

Um Estudo Sobre a Resolução do Problema de Torcedores e Otimização de Passagens Aéreas

Guilherme A. Bianeck¹, Ricardo Stoklosa¹

¹Bacharelado de Ciência da Computação (UDESC - CCT)

Resumo. Atualmente a necessidade de otimização de caminhos e fluxos é de extrema importância para a redução de custos e melhora de resultados. O presente artigo tem como objetivo resolver um problema de otimização de caminhos, fluxos e preços de passagens aéreas para um time de futebol, através de algoritmos para o cálculo de menor caminho é possível encontrar a viagem que possui o menor custo para o grupo considerando o horário máximo para chegar no local, a quantidade máxima de trocas de aviões e relevando também a capacidade máxima por avião. Será demonstrado também os grafos dado o problema em questão, assim como as suas formas reduzidas levando em consideração os algoritmos implementados, desta forma, é possível visualizar através dos grafos os processos realizados para encontrar as alternativas otimizadas para a compra de passagens.

1. Introdução

Muitas empresas demandam de processos tecnológicos para o seu funcionamento, estes meios tecnológicos requerem uma grande infraestrutura e manutenção para o seu pleno funcionamento o que acaba tornando inviável para muitas empresas, visto que, os custos podem não compensar o investimento [Armbrust et al. 2010]. Então, é necessário realizar o aperfeiçoamento de seus sistemas, encontrando os caminhos com menores custos e maiores fluxos. Com isto, é possível relacionar atividades e problemas diários com meios profissionais, aos quais se aplicam, como no presente trabalho ao qual deve-se realizar a otimização procurando o menor custo de passagens aéreas relevando também suas paradas, distâncias e quantidades de passagens disponíveis.

Neste trabalho é abordado na seção 2 os conceitos utilizados para a otimização das passagens aéreas ao qual tem por objetivo informar a base deste tema, a seção ?? é fundamentada nas bases do problema proposto ao qual explana a forma de resolução do problema proposto e seu algoritmo implementado. A última seção 5 aborda as considerações finais deste artigo, demonstrando de forma conclusiva os temas abordados no artigo e do conhecimento adquirido após a leitura do mesmo.

2. Conceitos

Os conceitos de grafos e suas otimizações tem sido fortemente utilizadas para o mercado atual, algoritmos tais como o Dijkstra é fortemente usado para a otimização de percursos na indústria [Aviram and Shavitt 2015].

3. Implementação

Para a implementação do código, foi necessário a criação de uma lista de adjacência adaptada ao qual seria interpretada pelo Dijkstra posteriormente como demonstrada na

Lista Adjacência 1, onde características como os horários, custos e capacidades deveriam ser relevadas. Como auxílio do algoritmo de Dijkstra achamos o caminho de menor custo. Armazenamos o gargalo deste caminho mínimo, no caso o valor com o menor números de reservas disponíveis, então percorremos este caminho subtraindo este valor dos números de vagas e adicionamos ao contador de pessoas que conseguiram viajar. Na implementação do Dijkstra uma aresta só pode ser relaxada se tiver 0 ou mais reservas disponíveis, portanto esta aresta ficara indisponível.

3.1. Dijkstra

O algoritmo de Dijkstra é muito utilizado para a resolução de problemas de caminhos mínimos entre pontos, representados através de grafos dirigidos ou não e de arestas com peso positivo [Javaid 2013]. Para a resolução do problema, Dijkstra possui alguns passos, tais como:

- Iniciar os caminhos com a pior hipótese, portanto, o maior valor possível, foi escolhido arbitrariamente a constante ($INF = 1000000000$) no algoritmo proposto.
- Determinar o menor custo em relação a um vértice Q.
- Realizar relaxamentos nas arestas, onde, um *Heap* para custo mínimo deve ser executado verificando se o horário de chegada da aresta anterior antecede o horário de saída da aresta em questão, sendo que devem ter 1 hora de diferença, e se a aresta possui reservas disponíveis.

4. Problema Proposto

O problema proposto para última atividade a matéria de Teoria dos Grafos "Torcedores" se diz respeito a um grande grupo de 150 torcedores do Coringas, este grupo se encontra em São Paulo e deseja assistir a partida na cidade de Natal, localizado no Rio Grande do Norte, o grupo deseja chegar até as 19:00h na cidade desejada. Para o problema proposto, todos os 150 torcedores vão dividir as despesas do voo como um todo, portanto, é necessário buscar o esquema de passagens mais baratos, chegando antes do horário estimado e realizando até cinco trocas de aviões, necessita-se considerar a pausa entre conexões de ao menos uma hora também. É importante notar que, destinos de saída e entrada para Brasília recebem descontos de 50%, respeitando o problema proposto.

