazená-los, pois para este exemplo está sendo utilizado o *k* = 2)

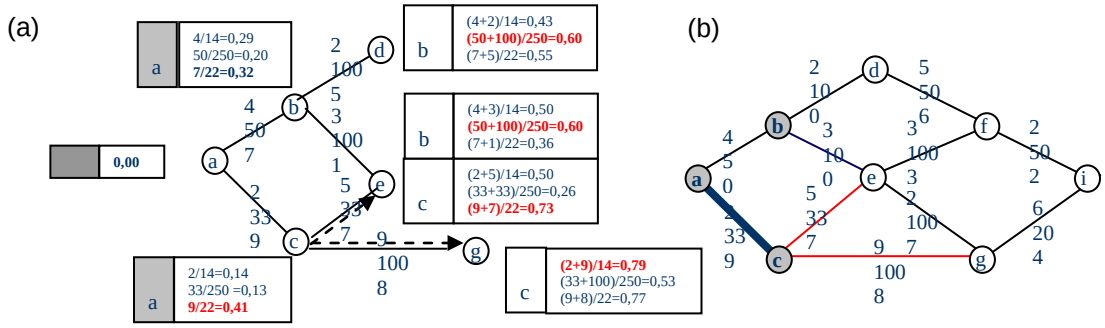


FIG. 3.10 : Busca dos vizinhos do nó *c.*

Na Figura 3.11, o algoritmo seleciona o primeiro menor caminho na fila, o nó *d* com o valor de 0,60 e verifica os vizinhos deste nó.

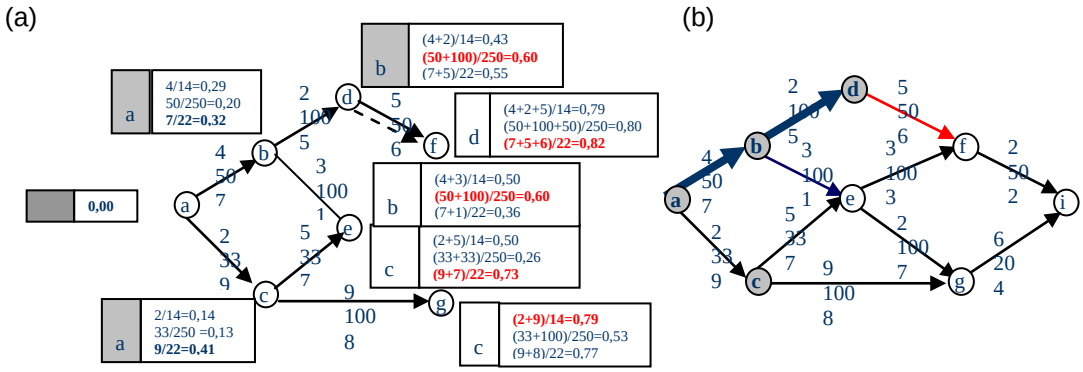


FIG. 3.11 : Busca dos vizinhos do nó *d.*

Na Figura 3.12, é selecionado o próximo menor caminho, neste caso em que o nó *e* possui dois valores, trabalha-se primeiro com o menor elemento do nó *e* que possui o valor de 0,60 e verifica os vizinhos. Observe que os valores a serem armazenados no nó *f* e *g* são descartados. Ainda não atingiram o nó destino e já obtiveram o limite determinado para o segundo critério.

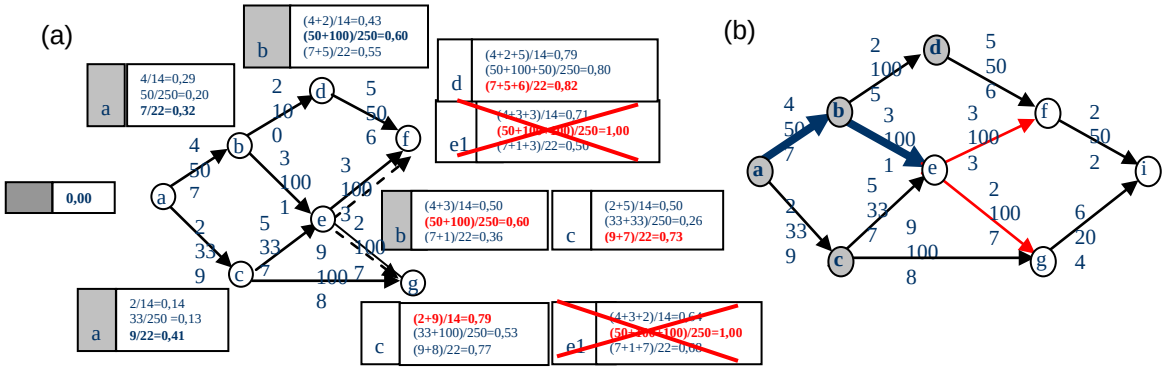


FIG. 3.12 : Busca dos vizinhos do 1º elemento do nó *e*.

Na Figura 3.13, o algoritmo seleciona o segundo menor elemento do nó *e* (0,73) e verifica os vizinhos com os novos valores. Observe que o caminho pelo nó *g* ultrapassa os limites do 3º critério. Este caminho é desconsiderado pelo algoritmo.

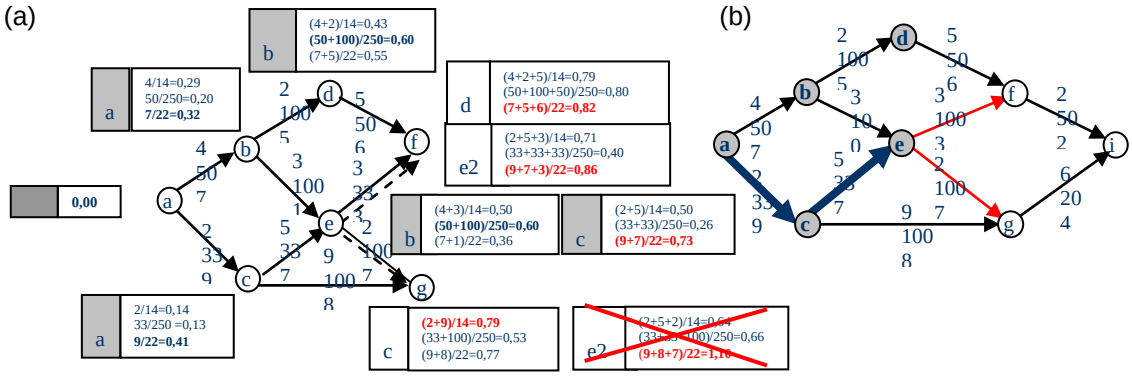


FIG. 3.13 : Busca dos vizinhos do 2º elemento do nó *e*.

Na Figura 3.14 o algoritmo seleciona o próximo menor caminho na fila. Através do nó *g* (0,79) busca seus vizinhos e encontra o nó destino *i*. O caminho através do nó *g* é descartado por não ser um caminho praticável (1º critério > 1).

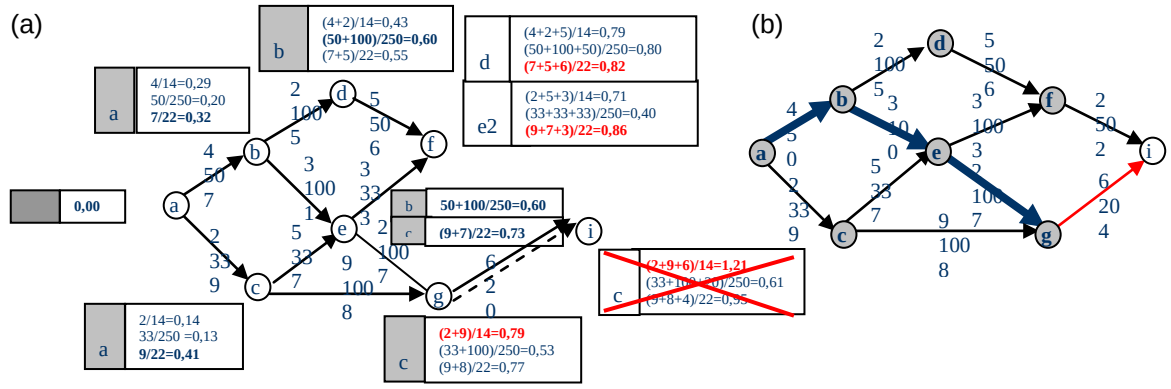


FIG. 3.14 : Busca dos vizinhos do do nó *g*.

Na Figura 3.15, o algoritmo seleciona o próximo menor caminho que é o 1º elemento do nó *f* (0,82) e na busca de seus vizinhos encontra o primeiro caminho praticável até o nó destino *i*.

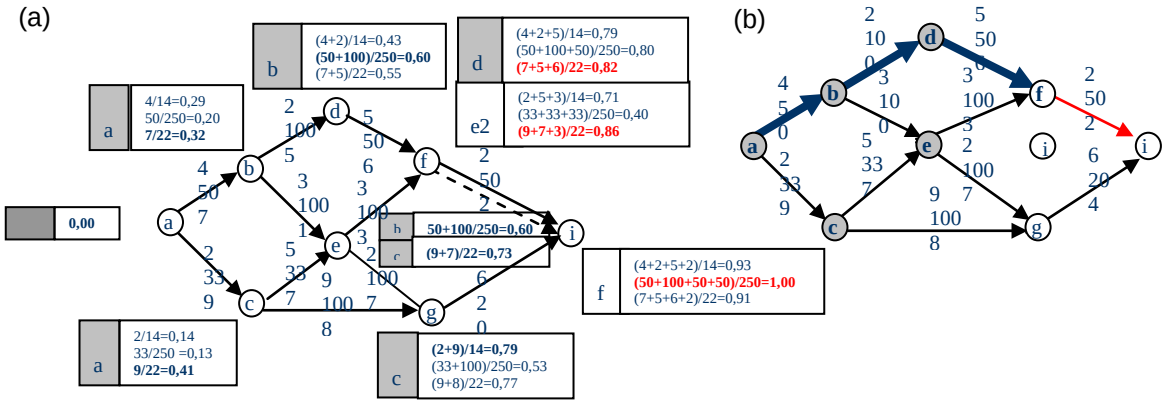


FIG. 3.15 : Busca dos vizinhos do 1º elemento do nó *f*.

Na figura 3.16, o segundo elemento do nó f é selecionado (0,86) e encontra um segundo caminho praticável até o nó destino i. Observe que este caminho é menor do que o primeiro.

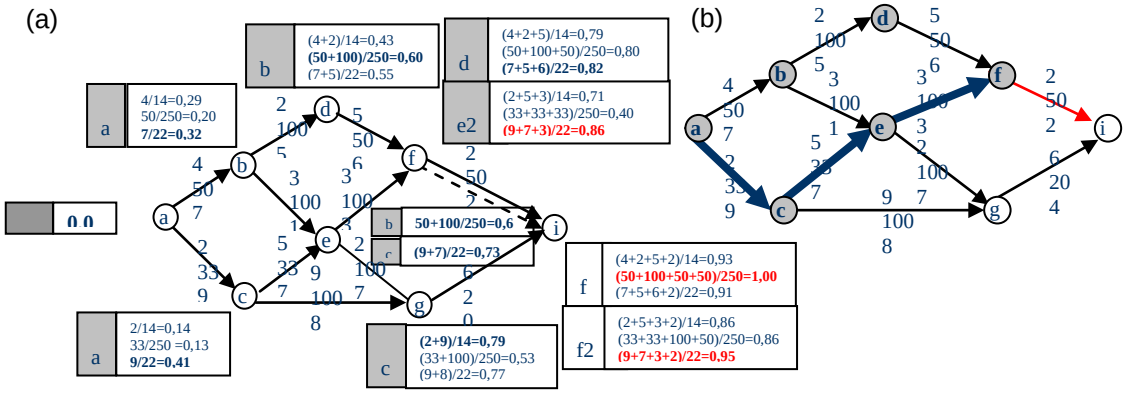


FIG. 3.16 : Busca dos vizinhos do 2º. elemento do nó *f*.

Finalmente na Figura 3.17, o algoritmo seleciona o próximo menor caminho, que é a segunda entrada do nó destino *i* (0,95). O algoritmo verifica que alcançou o nó destino e finaliza apresentando o menor caminho praticável encontrado com *k* = 2.

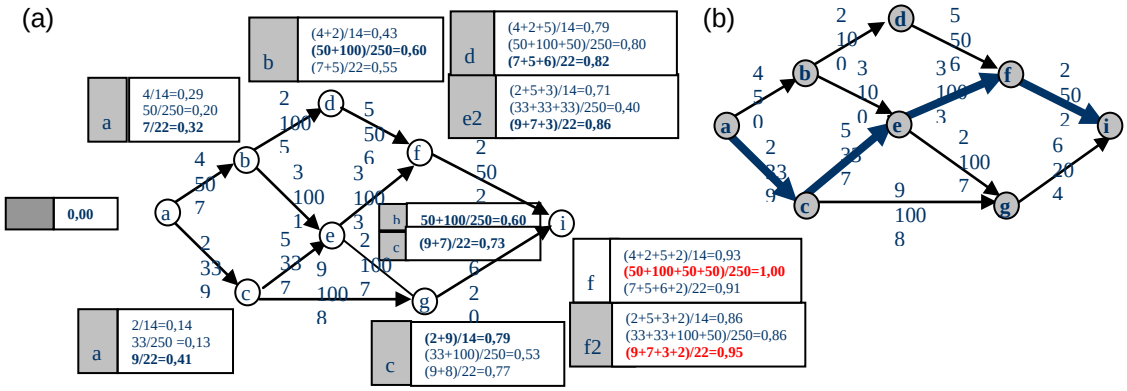


FIG. 3.17 : Encontra o nó destino *i*.

Conforme verificado no exemplo acima, o menor caminho praticável global foi encontrado. Seguindo a indicação do ponteiro do nó antecessor através de (*i,f,e,c,a*) é verificado o menor caminho. Desta maneira, o problema de encontrar um caminho utilizando múltiplos critérios é encontrado com *k* = 2.

Para efeitos de comparação, se for realizada a busca do menor caminho com o *Dijkstra*, ou seja, se considerar as definições apresentadas na Seção 2.2, onde os subcaminhos dos caminhos mais curtos também são menores caminhos, não estaríamos encontrando o menor caminho praticável para o problema de múltiplos critérios.

Na Figura 3.18 é apresentado o resultado do menor caminho praticável (0,95) utilizando o algoritmo *TAMCRA* com *k* = 2. Na Figura 3.19 é apresentado o resultado da aplicação do menor caminho utilizando múltiplos critérios com o *Dijkstra* (1,00).

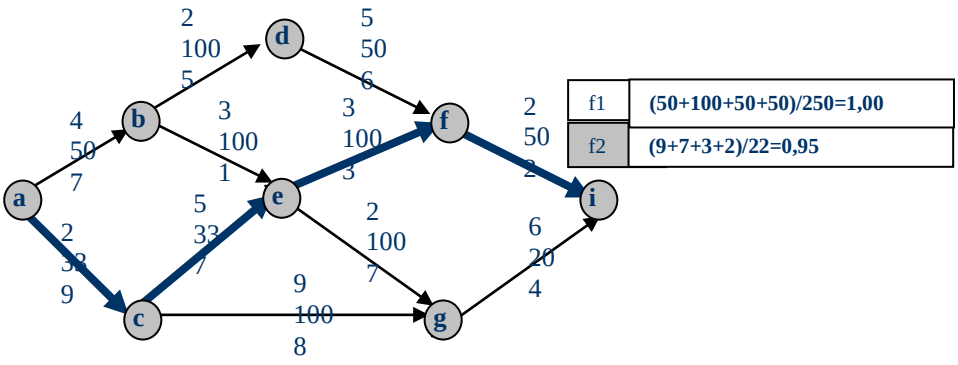


FIG. 3.18: Menor caminho encontrado pelo TAMCRA com *k* = 2.

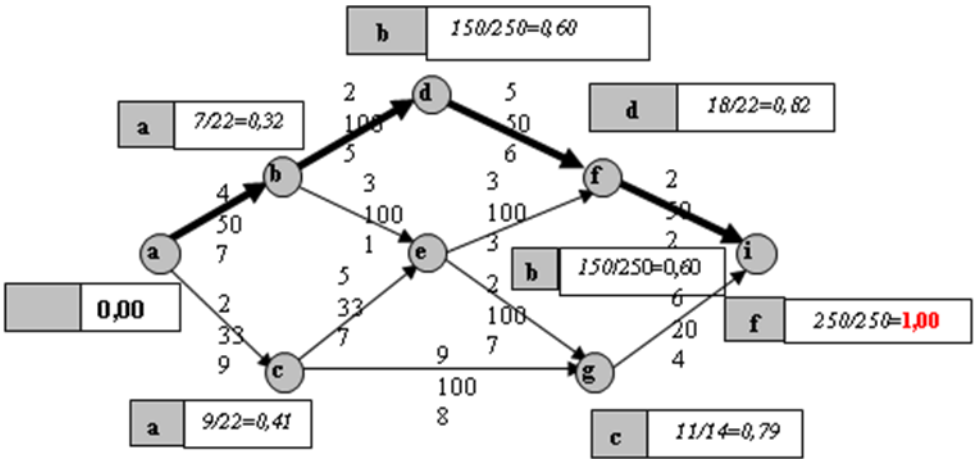


FIG. 3.19: Menor caminho encontrado com o *Dijkstra*.

