Estudo do Problema de Árvores Geradoras de Rótulos Mínimos*

[Implementação e análise de instâncias usando JPSO] †

Paulo Batista da Costa
Universidade Tecnológica Federal do Paraná
2015 Campo Mourão
Campo Mourão, Brasil
pauloc@alunos.utfpr.edu.br

Felipe Veiga Ramos
Universidade Tecnológica Federal do Paraná
2015 Campo Mourão
Peabiru, Brasil
fveigaramos@gmail.com

ABSTRACT

Um dos assuntos fundamentais da área da Teoria dos grafos são as árvores geradoras de rótulos mínimos. Elas tem por objetivo encontrar o menor número possível de rótulos de um grafo não dirigido em conexo que cubram todos os vértices possíveis. Os problemas de árvores geradoras de rótulos mínimos ou PAGRM são de grande importância para áreas que projetam redes de comunicação, elétrica, transporte e dentre outros. Desse modo, para estudar e ampliar os conhecimentos na área este trabalho foi proposto. Ele consiste na implementação de um algoritmo – no caso o JPSO – e na sua comparação com demais algoritmos propostos pela literatura de acordo com uma base comum. Assim, este trabalho é uma oportunidade de compreender e estudar na prática conteúdos que envolvem teoria dos grafos.

Keywords

PAGRM; Rótulos mínimos; Teoria dos Grafos; JPSO;

1. INTRODUÇÃO

Um dos assuntos fundamentais da área de Teoria dos grafos são as árvores geradoras de rótulos mínimos. Esta pode ser entendida como uma árvore geradora obtida através de um grafo conectado não direcionado . Os Problemas das árvores geradoras de rótulos mínimos (PAGRM) podem ser formalmente definidos como: Dado um grafo rotulado não dirigido $G = (V, E, L)$ no qual V é o conjunto n de vértices, E é o conjunto de m arestas e L é o conjunto de l rótulos, encontrar uma árvore geradora T de G tal que $|L_T|$ é minimizado. Define-se $|L_T|$ como o conjunto de diferentes rótulos das arestas em uma árvore geradora T. [6]

Tal é utilizado para representar situações presentes no mundo real como redes de comunicação, elétrica, transporte e dentre outros. Sua compreensão e estudo é fundamental para a elaboração de soluções que visam reduzir custos e ampliar eficiência. Em outras palavras, o objetivo do estudo e aplicações de algoritmos para resolver o PAGRM é gerar - a partir de um grafo não dirigido - uma árvore que contenha o menor número possível de rótulos e ao mesmo tempo cobrindo todos os seus vértices.

Este trabalho tem por finalidade analisar o conjunto de instâncias – o mesmo conjunto utilizado em [2] – através da implementação (realizada na linguagem de programação JAVA) e execução do algoritmo JPSO (*Jumping Particle Swarm Optimization*) [3], e dessa forma estabelecer uma comparação entre os resultados obtidos desta aplicação e aqueles provindos das heurísticas apresentadas em [2].

Tal comparação levou em consideração fatores como a complexidade dos grafos (número de vértices e rótulos) , valor médio da função objetivo e o tempo computacional. Dessa forma, os resultados são comparados da mesma forma como reportado em [2].

O restante deste trabalho está organizado em seções da seguinte forma: Na seção 2 é realizada uma revisão literária, na 3 é apresentada a proposta de algoritmo exato para solução do PAGRM (no caso o JPSO), em 3.2 os resultados da implementação são mostrados, em 4 é explicitada uma comparação entre os resultados da literatura e os obtidos neste trabalho e por último são apresentadas as conclusões em 5.

2. REVISÃO LITERÁRIA

Algoritmos propostos para cobrir o máximo número de vértices, do inglês *Maximum vertez covering algorithm – MVCA*, tem sua origem da necessidade de criar sistemas em rede com menor custo e maior eficiência. Em 1997, Chang and Leu (1997) introduziram tal assunto e hoje este é conhecido como problema de árvore geradora de rótulos mínimos (PAGRM). Eles introduziram discussões sobre PAGRM expõem sua primeira solução e provam ser um problema NP completo [5], ou seja, que é um problema cuja resolução pode ser determinada em um tempo polinomial. Eles propõem inicialmente a seguinte heurística: Começa-se com um grafo vazio e adicione sempre arestas que cubram o maior número de vértices não visitados. Tal solução gera uma árvore

*(Produces the permission block, and copyright information). For use SIG-ALTERNATE.CLS. Supported by ACM.

