Compromisso entre algoritmos de roteamento em redes tolerantes a atrasos e desconexões

8 de fevereiro de 2010

Trabalho de Formatura Supervisionado

Aluno: Cássia Garcia Ferreira
Supervisor: Prof. Dr. Alfredo Goldman vel Lejbman

Sumário

Parte I

Parte Técnica

Observação

A pesquisa sobre Redes Tolerantes a Atrasos e Desconexões está sendo feita em um grupo de cinco alunos. Este grupo consiste nos alunos Adriano Tabarelli, Caio Cestari Silva, Cássia Garcia Ferreira, César Gamboa Machado e Paulo Henrique Floriano. Apesar de termos dividido os assuntos pesquisados para o desenvolvimento das monografias em grupos menores, todos tem uma base em comum.

1 Introdução

Com o desenvolvimento das redes sem fio, novas estruturas de conexão começaram a surgir. Um possível cenário são as denominadas Redes Tolerantes a Atrasos e Desconexões ou DTNs (Delay-and-Disruption Tolerant Networks)[1]. Uma DTN é uma rede na qual os nós não estão necessariamente conectados o tempo todo, isto é, uma conexão entre dois nós pode estar ativa ou não em um dado instante. Se o comportamento desta rede for conhecido, ou seja, se soubermos exatamente os períodos em que cada conexão está ativa, é possível modelá-la através de Grafos Evolutivos (Evolving Graphs)[6].

Um grafo evolutivo é um conjunto de vértices e arestas. Cada aresta possui um conjunto de intervalos nos quais esta aresta está ativa. Neste cenário, um simples caminho no grafo não é suficiente para conectar dois nós. Para isso, esse caminho no tempo é definido como jornada. Uma jornada é um conjunto de arestas e um conjunto de tempos. Para cada aresta há um instante de tempo na qual esta é atravessada.

Nestes grafos, o problema de encontrar uma jornada ótima entre dois nós é dividido em três subproblemas, pois podemos otimizar três parâmetros:

- Jornada mais curta, *Shortest Journey*, que minimiza o número de arestas utilizadas;
- Jornada mais rápida, *Foremost Journey*, que minimiza o tempo de chegada, isto é, chega no instante mais cedo;
- Jornada de menor tempo em trânsito, *Fastest Journey*, que minimiza o tempo entre a saída e a chegada.

Algoritmos que encontram essas jornadas foram propostos em [2]. Como o algoritmo para encontrar a jornada Fastest, até este estudo, não foi implementado, estudaremos apenas o comportamento dos demais.

Muitas vezes as jornadas Shortest e Foremost podem possuir valores impraticáveis. Por exemplo: uma jornada curta que demora demais ou uma jornada rápida que passa por um número muito grande de nós, e seria de maior interesse utilizar uma jornada intermediária que poderia ser encontrada caso existisse um compromisso entre as jornadas conhecidas.

Para encontrar este compromisso usaremos o conceito de Eficiência de Pareto e construiremos uma Curva de Pareto para mostrar que, além destas duas jornadas, podemos encontrar outras jornadas ótimas intermediárias que podem ser melhores para certos contextos. Essas jornadas são consideradas ótimas pois a Curva de Pareto otimiza dois parâmetros simultaneamente. Para o caso, otimizamos o número de arestas e o instante de chegada das jornadas.

Após a obtenção de todas as jornadas ótimas, ainda temos o problema de determinar uma única para o contexto. Caso seja possível dar custos para as arestas e para os instantes

de tempo, podemos encontrar uma jornada ótima de menor custo. Assim, encontramos a melhor jornada a ser usada.

2 Redes Tolerantes a Atrasos e Desconexões

Com o desenvolvimento das redes móveis sem fio, novos modelos começaram a surgir. Uma deles são as chamadas Redes Tolerantes a Atrasos e Desconexões, também conhecidas como DTNs (Delay-and-Disruption Tolerant Networks).

Nas DTNs, as conexões não são perenes e a mobilidade causa frequentes desconexões entre os nós. Além disso, pode haver alguma transmissão de mensagens que não se complete totalmente, sendo esta outra característica que recebe tratamento especial por parte deste tipo de rede.

Um exemplo clássico no qual ocorrem atrasos e desconexões é o contexto interespacial. Nele, desconexões são frequentes devido à posição física dos elementos que trocam as informações (periodicamente visíveis e invisíveis um ao outro) e os atrasos têm origem na grande distância entre os nós da rede.

Para resolver estes problemas, é necessário que haja custódia de mensagens, ou seja, transferência da responsabilidade de entrega de um nó para outro, além de se assumir periodicidade e ciclicidade de posições relativas entre os nós, tornando a representação das Redes Tolerantes a Atrasos mais facilmente alcançável. Quando se conhece ou se assume o comportamento de uma rede específica, os Grafos Evolutivos a representam de forma satisfatória, o que será visto na próxima seção.

3 Grafos Evolutivos

Dada uma Rede Tolerante a Atrasos e Desconexões da qual são conhecidos todos os períodos de conexões ativas, podemos guardar essas informações num Grafo Evolutivo.

Definição 1

Sejam $G = (V_G , E_G)$ um grafo e $S_G = G_0 , G_1 , \ldots , G_\tau$ ($\tau \in N$) um conjunto ordenado de subgrafos de G tal que $G = \bigcup_{i=1}^{\tau} G_i$. O sistema $\mathcal{G} = (G, S_G)$ é chamado ***Grafo Evolutivo****.*

Os vértices do grafo evolutivo $\mathcal{G}$ são correspondentes aos nós da DTN. Assim, podemos dizer que cada subgrafo G_τ é o conjunto dos vértices de $\mathcal{G}$ com arestas que representam as conexões ativas no instante de tempo τ.

O grafo evolutivo pode então ser representado por um grafo em que cada aresta tem uma lista de instantes que ela está ativa, como podemos ver na figura 1.