Como pode ser verificado em MIEGHEM (2001), através dos resultados apresentados acima, é demonstrado como o *TAMCRA* pode otimizar a busca para o MCP. Este algoritmo possui um valor inteiro ajustável *k* que representa o número de caminhos considerados na lista de cada nó. Esta lista armazena os *k* menores caminhos a serem considerados na seleção global. O aumento do valor de *k*, incrementa a probabilidade do algoritmo obter o caminho ótimo para o MCP, porém também nos conduz ao aumento de complexidade do algoritmo.

3.3 ANÁLISE DE COMPLEXIDADE

Com objetivo de analisar a complexidade do algoritmo *TAMCRA*, abaixo é apresentado o algoritmo obtido em KUIPERS (2000). Na sequência, são descritas as rotinas e os conceitos explorados explicando a composição da complexidade do algoritmo.

```
1  counter = 0 for all nodes
2  endvalue = 1.0
3  s = source node, d = destination node
4  path(s[1]) = NULL and length(s[1]) = 0
5  s[1] = marked
6  put s[1] in queue
7  while(queue ≠ empty)
8     EXTRACT_MIN(queue) -> u[i]
9     u[i] = marked
10    if(u = d)
11          STOP
12    else
13          for each v ≠ adjacency_list(u)
14                if((v ≠marked) and (v ≠ previous node of u[i]))
15                      PATH = path(u[i]) + (u,v)
16                      LENGTH = length(PATH)
17                      check if PATH is non-dominated
18                      if(LENGTH < endvalue and non-dominated)
19                            if(counter(v) < k)
20                                  counter(v) = counter(v) + 1
21                                  j = counter(v)
22                                  path(v[j]) = PATH
23                                  length(v[j]) = LENGTH
24                                  put v[j] in queue
25                            else
26                                  max_length(v) -> v[j]
27                                  if(LENGTH < length(v[j])
28                                        path(v[j]) = PATH
29                                        length(v[j]) = LENGTH
30                                        put v[j] in queue
31                            if(v = d)
32                            endvalue = LENGTH
```

Algoritmo 3.1 – Algoritmo *TAMCRA*.

Fonte: KUIPERS (2000)

Para a execução do *TAMCRA*, o algoritmo tem como entrada cinco informações básicas: o Grafo, o nó inicial, o nó destino, os critérios e o tamanho da fila. Como saída, o algoritmo fornece o menor caminho praticável para o problema MCP.

Conforme observado no Algoritmo 3.1, da linha 1 a 5 são inicializadas variáveis e vetores do algoritmo. Dentre estes vetores, na linha 1 é inicializada a variável *counter* para todos os nós do grafo. Esta variável é a responsável em contabilizar a entrada na fila de cada nó. A quantidade desta variável é analisada na linha 19 e não pode ultrapassar o valor de *k*.

Para inicializar o algoritmo, o nó origem é inserido na fila através da linha 6.

Considerando o grafo *G(V,E),* devido à utilização dos *k* menores caminhos, nunca será obtido mais do que *kV* caminhos na fila.

Pelo fato da implementação do algoritmo ter utilizado o *heap de Fibonacci,* definido em CORMEN (2002), para a seleção do menor caminho dentre os *kV* caminhos, a complexidade no pior caso será O(*log(kV))*. Tendo em vista o loop do *while* na linha 7, cada nó no máximo poderá ser selecionado *k* vezes na fila, obtendo-se, assim, a complexidade *O*(*kVlog*(*kV*))*.*

Após selecionar o menor tamanho de caminho na fila através da função *EXTRACT_MIN* da linha 8, o elemento extraído é colocado no vetor *u*[*i*] e marcado na linha 9. Se o elemento extraído for igual ao nó destino, o caminho praticável foi encontrado e o algoritmo termina. Caso isto não ocorra, o procedimento vai para o *loop* da linha 13. Este *loop* poderá ser chamado *k* vezes de cada lado das arestas (vizinhos) no grafo. Em cada busca, na linha 15 é calculado o tamanho de cada critério (*z*) obtendo a complexidade no pior caso *O*(*kz*)*.*

Antes da inclusão do elemento na fila, o algoritmo certifica estar atendendo os 3 conceitos: se a entrada compõe um caminho não dominado (linha 17), se o tamanho do caminho atende ao critério (linha 18) e se a fila de menores caminhos do nó ultrapassa o tamanho de *k* (linha 19). Caso todas as condições sejam verdadeiras, é adicionado o nó com complexidade no pior caso de *O*(*1*). Caso o tamanho da fila já tenha o tamanho de *k*, se faz necessário encontrar o maior valor das *k* entradas, obtendo, assim, a complexidade no pior caso de *O*(*k*). Assim é verificada a complexidade *2kE(O(z)+O(kz)+O(k))* = *O*(*kEkz*) para o *loop de for* iniciado na linha 13. A complexidade do pior caso do *TAMCRA* é *O(kV.log(kV)+k*^{2}*zE)*.

Vale a pena ressaltar que a utilização de um único critério (*z* = 1) e a limitação da fila dos nós a uma entrada (*k* = 1), fará com que o algoritmo obtenha a complexidade do *Dijkstra* dada por *O*(*V logV* + *E*)*.*

4 A HEURÍSTICA LGS

4.1 INTRODUÇÃO

Conforme descrito em SHAPIRO (1992), a heurística *LGS* (*Level Graph Search*) é apresentada utilizando grafos em níveis não direcionados. Apesar do *Dijkstra* ser conhecido como algoritmo ótimo para o problema de menor caminho, utilizando grafos em níveis, Shapiro introduz uma proposta de busca em grafos com níveis, que além de reduzir a complexidade da geração de menores caminhos entre dois pontos no grafo, pode ser considerada como um critério adicional para a busca do caminho no grafo.

Quando se propõe a utilização do algoritmo *LGS* ao contexto rodoviário, são consideradas restrições de orientação e otimalidade.

A primeira restrição de orientação está ligada ao fato do algoritmo levar em consideração o uso de um grafo não orientado. Um mapa rodoviário de uma cidade não pode ser abstraído como um grafo não direcionado, e sim como um digrafo, pois nem todas as ruas possuem mão dupla.

A segunda restrição se deve à heurística usada pelo algoritmo *LGS*. Como se trata de uma heurística, o algoritmo pode conduzir à busca para uma rua principal na direção contrária ao destino que se pretende alcançar. Em determinadas situações, o caminho gerado como resultado possui uma rota com um custo determinado pela distância bem maior do que o custo encontrado pelos algoritmos ótimos.

Com objetivo de contornar estas restrições originais do *LGS*, verifica-se em BRAZ(2005) modificações que adequaram melhor o uso do *LGS* no contexto dos mapas rodoviários. A solução modificada conhecida como *LGS**, visa a implementação em um módulo de roteamento em localidades urbanas de um SIG (Sistema de Informações Geográficas) considerando métricas de hierarquia das ruas e segurança. A análise da complexidade de pior caso do *LGS** mostra que as modificações inseridas não alteraram a complexidade original do *LGS*.

4.2 GRAFOS EM NÍVEIS

Com objetivo de analisar algoritmos e segmentar a busca de caminhos em rodovias mais rápidas, em SHAPIRO (1992) é introduzida uma nova estrutura de grafos em níveis, onde é possível segmentar os nós e arestas conforme o nível. Abaixo são apresentadas as definições encontradas.

Definição 4.1 - *Um grafo em nível, definido por G=(V, E, k, f), é uma estrutura que consiste em um conjunto de nós V, um conjunto de arestas E, um inteiro positivo k e uma função f: E → {1,2,3,...,k}.*

Observe que quando $k = 1$ o grafo em nível é um grafo não orientado padrão. A função $f^{-1}(x)$ é denotada por E_i, ou seja, pelo conjunto de arestas de nível i. Como f é uma função, temos que $E = \bigcup_{i=1}^{k} E_i$ e $E_i \cap E_j = 0$ se $i \neq j$.

Entende-se por $V(E_i)$ o conjunto de vértices do conjunto de arestas de nível i. Seguindo o padrão que está sendo usado, $V(H)$ é o conjunto de vértices do grafo H e $E(H)$ o conjunto de arestas. Seja $G_i = (V(E_i); E_i)$ para $1 \leq i \leq k$, temos que G_i será um subgrafo de G, se $V(E_i) \subseteq V$ e $E_i \subseteq E$.

Por $G \cup H$ entende-se que o grafo $(V(G) \cup V(H); E(G) \cup E(H))$. Além disso $G \subseteq H$ significará que G é subgrafo de H, se $G^i = \bigcup_{j=1}^{i} G_i$ então, teremos a cadeia de subgrafos $G^1 \subseteq G^2 \subseteq G^3 \subseteq ... \subseteq G^4$ induzida por f.

Definição 4.2 – *Para qualquer $(u,v) \in E$: seja $l((u,v)) = i$, sempre que $(u,v) \in E_i$, entende-se por i o nível da aresta (u,v). Por conveniência, foi retirado o duplo parêntesis e escreve-se l(u,v).*

O conjunto de níveis das arestas incidentes em um dado vértice, é importante no estudo das propriedades e dos grafos em níveis. Daí temos:

Definição 4.3 - *Seja $v \in V$ define-se o conjunto de níveis de arestas de um vértice v, denotado por els(v), como o conjunto dos níveis de todas as arestas incidentes em v. Formalmente, $els(v) = \{i \mid v \in V(E_i)\}$.*

Pode-se, também, definir o conceito de nível de um vértice v em termos de seu $els(v)$.

Definição 4.4 - *Seja $v \in V$ define-se $l(v) = min \{i \mid i \in els(v)\}$; entende-se por $l(v)$ por nível do vértice v.*

Informalmente o nível de uma dada aresta indica a classe de rua a qual ela pertence, onde E_1 representa o conjunto de ruas de classe 1; E_2 o conjunto de ruas de classe 2 e assim por diante. Por convenção quando for referido a classe da rua, sua preferência está na ordem decrescente, ou seja, ruas de classe 1 serão mais preferidas, ruas de classe 2 serão menos preferidas que as de classe 1 e assim por diante. Por outro lado, o nível do vértice é uma indicação da rua de nível mais baixo que a cidade possui.

O Lema 4.1 indica a conexão entre os níveis dos vértices e a cadeia $G^1 \subseteq G^2 \subseteq G^3 \subseteq ... \subseteq G^4$.

Lema 4.1 - *Seja $v \in V(G)$ e $l(v) = i$, então $v \in V(G^i)$ e $v \notin V(G^j)$ para qualquer $j < i$.*

Existem duas condições que devem ser observadas nos grafos em níveis:

Condição 1. *G_1 é conexo.*

Condição 2. *$\forall v \in V$, se $l(v) \neq 1$, então $\exists u \in V$ tal que $l(v) \in els(u)$, $l(u) < l(v)$ e existe um caminho de v para u em $G^{l(v)}$.*

Informalmente da primeira condição é entendido que o conjunto de arestas que representam as ruas de nível 1 no grafo deve ser conexo.

Da segunda condição é verificado que de qualquer vértice que não possua arestas incidentes de nível mais baixo, existe, pelo menos, um caminho para algum vértice que possua arestas de nível mais baixo. Por exemplo, para um grafo com $k = 3$, de qualquer nó de nível 3 existe pelo menos um caminho para algum nó de nível 2 ou 1 no grafo $G^3 = G_3 \cup G_2 \cup G_1$. De qualquer nó de nível 2 existe, pelo menos, um caminho para um nó de nível 1, no grafo $G_2 = G_2 \cup G_1$.

Da condição 2 e da definição de G^i, temos a seguinte observação:

Observação 1. Seja $v \in V$ com $l(v) = i \neq 1$. Então existe $u \in V(G^{i-1})$ e um caminho de v até u em G^i.

Definição 4.5 - *Seja $v \in V$. Um vértice $w \in V$ é chamado de vértice objetivo de v se $l(v) \in els(w)$ e $l(w) < l(v)$.*

Ou seja o vértice objetivo de um dado vértice é aquele de nível mais baixo, desde que o nível daquele, pertença ao conjunto de níveis deste vértice.

Lema 4.2 - *Seja $v \in V$ e $l(v) \neq 1$. Então existe um vértice $u \in V(G_{l(v)})$ no qual é um vértice objetivo de v e um caminho de v para u em $G_{l(v)}$.*

Teorema 4.1 - *Para qualquer $1 \leq i \leq k$, G^i é um grafo conexo. Em particular $G(V,E) = G^k$ é conexo.*

Deste teorema, concluímos que o grafo *G,* definido por $G = \bigcup_{i=1}^{k} G_i$, onde *k* é o nível mais alto do grafo, é conexo.

A Figura 4.1 ilustra um grafo em níveis. Nesta ilustração é possível observar a densidade das arestas representando os níveis. Esta densidade da aresta nos remete à hierarquia das rodovias introduzidas em SHAPIRO (1992). O grafo abaixo apresenta 3 níveis de hierarquias. Adicionalmente, através de implementação encontrada em BRAZ (2005) foram atribuídos os níveis de segurança, representados pelos números nos centros das arestas. É verificado neste grafo, que além da tradicional utilização de métricas nas arestas ao longo de um caminho, também é possível fazer uso do nível como outra forma de métrica para a busca dos caminhos.

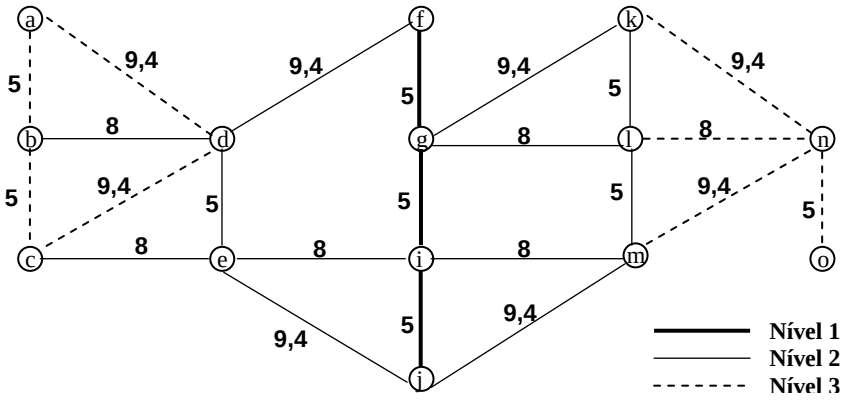


FIG.4.1: Grafo com 3 níveis.
Fonte: BRAZ(2005).

4.3 DESCRIÇÃO

Conforme apresentado em SHAPIRO (1992), o objetivo do algoritmo *LGS* é minimizar o uso de ruas de nível mais alto, isto é, arestas de nível maior que 1. Sejam v_1 e v_2 dois nós com seus respectivos níveis $l(v_1)$ e $l(v_2)$, o algoritmo gera caminhos de v_1 e v_2 para um vértice w tal que $l(w) \leq min\{l(v_1), l(v_2)\}$, ou para vértices de nível 1. Em seguida, o algoritmo irá conectar os dois vértices de nível 1 resultantes de um caminho que possui somente arestas de nível 1, usando para isso o *Dijkstra*.

Para facilitar o entendimento, raciocina-se com um grafo de apenas 2 níveis e tanto a origem como o destino vértices de nível 2. Neste caso, o algoritmo *LGS* funciona resumidamente da seguinte forma:

Dados dois vértices v_1 e v_2, origem e destino respectivamente, onde $l(v_1) = l(v_2) = 2$. A busca de um vértice objetivo w onde $l(w) = 1$, é iniciada pelo vértice v_1, pois $l(v_1) = l(v_2)$. Caso $l(v_1) < l(v_2)$, a busca iniciaria por v_2. O procedimento *level_path* irá gerar um caminho p de v_1 para w, o qual é o menor dos caminhos existentes entre v_2 e seus respectivos vértices objetivos. Em seguida $l(v_2)$ passa a ser igual a $l(w')$, ficando o vértice destino v_2 com nível 1.

Como os dois vértices, origem e destino possuem agora o nível mais baixo do grafo, é intitulado procedimento *shortest_path* entre os dois vértices objetivos remanescentes w e w'.

No final o *LGS* concatena os caminhos encontrados gerando um único caminho.

Conforme verificado acima, o *LGS* faz uso de dois procedimentos: *Level_Path* e o *Shortest_Path*.

O *Level_Path* é um algoritmo de busca em profundidade modificado que toma um nó v como parâmetro de entrada e retorna um nó w como parâmetro de saída e um caminho como valor de saída. Este nó w representa o nó objetivo mais próximo de v e o caminho retornado como valor é o menor caminho de v para w tal que todas as arestas deste caminho pertençam a $G_{l(v)}$.

As variáveis que recebem o caminho como valor são denotadas por *ipath* e *fpath*. A variável *ipath* recebe os caminhos gerados a partir da origem até o nó

objetivo mais próximo e a variável *fpath* recebe os gerados a partir do destino até o nó objetivo mais próximo. Como o algoritmo encontra o caminho do destino até o nó objetivo mais próximo, é necessária a inversão de *fpath*.

Em *Shortest_Path* pode ser utilizado qualquer algoritmo de menor caminho. Como exemplo, o algoritmo de Dijkstra é utilizado para o *LGS* em SHAPIRO (1992) ou também pode ser verificada a utilização do algoritmo A* para o *LGS** em BRAZ (2005). A variável *spath* recebe o caminho gerado pelo algoritmo de menor caminho entre *w* e *w'*, ou seja, entre os dois vértices objetivos de nível 1.

