INF05010 Otimização Combinatória

Trabalho Final:

Simulated Annealing para o Problema da Coloração Mais Balanceada

Prof. Marcus Ritt
Eduarda Trindade

04 Dezembro 2018

1. Introdução

Este trabalho teve como objetivo utilizar uma meta-heurística para resolver o problema da coloração mais balanceada. A meta-heurística implementada foi o Simulated Annealing, cujos detalhes serão mostrados na seção 4.

A definição do problema da coloração mais balanceada é a seguinte: dado um grafo G = (V, E) com pesos $w_i \in \mathbb{R}^+$ para cada $i \in V$, e um inteiro positivo k, deseja-se encontrar uma k-coloração de G que minimize o peso máximo de uma cor $\max_{i \in [k]} \sum_{v \in Ci} w_i$.

2. Formulação do programa inteiro misto

Variáveis:

- $\boldsymbol{x_{ij}} \in \{0, 1\} \quad \forall i \in V,\ j \in [k]$, tal que:

  $\boldsymbol{x_{ij}} = \mathbf{1}$, se vértice $i \in V$ pertence a cor $j \in [k]$.

  $\boldsymbol{x_{ij}} = \mathbf{0}$, caso contrário.

- $\boldsymbol{m} \in \mathbb{R}^+$ = peso máximo de uma cor.

Função Objetivo:

$$min.\ m$$

Restrições:

$$\sum_{j \in [k]} x_{ij} = 1, \qquad \forall i \in V \tag{1}$$

$$x_{uj} + x_{vj} \leq 1, \qquad \forall (u,v) \in E, j \in [\mathrm{k}] \tag{2}$$

$$m \geq \sum_{i \in V} x_{ij} * w_i, \qquad \forall j \in [k] \tag{3}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in V, \forall j \in [k] \tag{4}$$

$$m \in \mathbb{R}^+ \tag{5}$$

(1) Garante que um vértice tenha apenas uma cor.

(2) Garante que dois vértices que possuem uma aresta não tenham a mesma cor.

(3) Garante que ***m*** seja o peso da cor de maior peso.

3. Elementos e parâmetros do Simulated Annealing

O Simulated Annealing é um algoritmo que simula um processo de recozimento, que é um processo físico no qual se aquece um material a uma temperatura bem alta e resfria aos poucos, dando tempo para o material alcançar seu estado de equilíbrio. Nessa meta-heurística, a temperatura inicia alta e vai diminuindo ao ser multiplicada por um fator de resfriamento. Quanto mais alta estiver a temperatura mais ela vai permitir a chance de que soluções geradas por vizinhos de valores piores sejam aceitas com uma probabilidade e $^{-\Delta/T}$, isso é o que permite o algoritmo de escapar de mínimos ou máximos locais.

Os parâmetros do Simulated Annealing são:

s - Solução inicial

r - Fator de resfriamento, número real entre 0 e 1.

I - Iterações, número inteiro ≥ 1. Para uma mesma temperatura, vai gerar I vizinhos.

pi - Probabilidade inicial, número real entre 0 e 1. É utilizada para determinar uma temperatura inicial, permite movimentos piores com essa probabilidade.

pf - Probabilidade final, número real entre 0 e 1. É utilizada para critério de parada. Detalhado na seção 4.6.

4. O algoritmo

```
SimulatedAnnealing(r, I, pi, pf, n, e, k, W, E)
1:  s ← GeraSoluçãoInicial(n, e, k)
2:  t ← GeraTemperaturaInicial(s, r, I, pi, n, e, k, W)
3:  contador ← 0
4:  valor_atual ← ValorDaSolução(s, n, e, k, W)
5:  while contador < 5 do
6:          movimentos_aceitos ← 0
7:          tentativas_de_movimento ← 0
8:          for i in 0 to I do
9:                  s' ← GeraVizinhoAleatório(s, n, k)
10:                 valor_candidato ← ValorDaSolução(s', n, e, k, W)
11:                 delta ← valor_candidato - valor_atual
12:                 if delta <= 0 then
13:                         s ← s'
14:                         valor_atual ← valor_candidato
15:                         contador ← 0
16:                 else
17:                         tentativas_de_movimento ← tentativas_de_movimento + 1
18:                         if GeraNúmeroRandômico(0,1) < e^(−Δ/T) then
19:                                 movimentos_aceitos ← movimentos_aceitos + 1
20:                                 s ← s'
21:                         valor_atual ← valor_candidato
22:                         end if
23:                 end if
24:         end for
25:         if tentativas_de_movimento > 0 then
26:                 if pf > (movimentos_aceitos/tentativas_de_movimento) then
27:                         contador ← contador + 1
28:                 end if
29:         end if
30:         t ← t * r
31: end while
32: return s
```