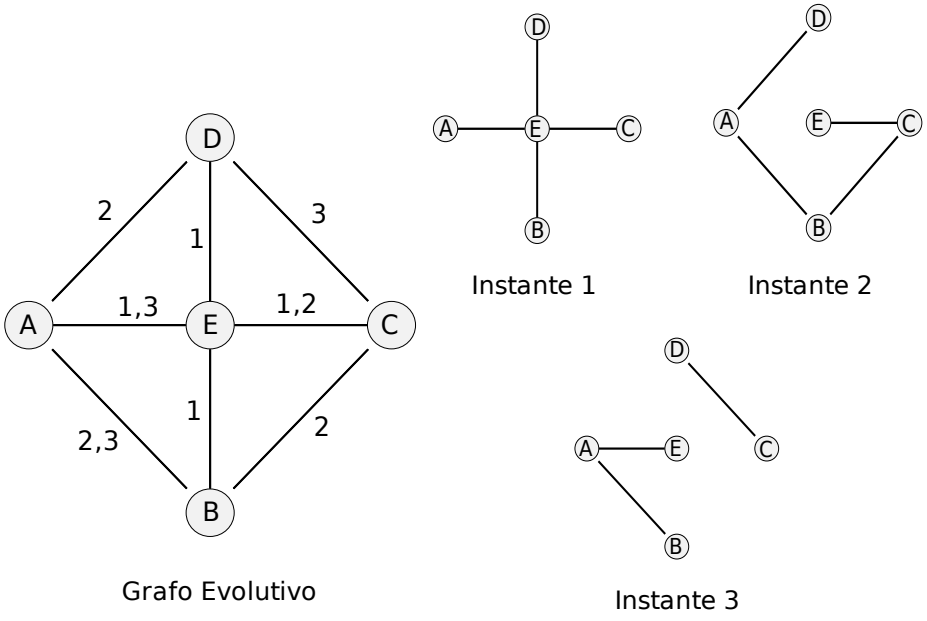

Figura 1: Grafo Evolutivo e seus instantes

Um caminho de um vértice s a t em um Grafo Evolutivo é um conjunto de arestas com seus respectivos instantes a serem percorridas. Chamamos esse caminho de *Jornada*.

4 Jornadas

Uma *Jornada* em um Grafo Evolutivo é formado por um conjunto de arestas e instantes de tempo nos quais cada aresta deve ser percorrida. Este conjunto de instantes é não decrescente.

Definição 2

Seja $R = e_1, e_2, \ldots, e_k$ ($e_i \in E_G$) um caminho em G, e $R_\sigma = \sigma_1, \sigma_2, ..., \sigma_k$ ($\sigma_i \in [1, \tau]$, $\sigma_i \leq \sigma_j \ \forall i < j$) um agendamento indicando quando cada arco de R deve ser atravessado. Então dizemos $\mathcal{J} = (R, R_\sigma)$ uma jornada em $\mathcal{G}$.

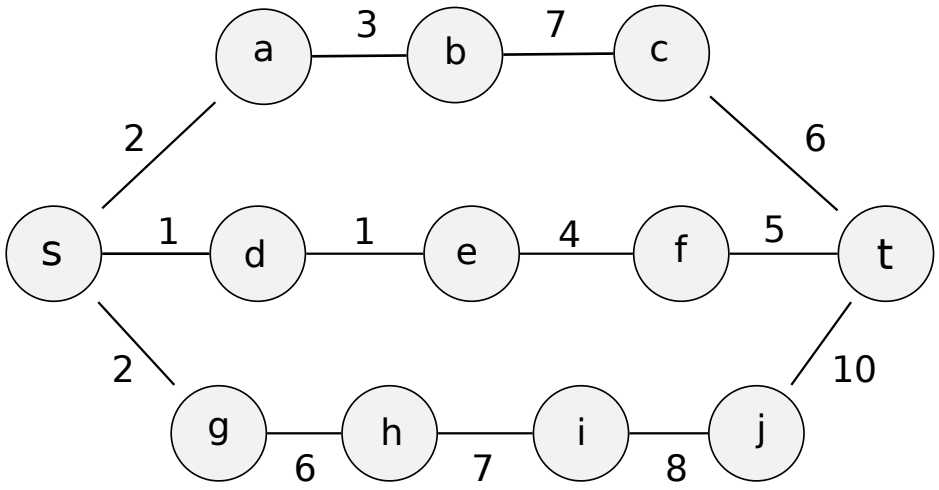

Figura 2: Grafo Evolutivo e algumas jornadas

Note que:

- $s \xrightarrow{2} a \xrightarrow{3} b \xrightarrow{7} c \xrightarrow{6} t$ não é uma jornada de s a t pois os instantes de tempo não estão em ordem não decrescente.
- $s \xrightarrow{1} a \xrightarrow{1} b \xrightarrow{4} c \xrightarrow{5} t$ é uma jornada de s a t.
- $s \xrightarrow{2} g \xrightarrow{6} h \xrightarrow{7} i \xrightarrow{8} j \xrightarrow{10} t$ é uma jornada de s a t.

4.1 Jornada Shortest

Chamaremos de *Shortest* a jornada que otimiza o número de arestas, ou seja, a jornada de menor tamanho.

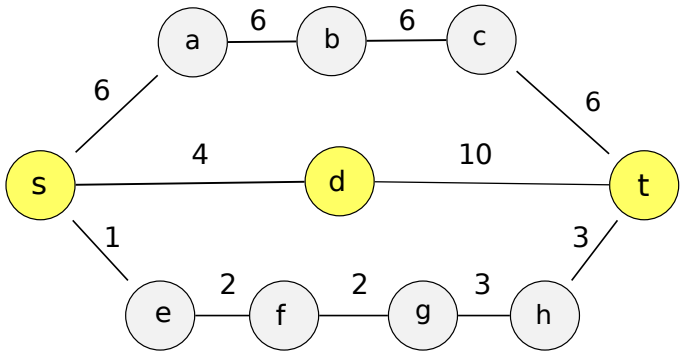

Figura 3: Grafo Evolutivo e jornada Shortest $s \xrightarrow{4} d \xrightarrow{10} t$

4.2 Jornada Foremost

Chamaremos de *Foremost* a jornada que otimiza o tempo de chegada, ou seja, a jornada que termina no menor instante de tempo.