Na variável *path é concatenado* o resultado final do algoritmo *LGS,* recebendo as variáveis *ipath*, *fpath* e *spath*. O símbolo $\oplus$ representa concatenação de caminhos, desta forma *path = ipath* $\oplus$ *spath* $\oplus$ *fpath*

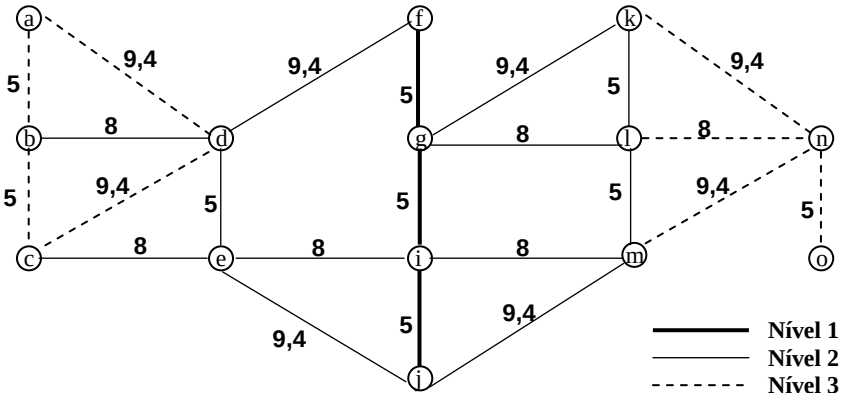


FIG.4.2: Grafo *G* com 3 níveis em momento inicial.
Fonte: BRAZ(2005).

Como exemplo, suponhamos que no grafo *G* da figura acima, os vértices *a* e *o* sejam a origem e destino, respectivamente. O algoritmo *LGS*(*G,a,o*) inicia sempre pelo nó que possui maior nível. Como neste caso *l*(*a*) = *l*(*o*) = 3, então o algoritmo inicia pelo vértice origem *a*. Os nós objetivos de *a* são *b, c, d, k, l, m* e o mais próximo de *a* é o nó *b*. A variável *ipath* irá armazenar a aresta (*a,b*) e o nó *b* passa a ser o nó origem. Como *l*(*b*)< *l*(*o*)*,* então o *LGS* toma o nó destino *o* como nó corrente. Os nós objetivos de *o* são *b, c, d, k, l, m* e o mais próximo de *n* é o nó *l*. A variável *fpath* irá armazenar as arestas *((o,n), (n,l))* e nó *l* passa a ser o nó destino.

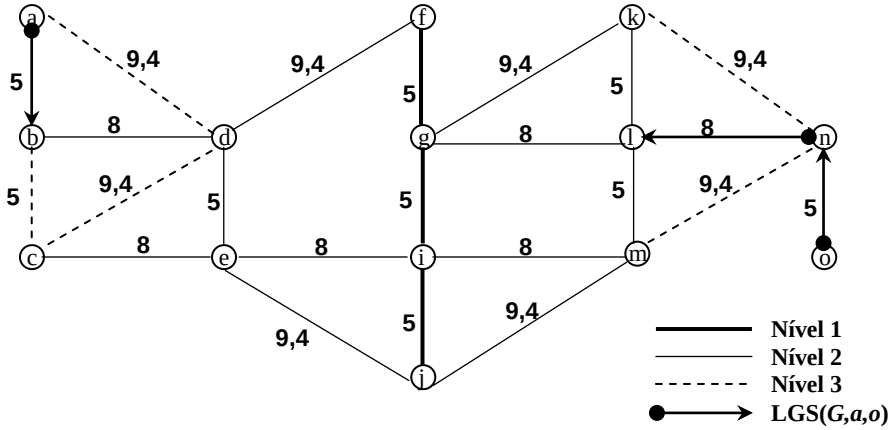


FIG 4.3: Procedimento *Level_Path* ao término do Nível 3.
Fonte: BRAZ(2005).

Novamente como $l(b) = l(l) = 2$, é iniciada a busca a partir do nó *b*, que passou a ser o nó origem. Observe que o procedimento de busca agora é no grafo $G_{l(b)}$. Os nós objetivo de *b* são *f, g, i, j* e o mais próximo de *b* é o nó *f*. A variável *ipath* irá armazenar as arestas ((*b,d*),(*d,f*)), isto é, *ipath* = ((*a,b*),(*b,d*),(*d,f*)) e o nó *f* passa a ser o nó origem. Como $l(f)<l(l)$, então o *LGS* toma o nó destino *l* como nó corrente. Novamente o procedimento de busca agora é o grafo $G_{l(l)}$. Os nós objetivos de *l* são *f, g, i, j* e o mais próximo de *l* é o nó *g*. A variável *fpath* irá armazenar a aresta ((*l,g*)), isto é, *fpath* = ((*o,n*),(*n,l*),(*l,g*)) e nó *g* passa a ser o nó destino.

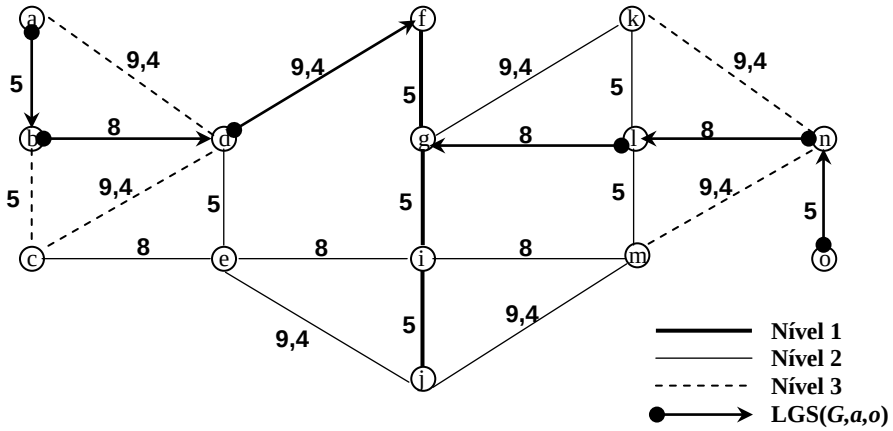


FIG.4.4: Procedimento *Level_Path* ao término do Nível 2.
Fonte: BRAZ(2005).

Até o momento, o *LGS* somente fez uso do procedimento *Level_Path*. Agora como *l*(*f*) = *l*(*g*) = 1, o algoritmo aplica no grafo G_1 o procedimento *Shortest_Path* realizando uma busca de menor caminho utilizando o *Dijkstra*. Assim o *Shortest_Path* = *Dijkstra*(G_1,*f*,*g*), armazena na variável *spath* o menor caminho entre *f* e *g*. Para o exemplo *spath* = ((*f*,*g*)).

Por fim o algoritmo *LGS* termina, guardando na variável path a concatenação de *ipath*, *spath* e *fpath*, onde *fpath* = ((*g,l*), (*l,n*), (*n,o*)), assim *path* = ((*a,b*), (*b,d*), (*d,f*), (*f,g*), (*g,l*), (*l,n*), (*n,o*)). A Figura 4.5 mostra o caminho gerado pelo *LGS* sobre o grafo.

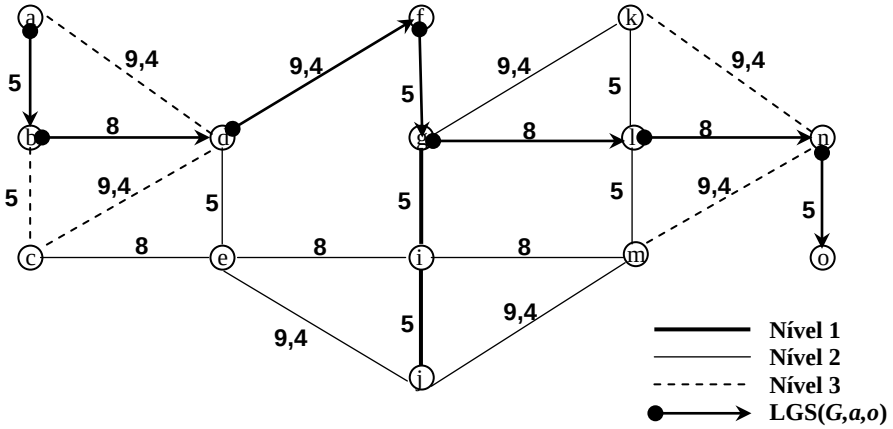


FIG.4.5: Procedimento *Shortest_Path* de *f* ao *g* e concatenação dos caminhos ao término do algoritmo.
Fonte: BRAZ(2005).

4.4 ANÁLISE DE COMPLEXIDADE

Conforme pode ser verificado em SHAPIRO(1992) e BRAZ(2005), a complexidade do algoritmo *LGS* é determinada pela complexidade do procedimento *Level_Path* e da complexidade do procedimento *Shortest_Path*.

Para realizar a análise foi utilizada a treliça estruturada uniforme descrita na Seção 5.2. Esta estrutura além de possuir propriedades que facilitam a análise, se assemelha a distribuição dos níveis de ruas que geralmente existem em um mapa rodoviário urbano.

De posse das propriedades da treliça é feita a análise de complexidade do algoritmo, a qual iniciará pelo procedimento *Shortest_Path* seguida do procedimento *Level_Path*.

Observe que o procedimento *Shortest_Path*, usa o grafo $G_{1,1}(V(E_{1,1}), E_{1,1})$ para realizar a busca, onde $V(E_{1,1})$ são os vértices de nível 1 de rótulo(1,1) e $E_{1,1}$ as arestas de nível 1, ligadas apenas por vértices de nível 1, suprimindo os de rótulo (1,*a*) onde $1 < a \leq k$. Considerando a utilização do *Dijkstra* que possui complexidade de pior caso $O(N^2)$. Substituindo o valor de *N*, temos que a complexidade do procedimento *Shortest_Path* é $O(n^4)$.

A complexidade do procedimento *Level_Path* irá depender do rótulo do vértice corrente e do número de vezes que é inserido um vértice no conjunto *Open*. Existem dois casos :

a) Caso1 - Rótulo(*a,a*). Na treliça estruturada uniforme, todos os vértices do perímetro de cada bloco são vértices objetivo. O número de vértices distintos que entraram no conjunto *Open* é limitado por $(n - 1)^2$: número de vértices possíveis, pois os sucessores ou predecessores não serão considerados pelo *Level_Path*. O número de vezes que cada nó pode ser colocado em *Open* é no máximo $(n - 1)^2$, logo a complexidade para esse caso é $O(n^4)$, com *n* sendo a densidade da treliça.

b) Caso2 - Rótulo(*a,b*). Com $a < b$ no pior caso, o grafo a ser varrido agora pelo *Level_Path* é $G_{a,b.}$ O número de nós distintos que podem ser colocados em *Open* é limitado por *n*-1: número de nós possíveis. O número de vezes que poderá ser colocado será limitado pelo número de nós possíveis, isto é, *n*-1. Logo a complexidade neste caso é $O(n^2)$, com *n* sendo a densidade da treliça.

Agora, resta examinar o número de vezes que o procedimento *Level_Path* é chamado. Observe que este número, também irá variar de acordo com o rótulo dos nós origem e destino onde, no pior caso, estes rótulos são iguais a *k*. Nota-se que após uma execução do procedimento, o valor do rótulo de ambos cai para *k*-1 e portanto, o número máximo de vezes que o procedimento será chamado é de 2(*k*-1), considerando que ele deve obter as variáveis *ipath* e *fpath*. Desta forma como o limite superior de complexidade do *Level_Path* é $O(n^4)$ e ele é chamado no máximo 2(*k*-1), a complexidade final do *Level_Path* é $O(k.n^4)$. Como $O(k.n^4) + O(n^4) = O(k.n^4)$, a complexidade do *LGS* é $O(k.n^4)$, onde *n* é a densidade da treliça e *k* a número de

níveis.

Colocando em função de *N*, número de nós total da treliça, a complexidade é $O(k \cdot N^{2/k})$. Observe que para *k* > 1, esta complexidade, é melhor que a complexidade do algoritmo de Dijkstra.

5 EXPERIMENTO COMPUTACIONAL

5.1 INTRODUÇÃO

O experimento computacional descrito abaixo, foi produzido conforme as recomendações obtidas em JOHNSON (2002). Através das lições aprendidas por JOHNSON em uma década de suas experimentações e discussões com outros experimentalistas, são utilizadas todas as orientações descritas nesta seção.

Para analise experimental dos algoritmos, geralmente são utilizadas três formas de análise.

a) Análise do pior caso,
b) Análise do caso médio,
c) Análise experimental.

Recentemente, houve o aumento no interesse no trabalho experimental tendo em vista o crescente reconhecimento dos resultados. Em conseqüência, diversos forums incluem o trabalho experimental e teórico (ACM em Algoritmos experimental, o simpósio de ACM-SIAM em algoritmos discretos (SODA), o simpósio europeu dos algoritmos (ESA), a oficina na engenharia do algoritmo (WAE) e a oficina na engenharia do algoritmo e na experimentação (ALENEX)).

O valor adicionado envolvendo cientistas revigora a agenda teórica estabelecendo conexões às aplicações. Neste trabalho, os algoritmos são analisados de forma teórica para que ajudem a realizar a experimentação. Segundo JOHNSON (2002), recomendações suportadas por pesquisadores como Bentley, Goldberg e outros, o interesse teórico relaciona uma aproximação mais científica à experimentação.

A execução de um algoritmo é fácil, a parte difícil é produzir significado e o valor do resultado na pesquisa. Com objetivo de evitarmos falhas no campo da análise experimental e produzirmos análises importantes para o estudo de rotas rodoviárias utilizando múltiplos critérios, nos baseamos em 10 princípios definidos por Johnson vistos a seguir.

a) “Realizar experimentos notáveis” – Experimentos que valem a pena publicar. Os resultados devem ser novos e de interesse e/ou uso de um conjunto de leitores.
b) Amarrar o trabalho à literatura – Deve-se fazer uma vasta pesquisa bibliográfica antes de realizar análises experimentais.
c) Usar conjunto de experimentos de forma a suportar conclusões gerais – Os dados podem ser reais ou gerados. Os dados gerados podem ser aleatórios, porém, devem ser estruturados refletindo o mundo real.
d) Usar projetos experimentais eficientes e eficazes – Existem técnicas para realizar análises experimentais que não só reduzem o tempo computacional exigido pelo experimento, mas também aumentam a acurácia das respostas obtidas.
e) Implementações razoavelmente eficientes – A implementação não tem a necessidade de ser muito complicada. Realizar uma implementação rápida e eficiente o bastante de modo a permitir finalizar o estudo rapidamente.
f) Assegurar a possibilidade de reprodução – Se o leitor utilizar o mesmo código fonte, as mesmas instâncias, a mesma máquina, o mesmo sistema operacional e o mesmo compilador ele obterá os mesmos resultados.
g) Assegurar a comparabilidade – Permitir que outros pesquisadores possam comparar os resultados com os deles. Documentar cada detalhe do experimento é de fundamental importância.
h) Relatar toda a história – Relatar na íntegra todos os dados usados explicando a maneira de como foram empregados e gerados.
i) Extrair conclusões bem justificadas e procurar explanações – Escrever de forma a fornecer todo o suporte ao entendimento das conclusões.
j) Apresentar seus dados em maneiras informativas – Fornecer tabelas e gráficos. Colocar somente um destes dois, pode não esclarecer o objetivo do experimento na íntegra.

Nesta seção apresentamos como pretendemos conduzir o desenvolvimento de nossa análise experimental. Na Seção 5.2 descrevemos o modelo de referência

que foi utilizado para este experimento e poderá ser utilizado em outros experimentos que estudam o problema MCP.

5.2 MODELO DE REFERÊNCIA

Tendo em vista que a natureza deste trabalho visa analisar experimentalmente os algoritmos *LGS* e *TAMCRA*, ao invés de optarmos pela utilização de um modelo de carta real, optamos pela mesma treliça estruturada definida por SHAPIRO (1992) e utilizada em BRAZ (2005). Esta treliça é baseada em um modelo matemático chamado *Manhattan Street Network* (MSN). Pode ser aplicada a várias cidades e se adequa ao gerador de grafos em níveis para a realização da análise com o algoritmo *LGS*.

A treliça estruturada consiste de um conjunto de nós e um conjunto de arestas onde para todos os nós e arestas é atribuído um rótulo, como, por exemplo, um nó *u* que possua arestas incidentes de rótulo 1 (nível 1) e rótulo 2 (nível 2), será rotulado como *u(1,2)*. O processo de construção da treliça é descrito em SHAPIRO (1992), o qual é caracterizado por um parâmetro *k*, que determina o número de níveis do grafo, e de um conjunto de *k* pares (n_i,m_i) para $1 \leq i \leq k$ o qual representa um fator de replicação do bloco inicialmente criado, isto é, será criada uma estrutura de n_i x m_i unidades de bloco, sendo este processo repetido *k* vezes com *i* variando de *k* até 1. Durante o processo rotulam-se os nós e as arestas.

Quando $n = n_k = n_{k-1} = n_{k-2} = ... = m_{k-2} = m_{k-1} = m_k$, assume-se que a treliça estruturada é uniforme onde *n* é a densidade do grafo. A Figura 5.1 representa uma treliça estruturada uniforme com $n = 3$ e $k = 2$.