†A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2ε and BibTeX* at `www.acm.org/eaddress.htm`

DOI: 10.1145/1235

geradora de rótulos mínimos. Entretanto, a solução inicial apresentou falhas – gerando grafos não conexos – e posteriormente sofreu modificações. No ano seguinte, Krumke e Wirth [4] propuseram a correção que gerava apenas grafos conexos que minizava o nímero de componentes. Essa correção alcançou complexidade $(1 + 2\log n)$. Entretanto,o trabalho mais recente [7] melhorou tal complexidade para $(1 + \log(n + 1))$. Em [8], os autores utilizaram outra aproximação para resolução: Algotimos Genéticos que utilizam conceitos provindos da biologia para resolver PAGRM. Em seguida, Cerulli et al em [1] aplicaram o *Pilot Method* e tal foi testado em conjuntos de instâncias obtendo a melhor resposta na maioria dos casos. Por sua vez, em 2006 Xiong et al em [9] implementou versões simplificadas do *Pilot Method* com modificações de algoritmos genéticos (MGA) e obteve uma performance melhor quanto a obtenção de respostas (capacidade de diversificação).

3. PROPOSTA DE ALGORITMO – O JPSO

3.1 Heurística - A ideia do JPSO

O PSO (Do inglês *Particle Swarm Optmization*) é um algoritmo que simula o comportamento de revoada de indivíduos na natureza. Cada solução do problema é tido como um indivíduo que pode voar dentro do espaço de soluções. A melhoria das posições, ou seja, as melhores respostas são obtidas através da continua movimentação das partículas que constituem tal revoada. É um algoritmo proposto por [3] que utiliza o conceito de inteligência coletiva e é capaz de auxiliar na busca por respostas em problemas de PAGRM. Entretanto, esse algoritmo é formulado para resolver problemas contínuos, o que contradiz o uso para tal. Isto pois, um PAGRM constitui um cenário de resolução discreta.

Portanto, para a utilização da ideia provinda do PSO é necessário que o algoritmo foque problemas discretos. Assim, uma adaptação é utilizada: O JPSO (*Jumping Particle Swarm Optimization*). O JPSO é um DPSO (Do inglês,*Discrete Particle Swarm Optimization*) [] , ou seja, possui a mesma essência da heurística aplicada pelo PSO com a diferença de ser voltado para problemas discretos. A ideia é que a cada iteração as possíveis resposta estejam mais próximas do que seria considerado como ótimo.

3.2 Explicações da implementação do JPSO

A implementação para este trabalho está disposta na linguagem de programação JAVA e está de acordo com o pseudocódigo descrito em []. Assim, a entrada para o JPSO é um grafo não dirigido e conexo $G = (V, E, L)$, com n vértices, m arestas, l rótulos e $Q \subseteq V$ nós. Além disto, tem como saída uma árvore geradora a qual objetiva ser mínima, ou seja, conter o menor número de rótulos possível e ao mesmo tempo cobrir todos os vértices do grafo.

Inicialmente foram carregadas as instâncias presentes nos arquivos disponibilizados em ambiente virtual referente aos grafos sobre os quais a análise foi aplicada e que são as mesmas utilizadas por [2]. Estas foram carregadas em estruturas as quais retornam um grafo não dirigido e conexo provindo da base de dados.

Em seguida, a partir de cada grafo abstraído da base de dados foram geradas 100 partículas. Estas partículas são a representação das possíveis soluções e são inicialmente formadas de forma randômica: Para cada partícula é induzido um grafo aleatoriamente a partir dos rótulos do grafo original.

A partir desse momento iniciam-se as iterações que buscam a melhor resposta, elas são caracterizadas pelos seguintes passos: 1. atualiza-se a melhor resposta (componente conexa com menor número de rótulos) para cada partícula , 2. atualiza-se a melhor resposta global já obtida, 3. Para cada partícula há a tentativa de reduzir o número de rótulos mantendo as propriedades necessárias, 4. tentativa de combinar duas soluções gerando um grafo com menor número de rótulos.

As tentativas de modificações de cada partículas são randômicas. Além disso é importante ressaltar que a cada iteração as respostas possíveis nunca apresentarão maior número de rótulos do que na iteração anterior, pois toda vez que ocorrer aumento no número de rótulos há o descarte da solução mantendo a anterior. Outro ponto importante é o critério de parada para o algoritmo:É finalizado a partir do momento em que se completam 100 iterações sem modificação no melhor resultado global.

Assim, o JPSO deve retornar a melhor resposta obtida durante sua execução. Apesar de garantir o retorno da melhor solução, não garante que esta seja a solução ótima. Em outras palavras, não retorna necessariamente a arvore com menor rótulos possível de ser formada. Isto devido ao fato de basear-se na aleatoriedade inicial das soluções, o que também implica que execuções distintas sobre a mesma base de dados retorne uma soluções diferentes.

Resultado da execução - O Algoritmo e sua Ação sobre as Instâncias

A aplicação do JPSO para PAGRM (que é um problema NP completo [5]) sobre a base disponibilizada gerou dados de saída que foram organizados em um arquivo texto em formato *txt*. Eles podem ser verificados de forma organizada nas tabelas a seguir em 1, 2, 3 e 4 e posteriormente são comparados aos resultados dos algoritmos exibidos em [2], estes sobre a mesma base de dados.