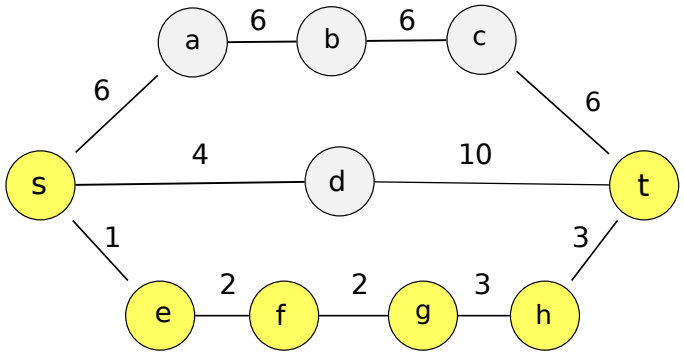

Figura 4: Grafo Evolutivo e jornada Foremost $s \xrightarrow{1} e \xrightarrow{2} f \xrightarrow{2} g \xrightarrow{3} h \xrightarrow{3} t$

4.3 Jornada Fastest

Chamaremos de *Fastest* a jornada que otimiza o tempo em percurso, ou seja, a jornada que apresenta menor tempo em trânsito.

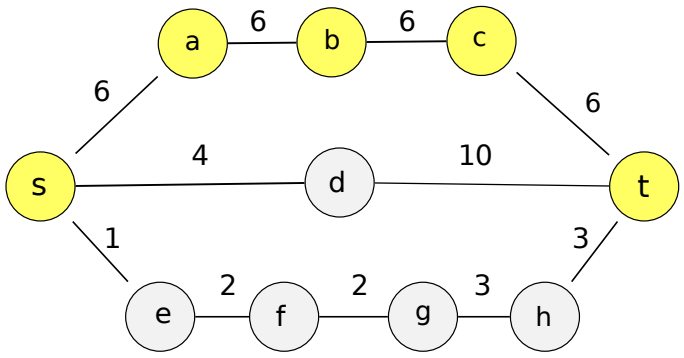

Figura 5: Grafo Evolutivo e jornada Fastest $s \xrightarrow{6} a \xrightarrow{6} b \xrightarrow{6} c \xrightarrow{6} t$

Foram propostos algoritmos para calcular as jornadas ótimas *Foremost*, *Shortest* e *Fastest* [2] de um vértice a outro usando Grafos Evolutivos. Porém, existem outras jornadas que não são encontradas por esses algoritmos que, dependendo do contexto, poderiam ser consideradas melhores.

Como o algoritmo *Fastest Journey* não foi implementado em um contexto prático até este estudo, apenas proposto de forma teórica, observaremos o comportamento do *Foremost Journey* e *Shortest Journey*.

5 Jornadas ótimas

Como já dito, observaremos apenas o comportamento de duas jornadas ótimas: a jornada *Shortest* e a jornada *Foremost*. Dissemos que, dependendo do contexto, poderia ser de maior interesse usar outra jornada ainda não conhecida. Vamos então ao seguinte exemplo:

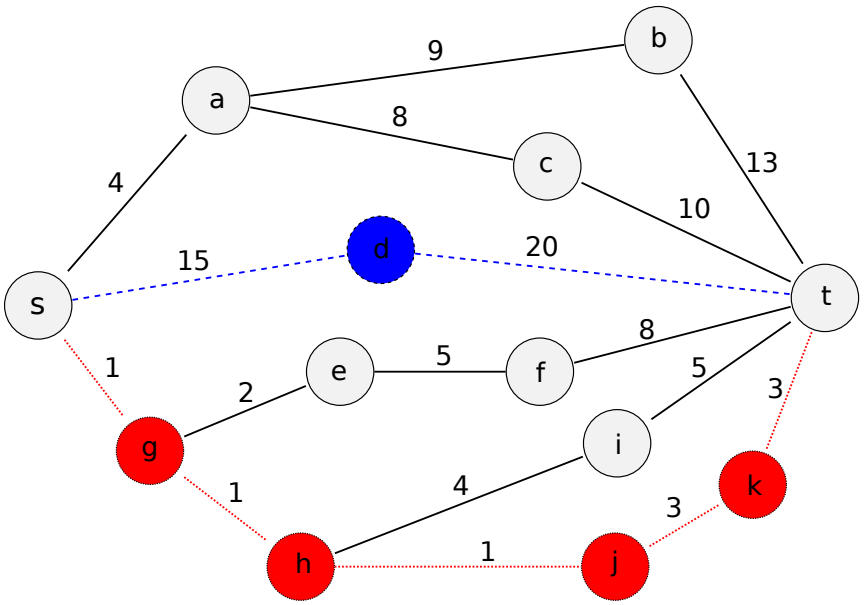

Figura 6: Grafo Evolutivo e jornadas *Shortest* (acima em azul tracejado) e *Foremost* (abaixo em vermelho pontilhado) de s a t

Suponha que queremos uma jornada com no máximo tempo de chegada 10 e no máximo 4 saltos (número de arestas), que chegue o mais cedo possível com o menor número de arestas possíveis. As jornadas Shortest e Foremost não satisfazem o contexto:

- Jornada Shortest demora demais (instante 20).
- Jornada Foremost é muito longa (5 saltos).

Assim, as jornadas que satisfariam seriam:

- $s \xrightarrow{4} \bigcirc \xrightarrow{8} \bigcirc \xrightarrow{10} t$
- $s \xrightarrow{1} \bigcirc \xrightarrow{2} \bigcirc \xrightarrow{5} \bigcirc \xrightarrow{8} t$
- $s \xrightarrow{1} \bigcirc \xrightarrow{1} \bigcirc \xrightarrow{4} \bigcirc \xrightarrow{5} t$

Observe que as duas últimas jornadas possuem mesmo tamanho (4 arestas) porém a $s \xrightarrow{1} \bigcirc \xrightarrow{2} \bigcirc \xrightarrow{5} \bigcirc \xrightarrow{8} t$ tem instante de chegada maior que a $s \xrightarrow{1} \bigcirc \xrightarrow{1} \bigcirc \xrightarrow{4} \bigcirc \xrightarrow{5} t$. Logo,

a última é ótima para esse tamanho e tempo de chegada (não há outra jornada de menor tamanho com mesmo instante de chegada nem outra com mesmo tamanho e menor instante de chegada).