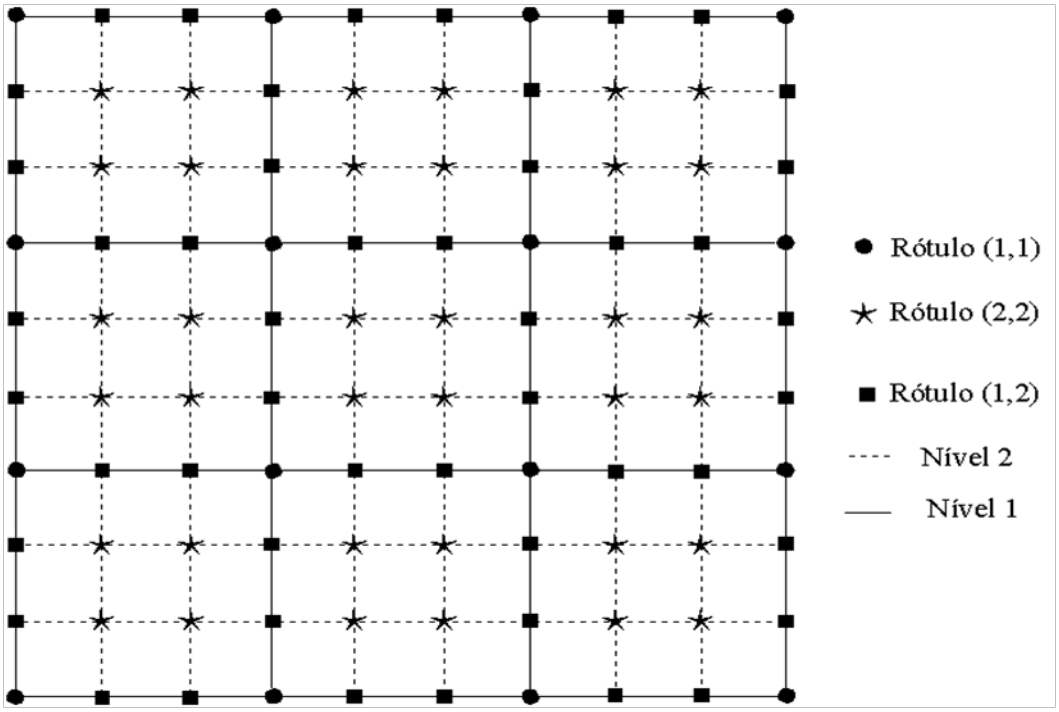


FIG 5.1 – Treliça Estruturada Uniforme com *n* = 3 e *k* = 2

Fonte: BRAZ(2005)

Para a treliça estruturada uniforme, temos as seguintes propriedades:

(a) $N_1=(n+1)\cdot(2\cdot n^k-n+1)$;

(b) $N_i=(n-1)\cdot n^{i-1}\cdot(2\cdot n^k-n^i-n^{i-1})$;

(c) $N=(n^k+1)^2$;

(d) $N_{1,1}=(n+1)^2$;

(e) $N_{i,i}=(n-1)^2\cdot n^{2(i-1)}\ para\ 1<i\leqslant k$;

(f) $N_{1,j}=2n^{j-1}\cdot(n-1)\cdot(n+1)\ para\ 1<j\leqslant k$;

(g) $N_{i,j}=2\cdot(n-1)^2\cdot n^{j-i}\cdot n^{2(i-1)}\ para\ 2\leqslant i<j\leqslant k$. onde

$N_i=N_{i,i}+\sum_{j=i+1}^{k}N_{i,j}$ é o número de nós do nível *i*;

$N_{i,j}$ é o número de nós com rótulo (*i,j*) e

$N=\sum_{i=1}^{k}N_i$ representando o número total de nós na treliça.

Por exemplo, para a Figura 5.1 onde n = 2 e k = 3, temos:

$N = 100\,nós$

$N_1 = 64\,nós\,de\,nível\,1, com\,N_{1,1} = 16\,nós\,de\,rótulo\,(1,1),\ N_{1,2} = 48\,nós\,de\,rótulo\,(1,2)$

$N_2 = 36\,nós\,de\,nível\,2, com\,N_{2,2} = 36\,nós\,de\,rótulo\,(2,2)$

Além disso, pode-se notar que ao longo do perímetro de cada bloco da treliça, todos os nós são nós objetivos e $N_{i,i} = (n\text{-}1)^2$, está inserido no bloco.

Entre os nós de rótulo (*a,b*), existiram apenas dois nós objetivos de rótulo (*a, b-1*), onde o número de nós com rótulo (*a,b*) é igual a *n*-1. A Figura 5.2 exemplifica esta configuração.

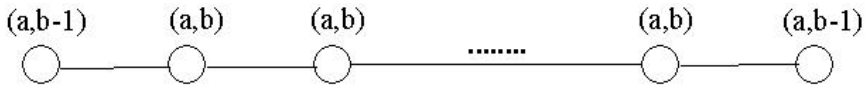


FIG.5.2 – Configuração de um bloco.
Fonte: BRAZ(2005)

5.3 DESCRIÇÃO DOS EXPERIMENTOS

O objetivo de nosso experimento é realizar uma análise dos caminhos mínimos gerados pelos algoritmos *LGS* e o *TAMCRA* aplicados ao roteamento de veículos com múltiplos critérios.

Como já verificamos anteriormente, dois algoritmos podem ser aplicados ao contexto rodoviário, levando em consideração a utilização de ruas de maior hierarquia, segurança e outros critérios.

Conforme definido na Seção 5.1, este trabalho tem a finalidade de descrever o impacto do algoritmo e a aplicação de seus conceitos no contexto rodoviário com múltiplos critérios. É necessário lembrar que estamos primeiramente interessados no caminho produzido pelo algoritmo e não em sua eficiência.

Conforme as definições aplicadas aos grafos, no contexto rodoviário, cada nó deve ser entendido como um cruzamento de ruas e suas arestas como as rodovias de uma cidade. Informações da treliça estruturada, atribuídas aos nós e arestas do grafo, nos fornecerão informações para considerarmos a hierarquia das ruas em

nosso roteamento.

Outro critério utilizado no roteamento de veículos é a distância. Tendo em vista que ambos os algoritmos consideram a hierarquia e o critério distância, são realizadas pesquisas em caminhos de curta, média e longa distância.

Para definir o que são caminhos curtos, médios e longos é utilizada a sobreposição na treliça estruturada com 2 níveis e 100 nós (Figura 5.3) e definidos alguns conceitos para o melhor entendimento de nosso experimento.

Definição 5.1 - Uma região de tráfego é um conjunto de ruas secundárias (de nível 2) limitadas por ruas primárias (de nível 1).

Na Figura 5.3 podemos verificar as regiões de tráfego delineadas pela linha contínua que recebem as letras *a, b* e *c*. No contexto rodoviário, podemos fazer a analogia a um bairro, onde existem ruas menores internamente e avenidas ao seu redor.

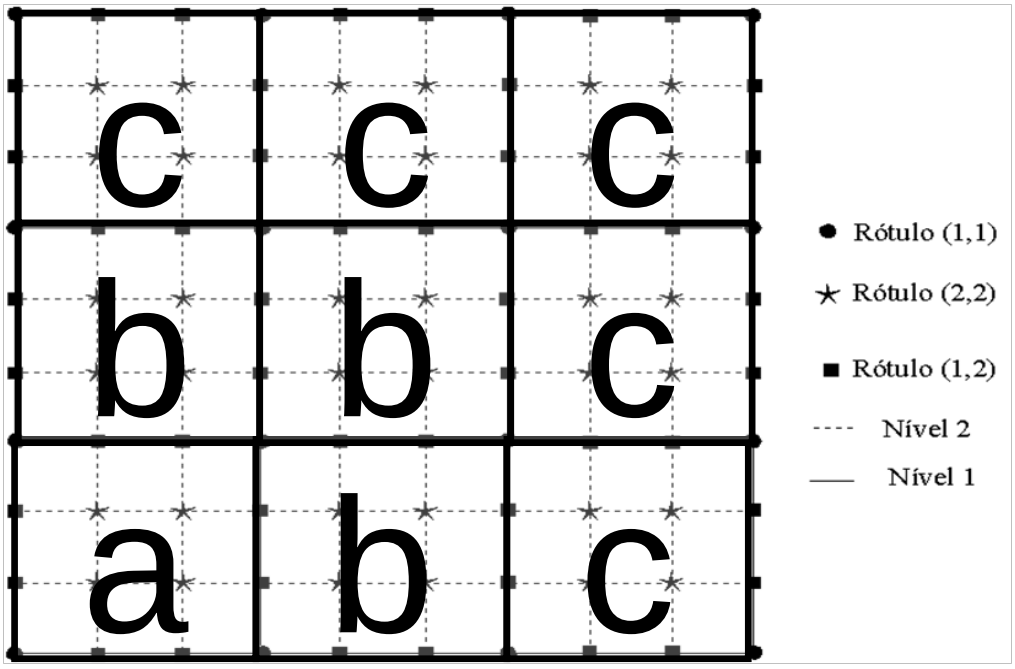


FIG 5.3 – Exemplo de regiões de tráfego.

Definição 5.2 - Seja o grafo *G*(*V,E*) uma treliça estruturada. Sejam *u* e *v* pertencentes a *V*, tais que *u* e *v* pertençam à mesma região de tráfego.

Exemplo 5.1: Na figura 5.3 ambos os nós poderiam estar inseridos na região demarcada com a letra *a*.

Definição 5.3 – Seja o grafo $G(V,E)$ uma treliça estruturada. Sejam u e v pertencentes a V, tais que u e v pertençam a duas regiões de tráfego distintas. O caminho $P_{u,v}$ é um caminho de média distância se todos vértices de nível 1 em $P_{u,v}$ se configurarem vértices de interseção entre as duas regiões de tráfego.

Exemplo 5.2: Na figura 5.3, o primeiro nó poderia estar dentro da região demarcada com a letra *a* e o segundo nó poderia estar dentro de qualquer região de tráfego demarcada com a letra *b*.

Definição 5.4 - Seja o grafo $G(V,E)$ uma treliça estruturada. Sejam u e v pertencentes a V, tais que u e v pertençam a duas regiões de tráfego distintas. O caminho $P_{u,v}$ é um caminho de longa distância se todos vértices de nível 1 em $P_{u,v}$ não se configurarem vértices de interseção entre as duas regiões de tráfego.

Exemplo 5.3: Na figura 5.3, o primeiro nó poderia estar dentro da região demarcada com a letra *a* e o segundo nó poderia estar dentro de qualquer região de tráfego demarcada com a letra *c*.

5.4 IMPLEMENTAÇÃO

Um dos objetivos desta dissertação é implementar um algoritmo que utilize múltiplos critérios capaz de maximizar o uso de rodovias de maior hierarquia. Outro objetivo é a implementação do *LGS* utilizando grafos em níveis de forma a viabilizar uma análise comportamental entres os algoritmos. Desta forma, nesta seção descrevo em linhas gerais como foi implementado os algoritmos *TAMCRA* e *LGS*.

Para o desenvolvimento e implementação dos códigos foi utilizada a linguagem C. Para as análises do caminho foram utilizados dois programas gratuitos. O primeiro denominado *dev C++*, versão 4.9.9.2 traduz-se em um ambiente de desenvolvimento integrado gratuito para programação na linguagem C. O segundo software foi o *GraphViz* na versão 1.0. Este nos auxiliou na geração de grafos para realizar a análise comportamental dos algoritmos. Trata-se de um conjunto de programas, de código fonte aberto, para a automatização do desenho

de grafos.

Todos os algoritmos e softwares foram implementados em um microcomputador Pentium 4 de 3 Ghz, com 2 GB de RAM e sistema operacional Windows XP Professional versão 2002.

Referente à implementação dos algoritmos, na Subseção 5.4.1 é apresentado o algoritmo de construção do grafo aderente ao nosso modelo de referência. Na Subseção 5.4.2 está descrita a implementação do algoritmo *LGS*. Finalmente na Subseção 5.4.3 o *TAMCRA* é comentado.

5.4.1 IMPLEMENTAÇÃO DA TRELIÇA ESTRUTURADA UNIFORME

Conforme definido na Seção 5.2 Capítulo 5, utilizamos um modelo de referência existente introduzido em SHAPIRO (1992). Através de nossa implementação é gerado um grafo com 100 nós onde os algoritmos *LGS* e *TAMCRA* se basearam para a busca de caminhos mínimos.

Conforme verificado trata-se de um processo iterativo de encaixe de uma matriz finita onde é gerado um conjunto de nós e de arestas. Para as arestas são atribuídas rótulos de níveis. Para esta implementação utilizamos dois níveis. Logo, podemos ter arestas com rótulos de nível 1 ou 2. Arestas rotuladas com o nível 1, representam no contexto rodoviário, ruas de maior hierarquia do que as rotuladas com nível 2. Da mesma forma, no cruzamento de arestas teremos a rotulação para os vértices. Para um vértice entre duas arestas de nível 1 será atribuída a rotulação (1,1), para um vértice entre duas arestas de nível 2 será atribuída a rotulação (2,2) e para qualquer combinação de arestas de nível 1 e 2 será atribuída a rotulação (1,2).

No início de cada algoritmo implementado, foi utilizado um processo de construção da treliça através da função *Lattice*(*k*). O *k* = 2 atribuído para esta implementação, determinou o número de níveis desejados.

O processo de construção pode ser verificado em SHAPIRO (1992) e consiste em 2 partes:

a) Parte 1 – Gerar os nós e assinalar os rótulos.
b) Parte 2 – Gerar as arestas e assinalar os rótulos.

Os principais passos dos processos da implementação são:

a) Um quadrado (*k* X *k*) é construido no canto inferior esquerdo da treliça;
b) Os 4 vértices dos cantos são atualizados com o rótulo (*k,k*);
c) Utilizando o quadrado como bloco, o retângulo é construído conforme um fator de replicação pré-determinado (*nk* X *mk*);
d) Exceto para os 4 cantos, todos os rótulos dos nós do perímetro são atualizados para (*k*-1, *k*). Aos rótulos dos cantos são atribuídos (*k*-1, *k*-1);
e) No passo seguinte, a estrutura definida nos passos *a* ao *d* é utilizada para construir outro retângulo.
f) Este processo se repete *k* vezes.

5.4.2 IMPLEMENTAÇÃO DO LGS

Após a função *Lattice(k)* ter sido chamada para a criação da treliça estruturada com 2 níveis, a função *LGS*(*VerticeInicial,VerticeFinal*) é solicitada. Através da definição das variáveis *VerticeInicial e VerticeFinal* no procedimento principal do algoritmo, estas são repassadas para a função *LGS* respectivamente como nó origem e nó destino do caminho desejado.

Conforme descrito no capítulo 3 e aderente à definição encontrada em SHAPIRO (1992), a função *LGS(VerticeInicial,VerticeFinal)* ainda faz uso de duas outras funções internas.

A primeira função *level_path*(*v,w*) é utilizada em dois sentidos. Hora faz a busca a partir do nó origem, outra hora busca o caminho a partir do nó destino. Desta forma, em um primeiro momento utiliza o *v* com o valor de *VerticeInicial,* em um segundo momento com o valor do *VerticeFinal*.

A função *level_path*(*v,w*) também utiliza um parâmetro de saída atribuído a *w*. Após a busca e o *nó gol* encontrado, este vértice é atribuído a *w*. A função tem sua primeira busca em profundidade modificada. A busca inicia em um nó de nível ≥ 2 e encontra o caminho mais curto até um nó de nível 1. Se esta busca iniciou no vértice inicial, um valor w_1 é atribuído a *w* e teremos o subcaminho no vetor *ipath*. No sentido contrário, *w* terá o valor w_2 e um subcaminho invertido no vetor *fpath*.

Após o algoritmo obter o w_1 *e* w_2 como parâmetro de entrada, uma segunda

função *shortest_path*(w_1,w_2), retorna o menor caminho dentre todos os nós de nível 1 utilizando o algoritmo de Dijkstra. Este subcaminho obtido é atribuído ao vetor *spath*.

Após o término das funções *level_path e shortest_path*, o subcaminho contido do vetor *fpath* é invertido e o caminho final é obtido através da respectiva concatenação de *ipath, spath e fpath.*

5.4.3 IMPLEMENTAÇÃO DO TAMCRA

Aderente às recomendações de implementação encontradas em KUIPERS (2000), após gerar a treliça estruturada com 2 níveis, a função *TAMCRA*(*Grafo,VerticeInicial,VerticeFinal,criterio1,criterio2,kcaminhos,Praticavel*) é solicitada. Primeiramente o parâmetro *Grafo* é atribuído ao nome da estrutura do *heap* de Fibonacci encontrada em CORMEN (2002). As variáveis *VerticeInicial e VerticeFinal* são repassadas para a função, respectivamente como nó origem e nó destino do caminho desejado. Os parâmetros *criterio1 e criterio2* são os respectivos limites de cada critério atribuídos à busca. O parâmetro *kcaminhos* definido como 2 será o número de caminhos a serem guardados em cada nó. Este parâmetro é o responsável pela determinação do tamanho máximo da fila do algoritmo.

Como saída, através de uma matriz de saída chamada de *Praticável,* o algoritmo fornece o menor caminho praticável para o problema MCP. A descrição dos procedimentos e funções principais do algoritmo estão descritas na Seção 3.3 Capítulo 3.

6 RESULTADOS EXPERIMENTAIS

6.1 INTRODUÇÃO

Este capítulo apresenta a análise experimental realizada sobre os algoritmos *LGS* e *TAMCRA*, anteriormente conceituados nesta dissertação. Seu objetivo é fornecer dados experimentais ao leitor e avaliar o comportamento, acurácia e desempenho dos algoritmos para a aplicação de múltiplos critérios ao contexto rodoviário. A análise experimental está baseada nos algoritmos *LGS* e *TAMCRA*.

Este capítulo está dividido em duas seções. Em cada seção, as informações referentes a curta, média e longa distância são detalhadas.