4.1. Instância

Para a utilizar o algoritmo de Simulated Annealing para o problema de Coloração mais balanceada foi necessário utilizar mais alguns parâmetros:

num_vertices: número de vértices do grafo da instância, número inteiro entre 1 e *n*.

num_edges: número de arestas do grafo da instância, número inteiro entre 1 e *e*.

num_colors: número cores da instância, número inteiro entre 1 e *k*.

weights: lista de pesos dos *n* vértices, número real ≥ 0.

edges: matriz de adjacências.

4.2. Representação da solução

A solução é uma k-coloração do grafo representada por uma matriz $S_{n \times k}$ contendo valor 0 ou 1, na qual os índices das linhas representam os vértices e os índices das colunas representam as cores. Se uma posição [*v*][*c*] da matriz tem o valor 1, significa que o vértice *v* é colorido com a cor *c*.

4.3. Solução inicial

A solução inicial é gerada aleatoriamente. Para cada vértice da instância é atribuída uma cor aleatória. Isso pode levar a soluções não factíveis, gerando vértices adjacentes coloridos com a mesma cor. Se a solução for factível seu valor, que é dado pelo peso máximo de uma cor como visto na seção 1, será calculado normalmente. Se a solução não for factível seu valor é somado com penalidade de valor 10000 multiplicada pelo número de conflitos de cores entre vértices adjacentes. Assim, uma solução com menos conflitos ou até mesmo nenhum terá necessariamente um valor melhor.

4.4. Vizinhanças

Para gerar um vizinho um vértice aleatório é selecionado e tem sua cor mudada para uma nova cor aleatória diferente da anterior. Se o vizinho gerado não for factível terá seu valor somado à mesma penalidade apresentada na seção 4.3.

4.5. Temperatura inicial

É definida uma probabilidade *pi* que é passada para uma versão rápida (I = 100) do algoritmo de Simulated Annealing que serve para determinar uma temperatura inicial tal que um movimento para valor pior que o atual é aceito com probabilidade de aproximadamente *pi*.

4.6. Critério de parada

É definida uma probabilidade *pf,* para cada nível de temperatura em que os movimentos para valor pior foram aceitos com probabilidade menor que *pf* incrementa um contador. Zera o contador se uma nova melhor solução é encontrada e se o contador chegar em 5 considera-se que o algoritmo foi resfriado e a execução termina.

5. Implementação

5.1. Plataforma de implementação

O trabalho foi implementado e testado em sistema operacional Windows 10 Home (64-bit), com um processador Intel(R) Core(TM) i7-8700 CPU com 6 núcleos físicos e 12 virtuais de 3,20 Ghz, com 32KB de cache L1 e 256KB de cache L2 e 16GB de memória. A linguagem de programação utilizada foi Python 3.5.3.

5.2. Estruturas de dados utilizadas

Os pesos dos vértices foram implementados em uma lista do tamanho do número de vértices, na qual cada posição de índice i tem o peso do vértice de mesmo índice. As arestas foram representadas por uma matriz de adjacências

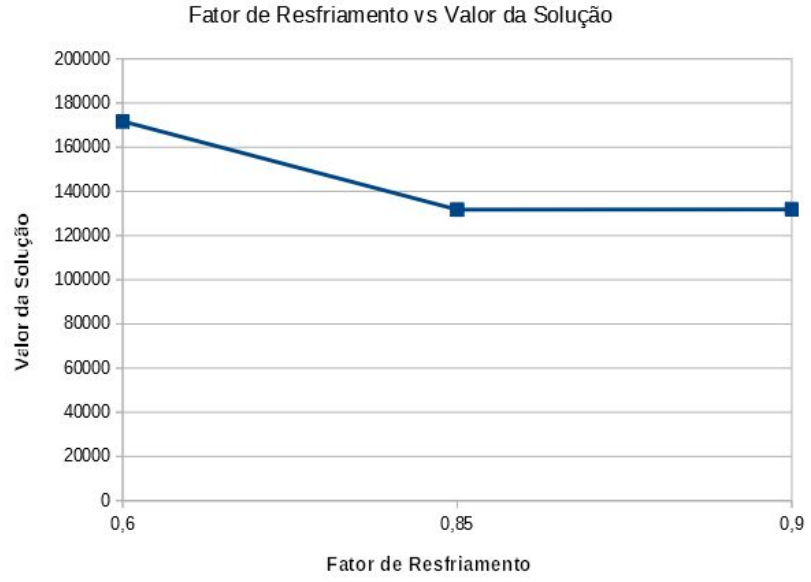
sem armazenar os dados simétricos do grafo não direcionado. E a solução foi estruturada como mencionado na seção 4.3.