Logo, para encontrar estas outras jornadas, a fim de chegar em uma que satisfaça o contexto, caímos primeiramente no seguinte problema: otimizar dois parâmetros ao mesmo tempo, ou seja, otimizar o número de saltos (arestas) e o tempo de chegada das jornadas ao mesmo tempo. Podemos encontrar estas jornadas ótimas caso exista um compromisso entre as jornadas ótimas *Shortest* e *Foremost*.

6 Compromisso entre jornadas Shortest e Foremost

Caso seja possível encontrar jornadas de um vértice s a um vértice t nas quais não é possível diminuir o número de arestas sem aumentar o tempo de chegada nem diminuir o tempo de chegada sem aumentar o número de arestas, encontramos jornadas que são consideradas ótimas em relação a estes dois parâmetros.

Primeiramente vamos encontrar essas jornadas teoricamente a partir do compromisso entre as jornadas *Shortest* e *Foremost*. Para isso, usaremos o conceito de *Eficiência de Pareto* e construiremos uma *Curva de Pareto*[3].

6.1 Curva de Pareto

Definição 2

Eficiência de Pareto *ou Ótimo de Pareto corresponde a uma situação na qual é impossível melhorar um parâmetro sem prejudicar algum outro.*

Definição 3

Curva de Pareto *é uma curva formada por pontos ótimos propostos pela Eficiência de Pareto (ou Ótimo de Pareto), onde para qualquer situação não é possível melhorar um parâmetro sem prejudicar o outro.*

Para encontrar o nosso compromisso, a Curva de Pareto terá como coordenadas o número de arestas e o tempo de duração da jornada em questão, que são os dois parâmetros que queremos otimizar.

A partir de todas as jornadas possíveis, a curva conterá apenas os pontos ótimos como explicados na Definição 2. Não é possível andar pela curva em direção a otimizar uma das coordenadas de modo a não prejudicar a outra.

6.1.1 Construção da Curva de Pareto: passo-a-passo

Temos o seguinte Grafo Evolutivo:

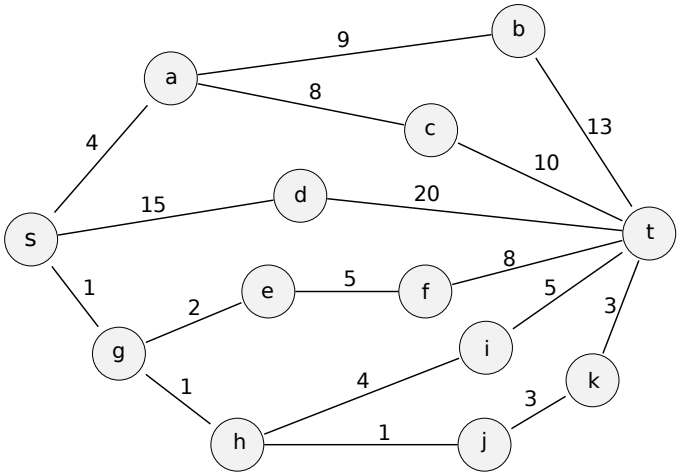

Figura 7: Grafo Evolutivo $\mathcal{G}$

- Colocamos primeiro todos os pontos correspondentes a todas as jornadas $\mathcal{J}$ do nó s a t do Grafo Evolutivo $\mathcal{G}$.

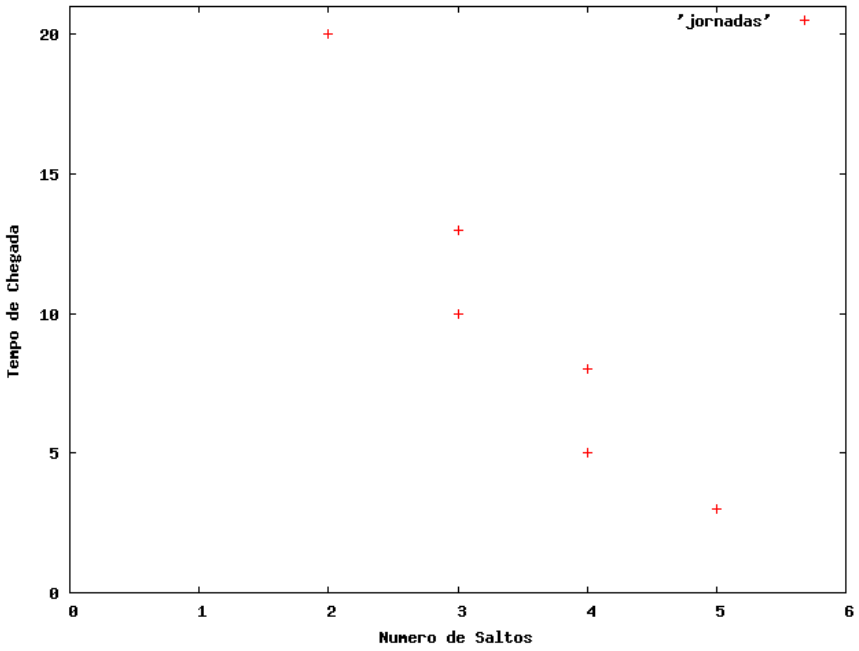

Figura 8: Pontos correspondentes a todas as jornadas

Temos então que verificar quais são aqueles que correspondem à Eficiência de Pareto (Definição 2).

- Note que apenas os pontos marcados pela curva não podem ser melhorados. Para os demais, existe algum outro ponto pertencente a curva que melhora um dos parâmetros sem prejudicar o outro: estes não farão parte da Curva.

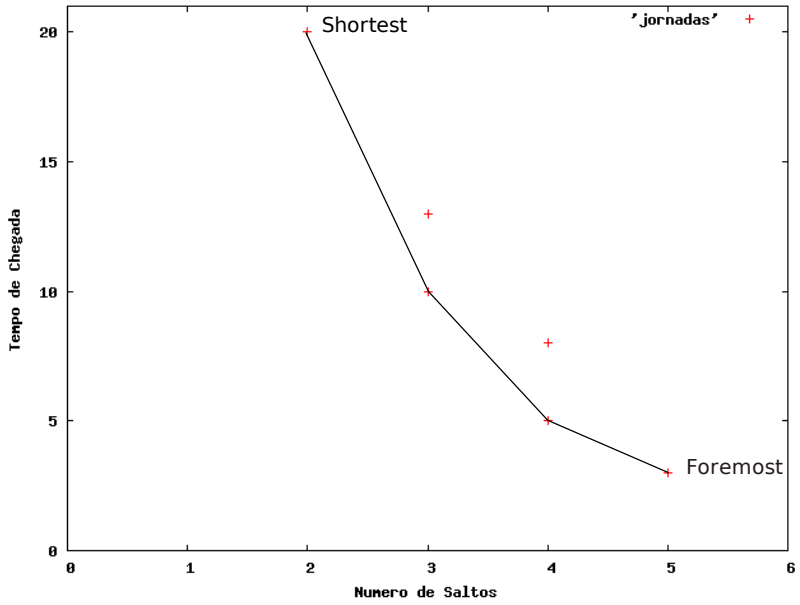

Figura 9: Pontos correspondentes a todas as jornadas e curva de pareto

- Por fim, obtemos a seguinte Curva de Pareto de acordo com a Definição 3.

