Universidade Federal do Rio Grande do Sul
Otimização Combinatória

Meta-heurística Simulated Annealing aplicada ao problema de Ordenação Linear

Otávio Carvalho

Junho de 2014

1 – Introdução

O problema de Ordenação Linear (Linear Ordering Problem), é um dos problemas clássicos de otimização combinatória que foi classificado como NP-hard em 1979 por Garey e Johnson. Ele recebeu considerável atenção em várias áreas de aplicação, que vão desde arqueologia e escalonamento, passando por economia e chegando até ao campo da psicologia matemática. Métodos de solução para o problema de Ordenação Linear tem sido propostos desde 1958, quando Chenery e Watanabe definiram algumas idéias sobre como obter soluções para este problema. **[4]**

Neste relatório, descreveremos o processo de adaptação da meta-heurística Simulated Annealing aplicada ao problema de Ordenação Linear, visando atingir resultados o mais próximo possíveis dos lower bounds conhecidos, para um conjunto de problemas propostos.

Na seção 2, iremos definir o problema, explicando-o e definindo a sua formulação matemática. Na seção 3 iremos propor uma forma de resolver o problema utilizando a meta-heurística que foi proposta. Os resultados, bem como uma análise sobre os mesmos, serão apresentados na seção 4, seguidos pela conclusão na seção 5.

2 – O Problema

Dado como entrada um grafo **G = (V,E)** e uma função de mapeamento **φ: V → [1..|V|]**, que mapeia cada vértice do grafo de entrada para um novo valor determinado por essa função.

Queremos saber: ***Qual o mapeamento que minimiza o custo das arestas do grafo?*** **[5]**

Considerando que, dado um grafo, a instância **φ** corresponde ao mapeamento da ordenação linear que minimiza o custo das arestas do grafo. Podemos formular a função objetivo no formato abaixo:

$$\mathrm{LA}(G,\varphi) = \sum_{uv \in E} |\varphi(u) - \varphi(v)|.$$

O problema pode ser melhor ilustrado através das imagens abaixo:

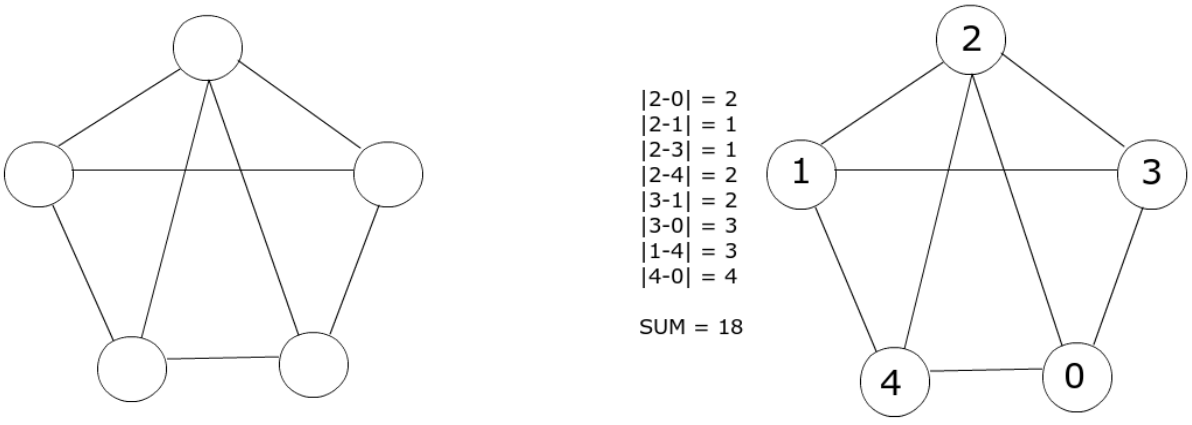

Figura 1 – Grafo e Instância de Ordenação Linear

Uma vez que tenhamos um grafo como o da esquerda da *Fig. 1*, podemos mapear as suas arestas para uma instância como a da direita da *Fig. 1*, resultando em uma solução de custo 18. Por outro lado, podemos logo pensar em uma outra solução com um custo melhor, como a seguinte:

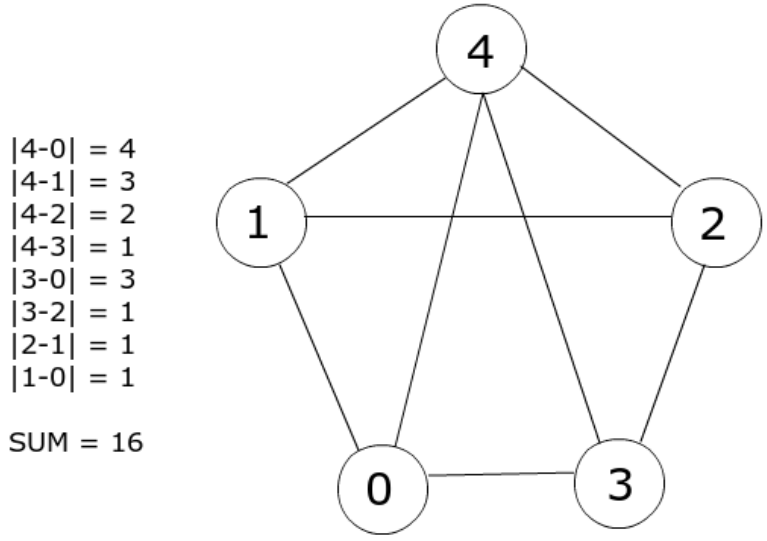

Figura 2 – Instância de Ordenação Linear Melhorada

Portanto, o problema da Ordenação Linear consiste em encontrar a instância que minimize o custo de um grafo, gerando para o mesmo um conjunto de valores de vértices que realize essa operação. Neste trabalho, visando atingir mapeamentos desse tipo, iremos utilizar a meta-heurística de Simulated Annealing.

Para a construção do algoritmo da meta-heurística de Simulated Annealing, que aproximam resultados ótimos da Ordenação Linear, optamos por representar o grafo por uma lista de adjacências, uma vez que as matrizes desse problema podem ser exparsas, o que acarretaria em um custo desnecessário de memória. Utilizamos também mapas baseados em tabelas hash, para o mapeamento dos vértices, o que nos proporcionou as facilidades de inserção e consulta no mapa com complexidade O(n).

3 – Solução do Problema

3.1 – Simulated Annealing

A meta-heurística de Simulated Annealing pertence à classe das heurísticas de busca local randomizadas. Ela é baseada em uma analogia entre um processo físico conhecido como anelamento de sólidos e o processo algoritmico de resolver um problema de otimização.

O princípio básico da busca local é iterativamento melhor uma dada solução por executar mudanças locais na sua estrutura combinatória. Em algoritmos de *"Hill climbing"*, mudanças que melhoram a solução são aceitas, e aquelas que não melhoram a solução diretamente são rejeitadas. Por esse motivo, algoritmos desse gênero terminam em um ótimo local.

Figura 3 – Problema dos ótimos locais em uma heurística de Hill Climbing

Visando contornar o problema dos valores ótimos locais, algoritmos como o de Metropolis foram desenvolvidos, a fim de buscar soluções ótimas no restante do espaço de soluções, mesmo que para isso seja necessária a aceitação de movimentos que pioram a solução corrente. O algoritmo de Metropolis propõe uma parametrização por uma temperatura **t**. Um movimento que produz um ganho de **δ** no custo é aceito com uma certa probabilidade. **[3]**

O algoritmo de Simulated Annealing é extremamente semelhante ao de Metropolis, porém consiste em uma sequência de execuções de um algoritmo de Metrópolis, com um decremento progressivo da temperatura **t**. O pseudo-código abaixo descreve a idéia geral da solução, onde devemos considerar que **LA** representa a função que calcula o custo de uma solução **φ** proposta:

função SimulatedAnnealing(**G**):
    φ := Gera um mapeamento inicial
    t := $\mathbf{t_0}$:= Selecione uma temperatura inicial
    enquanto não congelar *faça*:
        φ := Selecione um vizinho de **φ**
        δ := LA(**φ**,G) – LA(**φ'**,G)
        com probabilidade min(1, $\mathbf{e^{-\delta/t}}$) *faça* **φ** := **φ'**
    fim enquanto
    t := **α*t**
    retorna **φ**
fim função

Podemos perceber que o algoritmo tem por finalidade selecionar soluções que melhorem o custo mínimo da função de otimização. Porém, ele também é capaz de dar chances probabilisticas de tentarmos atingir valores melhores, através de soluções locais piores (com relação à atual), buscando fugir de ótimos locais. É importante notar a semelhança com o fenômeno físico, uma vez que conforme

a temperatura vai diminuindo, as chances de mudança vão se tornando mais escassas, simulando o comportamento de um anelamento real.