4. COMPARAÇÃO DE RESULTADOS

Dentre as tarefas necessárias para a realização deste trabalho está a realização da comparação entre os algoritmos presentes em [2] e o escolhido para a realização de implementação. Em [2] os algoritmos discutidos e executados sobre a mesma base de dados utilizada neste trabalho são: PILOT, MGA, GRASP e VNS. Assim, como os algoritmos são aplicados em uma base comum a deste trabalho, é possível estabelecer uma comparação plausível.

Tal comparação está divida em 3 grupos, baseados nas características de suas instâncias: grupo 1 é referente a instâncias pequenas que possuem número de vértices iguais ao número de rótulos, o grupo 2 considera instâncias maiores com o número de rótulos definidos por $0,25 * n$, onde n é o número de vértices e os rótulos varia de 25 a 125. Por último, o grupo 3 são as maiores instâncias presentes na base, pois é composta por parâmetros como número de vértices igual a 200 e número de rótulos que variam de 50 a 250 como visto na tabela 3 e número de vértices igual a 500 com número de rótulos variando de 125 a 625 como visto na tabela 4.

4.1 Comparações no grupo 1

N	L	D	F. Objetiva	Tempo (ms)
20	20	0.8	2.4	1002
		0.5	3.1	1543
		0.2	6.8	2156
30	30	0.8	2.8	1890
		0.5	3.7	4291
		0.2	7.5	6191
40	40	0.8	2.9	3021
		0.5	3.7	8018
		0.2	7.7	14583
50	50	0.8	3.0	6729
		0.5	4.5	15001
		0.2	8.8	3113
		Total	56.9	67538

Table 1: N é o número de nós, L é o número de rótulos e D é a densidade. A tabela mostra o tempo de execução de cada

N	L	D	F. Objetiva	Tempo (ms)
100	25	0.8	1.8	11440
		0.5	2.0	16664
		0.2	4.6	46836
	50	0.8	2.0	31146
		0.5	3.0	34667
		0.2	6.9	107502
	100	0.8	3.8	37138
		0.5	5.3	96218
		0.2	11.3	192166
	125	0.8	4.1	60090
		0.5	6.2	113139
		0.2	13.1	239019
	Total –		64.1	986025

Table 2: N é o número de nós, L é o número de rótulos e D é a densidade. A tabela mostra o tempo de execução de cada

No grupo 1 (120 instâncias), dentre os algoritmos apresentados em [2] os algoritmos MGA e PILOT são antagônicos: O segundo é mais rápido que o primeiro, entretanto provendo respostas piores. Enquanto isso, o segundo é mais demorado quanto a execução, mas gera soluções melhores. Por sua vez, GRASP e VNS são os mais rápidos e promovem a solução exata. Estes são excelente para essas instâncias. Comparando o JPSO implementado é perceptível que este é mais rápido que MGA (em função do tempo de execução) e mais lento que os demais. Além disso, o JPSO não promove a certeza de que a solução obtida é a exata (ótima).

O MGA é o mais lento por ser o que mais diversifica as possíveis soluções (no objetivo de encontrar a melhor), porém retornando uma resposta melhor que o PILOT que diversifica pouco, mas apresenta uma resposta pior. Assim, o JPSO é mais executa mais rápido do que o algoritmo mais lento apresentado em [2] , levando em considerações as instâncias do grupo 1 (cujas características podem ser vistas na tabela 1).

4.2 Comparações no grupo 2

No grupo 2, instâncias com número de vértices igual a 100 (360 instâncias), a melhor performance é tida pelo algoritmo VNS. Isto, pois apresenta como solução a melhor resposta (exata) e no menor tempo dentre os algoritmos apresentados. GRASP por sua vez também pode ser enquadrado como um algoritmo bom e obtém a mesma solução que o VNS. Aqui PILOT apresenta o segundo maior tempo de execução, tendo respostas piores que os demais. MGA continua sendo aquele de maior tempo. Aqui é possível perceber que ao aumentar o número de instâncias, o PILOT passa a perder aquilo que ele tinha de vantagem em relação aos demais que é a rapidez de execução. O JPSO aqui implementado possui um tempo de maior e ainda não produz uma resposta com a garantia que seja a solução ótima. Portanto a melhor opção para resolver um PAGRM é o uso do VNS.