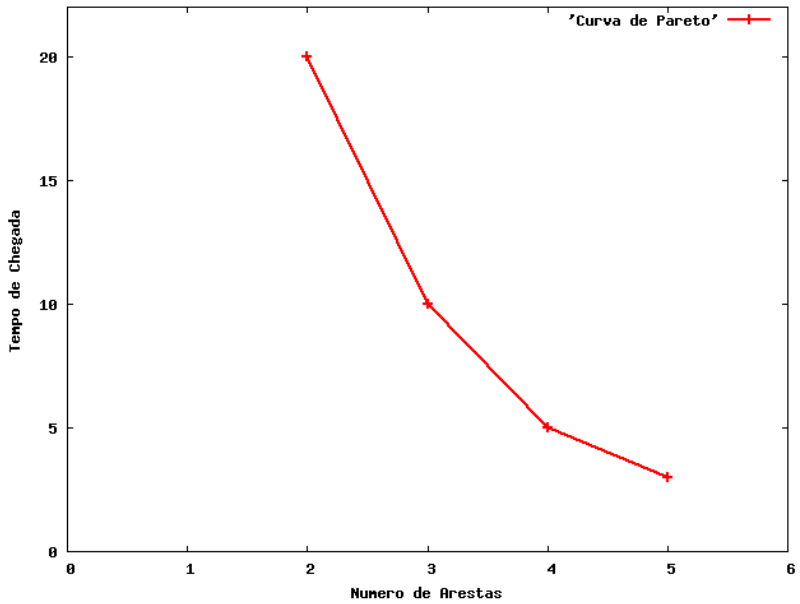

Figura 10: Curva de Pareto

Após obtermos a Curva de Pareto, temos então todas as jornadas Ótimas do nó s ao nó t. Concluímos que estas são:

- *Shortest*: $s \xrightarrow{15} d \xrightarrow{20} t$
- *Intermediária*: $s \xrightarrow{4} \bigcirc \xrightarrow{8} \bigcirc \xrightarrow{10} t$
- *Intermediária*: $s \xrightarrow{1} \bigcirc \xrightarrow{1} \bigcirc \xrightarrow{4} \bigcirc \xrightarrow{5} t$
- *Foremost*: $s \xrightarrow{1} g \xrightarrow{1} h \xrightarrow{1} j \xrightarrow{3} k \xrightarrow{3} t$

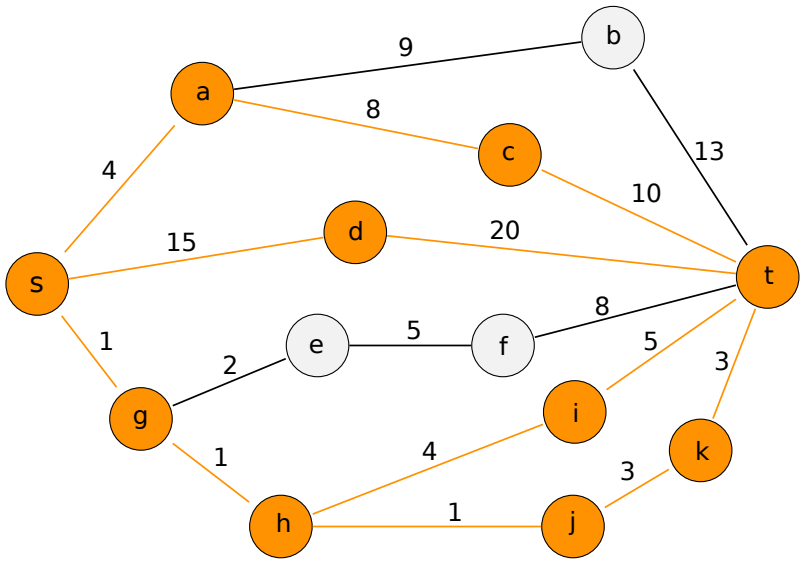

Figura 11: Grafo Evolutivo $\mathcal{G}$ e Jornadas Ótimas de s a t em destaque

Como obter estas Jornadas Ótimas na prática? A solução encontrada foi a proposta de um algoritmo que as obtenha a partir do compromisso entre a Shortest e a Foremost. Este algoritmo foi baseado em um algoritmo já proposto para encontrar a jornada Shortest[2].

7 Algoritmo para jornadas ótimas

Este algoritmo funciona basicamente da seguinte maneira: para cada possível valor de k, guardamos todas as jornadas *Foremost*, a partir de s, dentre as que possuem tamanho k, ou seja, guardamos as jornadas de menor tempo para todos os nós encontrados com k saltos.

7.1 Exemplo de simulação do algoritmo

Temos então o seguinte grafo:

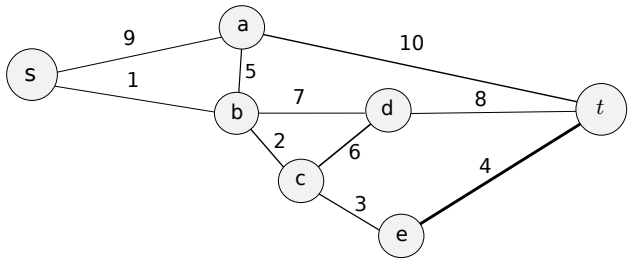

Figura 12: Grafo Evolutivo $\mathcal{G}$

Uma jornada $\mathcal{J}$ de s a s' de tamanho $\mathcal{K}$ e instante de chegada $\mathcal{T}$ é ótima se, e somente se, não existe outra jornada $\mathcal{J}'$ de tamanho $\mathcal{K}' \leq \mathcal{K}$ e instante de chegada $\mathcal{T}' < \mathcal{T}$ OU de tamanho $\mathcal{K}' < \mathcal{K}$ e instante de chegada $\mathcal{T}' \leq \mathcal{T}$.