Primeiramente a Seção 6.2 apresenta análises experimentais como: menores caminhos praticáveis com dois critérios, tempo de CPU e informações sobre a quantidade de arestas utilizadas na busca. Na Seção 6.3 são apresentados estudos comportamentais dos grafos utilizando múltiplos critérios aplicados a uma malha rodoviária.

6.2 AVALIAÇÃO COMPARATIVA DA PERFORMANCE

Esta avaliação levou em consideração o efeito causado por três variáveis:

a) a aplicação de critérios justos;
b) o menor caminho praticável encontrado;
c) a quantidade de subcaminhos analisados;
d) o tamanho do grafo;

Para facilitar a avaliação comparativa e imparcial entre os dois algoritmos, primeiramente foi utilizado um grafo de 100 vértices aderente à treliça estruturada uniforme definida em SHAPIRO (1992). Este grafo possui 64 vértices de nível 1 (ruas de maior hierarquia) e 36 vértices de nível 2 (ruas de menor hierarquia) conforme apresentado no Capítulo 5.2.

Tomando como premissa a utilização do mesmo grafo, os algoritmos *LGS* e

TAMCRA tiveram seu desempenho médio obtido através de 40 tomadas de tempo. Conforme apresentado na Figura 5.3, a partir de cada nó da região de tráfego *a*, realizamos buscas para todos os nós de cada região de tráfego. Foram geradas buscas de caminhos em curta distância (dentro da região *a*), em média distância (da região *a* até a região *b*) e longa distância (da região *a* até a região *c*).

Com objetivo de avaliarmos a aplicação de critérios com limites rígidos e folgados no *TAMCRA*, este algoritmo foi analisado em dois momentos distintos. Em um primeiro momento, atribuímos 1000 para ambos os critérios aditivos implementados. Denominamos esta atribuição como limites frouxos devido ao alto custo atribuído aos critérios. Em um segundo momento, implementamos um limite justo para cada critério, ou seja, utilizamos o menor valor possível para os critérios de forma que o algoritmo possa considerar a busca de caminhos por arestas de nível 1 (ruas de maior hierarquia).

A seguir, na Subseção 6.2.1 são apresentadas as médias dos menores caminhos praticáveis obtidos, na Subseção 6.2.2 são quantificadas o número de arestas, na Subseção 6.2.3 é apresentado o tempo de cada algoritmo e finalmente na Subseção 6.2.4 é apresentado o efeito causado pela variação do tamanho do grafo.

6.2.1 EFEITO NOS MENORES CAMINHOS PRATICÁVEIS

Para avaliarmos o efeito causado nos menores caminhos praticáveis, a partir de cada vértice da região *a*, foram utilizadas 40 tomadas de tempo para cálculo da média de cada caminho entre as regiões de curta, média e longa distância, conforme definido na Seção 5.3 e demonstrado na Figura 6.1.

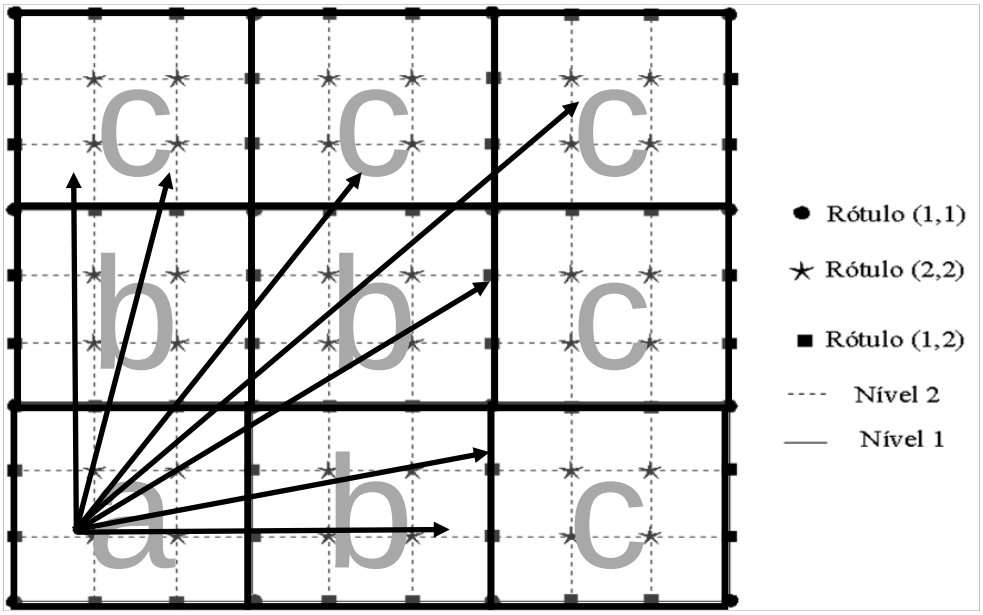


FIG 6.1 – Regiões e direção das buscas.

A cada aresta foram atribuídos dois custos. Para a primeira métrica foram atribuídos o custo 1 nas arestas de maior hierarquia e custo 2 para as arestas de menor hierarquia. Com objetivo de avaliar o tamanho do caminho percorrido no grafo, ao segundo critério foi aplicado um custo unitário e uniforme. Abaixo na Figura 6.2 e Tabela 6.1 são apresentadas as médias dos tamanhos dos menores caminhos para curta, média e longa distância.

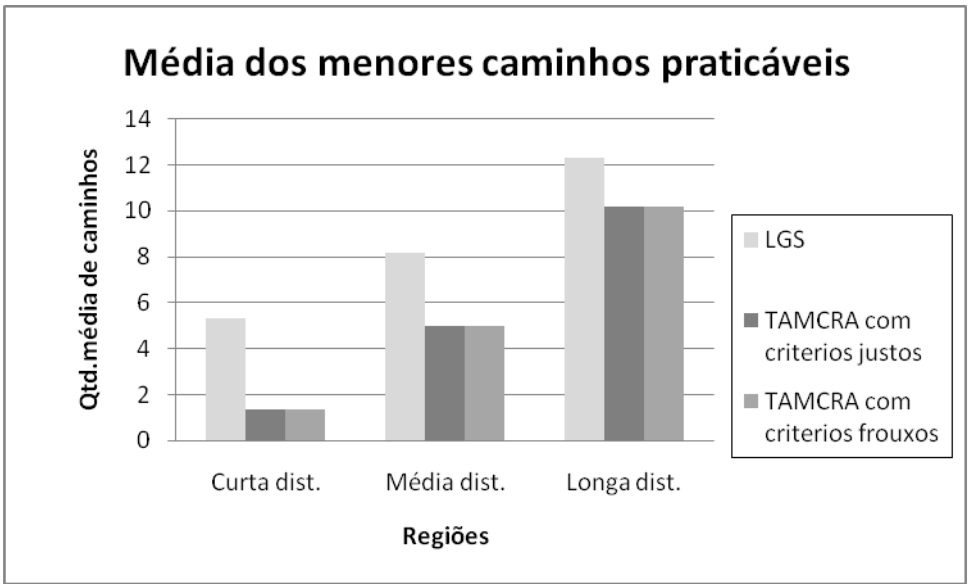


FIG 6.2 – Médias dos tamanhos dos menores caminhos.

Média dos menores caminhos praticáveis			
Algoritmos/regiões	**Curta dist.**	**Média dist.**	**Longa dist.**
LGS	5,33	8,17	12,30
***TAMCRA* com critérios justos**	1,33	5,00	10,20
***TAMCRA* com critérios frouxos**	1,33	5,00	10,20

TAB 6.1 – Efeito das médias dos tamanhos nos menores caminhos gerados.

Confirmando o que foi demonstrado no Capítulo 3, Seção 3.3, o algoritmo *TAMCRA* com critérios justos ou frouxos, otimiza a busca de menores caminhos praticáveis com múltiplos critérios. A diferença significativa do tamanho do *LGS*, está diretamente ligada às restrições citadas no Capítulo 4 na Seção 4.1. Na Seção 6.2 são demonstradas de forma comportamental as restrições de orientação que foram determinantes para o aumento do tamanho do *LGS*.

No que se refere a média dos menores caminhos praticáveis, observa-se também que ao aumentar a distância da busca, a diferença do algoritmo *LGS* em relação ao *TAMCRA* diminui. Esta redução deve-se ao aumento do número de arestas sendo tratadas pelo procedimento *Shortest_Path* do algoritmo *LGS*. Abaixo na Tabela 6.2 é apresentado o ganho percentual encontrado na utilização do *TAMCRA* em relação ao menor caminho praticável encontrado pelo *LGS*.

TAMCRA*/*LGS	**Curta dist.**	**Media dist.**	**Longa dist.**
***TAMCRA* com critérios justos**	75,00%	38,78%	17,07%
***TAMCRA* com critérios frouxos**	75,00%	38,78%	17,07%

TAB 6.2 – Diferença percentual dos menores caminhos do *TAMCRA* em relação ao *LGS*.

6.2.2 EFEITO NO NÚMERO DE ARESTAS UTILIZADO NA BUSCA

A segunda avaliação trata da comparação do número de arestas durante a busca de caminhos praticáveis. Foi coletada a quantidade de arestas analisadas pelos algoritmos *LGS* e *TAMCRA* entre as regiões de curta, média e longa distância definida na Seção 5.3.

Com o objetivo de verificarmos o efeito da aplicação dos principais conceitos do *TAMCRA*, *k menores caminhos*, *caminho não dominado* e o *tamanho do caminho não linear*, são aplicados critérios justos e frouxos para avaliar a variação do número de arestas durante a busca. Na Figura 6.3 e Tabela 6.3 são apresentadas as médias do número de arestas analisadas durante a busca de caminhos de curta, média e longa distância.

Algoritmos/regiões	Curta dist.	Média dist.	Longa dist.
LGS	14,00	25,75	48,60
***TAMCRA* com critérios justos**	6,67	32,25	78,20
***TAMCRA* com critérios frouxos**	16,67	49,25	85,60

TAB 6.3 – Efeito das médias do número de arestas analisadas durante a busca.

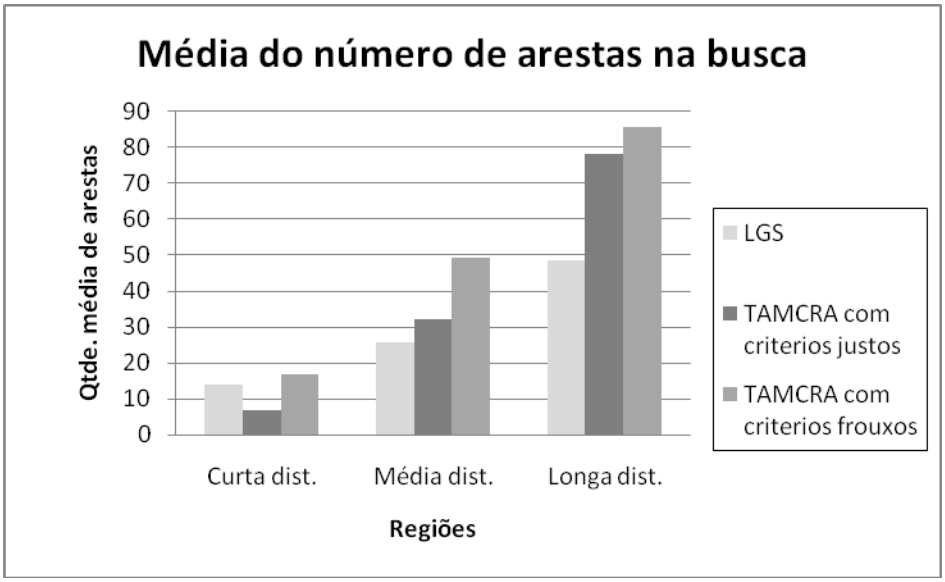


FIG 6.3 – Médias do número de arestas analisadas na busca.

Conforme esperado, foi verificado o gradativo aumento do número de arestas conforme a ampliação da distância do caminho. Conforme definido para o *TAMCRA* no Capítulo 4 Seção 4.2.2, este aumento de arestas deve-se ao conceito dos *k* menores caminhos. Na definição do $k = 2$ para esta implementação, em cada vértice intermediário é verificada a adição de cada aresta com até 2 menores caminhos. Pode-se observar na Figura 6.3, que em caminhos de curta distância onde ainda não existe 2 menores caminhos, a média de arestas analisadas pelo *TAMCRA* é

bem menor do que a média para longa e média distância.

Para analisar a medida de tamanho não linear definida no Capítulo 4 Seção 4.2.1, é realizada a comparação do *TAMCRA* com critérios frouxos e justos. Pode ser observado na Figura 6.3, que a aplicação de critérios estritos elimina a quantidade de arestas. Esta eliminação de menores caminhos praticáveis durante a busca, evita a avaliação de caminhos que já ultrapassam os critérios antes de alcançar o vértice destino.

Para o *LGS*, conforme o aumento da distância é verificado um número de arestas cada vez menor em relação ao *TAMCRA*. Apesar do crescimento uniforme do *LGS*, o número reduzido de arestas deve-se à aplicação do conceito do grafo em níveis. Conforme a distância é ampliada, maior é o número de arestas de nível 2 descartadas da busca. Tendo em vista que o procedimento *Shortest_Path* considera somente a análise das arestas de nível 1, o *LGS* apresenta um número cada vez menor de arestas analisadas.

Abaixo na Tabela 6.4 é apresentado os ganhos e perdas percentuais na utilização do *TAMCRA* em relação ao número de arestas analisadas pelo *LGS*.

TAMCRA/LGS	**Curta dist.**	**Media dist.**	**Longa dist.**
***TAMCRA* com critérios justos**	52,38%	-25,24%	-60,91%
***TAMCRA* com critérios frouxos**	-19,05%	-91,26%	-76,13%

TAB 6.4 – Ganhos e perdas percentuais do *TAMCRA* em relação ao LGS.

6.2.3 EFEITO NO TEMPO DE BUSCA

A terceira avaliação compara os tempos obtidos durante a busca de caminhos praticáveis. Da mesma forma que as seções anteriores, a média dos 40 tempos de cada caminho, foi coletada pelos algoritmos *LGS* e *TAMCRA* entre as regiões de curta, média e longa distância definida na Seção 5.3.

Na Figura 6.4 e Tabela 6.5 são apresentadas as médias dos tempos obtidos durante a busca dos caminhos.

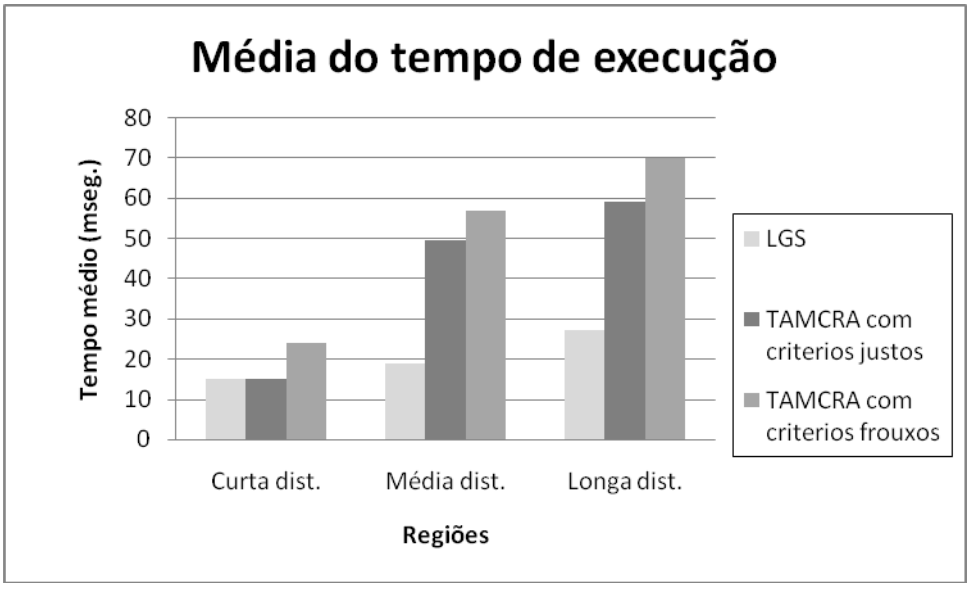


FIG 6.4 – Médias do tempo de execução por caminhos praticáveis.

Algoritmos/regiões	**Curta dist.**	**Média dist.**	**Longa dist.**
LGS	15,00	19,00	27,10
***TAMCRA* com critérios justos**	15,00	49,58	59,30
***TAMCRA* com critérios frouxos**	24,00	57,08	70,00

TAB 6.5 – Efeito das médias do tempo de busca por caminhos praticáveis.

Conforme esperado, foi verificado o aumento de tempo conforme a ampliação da distância do caminho. Apesar do crescimento não uniforme do número de arestas apresentados na Subseção 6.2.2, foi verificada uma redução de aproximadamente 42% no tempo de caminhos de longa distância em relação aos de média distância. Conforme definido para o algoritmo *TAMCRA* na Seção 4.2.3, é verificado que grande parte desta redução é devida à eliminação dos caminhos dominados.

É verificada na Tabela 6.6 uma grande diferença percentual do tempo do TAMCRA em relação ao *LGS*. Além desta diferença entre os dois algoritmos, também é observada a significativa redução de tempo quando são definidos critérios justos para o *TAMCRA*. Esta diferença é de 60% para caminhos de curta distância e de aproximadamente 49% para caminhos de média e longa distância.