Os números dos voos, origens, destinos, saídas, chegadas, custos e reservas foram dados como a imagem 2, entretanto, para a resolução do problema proposto foi necessário desenvolver a matriz 1 e para fins de visualização e compreensão o grafo da imagem 1.

Após executar o algoritmo com a entrada proposta, é possível reescrever o grafo novamente, demonstrando seus caminhos de menor custo, a saída resultante do programa respeita a imagem 3 e o grafo 4

4.1. Funções

Para o desenvolvimento do código, algumas funções necessitaram ser implementadas, tais quais:

Tabela 1. Lista Adjacência

Departure	Arrival	Dep Time	Arriv Time	Cost	Capacity
SP	GOI	0800	1000	120	120
GOI	SAL	1100	1340	120	70
SAL	REC	1550	1700	100	80
REC	NA	1800	1845	80	100
SP	BRA	0900	1050	50	180
BRA	FOR	1145	1500	180	20
FOR	REC	1550	1620	100	25
CUR	MAN	0650	1020	320	180
MAN	BEL	1050	1200	100	20
BEL	FOR	1250	1420	100	35
RJ	SP	0600	0750	80	160
SP	MAC	0830	1100	180	100
MAC	REC	1200	1300	80	120
REC	NA	1500	1545	80	10
POA	CUR	0600	0730	100	100
CUR	BRA	1000	1145	150	140
BRA	REC	1400	1630	150	120
BRA	MAC	1200	1430	100	180
RJ	BRA	0900	0945	100	110
RJ	VI	0500	0600	100	130
VI	SAL	0830	0945	80	20
SAL	MAC	1045	1145	80	100
MAC	NA	1645	1800	100	50
REC	NA	1730	1815	50	120