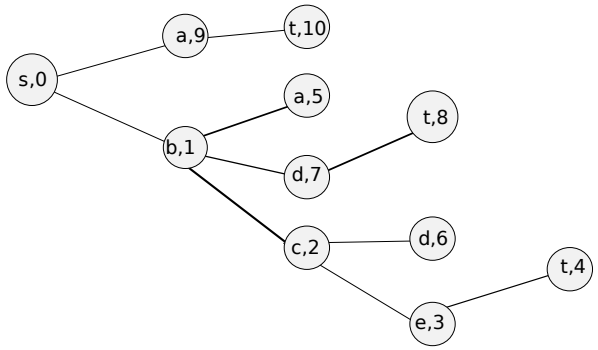

Figura 13: Árvore de predecessores para $\mathcal{G}$

Para o Grafo Evolutivo $\mathcal{G}$, o algoritmo encontra primeiro todas as jornadas ótimas de tamanho 1, depois todas as jornadas ótimas de tamanho 2, até todas as jornadas ótimas de tamanho n-1 que seria o caso onde todos os nós de $\mathcal{G}$ fazem parte da jornada. Com isso, obtemos uma árvore (ver Figura 13) onde cada nó s' aparece apenas uma vez por nível, e a jornada $\mathcal{J}$ de s a s', com os instantes de tempo encontrados na árvore, é ótima.

7.2 Algoritmo proposto

SELECAO-DE-ARESTAS-E-HORÁRIOS($\mathcal{G}, t_{LBD}$)

```
1  para v ∈ V_G
2      faça e_min[v] ← nil
3           t_min[v] ← ∞
4           t_chegada[v] ← t_LBD[v]
5  para (u, v) ∈ E_G
6      faça t ← f((u, v), t_LBD[u])
7           se t + ζ(u, v) < t_chegada[v]
8              então e_min[v] ← (u, v)
9                    t_min[v] ← t
10                   t_chegada[v] ← t + ζ(u, v)
11 devolva (e_min, t_min)
```

JORNADAS-ÓTIMAS($\mathcal{G}, s$)

```
1  para v ∈ V_G
2      faça t_LBD[v] ← ∞
3           J[v] ← ∅
4           J_otimas[v] ← ∅
5  t_LBD[s] ← 0
6  k ← 0
7  enquanto k < N
8      faça k ← k + 1
9           (e_min, t_min) ← SELECAO-DE-ARESTAS-E-HORÁRIOS(G, t_LBD)
10          para v ∈ V_G tal que e_min[v] ≠ nil
11             faça Seja (u, v) = e_min[v]
12                  Seja (R, R_σ) = J[u]
13                  J[v] ← (R ∪ e_min[v], R_σ ∪ t_min[v])
14                  J_otimas[v] ← (J_otimas[v] ∪ J[v])
15                  t_LBD[v] ← t_min[v] + ζ(e_min[v])
16 devolva J_otimas
```

Temos as seguintes estruturas de dados indexadas por $v \in V_G$:

- e_{min} e t_{min} guardam respectivamente a última aresta aresta usada para chegar ao nó v e último instante que chegamos ao nó v.
- t_{LBD} guarda o tempo de chegada da última jornada ótima que alcança o nó v.
- $t_{chegada}$ guarda o tempo de chegada da última jornada de tamanho k que alcança o nó v.
- $\mathcal{J}$ guarda a última jornada ótima encontrada que alcança o nó v.
- $\mathcal{J}_{otimas}$ guarda todas as jornadas ótimas encontradas que alcançam o nó v.

E a seguinte função:

- $f((u,v), t_{LBD}[u])$ que recebe uma aresta e um instante de tempo, e devolve o próximo instante de tempo a partir de $t_{LBD}[u]$ que esta aresta (u,v) pode ser atravessada.

A rotina SELECAO-DE-ARESTAS-E-HORÁRIOS verifica, a partir dos nós u encontrados com k saltos, quais nós v podem ser encontrados com $k+1$ saltos. Ao encontrarmos um nó v com $k+1$ saltos pela aresta (u,v), verificamos se o tempo de chegada a v por essa aresta é menor que o tempo de chegada da última jornada ótima para v. Se for menor, a jornada $\mathcal{J}$ de s a v de tamanho k+1 é considerada ótima. Esta aresta (u,v) é guardada em e_{min} e o instante de chegada em t_{min} e serão usados pela rotina JORNADAS-ÓTIMAS.

A complexidade desta rotina é $O(Mlog\delta_E)$ pois o laço da rotina tem complexidade M e a função $f((u,v), t_{LBD}[u])$ tem complexidade $log\delta_E$.

A rotina JORNADAS-ÓTIMAS calcula, para cada k, todas as Jornadas Ótimas de tamanho k. Sabemos que a única rota de tamanho 0 (zero) é de s para s, e a cada iteração usamos a rotina SELECAO-DE-ARESTAS-E-HORÁRIOS para calcular as próximas jornadas ótimas de tamanho $k < N$. Sempre que encontramos uma jornada ótima para um nó v, a guardamos em $\mathcal{J}_{otimas}[v]$.

Por fim, $\forall v \in V_G$ temos todas as jornadas ótimas de s a v em $\mathcal{J}_{otimas}[v]$.

Como temos nesta rotina JORNADAS-ÓTIMAS um laço com consumo de tempo $O(N)$ (linha 10) e a chamada da rotina SELECAO-DE-ARESTAS-E-HORÁRIOS (linha 9) sendo executados dentro de um laço que os executa N vezes, o consumo de tempo do algoritmo é $O(N(Mlog\delta_E + N))$.

8 Jornada ótima para determinado contexto

8.1 O problema

Após a obtenção de todas as jornadas ótimas possíveis de um nó s a um outro nó t, reduzimos o nosso problema para a tarefa de escolher uma única jornada ótima para o contexto.