TAMCRA/*LGS*	Curta dist.	Media dist.	Longa dist.
***TAMCRA* com criterios justos**	0,00%	-160,96%	-118,82%
***TAMCRA* com criterios frouxos**	-60,00%	-200,44%	-158,30%

TAB 6.6 – Diferença percentual do tempo no *TAMCRA* em relação ao *LGS*.

6.2.4 EFEITO CAUSADO PELA VARIAÇÃO DO TAMANHO DO GRAFO

A quarta avaliação apresenta os efeitos causados com a variação do tamanho das regiões e do grafo. Os efeitos verificados nas Subseções 6.2.1, 6.2.2 e 6.2.3 são baseados em um grafo de 100 vértices e regiões com 4 vértices de nível 2. Nesta subseção é analisada o efeito causado em outro grafo com 256 vértices e regiões com 16 vértices de nível 2, conforme apresentado na Figura 6.5.

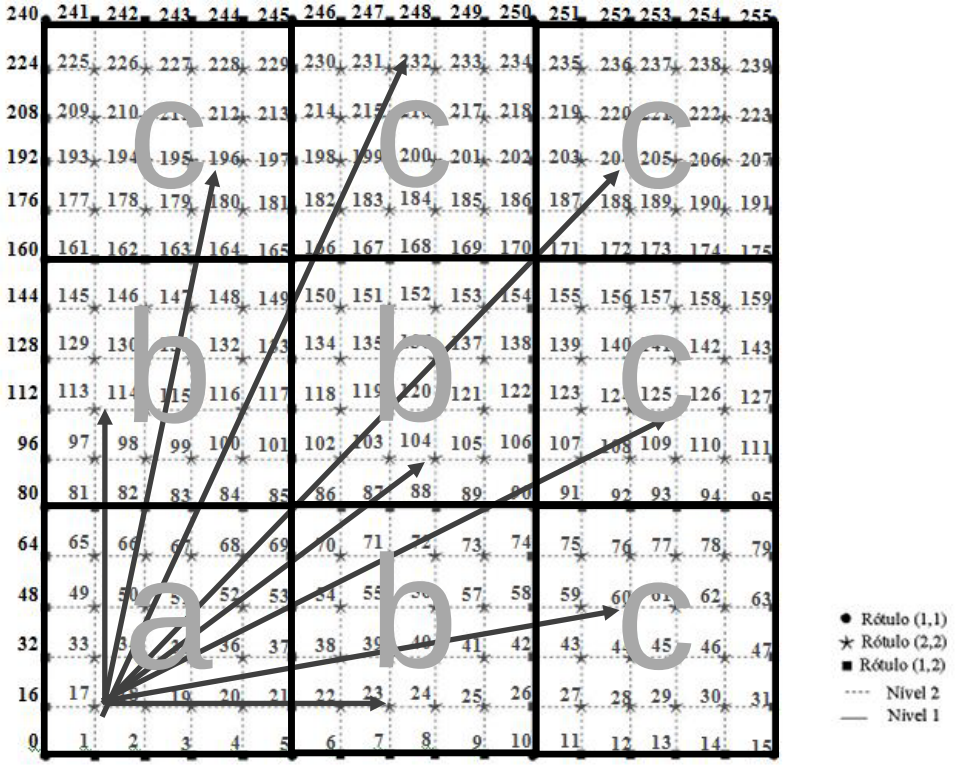


FIG 6.5 - Treliça estruturada uniforme com 256 vértices.

Da mesma forma que nas Subseções 6.2.1, 6.2.2 e 6.2.3, a média dos 40 tempos de cada caminho, foi coletada pelos algoritmos *LGS* e *TAMCRA* entre as

regiões de curta, média e longa distância definida na Seção 5.3.

Na Figura 6.6 e Tabela 6.7 são apresentadas as médias dos menores caminhos conforme os algoritmos, critérios, regiões e tamanho do grafo.

Devido ao aumento dos vértices de nível 2, internamente as regiões da treliça, foi verificado o aumento uniforme na média dos menores caminhos.

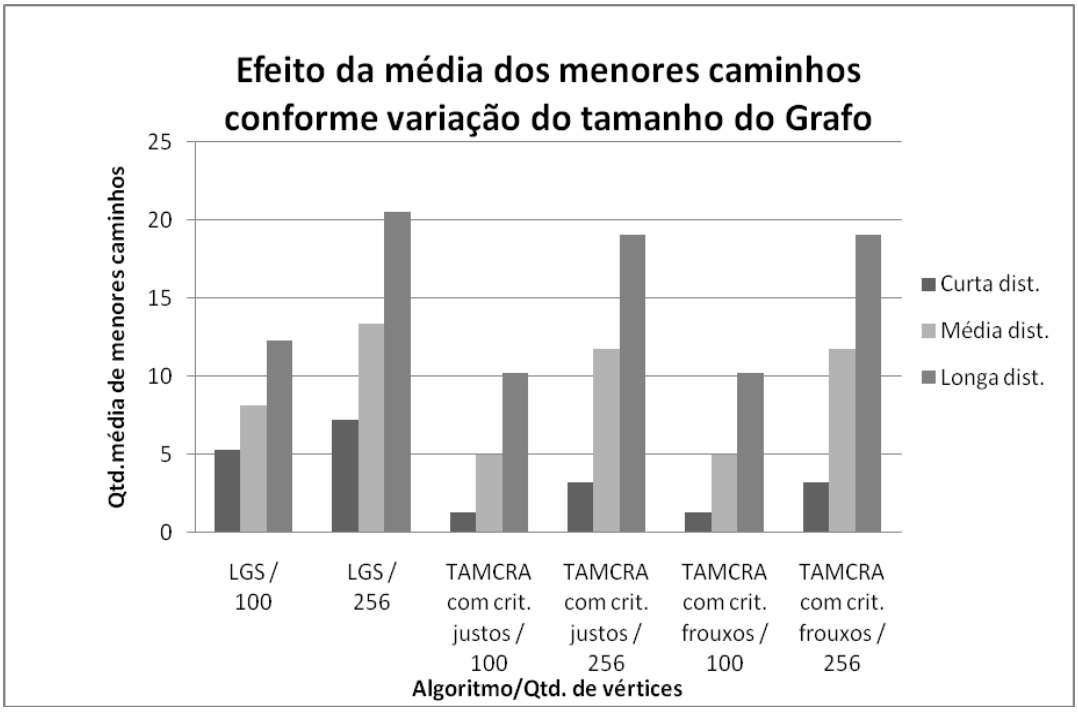


FIG 6.6 – Variação da média dos menores caminhos conforme o algoritmo e tamanho do grafo.

Média dos menores caminhos			
Algoritmos/regiões	**Curta distância**	**Média distância**	**Longa distância**
LGS com 100 vértices	5,33	8,17	12,30
LGS com 256 vértices	7,20	13,38	20,55
TAMCRA com crit. justos e 100 vértices	1,33	5,00	10,20
TAMCRA com crit. justos e 256 vértices	3,20	11,79	19,10
TAMCRA com crit. frouxos e 100 vértices	1,33	5,00	10,20
TAMCRA com crit. frouxos e 256 vértices	3,20	11,79	19,10

TAB 6.7 – Efeito das médias dos tamanhos nos menores caminhos gerados conforme o algoritmo e regiões.

Na Figura 6.7 e Tabela 6.8 são apresentadas as médias da quantidade de arestas analisadas conforme os algoritmos, critérios, regiões e tamanho do grafo.

Devido ao aumento dos vértices de nível 2, internamente as regiões da treliça, foi verificado o aumento da quantidade de arestas no algoritmo LGS. Este aumento deve-se a maior utilização do procedimento *Level_Path*.

A variação do grafo para o algoritmo TAMCRA apresentou um aumento proporcional ao número de arestas. Na figura 6.7, pode ser observada que a aplicação de critérios justos pode reduzir o espaço de busca no algoritmo TAMCRA.

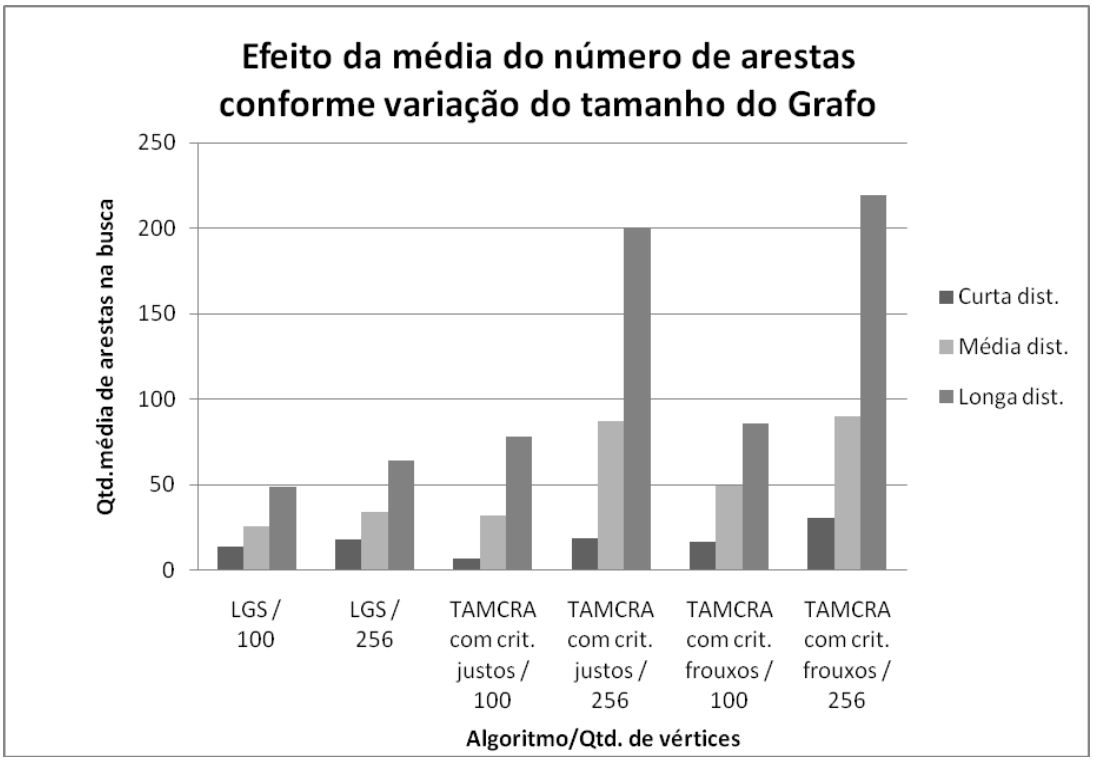


FIG 6.7 – Variação da média da quantidade de arestas conforme o algoritmo e tamanho do grafo.

Média da quantidade de arestas utilizadas na busca			
Algoritmos/regiões	**Curta distância**	**Média distância**	**Longa distância**
LGS com 100 vértices	14,00	25,75	48,60
LGS com 256 vértices	18,47	33,97	64,11
TAMCRA com crit. justos e 100 vértices	6,67	32,25	78,20
TAMCRA com crit. justos e 256 vértices	18,73	87,23	200,19
TAMCRA com crit. frouxos e 100 vértices	16,67	49,25	85,60
TAMCRA com crit. frouxos e 256 vértices	30,60	90,42	219,14

TAB 6.8 – Efeito das médias da quantidade de arestas conforme o algoritmo e regiões.

Na Figura 6.8 e Tabela 6.9 são apresentadas as médias do tempo conforme os algoritmos, critérios, regiões e tamanho do grafo.

Comparando a Figura 6.7 com a Figura 6.8, pode ser observada a relação de aumento de tempo conforme o incremento do número de arestas. Durante a busca de caminhos de média e longa distância, devido a não inclusão dos caminhos dominados para grafos, a redução do tempo de execução para o grafo de 256 vértices foi maior do que no grafo de 100 vértices.

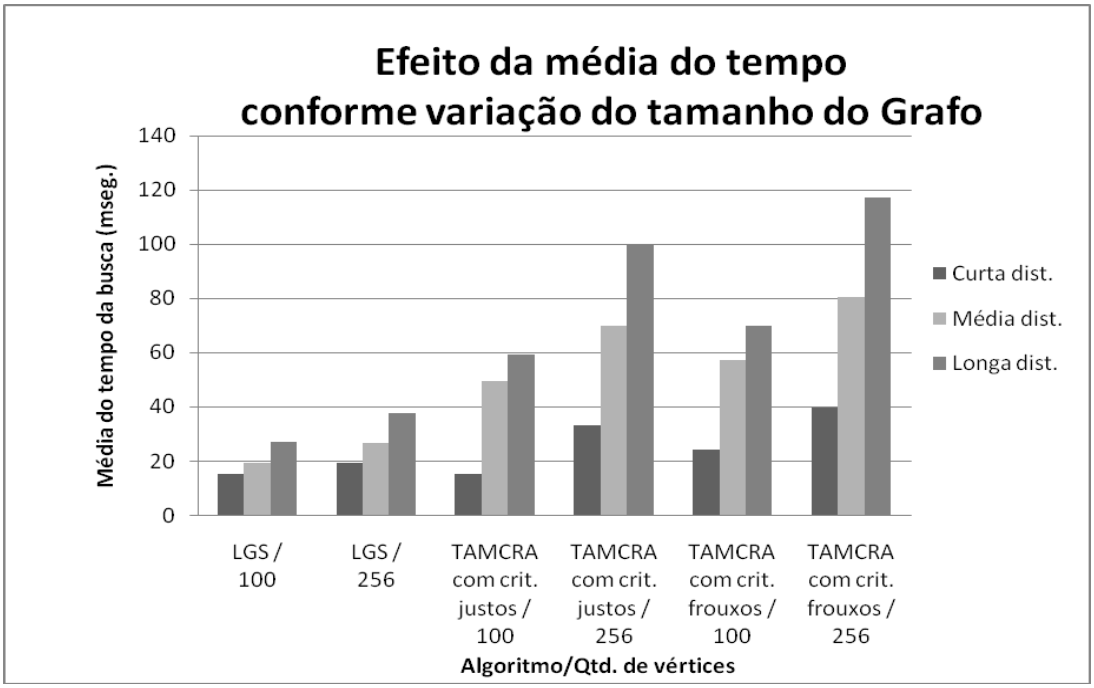


FIG 6.8 – Variação da média do tempo conforme o algoritmo e tamanho do grafo.

Média de tempo da busca (mseg.)			
Algoritmos/regiões	**Curta distância**	**Média distância**	**Longa distância**
LGS com 100 vértices	15,00	19,00	27,10
LGS com 256 vértices	19,27	26,33	37,56
TAMCRA com crit. justos e 100 vértices	15,00	49,58	59,30
TAMCRA com crit. justos e 256 vértices	33,23	69,69	99,40
TAMCRA com crit. frouxos e 100 vértices	24,00	57,08	70,00
TAMCRA com crit. frouxos e 256 vértices	39,47	80,23	117,33

TAB 6.9 – Efeito das médias de tempo da busca conforme o algoritmo e regiões.

6.3 AVALIAÇÃO COMPORTAMENTAL DOS ALGORITMOS

Considerando que o algoritmo *LGS* e o *TAMCRA* apresentaram resultados satisfatórios de desempenho para hierarquização do número de vias e um segundo critério aplicados ao contexto rodoviário, nesta seção são apresentadas avaliações comportamentais que podem gerar melhores caminhos praticáveis e sub-ótimos aderente aos aspectos comportamentais dos seres humanos apresentados em LIU (1996).

Para facilitar a interpretação do efeito comportamental e viabilizarmos a correlação com grafos em árvore, cada vértice foi numerado conforme a ordem apresentada na Figura 6.9. Após a rotulação de cada vértice, foi analisado o caminho entre as regiões de curta, média e longa distância, conforme definido na Seção 5.3.

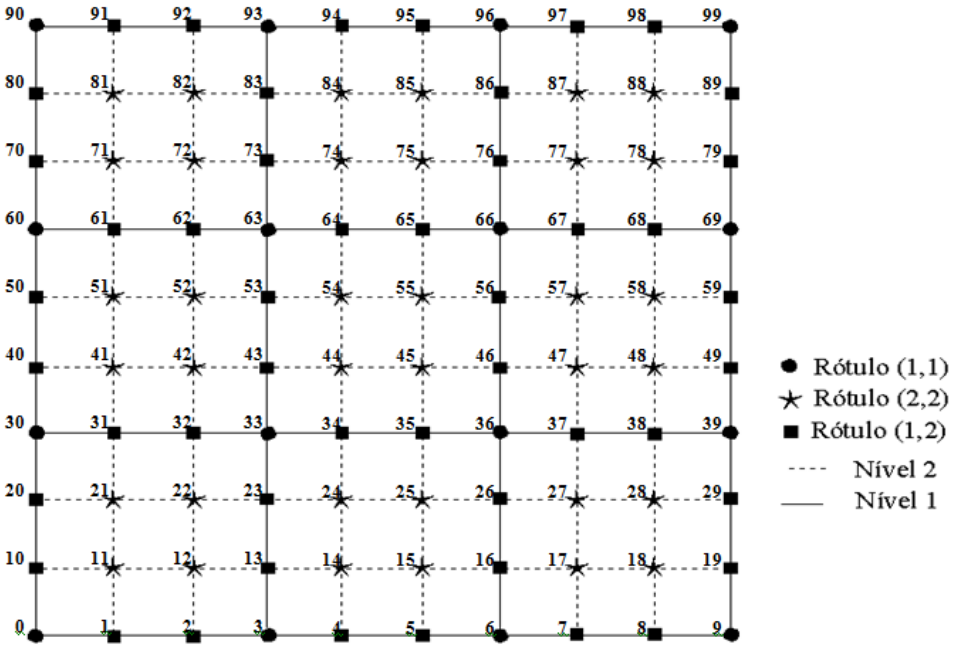


FIG 6.9 - Rotulação dos vértices da treliça estruturada uniforme.