6. Testes de parâmetros

Para os seguintes testes, variou-se cada parâmetro de entrada separadamente, para determinar procurar a configuração mais adequada. Os testes foram realizados com a instância cmb01, o valor inicial de teste para os parâmetros que não estavam sendo testados são: $r = 0.85$, $I = 200$, $pi = 0.85$, $pf = 0.1$. Para cada teste, mediu-se o tempo de execução, o valor da solução obtida e o número de conflitos de cores restantes para as soluções não factíveis.

6.1. Fator de resfriamento *r*

Foram testados valores de *r* para 0.60 , 0.85 e 0.90. As figuras e tabelas abaixo mostram os resultados obtidos.



Valor Inicial	Número Inicial de Conflitos de Cores
2983288.42	283

r	Valor Final	Número Final de Conflitos de Cores	Tempo (segundos)
0.60	171721.1500	**7**	5198.3706
0.85	131747.9300	3	6810.8182
0.90	131821.4199	3	10643.4064

Observa-se melhora do valor final e do número de conflitos de cores ao aumentar o valor de *r* de 0.60 para 0.85 com um acréscimo de tempo de aproximadamente 27 minutos. Porém ao analisar o valor de *r* de 0.90 o valor final foi aproximadamente igual ao de *r* = 0.85, o número de conflitos permaneceu o mesmo e o acréscimo de tempo foi de aproximadamente uma hora. Sendo assim, conclui-se que um valor razoável para *r* é 0.85.

Além desses testes, foi feito um com *r* = 0.85 e com uma temperatura inicial fixa igual a 50 ao invés de deixar a temperatura inicial ser gerada pela execução pequena do Simulated Annealing. A tabela abaixo mostra os resultados obtidos.

r	Valor Final	Número Final de Conflitos de Cores	Tempo (segundos)
0.85	171712.26	7	5568.1933

Observa-se que o valor final piorou em relação a execução com *r* = 0.85 e temperatura não fixa. Além disso, o número de conflitos aumentou em 4. Esse resultado ficou próximo do obtido no teste anterior com valor *r* = 0.60. Dessa forma, conclui-se que utilizar uma versão rápida do algoritmo de Simulated Annealing para determinar a temperatura inicial se mostrou melhor que partir de uma temperatura fixa.

6.2. Número de iterações *I*

Foram testados valores de *I* para 100 e 200. As tabelas mostram os resultados obtidos.

Valor Inicial	Número Inicial de Conflitos de Cores
2983288.42	283

I	Valor Final	Número Final de Conflitos de Cores	Tempo (segundos)
100	192110.9699	9	3125.6076
200	131747.9300	3	6810.8182

Observa-se melhora do valor final e do número de conflitos de cores ao aumentar o valor de *I* de 100 para 200 iterações. Apesar de ter um acréscimo de tempo de aproximadamente uma hora, o número de conflitos ficou três vezes menor. Portanto, conclui-se que um valor razoável para *I* é 200. Foram feitos testes para valores de *I* maiores porém o algoritmo executou por muitas horas sem gerar resultado, que não pode ser analisado.

6.3. Probabilidade Inicial *pi*

Foram testados valores de *pi* para 0.85 e 0.95. As tabelas mostram os resultados obtidos.

Valor Inicial	Número Inicial de Conflitos de Cores
2983288.42	283

pi	Valor Final	Número Final de Conflitos de Cores	Tempo (segundos)
0.85	131747.9300	3	6810.8182
0.95	131747.9300	3	7741.9357

Observa-se que o resultado foi igual para esses valores de *pi*, portanto por ter tempo menor foi selecionado o valor 0.85 para *pi*.

6.4. Probabilidade Final *pf*

Foram testados valores de *pf* para 0.01 e 0.1. As tabelas mostram os resultados obtidos.