Observaremos, então, as ótimas já obtidas. Caso seja possível determinar custos para as jornadas, podemos escolher a de menor custo como ótima.

Como otimizamos dois parâmetros, o número de arestas e o tempo de chegada da jornada, determinaremos custos para estes parâmetros e calcularemos o custo da jornada a partir da seguinte fórmula:

$$custo_{jornada} = custo_{\mathcal{K}} \times \mathcal{K} + custo_{\mathcal{T}} \times \mathcal{T},$$

onde $\mathcal{T}$ é o tempo de chegada e $\mathcal{K}$ é o tamanho da jornada.

Após obtermos os custos de todas as jornadas ótimas, determinamos como ótima para o contexto a jornada de menor custo.

8.2 Jornada ótima de menor custo

- Custo da aresta = 3, custo de instante de tempo = 1

Jornada	Arestas	Tempo	Custo
Shortest	2	20	26
Intermediária	3	10	19
Intermediária	4	5	17
Foremost	5	3	18

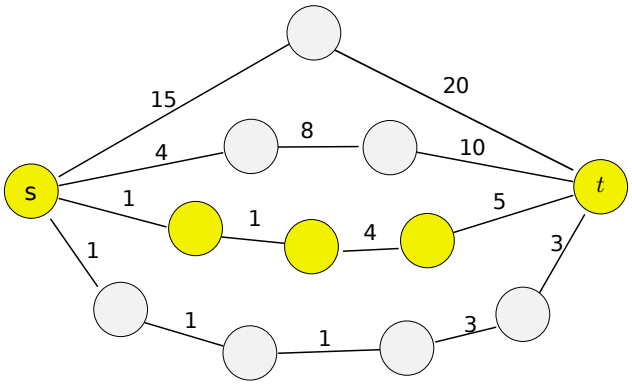

Figura 14: Jornada ótima em destaque

- Custo da aresta = 5, custo de instante de tempo = 1

Jornada	Arestas	Tempo	Custo
Shortest	2	20	30
Intermediária	3	10	25
Intermediária	4	5	25
Foremost	5	3	28

Neste caso, as duas jornadas em destaque, as intermediárias, são ótimas para o contexto.

- Custo da aresta = 6, custo de instante de tempo = 1

Jornada	Arestas	Tempo	Custo
Shortest	2	20	32
Intermediária	3	10	28
Intermediária	4	5	29
Foremost	5	3	33

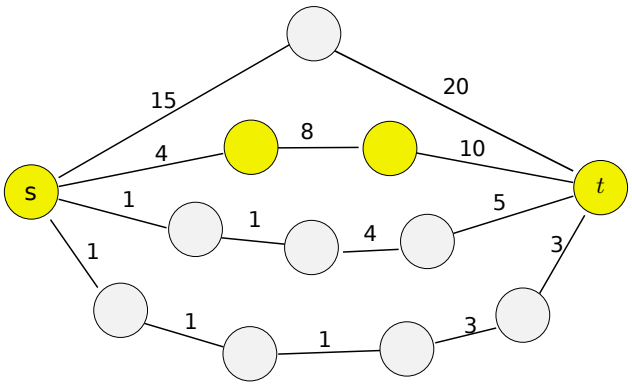

Figura 15: Jornada ótima em destaque

8.3 Algoritmo para jornada ótima de menor custo

O algoritmo para encontrar a jornada ótima de menor custo tem como base o algoritmo proposto para encontrar todas as jornadas ótimas. A diferença é que ao encontrar uma jornada ótima para v, esta só será armazenada caso seja a jornada ótima de menor custo encontrada até o momento. No final do algoritmo, temos guardadas apenas aquelas com origem em s que possuem menor custo. Em $\mathcal{J}_{otimas}[v]$ guardamos as jornadas ótimas de menor custo de s a v.

A complexidade deste algoritmo também é $O(N(Mlog\delta_E + N))$.

9 Conclusão

Com as redes móveis, novos cenários começaram a surgir. Um deles são as denominadas Redes Tolerantes a Atrasos e Desconexões ou Delay-and-Disruption Tolerant Networks (DTN). Aprofundando em rotas para entrega de mensagens em uma DTN, sabemos que, caso o comportamento desta rede seja conhecido, podemos modelá-la através de um Grafo Evolutivo e traçar as rotas a partir desse grafo.

Chamaremos um caminho de um nó s a t do Grafo Evolutivo de Jornada, e podemos otimizar apenas três parâmetros: tempo de chegada, tamanho e tempo de trânsito da jornada. Porém, dependendo do contexto, essas rotas ótimas podem possuir valores impraticáveis e seria de maior interesse o uso de jornadas intermediárias a estas. Levando em conta o tempo de chegada e o tamanho da jornada, temos o desafio de encontrar todas as jornadas ótimas a partir destes dois parâmetros.

A teoria escolhida *Eficiência de Pareto* nos permite otimizar dois parâmetros distintos ao mesmo tempo e construir uma *Curva de Pareto*. A partir destes dois conceitos, é possível encontrar todas as jornadas ótimas de um nó s a um nó t para depois escolhermos uma única ótima dependendo do contexto aplicado.

O algoritmo proposto para encontrar as jornadas ótimas foi baseado no algoritmo proposto por [2] para encontrar a jornada Shortest. Este algoritmo Jornadas-Ótimas foi implementado, testado e, definido um nó s, ele encontra todas as jornadas ótimas com origem em s.

Após encontrar todas as jornadas ótimas de um nó s a t, podemos colocar custos para os parâmetros e determinar uma jornada ótima para o contexto, no caso as de menor custo. Podemos mudar poucas coisas do algoritmo proposto e encontrar a jornada de menor custo. Este algoritmo também foi implementado.

Referências

[1] C.T. Oliveira, M.D.D. Moreira, M.G. Rubinstein, L. Costa, and O. Duarte. Redes tolerantes a atrasos e desconexões. In *Minicursos do Simposio Brasileiro de Redes de Computadores (SBRC 2007)*, 2007.

[2] B. Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. 2002.

[3] Ross D. Game Theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. 2009.

[4] Fudenberg, D. and Tirole, J. Game Theory. MIT Press. 1983.

[5] Monteiro, J. and Goldman, A. and Ferreira, A.. Using Evolving Graphs Foremost Journey to Evaluate Ad-Hoc Routing Protocols, *Procedings of 25th Brazilian Symposium on Computer Networks (SBRC'07), Blem, Brazil*. 2007.

[6] Ferreira, A. On models and algorithms for dynamic communication networks: the case for evolving graphs, In Proc. ALGOTEL, 2002.

Parte II

Parte Subjetiva

1 Desafios e frustrações

Em setembro de 2008 estava querendo começar uma iniciação cientifica. Fui falar com o professor Alfredo Goldman que perguntou se eu tinha alguma área de interesse e eu logo falei "redes móveis". Ele me prometeu pensar em algum projeto e em menos de um mês veio com a proposta de iniciar um projeto junto com mais cinco alunos.

Não demorou muito para encontrar outros alunos e fecharmos o grupo para o projeto. Começamos a nos reunir frequentemente e debater sobre Redes Tolerantes a Atrasos e Desconexões. Éramos eu, Adriano Tabarelli, Caio Cestari, Carlos Eduardo Moreira dos Santos, César Gamboa Machado e Paulo Floriano. Após algumas semanas o Carlos recebeu uma proposta do professor Fábio Kon, e mudou de projeto. Os demais prosseguiram.

Após o estudo sobre as DTNs e algoritmos de roteamento nessas redes, começamos a estudar grafos evolutivos e suas jornadas ótimas. Desde o começo tinhamos a intenção de fazer o TCC baseado no projeto. Fomos percebendo que essa área tinha diversos ramos e resolvemos dividir o grupo em três menores. Eu fiquei sozinha e formaram-se duas duplas: Caio e Adriano, César e Paulo. Continuamos estudando em conjunto e sempre um grupo ajudando o outro.

Eu fiquei com essa parte do projeto, e a princípio foi difícil entender o que seria esse tal "compromisso". Li muito sobre Teoria dos Jogos para entender o conceito Eficiência de Pareto e Curva de Pareto. Como nunca tinha contato com essa matéria, o entendimento demorou um tempo. Eu ficava olhando horas e horas para alguns grafos e pensando quais seriam os pontos da curva de Pareto. Mesmo tendo explicado essa parte de forma simples na monografia, acredito que seja a parte que mais me deu trabalho.

Após entender o conceito de Curva de Pareto e entender quais pontos fariam parte dela, tive o desafio de encontrar o algoritmo exato para encontrar os pontos que pertencem a essa curva. Como o César e o Paulo haviam implementado os algoritmos Shortest e Foremost, eles me deram a dica que a partir do algoritmo Shortest eu conseguiria encontrar estes pontos. Foi aí que foquei os meus estudos no algoritmo Shortest. Li várias vezes papers que tratavam o assunto e parti para o código. Mexi e remexi no código várias vezes, e quando achei que tinha resolvido o problema cheguei no seguinte caso: como mudei a condição de parada do algoritmo, ele estava encontrando jornadas que podiam passar duas vezes por uma mesma aresta. Demorei muito tempo para resolver esse problema até que encontramos um erro na implementação do Shortest. Resolvido o problema, consegui finalmente encontrar todas as jornadas ótimas.

A minha base de testes foi o ONE (Opportunistic Network Environment simulator). Gerava um grafo evolutivo no ONE e testava o meu algoritmo. A idéia até aí era escolher uma jornada ótima obtida por esse algoritmo, coloca-la como rota no ONE, e ver uma DTN usando as rotas escolhidas. Foi quando caímos no seguinte problema: como escolher uma única jornada ótima? Aí que veio a idéia de utilizar custos. Terceiro grande desafio: algoritmo com custos. Algumas dúvidas começaram a surgir: o que vai ter custos? custos ou pesos? Defini que o número de arestas da jornada teria um custo (cada aresta um custo, todos os mesmos custos) e os instantes de tempo teriam outro

(cada instante de tempo um custo, todos o mesmo custo). Assim, seria fácil determinar a jornada de menor custo.

Não tão fácil assim. Mudar o algoritmo para encontrar as jornadas de menor custo, embora pareça simples, me tomou muito tempo até encontrar a solução. A idéia inicial era fazer um algoritmo totalmente diferente, que comparava todas as possíveis jornadas e não apenas as ótimas. Foi quando eu percebi que a jornada de menor custo não seria apenas uma jornada qualquer, mas uma das jornadas ótimas (bingo!). Voltei então ao algoritmo para encontrar as jornadas ótimas e deu certo (este algoritmo também foi implementado).

Após encontrar uma única jornada ótima, a idéia era voltar ao ONE, testar as rotas e vê-las funcionando. Infelizmente não deu tempo, assim como não deu para fazer o compromisso entre as três jornadas ótimas conhecidas (shortest, fastest e foremost).

2 Trabalhos Futuros

Pretendo continuar o projeto, testar os resultados no simulador de DTN (ONE) e fazer o compromisso entre as três jornadas. Pretendo escrever um artigo sobre esses resultados e mandá-lo para o Simpósio Brasileiro de Redes de Computadores.

3 Disciplinas mais relevantes

- **MAC0110 - Introdução à Computação, MAC0122 - Princípios de Desenvolvimento de Algoritmos**: foram a base para o meu aprendizado. Entrei na faculdade sem saber nada de algoritmos e programação.

- **MAC0211 - Laboratório de Programação I, MAC0242 - Laboratório de Programação II**: nessas matérias que comecei a entenderam o que é um projeto. Aprendi LaTeX nestas disciplinas, a programar em grupo e a dividir tarefas.

- **MAC0328 - Algoritmos em Grafos, MAC0323 - Estruturas de Dados** : o aprendizado nestas disciplinas foram essenciais para o desenvolvimento do meu trabalho. Entender o que é uma estrutura de dados, grafos, e como funcionam algoritmos usando essas estruturas foi importantíssimo.

- **MAC0438 - Programação Concorrente**: nessa disciplina eu tive contato diretamente com projetos em Java. Passei a usar eclipse e isso ajudou muito na hora de desenvolver o projeto no TCC.

4 Agradecimentos

A todos que acreditaram na minha competência, no meu trabalho e me apoiaram em todos os momentos. Em especial ao: Alfredo, Adriano, Caio, César, Paulo, Lucas, BCC2006 e meus pais.