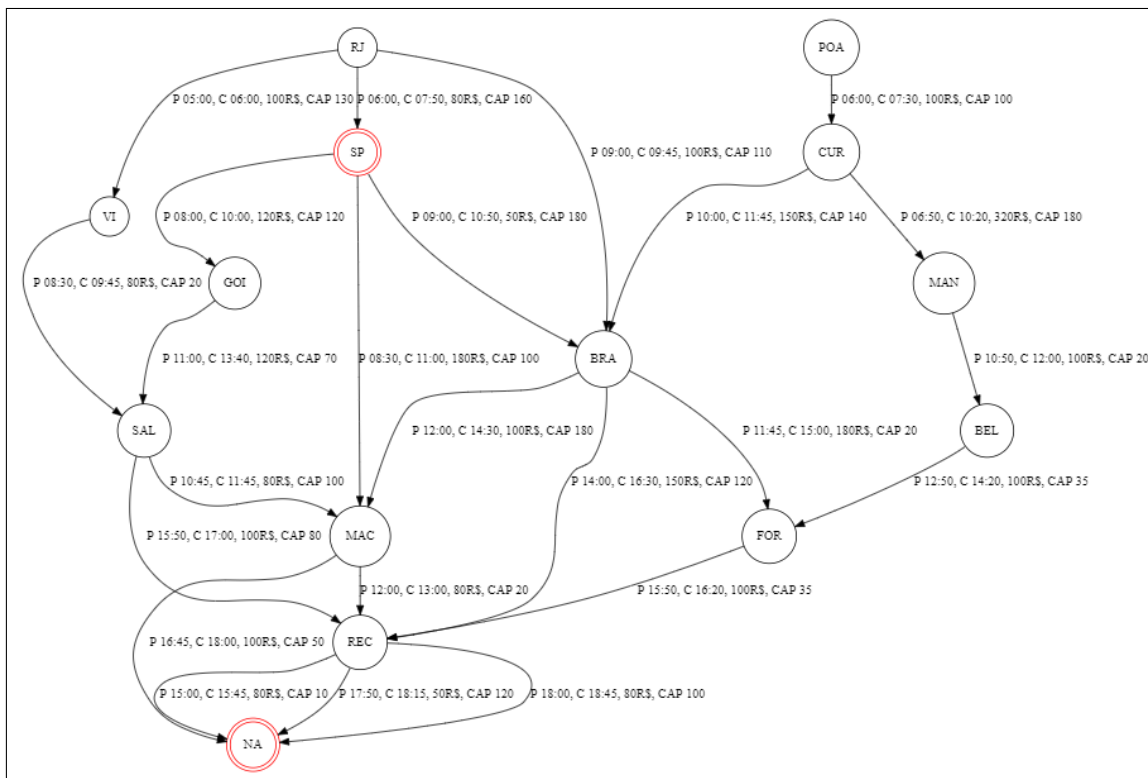


Figura 1. Grafo do Problema Proposto

# do Voo	Origem	Destino	Saída	Chegada	Custo Normal US	Reservas ✓
501	<u>São Paulo</u>	Goiânia	08:00	10:00	120,00	120
	Goiânia	Salvador	11:00	13:40	120,00	70
	Salvador	Recife	15:50	17:00	100,00	80
	Recife	Natal	18:00	18:45	80,00	100
513	<u>São Paulo</u>	Brasília	09:00	10:50	50,00	180
	Brasília	Fortaleza	11:45	15:00	180,00	20
	Fortaleza	<u>Recife</u>	15:50	16:20	100,00	25
581	<u>Curitiba</u>	Manaus	06:50	10:20	320,00	180
	Manaus	Belém	10:50	12:00	100,00	20
	Belém	<u>Fortaleza</u>	12:50	14:20	100,00	35
587	<u>Rio de Janeiro</u>	São Paulo	06:00	07:50	80,00	160
	São Paulo	Maceió	08:30	11:00	180,00	100
	Maceió	Recife	12:00	13:00	80,00	120
	Recife	<u>Natal</u>	15:00	15:45	80,00	10
590	<u>Porto Alegre</u>	Curitiba	06:00	07:30	100,00	100
	Curitiba	Brasília	10:00	11:45	150,00	140
	Brasília	<u>Recife</u>	14:00	16:30	150,00	120
592	<u>Brasília</u>	<u>Maceió</u>	12:00	14:30	100,00	180
593	<u>Rio de Janeiro</u>	<u>Brasília</u>	09:00	09:45	100,00	110
594	<u>Rio de Janeiro</u>	Vitória	05:00	06:00	100,00	130
	Vitória	Salvador	08:30	09:45	80,00	20
	Salvador	Maceió	10:45	11:45	80,00	100
	Maceió	<u>Natal</u>	16:45	18:00	100,00	50
595	<u>Recife</u>	<u>Natal</u>	17:30	18:15	50,00	120

Figura 2. Tabela do Problema Proposto


```
TEG/trabalhoFinal/src on  master
→ make
g++ graph.cpp main.cpp -o main
./main < entrada
>>20 pessoas
[ BRA 540 650 25 ]
[ REC 840 990 75 ]
[ NA 1050 1095 50 ]
Custo total 150
>>100 pessoas
[ MAC 510 660 180 ]
[ NA 1005 1080 100 ]
Custo total 280
>>30 pessoas
[ GOI 480 600 120 ]
[ SAL 660 820 120 ]
[ REC 950 1020 100 ]
[ NA 1080 1125 80 ]
Custo total 420
Pessoas150
```

Figura 3. Código de Saída

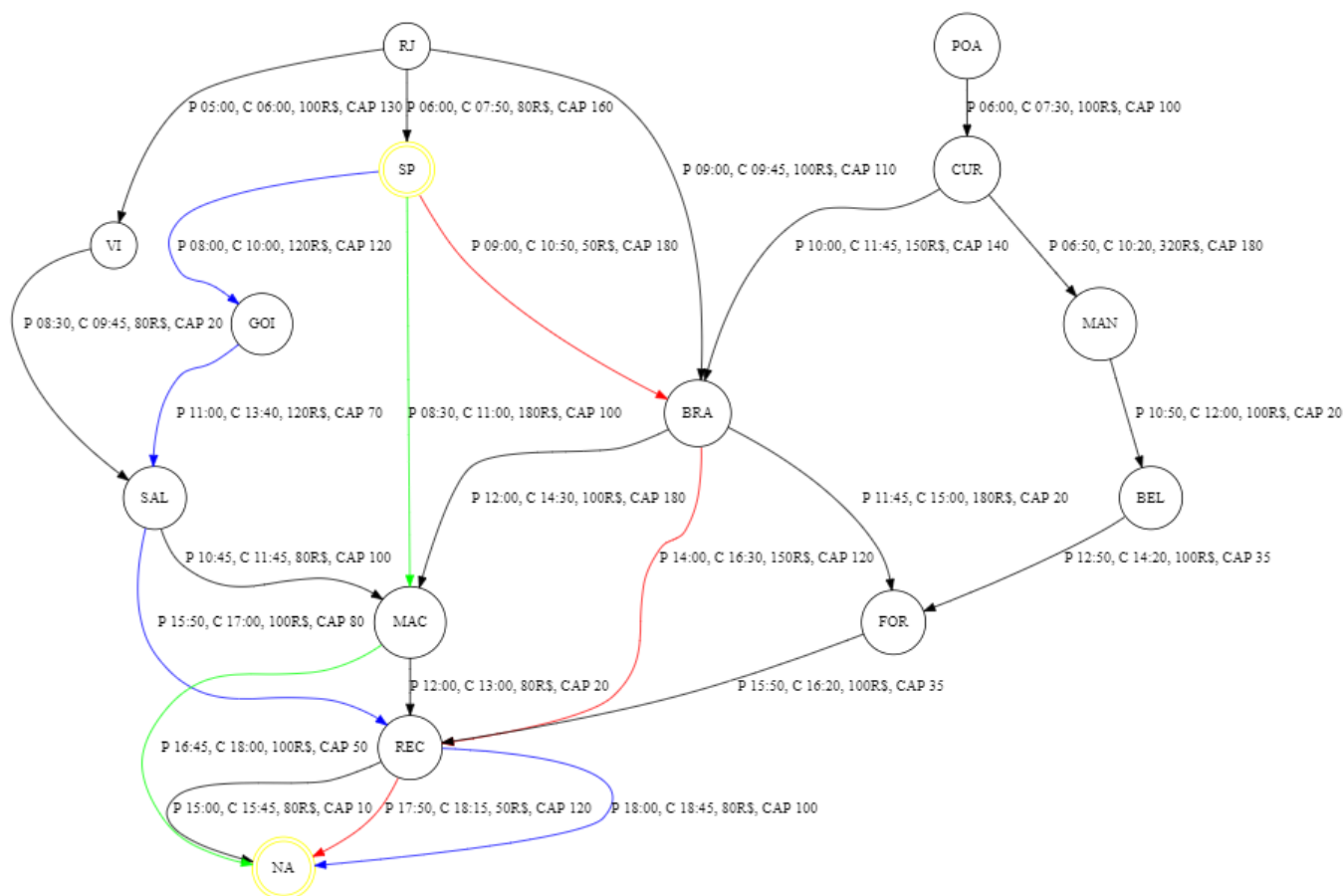


Figura 4. Grafo Resultante

- *getCity*: A função *getCity*, para todas as cidades disponíveis ("SP", "GOI", "FOR", "CUR", "MAN", "BEL", "REC", "POA", "BRA", "RJ", "VI", "SAL", "MAC", "NA"), primeiramente compara se as entradas correspondem as cidades cadastradas, caso não pertença, o programa retorna um aviso, caso pertença, a cidade retorna seu valor respectivo (1...14)
- *addEdge*: Esta função é responsável por assimilar as entradas aos seus respectivos campos da struct.
- *showAdjList*: A função *showAdjList* demonstra a lista de adjacência registrada.
- *readInput*: *readInput* é a função responsável por registrar os valores lidos, juntamente em conjunto com a *addEdge* e *getCity*, assim como aplicar o desconto dos voos relacionados ao aeroporto de Brasília.
- *dijkstra*: A principal função do programa, como explicado anteriormente, é responsável por realizar o cálculo de otimização para o sistema proposto
- *hourToMinutes*: Esta função é responsável por realizar a conversão de horas para minutos, com o intuito de facilitar os cálculos relacionados ao tempo.

5. Considerações Finais

Como consideração final compreendemos que a otimização de fluxos e caminhos são necessários para reduzir gastos e aumentar a eficiência de sistemas, tal como reduzir o custo de passagens aéreas entre um grupo de torcedores de futebol como no problema proposto.

Os algoritmos de otimização podem ser aplicados em diversas finalidade, apesar de ser fortemente utilizado na indústria, mais em específico de empresas relacionadas ao segmento de logística e redes, compreendemos que em atividades diárias se vê necessário a otimização e uso destas aplicações.

No problema proposto, foi possível concluir que os voos seguindo pelos respectivos trajetos São Paulo, Brasília, Recife, Natal e São Paulo, Maceió, Natal, e São Paulo, Goiânia, Salvador, Recife, Natal formam o menor custo para o trajeto calculado, respeitando regras tais como o horário de partida, quantidade de passagens disponíveis, desconto de 50% para trajetos que saem e entram em Brasília e a quantidade máxima de 5 conexões. Sendo assim, as pessoas participantes deverão dividir o valor total de 43.600R\$ entre si, resultando em um gasto de apenas 290R\$ por pessoa para viajar de avião de São Paulo até Natal, tal economia poderia acarretar em mais investimentos para o clube.

Referências

- Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.
- Aviram, N. and Shavitt, Y. (2015). Optimizing dijkstra for real-world performance.
- Javaid, A. (2013). Understanding dijkstra algorithm. *SSRN Electronic Journal*.