Valor Inicial	Número Inicial de Conflitos de Cores
2983288.42	283

pi	Valor Final	Número Final de Conflitos de Cores	Tempo (segundos)
0.01	131747.9300	3	7850.5063
0.10	131747.9300	3	6810.8182

Observa-se que o resultado foi igual para esses valores de *pf*, portanto por ter tempo menor foi selecionado o valor 0.10 para *pf*.

7. Testes das instâncias

Foi fornecido um conjunto de 10 instâncias, cmb01, cmb02, cmb03, cmb04, cmb05, cmb06, cmb07, cmb08, cmb09, cmb10 disponíveis em http://www.inf.ufrgs.br/~mrpritt/oc/cmb.zip. Também foi criada uma instância

pequena para teste chamada “my”. Os parâmetros escolhidos para esses testes foram $r = 0.85$, $I = 200$, $pi = 0.85$, $pf = 0.1$, com base nos testes feitos para os parâmetros mostrados na seção 6.

Utilizando a formulação com o solver GLPK com limite de tempo uma hora não foram encontradas soluções factíveis para nenhuma das instâncias fornecidas, embora com a instância pequena “my” tenha encontrado solução ótima.

Utilizando a meta-heurística do Simulated Annealing implementada foram obtidos resultados apenas para a instância cmb01 devido ao tempo de execução de muitas horas para os parâmetros estabelecidos.

Instância	Valor Inicial	Número Inicial de Conflitos de Cores	Valor Final	Número Final de Conflitos de Cores	Desvio para Sol. Inicial (%)	Tempo de Execução (segundos)	Seed
my	20039.4	2	32.6	0	99.8914798	0.5515	93563511
cmb01	2983288.42	283	131747.93	3	95.5838	6810.8182	10
cmb02	-	-	-	-	-	>14400	13766871
cmb03	-	-	-	-	-	>14400	775729445
cmb04	-	-	-	-	-	>14400	8943774
cmb05	-	-	-	-	-	>14400	166164842
cmb06	-	-	-	-	-	>14400	239822
cmb07	-	-	-	-	-	>14400	3453
cmb08	-	-	-	-	-	>14400	150650
cmb09	-	-	-	-	-	>14400	230590
cmb10	-	-	-	-	-	>14400	616

8. Conclusão

Considera-se que o algoritmo teve uma performance ruim pois não foram encontrados resultados factíveis em tempo de até quatro horas. Houve melhora em relação ao valor da solução inicial e quanto ao número de conflitos de cores, porém em tempo hábil o algoritmo não eliminou esses conflitos.

Através dos testes para cada parâmetro separadamente, propõe-se uma melhor configuração para o algoritmo dada por: $r = 0.85$, $I = 200$, $pi = 0.85$, $pf = 0.1$.

Se o algoritmo fosse executado por tempo maior talvez fosse possível encontrar melhores resultados utilizando o Simulated Annealing, possivelmente levando a soluções factíveis sem conflito de cores entre vértices adjacentes.

Algumas mudanças de escolhas feitas na implementação poderiam tornar o tempo de execução menor para instâncias grandes como as fornecidas, assim também possibilitando gerar mais testes de parâmetros por levar menos tempo. Representar a solução como uma matriz foi uma escolha computacionalmente custosa, pois muitas vezes é necessário consultar a matriz para calcular o valor da solução, talvez implementá-la como uma lista de

listas em que cada posição da lista representa uma cor que contém uma lista de vértices. Além disso a escolha de representação das *edges* foi custosa, pois é feita por uma matriz de adjacência que é consultada diversas vezes para contar o número de conflitos de cores de uma solução, poderia buscar outra forma mais otimizada de representar as arestas.

Seria interessante experimentar mudar a forma de gerar a solução inicial e as vizinhanças da meta-heurística. Criando uma solução inicial factível o algoritmo partiria de um valor inicial melhor do que criando uma solução aleatória com penalidades para conflitos. A geração de vizinhos também poderia garantir que sejam factíveis ou então melhorar a geração aleatória levando em conta a cor atual de maior peso no momento de selecionar um vértice aleatório.

Tendo em vista os pontos citados acima, considera-se que o trabalho, como um todo, não foi tão bem sucedido quanto o esperado e poderia ser melhorado com escolhas melhores de representação.

Referências

BURIOL, Luciana; RITT, Marcus; COSTA, Alysson M. INF05010 - Otimização combinatória Notas de aula. Instituto de Informática, Departamento de Informática Teórica, Universidade Federal do Rio Grande do Sul, Porto Alegre: [s.n.], 2018. cap. 5, 6, 9, 10.2.