4.3 Comparações no grupo 3

Para instâncias maiores, como as que compõem o grupo 3 (referente à tabela 3 e 4) as heurísticas referentes ao VNS e ao GRASP se mantiveram superiores nos quesitos qualidade da resposta e tempo de execução. O GRASP passa a ter menor tempo de execução em relação ao VNS quão maior for o PGRM em questão. Em relação ao JPSO, é perceptível através da tabela 3 que em instâncias com o número de vértices igual a 200 possui um tempo de execução melhor apenas que o MGA. já em instâncias com o número de vértices igual a 500 (as maiores instâncias) é notável que ele apresenta o maior tempo de execução sem garantir o resultado ótimo.

5. CONCLUSÃO

Em nosso trabalho implementamos o algoritmo JPSO para fim de execução da base de dados disponibilizada. Esta é a mesma aplicada em [2] e este fato permitiu que fossem comparados os resultados obtidos pela implementação aqui discutida e aqueles apresentados por demais soluções. O objetivos deste trabalho foi ampliar os conhecimentos a respeito do PAGRM que é o foco dos algoritmos apresentados.

N	L	D	F. Objetiva	Tempo (ms)
200	50	0.8	2.0	53185
		0.5	2.3	130555
		0.2	5.7	306403
	100	0.8	2.9	99256
		0.5	4.1	157123
		0.2	9.4	574387
	200	0.8	4.6	254051
		0.5	6.8	655515
		0.2	15.3	1184675
	250	0.8	5.1	566244
		0.5	8.0	933404
		0.2	17.9	1441719
		Total –	84.1	6856514

Table 3: N é o número de nós, L é o número de rótulos e D é a densidade. A tabela mostra o tempo de execução de cada

N	L	D	F. Objetiva	Tempo (ms)
500	125	0.8	2.0	580854
		0.5	3.1	1535064
		0.2	7.3	4508425
	250	0.8	3.1	2351650
		0.5	4.9	6289072
		0.2	12.9	9052493
	500	0.8	5.8	6751334
		0.5	8.7	16560601
		0.2	21.3	16661087
	625	0.8	6.7	18957465
		0.5	10.5	17878234
		0.2	25.4	18310678
		Total –		

Table 4: N é o número de nós, L é o número de rótulos e D é a densidade. A tabela mostra o tempo de execução de cada

Nota-se que para instâncias maiores os algoritmos VNS e GRASP são os mais compensativos, levando em consideração aspectos como tempo de execução e garantia de solução ótima. O JPSO implementado possui um baixo rendimento para resolver um JPSO quando comparado às duas melhores heurísticas, pois não garante a obtenção da solução exata e ainda tem um tempo de execução maior, sendo compensativo em alguns casos em relação ao MGA e ao PILOT. Acredita-se que o motivo do execução do JPSO implementado está atrelado ao critério de parada do laço de repetição contido no algoritmo: A execução continuará até que a melhor solução global não se altere por até 100 iterações, assim a redução desse critério influencia no tempo de execução. Além disso, foi possível notar que dependendo do objetivo e das condições apresentadas em mundo real, uma solução rápida de ser executada pode gerar uma resposta que não compensa – caso do PILOT em um conjunto de instâncias pequenas. Este trabalho foi fundamental para ampliar a visão a respeito do PAGRM e a forma como especialistas da área de grafos estudam heurísticas candidatas a serem boas alternativas de resolução.

.

6. REFERENCES

[1] R. Cerulli, A. Fink, M. Gentili, and S. Voß. Metaheuristics comparison for the minimum labelling spanning tree problem. In *The Next Wave in Computing, Optimization, and Decision Technologies*, pages 93–106. Springer, 2005.

[2] S. Consoli, K. Darby-Dowman, N. Mladenović, and J. M. Pérez. Greedy randomized adaptive search and variable neighbourhood search for the minimum labelling spanning tree problem. *European Journal of Operational Research*, 196(2):440–449, 2009.

[3] S. Consoli, J. M. Pérez, K. Darby-Dowman, and N. Mladenović. Discrete particle swarm optimization for the minimum labelling steiner tree problem. In *Nature inspired cooperative strategies for optimization (NICSO 2007)*, pages 313–322. Springer, 2008.

[4] S. O. Krumke and H.-C. Wirth. On the minimum label spanning tree problem. *Information Processing Letters*, 66(2):81–85, 1998.

[5] R. E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, Jan. 1975.

[6] S. Pettie and V. Ramachandran. An optimal minimum spanning tree algorithm. *J. ACM*, 49(1):16–34, Jan. 2002.

[7] Y. Wan, G. Chen, and Y. Xu. A note on the minimum label spanning tree. *Information Processing Letters*, 84(2):99–101, 2002.

[8] Y. Xiong, B. Golden, and E. Wasil. A one-parameter genetic algorithm for the minimum labeling spanning tree problem. *Evolutionary Computation, IEEE Transactions on*, 9(1):55–60, 2005.

[9] Y. Xiong, B. Golden, and E. Wasil. Improved heuristics for the minimum label spanning tree problem. *Evolutionary Computation, IEEE Transactions on*,

10(6):700–703, 2006.