A cada aresta foram atribuídos dois custos. Para a primeira métrica foram atribuídas o custo 1 para as arestas de maior hierarquia e o custo 2 para as arestas de menor hierarquia. Para avaliar o tamanho do caminho percorrido, ao segundo critério aplicamos um custo unitário e uniforme. Através de grafos conforme a Figura 6.8, além da ordem de busca atribuída às arestas, será possível visualizar

internamente aos nós, o rótulo, o cálculo da medida não linear definida na Seção 4.2 e os custos acumulados de cada métrica para as buscas. Uma segunda linha entre os vértices representará o menor caminho praticável.

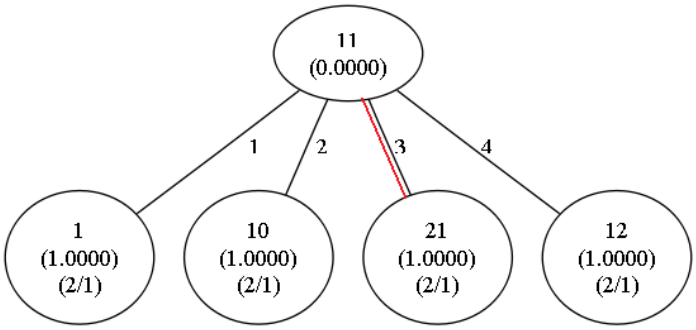


FIG 6.10 – Busca do *TAMCRA* a partir do vértice 11 com critérios 2 e

A verificação comportamental comparativa para a aplicação no contexto rodoviário revelou duas principais características:

- O *TAMCRA* não apresenta restrições de direção.
- O *TAMCRA* apresenta caminhos praticáveis variáveis conforme os critérios.

A seguir nas Subseções 6.3.1 e 6.3.2 serão demonstradas de forma comportamental as características listadas acima.

6.3.1 O TAMCRA NÃO APRESENTA RESTRIÇÕES DE DIREÇÃO.

Conforme esperado, o algoritmo LGS apresentou as restrições de direcionamento e otimalidade citadas em BRAZ(2005). É verificado que estas restrições se devem ao fato do LGS utilizar dois procedimentos distintos. O procedimento *Level_Path* busca os caminhos de nível 2. Posteriormente com informações do primeiro procedimento *Shortest_Path* busca os caminhos de nível 1.

Para ilustrar a falta de direcionamento encontrada em curta, média e longa distância, nas Figuras 6.11 e 6.12 é apresentado caminho mínimo e grafo da busca feita pelo LGS. Nas Figuras 6.13 e 6.14 é mostrado o menor caminho praticável e o grafo para a busca realizada com o *TAMCRA*.

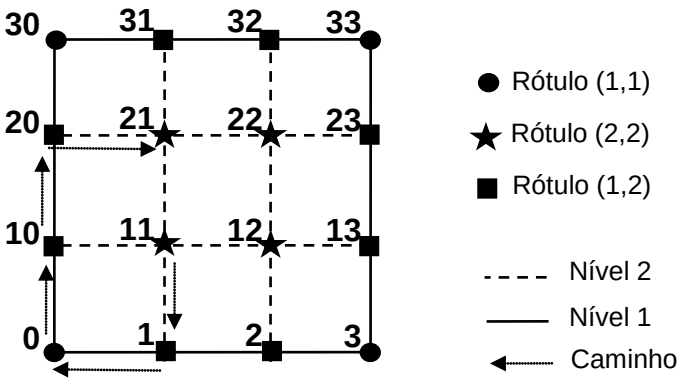


FIG 6.11 – Caminho do 11 ao 21 gerado pelo *LGS*.

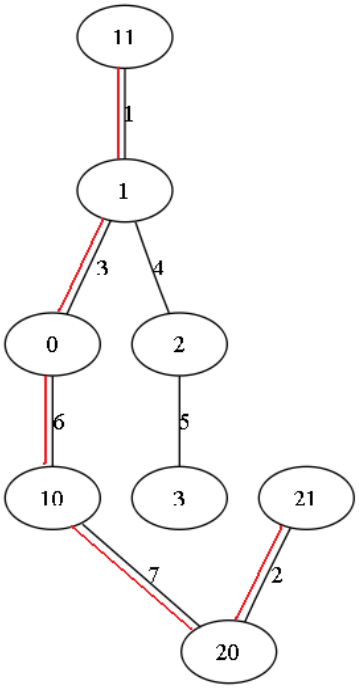


FIG 6.12 Grafo gerado da busca do vértice 11 ao 21 no *LGS*.

No grafo da Figura 6.12 é observado o procedimento *Level_Path*, que a partir do nó 11 encontra primeiramente o vértice 1. Aplicando o mesmo procedimento, no sentido contrário a partir do nó 21, o vértice 20 é encontrado. Tendo em vista que o algoritmo *LGS* já obteve um vértice origem e destino de nível 1, o *LGS* utiliza o procedimento *Shortest_Path* (*Dijkstra*) a partir do vértice 1 até o 20. Este último procedimento somente faz uso de vértices de nível 1. Observa-se que o caminho obtido pelo *LGS* na Figura 6.11 é bem maior do que o caminho praticável encontrado pelo *TAMCRA* na Figura 6.13.

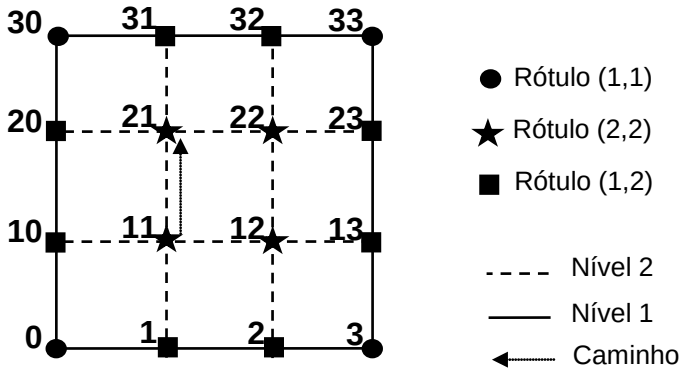


FIG 6.13 – Caminho do 11 ao 21 gerado pelo *TAMCRA*.

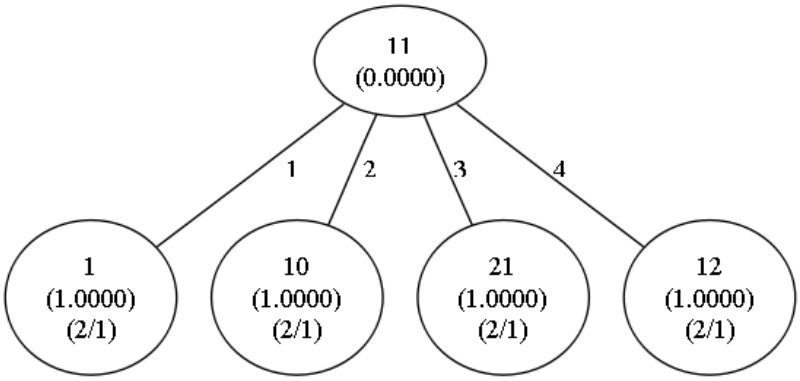


FIG 6.14 Grafo da busca do vértice 11 ao 21 com critérios 2 e 1 no *TAMCRA*.

Conforme pode ser verificado nesta subseção, o *TAMCRA* trata a hierarquia de vias em conjunto com seu segundo critério.

Devido ao fato do algoritmo analisar todos os vértices do grafo, o *TAMCRA* não apresentou restrições de direção.

Da mesma forma que na Subseção 6.2.1, o algoritmo otimiza a busca de caminhos com múltiplos critérios em relação ao *Dijkstra*, os caminhos mínimos encontrados em curta, média e longa distância são otimizados em relação ao *LGS*.

6.3.2 CAMINHOS PRATICÁVEIS VARIÁVEIS CONFORME OS CRITÉRIOS.

Conforme esperado, devido à utilização do conceito de medida não linear conceituado na Subseção 4.2.1, foi possível observar a dinâmica do *TAMCRA* na definição de menores caminhos praticáveis ao longo da execução do algoritmo. Devido a atribuição de pesos uniformes, principalmente em caminhos de longa distância foi possível verificar a determinação do menor caminho praticável conforme rigidez ou folga de um ou outro critério.

Para ilustrar a dinâmica na escolha do menor caminho praticável conforme a valoração dos critérios, as Figuras 6.15 à 6.18 apresentam os caminhos mínimos e o grafo da busca em longa distância do vértice 11 ao 71. Primeiramente, nas Figuras 6.15 e 6.16 são atribuídos os critérios 12 e 6, onde pretende-se apresentar a determinação do caminho aplicando uma maior rigidez do critério distância (2º critério). Por último, nas Figuras 6.17 e 6.18 são atribuídos os critérios 10 e 8. Com estes critérios pretende-se apresentar a dominância do critério de hierarquia das vias (1º critério) em relação à distância (2º critério).

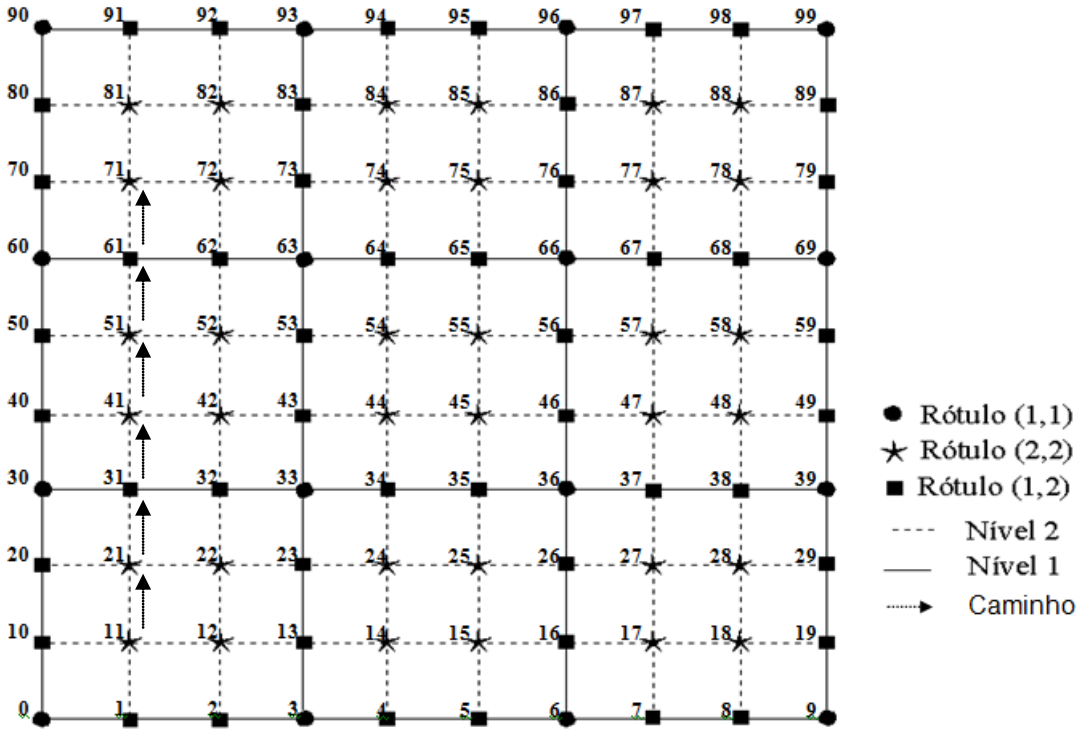


FIG 6.15 - Caminho do vértice 11 ao 71 com critérios 12 e 6.

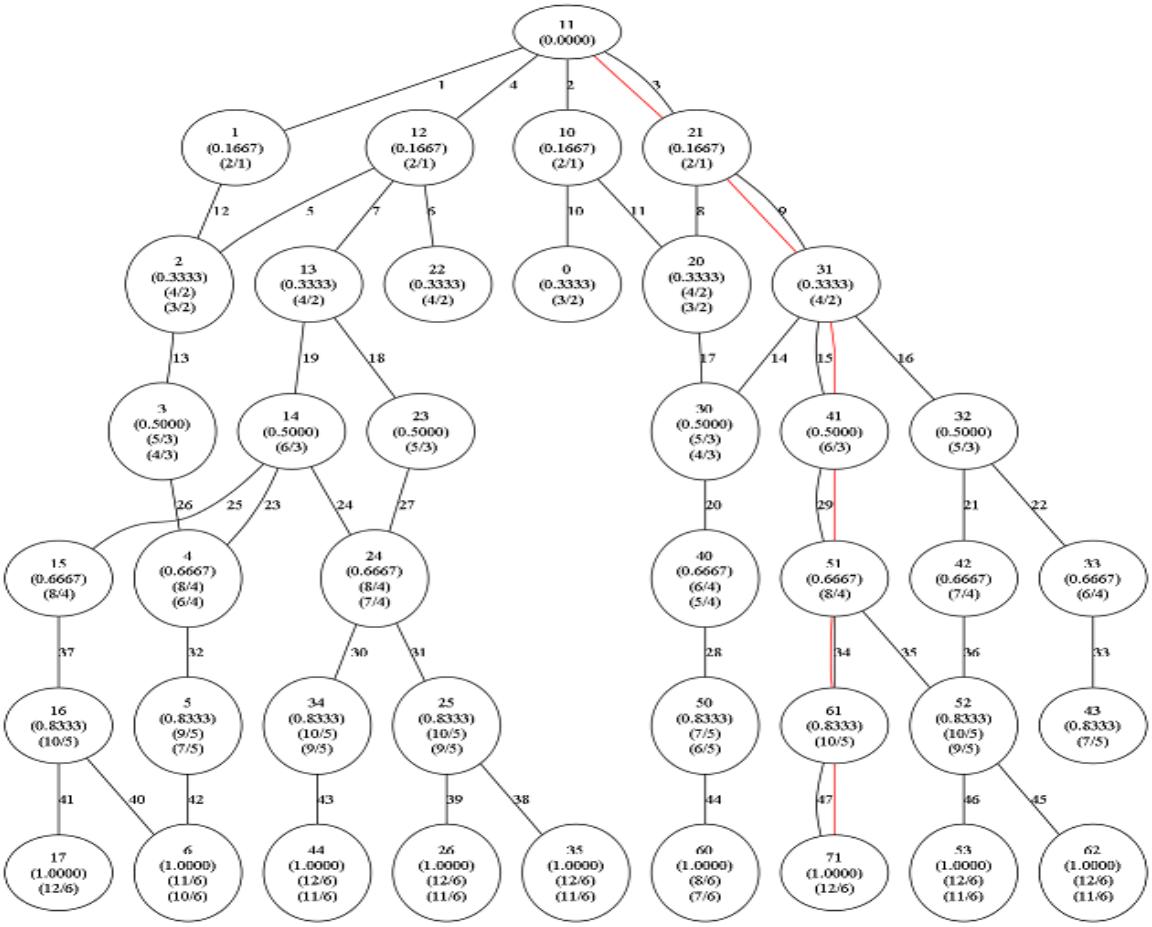


FIG 6.16 - Caminho praticável do vértice 11 ao 71 com critérios 12 e 6 gerado pelo *TAMCRA*.

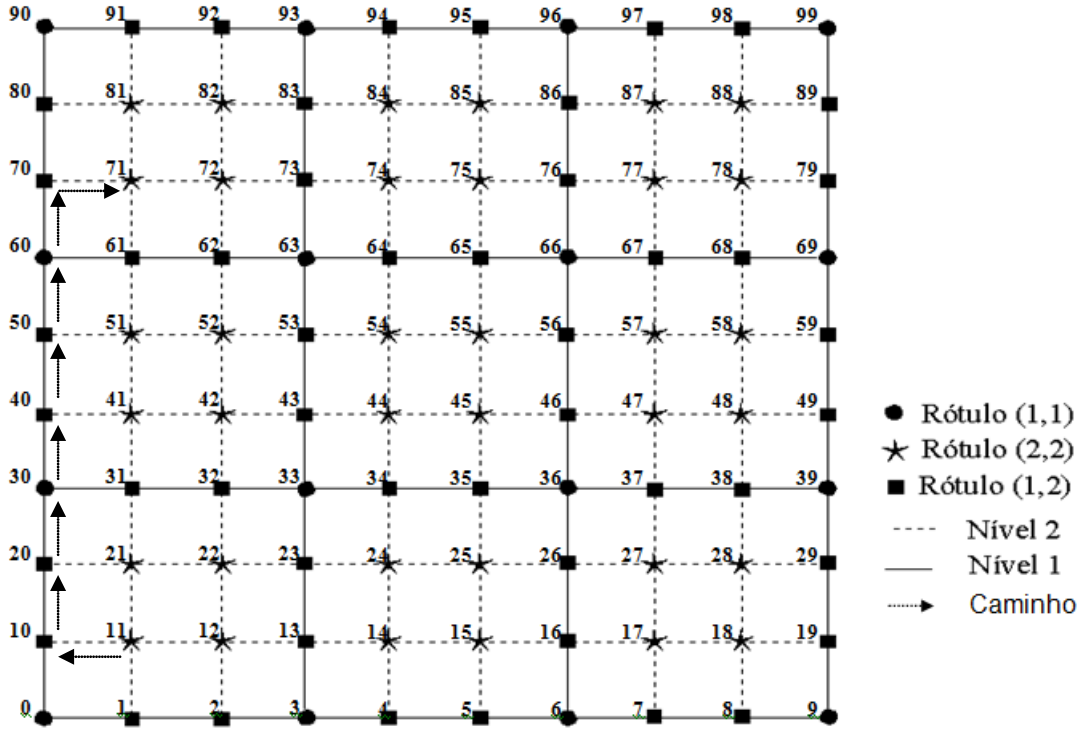


FIG 6.17 - Caminho do vértice 11 ao 71 com critérios 12 e 6.