3.2 – Solução Inicial

Para a definição da Solução Inicial, três heurísticas construtivas foram utilizadas: Ordenação Fixa, Ordenação Randômica e Ordenação por BFS-Search. **[2]**

Dos três algoritmos utilizados, o de Ordenação por BFS-Search foi o que demonstrou capacidade de gerar as instâncias mais próximas dos lower bounds conhecidos.

Abaixo, descrevemos brevemente o funcionamento de cada um deles:

3.2.1 – Ordenação Fixa

Os vértices são ordenados da maneira como eles são apresentados no grafo. Esta solução é fixa e não muda entre as execuções.

3.2.2 – Ordenação Randômica

Os vértices são escolhidos aleatoriamente, e nomeados na ordem na qual eles aparecem nessa seleção randômica. O algoritmo é custoso e a solução não considera a conectividade do grafo, além de variar entre as execuções.

3.2.3 – Ordenação por BFS-Search

Os vértices são ordenados escolhendo-se o primeiro aleatoriamente e, a partir deste, é executada uma busca em largura no grafo, até que todos os vértices tenham sido nomeados.

3.3 – Vizinhança

Após testes com alguns algoritmos de vizinhança, como o **Flip2**, **FlipE**, acabamos optando pela utilização da vizinhança **FlipE**, devido a sua utilização da estrutura conectiva do grafo, o que acreditamos que possa ser usado para melhorar localmente os resultados, para instâncias de problema com alta conectividade.

Consideramos que dois vértices são vizinhos se, dado um grafo **G(V,E)**, se existe uma aresta **E**, que liga dois vértices $\mathbf{V_1}$ e $\mathbf{V_2}$.

A vizinhança **FlipE** considera que a partir de dois vértices vizinhos, ligados por uma aresta, é possível obter uma nova vizinhança, apenas trocando os labels dos dois vértices. **[2]**

Esta solução foi escolhida principalmente devido à sua simplicidade, quando comparada com soluções complexas como a de Spectral Sequencing Method (SSQ), onde a vizinhança é obtida através do computo de eigenvectors da matriz Laplaciana de **G(V,E)**. **[1]**

4 – Resultados

4.1 - Legenda das tabelas

- **Instância**: Nome da instância de problema resolvida
- **NE**: Número de Execuções
- **SI**: Solução Inicial (a melhor encontrada)
- **MS**: Melhor Solução encontrada
- **MC**: Melhor solução Conhecida para o problema
- **T(s)**: Tempo de execução em segundos (tempo total das NE execuções)
- **Desvio**: Desvio percentual da Melhor solução Conhecida (MC)

4.2 – Solução Inicial com Ordenação Fixa

Instância	NE	SI	MS	MC	T(s)	Desvio
bintree10	20	262147	262147	4267	0.512	98.37
hc10	20	639336	639336	523776	0.848	18.07
mesh33x33	20	35474	35474	32703	0.569	7.81
3elt	20	770618	770618	431737	2.243	43.97
airfoil1	20	427742	427742	322611	1.993	24.57
whitaker3	20	6852980	6852980	1307540	4.285	80.92
c3y	20	623645	623645	124117	0.720	80.09
c4y	20	483903	483903	115144	0.748	76.20
c5y	20	459886	457426	96952	0.620	78.80
gd96a	20	489822	486609	96253	0.550	80.21

4.3 – Solução Inicial com Ordenação Randômica

Instância	NE	SI	MS	MC	T(s)	Desvio
bintree10	20	331853	331378	4267	55	98.71
hc10	20	1717850	1716970	523776	54	69.49
mesh33x33	20	757474	755821	32703	58	95.67
3elt	20	21437500	21431700	431737	249	97.98
airfoil1	20	17123500	17117100	322611	284	98.11
whitaker3	20	94045000	94042000	1307540	687	98.60
c3y	20	1209780	1206660	124117	72	89.71
c4y	20	1266350	1266310	115144	78	90.90
c5y	20	987132	986010	96952	60	90.16
gd96a	20	578513	575733	96253	55	83.28

4.4 – Solução Inicial com BFS-Search:

Instância	NE	SI	MS	MC	T(s)	Desvio
bintree10	100	136833	136824	4267	2.3	96.88
hc10	100	923695	923695	523776	3.7	43.29
mesh33x33	100	46968	46968	32703	2.95	30.37
3elt	100	951307	951307	431737	11	54.61
airfoil1	100	757023	757023	322611	10	57.38
whitaker3	100	1381480	1381480	1307540	22	5.35
c3y	100	487053	487053	124117	3.6	74.51
c4y	100	425349	425349	115144	3.8	72.92
c5y	100	394114	394114	96952	3.3	75.40
gd96a	100	308814	304390	96253	2.9	68.37

Os parâmetros utilizados para o Simulated Annealing foram:

0 <= T <= 1000000
r = 0,8

Onde **T** é o intervalo de temperatura utilizado pelo algoritmo, que inicia em 1000000 e é resfriado, por uma taxa de **r** = 0,8, até o valor atingir o congelamento em 0 e encerrar a sua execução.

Após implementado o algoritmo de Ordenação Fixa e percebida a sua dificuldade em propôr soluções próximas dos lower bounds conhecidos, foi tentada uma implementação randômica. A implementação randômica, por sua vez, mostrou custosa em tempo de execução e ofereceu resultados de maneira geral piores que os da implementação de Ordenação Fixa. Por fim, foi tentado um algoritmo de Solução Inicial baseado no BFS (Busca em Largura), que mostrou-se superior aos anteriores, atingindo valores bem próximos dos conhecidos para alguns casos, como é o da instância whitaker3, em que a solução encontrada teve um desvio de apenas 5.35% da melhor solução conhecida.

O nosso algoritmo apresentou dificuldade em melhorar as soluções iniciais geradas, e logo foram tentadas variações empíricas dos parâmetros do Simulated Annealing, e as vizinhanças Flip2 e FlipE, que não demonstraram variações significativas nos resultados do algoritmo.

Conforme a literatura pesquisada, os nossos resultados demonstraram-se satisfatórios para as mesmas instâncias **[1]**, ainda que não tenhamos conseguido nos equiparar com os resultados de Petit **[2]**. Esta não equiparação aos resultados da principal fonte de pesquisa deve-se, provavelmente, à tecnica adaptiva de definição dos parâmetros do Simulated Annealing, utilizada pelo mesmo.

Um possível melhoramento a ser feito, em trabalhos futuros, pode ser a aplicação do algoritmo de Two-Stage Simulated Annealing **[6]**, que conhecidamente supera os resultados do Simulated Annealing simples.

5 – Conclusões

O trabalho forneceu uma oportunidade interessante de realizar uma pesquisa em uma área específica da Otimização Combinatória, proporcionando uma experiência interessante de pesquisa e de implementação, tendo sido um processo desafiador buscar por soluções, na literatura, que melhorassem o algoritmo.

O processo foi interessante, pois a barreira dos primeiros algoritmos de Solução Inicial fez com que buscássemos por melhores alternativas para o problema, não só na literatura mas também na forma de implementar o software, o que fez com que otimizassemos também o nosso código, buscando estruturas de dados que melhorassem a performance.

Lamentamos a ausência dos resultados do solver GLPK para comparação com os resultados do algoritmo. Porém, após exaustivas tentativas, infelizmente não foi possível a definição do problema de maneira satisfatória.

Conforme os resultados obtidos através da pesquisa, acreditamos que os resultados estejam satisfatórios, pois equiparam-se àqueles obtidos por algoritmos homônimos em outros trabalhos.

6 – Referências

[1] Martí, Rafael, et al. "Scatter search and path relinking: a tutorial on the linear arrangement problem." *International Journal of Swarm Intelligence Research (IJSIR)* 2.2 (2011): 1-21.

[2] Petit, Jordi. "Experiments on the minimum linear arrangement problem."*Journal of Experimental Algorithmics (JEA)* 8 (2003): 2-3.

[3] Petit, Jordi. "Combining spectral sequencing and parallel simulated annealing for the MinLA problem." *Parallel Processing Letters* 13.01 (2003): 77-91.

[4] Martí, Rafael, and G. Reinelt. The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization. *Heidelberg: Springer*, 2011. Print.

[5] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of Np-Completeness*. W. H. Freeman & Co., New York, NY, USA.

[6] Rodriguez-Tello, Eduardo, Jin-Kao Hao, and Jose Torres-Jimenez. "An effective two-stage simulated annealing algorithm for the minimum linear arrangement problem." *Computers & Operations Research* 35.10. 2008.