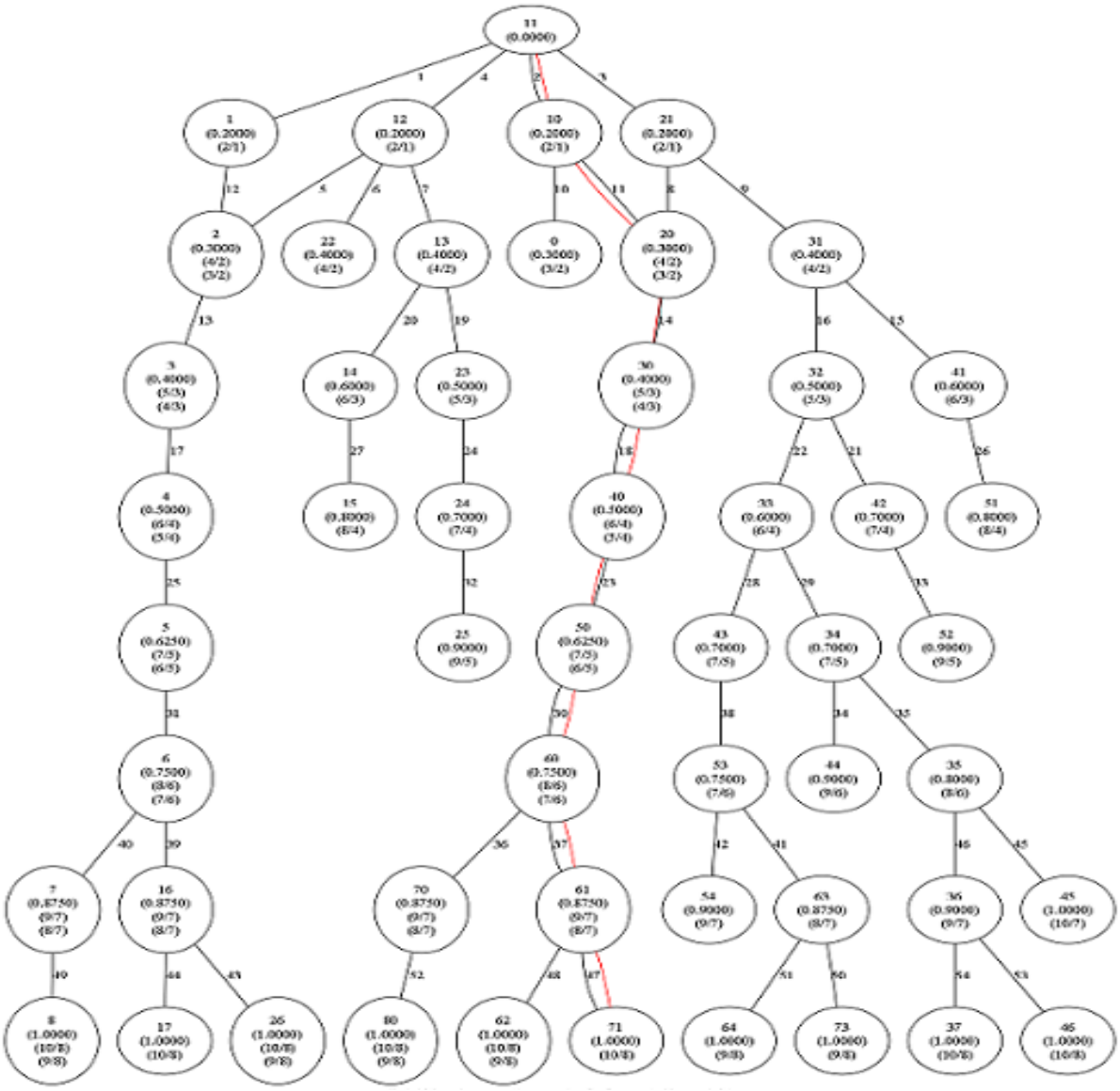


FIG 6.18 - Caminho praticável do vértice 11 ao 71 com critérios 10 e 8 gerado pelo *TAMCRA*.

Conforme verificado na Figura 6.15, tendo em vista a rigidez do segundo critério (distância=6), o *TAMCRA* encontra o menor caminho entre os nós 11 ao 71. Com objetivo de demonstrar que é possível priorizar a busca de caminhos praticáveis fornecendo prioridade às ruas de maior hierarquia, aumentamos a rigidez do primeiro critério (hierarquia da via = 10) e afrouxamos o segundo critério (distância=8). É possível observar na Figura 6.17, que através da determinação de critérios pré-definidos é possível atender aos aspectos comportamentais citados por LIU (1996), onde os seres humanos preferem dirigir por ruas principais e por ruas com as quais já estão familiarizados.

7 CONCLUSÕES

Esta dissertação apresenta como principal contribuição, um estudo do problema do roteamento com múltiplos critérios que possa ser aplicável a roteamento de veículos em malhas viárias urbanas. No Capítulo 2 são apresentados os principais conceitos, algoritmos e suas respectivas complexidades que abordam o problema de roteamento com múltiplos critérios na literatura.

Após a revisão bibliográfica e identificados os principais conceitos dos algoritmos aplicáveis ao roteamento de veículos com múltiplos critérios, é utilizado o algoritmo *TAMCRA* descrito no Capítulo 4. Com uma complexidade de tempo polinomial, este algoritmo reduz o problema NP completo a um problema mais simples que pode ser resolvido em tempo polinomial.

Após a verificação que o desempenho do *TAMCRA* pode ser previsível através da limitação dos *k* menores caminhos e que, na existência de caminhos, o algoritmo garante encontrar pelo menos um caminho praticável, definimos a heurística *LGS* como referência para nossas comparações. Apesar da heurística apresentar caminhos sub-ótimos, o *LGS* atende ao nosso principal interesse por possibilitar a priorização das vias através de grafos em níveis.

Para realizarmos a análise experimental comparando o *TAMCRA* ao *LGS*, no Capítulo 5 modelamos a solução do problema e definimos que a experimentação a ser realizada para caminhos de curta, média e longa distância em uma estrutura padrão e uniforme denominada treliça estruturada uniforme.

Após a definição de um modelo computacional semelhante às malhas rodoviárias urbanas, no Capítulo 6 finalmente é apresentada uma análise experimental utilizando múltiplos critérios. Para a realização da análise experimental, primeiramente foi implementado um gerador de grafo aderente ao modelo definido. Posteriormente, a implementação dos algoritmos *TAMCRA* e *LGS* consideraram o mesmo grafo como base para suas buscas.

A análise experimental se baseou em buscas de menores caminhos praticáveis entre um ponto de partida e um ponto de chegada das regiões definidas para o modelo. Através de análises comportamental e de performance apresentadas no Capítulo 6, os resultados obtidos comprovaram alguns efeitos já esperados:

a) O *TAMCRA* otimiza a busca de caminhos para o problema de roteamento com múltiplos critérios.
b) O *LGS* apresenta restrições de direcionamento e otimalidade.
c) O *TAMCRA* garante encontrar caminhos praticáveis, caso este exista.
d) O *TAMCRA* pode atender às expectativas comportamentais dos seres humanos através do equilíbrio do peso em seus critérios.

Tendo em vista que estamos trabalhando com um problema NP Completo e que a complexidade $O(kV.log(kV)+k^2zE)$ do TAMCRA está diretamente relacionada com o k, vale a pena ressaltar duas informações importantes. A primeira é que a implementação apresentada do TAMCRA nesta dissertação não é exata devido a utilização do $k = 2$. A segunda informação é que NEVE (1998) através de análises estatísticas demonstra, através de experimentos com 100 nós e duas métricas aditivas, que utilizado $k = 4$ é possível garantir 99% de chance em encontrar a solução para o menor caminho do problema MCP.

Como sugestão para trabalhos futuros, deixamos a criação de um algoritmo híbrido e dinâmico para o roteamento de veículos. Este algoritmo seria capaz de aprender o comportamento dos seres humanos em malhas rodoviárias. Unindo a aprendizagem ao conhecimento da estrutura viária de uma região real, é possível mapear perfis por usuário e por região.

8 REFERÊNCIAS BIBLIOGRÁFICAS

AHUJA, R.K., MAGNANTI, T.L. e ORLIN, J.B. **Network flows: Theory, algorithms, and applications.** Prentice Hall, 1993

BELLMAN, R. **Dynamic Programming.** Universidade de Princeton, Princeton University Press, New Jersey, 1957.

BRAZ, C. e ARAÚJO, L. **Adaptando um algoritmo de busca em grafos em níveis para uso em mapas rodoviários.** Em XXXVII Simpósio Brasileiro de Pesquisa Operacional, 2005.

BRAZ, C. **Algoritmos de roteamento em malhas rodoviárias com sistemas de informações geográficas.** Dissertação de mestrado, Instituto Militar de Engenharia, 2006.

CHEN, S., NAHRSTEDT, K. **On Finding Multi Constrained Paths**. Proceedings of ICC'98, p.874-879, 1998.

CHERKASSKY, B.V., GOLDBERG, A.V. e RADIZIK, T. **Shortest paths algorithms: theory and experimental evaluation.** Em SODA: ACM-SIAM Symposium on Discrete Algorithms, v.73, p.129-174, 1996.

CHERKASSKY, B.V., GOLDBERG, A.V. e SILVERSTEIN, C. **Buckets, heaps, lists and monotone priority queues.** Siam Journal on Computing, v.28, p.1326-1346, 1999.

CHONG, E.I., MADDILA, S. e MORLEY. S. **On finding single-source single-destination k shortest paths.** Journal of Computing and Information, special issue ICCI'95, p.40-47, Jul.1995.

CORMEN, T.H., LEISERSON, C.E., RIVEST, R.L.;STEIN,C. **Algoritmos: Teoria e Prática.**, Campus, Segunda Edição, 2002.

DIJKSTRA, E.W. **A note on two problems in connexion with graphs.** Numerische Mathematik, v.1, p.269-271, 1959.

FLOYD, R.W. **Algorithm 97 (shortest path)**. Communications of the ACM, v.5, p.345, 1962.

GALLO, G. E PALLOTTINO, **S. Shortest paths algorithms**, Annals of Operations Research, v.13, p.3-79, 1988.

GAREY , M.R. E JOHNSON, D.S. **Computers and Intractability - A Guide to the Theory of the NP-Completeness.**, W. H. Freeman & Co., San Francisco, CA, USA, 1979, ISBN:0716710447.

GOLUMBIC, M.C., **Algorithmic Graph Theory and Perfect Graphs.,** Academic Press Inc., 1980.

HARARY, F. **Graph Theory**. Addison-Wesley, 1ª edição, 1969.

HENIG, M.I., **The shortest path problem with two objective functions.** European Journal of Operational Research, v.25, n.2, p.281-291, 1986.

IWATA, A., IZMAILOV, R., LEE, D-S., SENGUPTA, B., RAMAMURTHY, G. E SUZUKI, H. **ATM routing algorithms with multiple QoS requirements for multimedia internetworking.** IEICE Trans. Commun., v. E79-B, n.8, 1996.

JAFFE, J.M. **Algorithms for finding paths with multiple constraints**. Networks, v.14, p.95-116, 1984.

JOHNSON, D.S. **A Theoretician's Guide to the Experimental Analysis of Algorithms**. Nas continuações do quinto e sexto "DIMACS Implementation Challenges", p.244-256, 2002.

KLEINROCK, L., **The Latency/Bandwidth Tradeoff in Gigabit Networks.** IEEE Communications, Abr. 1992.

KORKMAZ, T. E KRUNZ, M. **A randomized algorithm for finding a path subject to multiple QoS requirements.** Comput. Networks, v.36, p.251-268, 2001.

KORKMAZ, T. E KRUNZ, M. **Multi-constrained optimal path selection.** Proc. IEEE INFOCOM, v.2, p.834-843, 2001.

KORKMAZ, T., KRUNZ, M. **Routing Multimedia Traffic with QoS Guarantees: IEEE Transactions on Multimedia**, v.5, n.3, p.429-443, 2003.

KUIPERS F.A. ET AL. **An Overview of Constraint-Based Path Selection Algorithms for QoS Routing**. IEEE Commun. Mag., v.40, n.12, p.50-5, Dez. 2002.

KUIPERS F.A. **Hop-by-Hop destination based routing with quality of service constraints**, 2000, Tese de mestrado - Delft Univ. Technol., Jun. 2000.

KUIPERS, F.A. **Quality of Service Routing in the Internet. Theory, Complexity and Algrithms.** Tese de Pós-Doutorado na Univ.Netherlands, Delft University Press, Set. 2004, ISBN 90-407-2523-3.

KUIPERS, F.A., e MIEGHEM, P.V. **Bi-directional search in QoS routing**, Em Proc. of the fourth COST 263 International Workshop on Quality of Future Internet Services - QofIS2003, p.102-111, Out. 2003.

KUIPERS, F.A., KORKMAZ, T., KRUNZ, M. e MIEGHEM, P.V. **Performance Evaluation of Constraint-Based Path Selection Algorithms**. IEEE Network, v.18, n.5, p.16-23, 2004.

LEGNER, M. **Map-based geographic forwarding in vehicular networks.** Universidade de Stuttgart, 2002.

LIN S. **Computer solutions of the traveling salesman problem.** Bell Syst. Tech. J., v.44, n.10, p.2245-2269, 1965.

LIU, B. **Intelligent route finding: Combining knowledge and cases and an efficient search algorithm**. Em European Conference on Artificial Intelligence, p.380-384, 1996.

LIU, G. E RAMAKRISHNAN, K. G. **A*Prune: An Algorithm for Finding k-shortest Paths Subject to Multiple Constraints.** Proc. IEEE INFOCOM Conf., v.2, p.743-749, Abr. 2001.

LOUNDON K., **Dominando algoritmos com C,** Ciência Moderna, 2000, ISBN: 8573930764.

MEIGHEM, P.V., NEVE, H.D E KUIPERS, F.A. **Hop-By-Hop Quality of Service Routing**. The International Journal of Computer and Telecommunications Networking, v.37, p.407-423, 2001.

MIEGHEM, P. e KUIPERS F.A. **On the complexity of QoS routing**. Comput. Commun., v.26, n.4, p.376-387, Mar. 2003.

MIEGHEM, P. **Paths in the simple random graph and the Waxman graph**. Probability Eng. Inform. Sci. (PEIS), v.15, p.535-555, 2001.

MIEGHEM, P., NEVE, H., e KUIPERS F.A. **Hop-by-hop quality of service routing**. Comput. Networks, v. 37, n.3-4, p.407-423, 2001.

MIEGHEM, P.V., KUIPERS, F.A. **Concepts of exact QoS routing algorithms.** IEEE/ACM Transactions on Networking (TON), v.12, n.5, p.851-864, 2004.

NEVE H., MIEGHEM, P. **A multiple quality of service routing algorithm for PNNI** , IEEE ATM'98 Workshop, p.306-314, 1998.

NEVE, H. e MIEGHEM P.V. **TAMCRA: A tunable accuracy multiple constraints routing algorithm.** Comput. Commun., n.23, p.667-679, 2000.

NEWELL A. E ERNST G., **The search for generality.** Em Proc. IFIP Congr., v.1, p.17-24, 1965.

RODRIGUES, C. K. S. **O problema do caminho mais rápido.** Dissertação de mestrado, Instituto Militar de Engenharia, 1999.

SEDGWICK, R. E VITTER, J. **Shortest path in euclidean graphs.** Algorithmica, v.1, n.1, p.31-48, 1986.

SHAPIRO J., WAXMAN J. E NIR,D. **Level graphs and approximate shortest path algorithms.** Networks, n.22, p.691-717, 1992.

SHEKRAR s. e CHAWLA, S. **Spatial Database a Tour.** Prentice Hall, Pearson Education Inc., Upper Saddle River, New Jersey 07458, Primeira edição, 2003.

SZCZEŚNIAK, I. **Shortest Path Algorithms for Public Transport.** Socrates/Erasmus Exchange Programme, Tese de mestrado na Trent University, The Department of Computing, 2000.

SZWARCFITER, J. e MARKENZON, L. **Estruturas de Dados e Seus Algoritmos. ,** LTC, 2ª edição, 1994.

SZWARCFITER, J. **Grafos e Algoritmos Computacionais.** Editora Campus, 1986.

WANG, Z. E CROWCROFT, J. **Qos routing for supporting resource reservation.**, IEEE JSAC, Setembro de 1996.

WANG, Z. e CROWCROFT, J. **Quality-of-service routing for supporting multimedia applications**. IEEE J. Select. Areas Commun., v.14, n.7, p.1228-1234, Set. 1996.

WARSHALL, S., **A theorem on matrices.** Journal of the ACM, v.9, n.1, p.11-12, 1962.

YUAN X. **Heuristic algorithms for multiconstrained quality-of-service routing**. IEEE/ACM Trans. Networking, v.10, p.244-256, Abr. 2002.

ZHAN, F. B. e NOON, C.E. **Shortest Path Algorithms: An Evaluation Using Real Road Networks.** Transportation Science, v.32, n.1, p.65-73, 1996.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo