Universidade Federal do Rio Grande do Sul

Simulated Annealing Aplicado ao problema da Partição Mais Distante

Aluno:
Rafael Kehl

Professor:
Marcus Ritt

18 de Dezembro, 2017

Sumário

1 Introdução

Na computação existe a classe dos problemas NP. Para eles, não existe algoritmo eficiente que garanta a otimalidade da solução, portanto uma forma de atacar esse problema é através de heurísticas que muitas vezes nos dão um resultado satisfatório, mesmo sem garantia de otimalidade.

Neste trabalho foi modelado e implementado o problema da Partição Mais Distante para ser resolvido através do GLPK e de uma implementação do Simulated Annealing. Abaixo serão apresentados os resultados obtidos para cada uma das instâncias dadas.

Não. Como P \subset

2 Definição do problema

Seja $G = (V, A)$ um grafo completo não-direcionado, com distâncias $d_a \geqslant 0, \forall a \in A$ e pesos $p_v \geqslant 0, \forall v \in V$. Temos então que organizar g grupos G_k com peso alvo $M_k \geqslant 0, \forall k \in [g]$ e uma tolerância $\alpha \geqslant 0$ de tal forma que seja maximizada a distância mínima entre os vértices do mesmo grupo, ou seja:

$$\max_{G_1,\ldots,G_g \in \mathcal{P}(\mathcal{V})} \; \min_{k\in[g], u,v\in G_k | (u,v)\in\mathcal{A}} d_{(u,v)}$$

Assim, tomando $n = \|V\|$. Seja $x_{ik} = \begin{cases} 1, \text{se } v_i \in G_k \\ 0, \text{ caso contrário} \end{cases}$ e $z_{ijk} = \begin{cases} 1, \text{se } v_i, v_j \in G_k \\ 0, \text{ caso contrário} \end{cases}$ com $i, j \in [n]$ e $k \in [g]$, e M como um $big - M$. Com isso, podemos resolver o problema da Partição Mais Distante com o seguinte programa linear:

Seria interessante explicar essa parte do m

$$\begin{aligned}
\text{maximizar} \quad & u \\
\text{sujeito a} \quad & u \leqslant d_{ij} + M(1 - z_{ijk}), && \forall i, j \in [n],\ k \in [g] \\
& z_{ijk} \geqslant x_{ik} + x_{jk} - 1, && \forall i, j \in [n],\ k \in [g] \\
& z_{ijk} \leqslant x_{ik}, && \forall i, j \in [n],\ k \in [g] \\
& \sum_{i\in[n]} x_{ik} p_i \leqslant (1 + \alpha) M_k, && \forall k \in [g] \\
& \sum_{i\in[n]} x_{ik} p_i \geqslant (1 - \alpha) M_k, && \forall k \in [g] \\
& \sum_{k\in[g]} x_{ik} = 1, && \forall i \in [n] \\
& x_{ij} \in \{0, 1\},\ z_{ijk} \in \{0, 1\}
\end{aligned}$$

2.1 Simulated Annealing

O Simulated Annealing, ou Têmpera Simulada, é um processo físico que consiste no resfriamento controlado e gradual de um sólido, de forma que os átomos se componham numa estrutura uniforme de energia mínima, reduzindo os defeitos do material.

Mas teu problema

A meta-heurística simula esse processo, tomando cada solução possível como uma das possíveis estruturas atômicas, sendo a ótima a de energia mínima. O nível de energia e cada configuração seria o valor da função objetivo e a temperatura um parâmetro de controle para determinar se o sistema irá ou não para um de maior energia.

Note que, dependendo da temperatura, há uma probabilidade do sistema migrar para outro de maior energia, diminuindo a qualidade da solução. À primeira vista, isto não parece ser uma boa estratégia, porém é justamente isso que nos permite sair de mínimos locais da função e nos aproximarmos cada vez mais do mínimo global. Essa probabilidade é dada por $P(\Delta E) = e^{\frac{-\Delta E}{kT}} \cong e^{\frac{-\Delta E}{T}}$.

Como a formulação do problema consiste de uma maximização, definimos $\Delta E = E' - E$, onde E é o atual valor da função objetivo e E' é o valor da solução vizinha. Neste caso, adotamos como solução vizinha aquela que é obtida ao trocar dois vértices de grupo, ambos selecionados aleatoriamente.

Determinadas todos os parâmetros, o pseudo-código do simulated annealing é:

```
1  SimulatedAnnealing(s, T, k, r, I) :=
2    repeat sistema ``esfriado''
3      repeat I vezes
4        seleciona s' ∈ N(s) aleatoriamente
5        seja Δ := c(s') − c(s)
6        if Δ ≤ 0 then
7            s := s'
8          else
9            s := s' com probabilidade e^(−Δ/T):
10         end fi
11     end repeat
12     T := rT
13   end repeat
14 return s
```

Na implementação feita foram escolhidos como critérios de parada o tempo ou n^2 iterações sem melhora na função objetivo.

A solução inicial foi gerada selecionando vértices aleatóriamente e adicionando a um grupo, ordenados em ordem de capacidade, até que todos os grupos possuam o peso mínimo. Após isso, os vértices restantes são adicionados do mais pesado para o mais leve aos grupos que possuem maior capacidade livre, de forma que não ultrapasse a capacidade máxima, até que todos os vértices sejam distribuídos.

Ou seja, se tenta cons

3 Resultados (com bug)

A implementação do SA (Simulated Annealing) é bastante simples, porém a eficiência dele depende muito da escolha dos parêmetros. Para tal é necessário comparar as soluções obtidas com a ótima ou através de muitas execuções com variações nos parâmetros. Portanto, executei o algoritmo diversas vezes, sempre analisando as diferenças nos resultados finais e comparei os mesmos com os Best Known Value (BKV) dados.

Escolhi, primeiramente, a temperatura de tal forma que, nas primeiras iterações e no caso médio, a chance de ir para uma solução de maior energia fosse próxima de 80%. Assim, comecei meus testes com $T = 50$ e $r = 0.99$ e sempre diminuindo a temperatura quando pulasse para uma solução melhor que a atual. Testei também para as combinações possíveis de $T = 80$ e outros maiores, $T = 30$ e $r = 0.95$ e $r = 0.9$.

Seria bom ter esses re

Para os valores de r e T menores que os primeiros que tomei o sistema resfriou muito rápido e minha solução final dependia apenas da solução inicial, se tratando portanto de um máximo local. Para $T = 80$ ou maiores e $r = 0.99$ obtive exatamente os mesmos resultados que para $T = 50$. Portanto, abaixo podemos ver os resultados obtidos para $T = 50$, $r = 0.99$, $\alpha = 0.05$, $t = 2700seg$ para todas as soluções iniciais aleatórias distintas encontradas.

45 minutos para a heurística é muito tempo, principalmente considerando os resultados obtidos.

Tempos de exe

Instância	Solução Inicial	Solução Obtida	Melhora (%)	BKV	Nº Trocas	Nº Melhoras
300-5-0.75-1	1.76055572354	2.80610521686	59.4	0.501	82111	832
300-5-0.75-1	1.89198636362	2.80963766287	48.5	0.501	87745	806
300-5-1.00-1	0.472817157056	2.34897958119	396.8	0.530	81845	365
300-5-1.00-1	1.28811712098	2.34897958119	82.35	0.530	85422	472
400-11-0.75-1	1.87239074351	2.71878453232	45.2	1.110	95540	838
400-11-1.00-1	1.86888043899	3.62030111329	93.71	0.985	99705	835
500-10-0.75-1	1.69723601087	2.58862624532	52.52	0.748	66281	421
500-10-0.75-1	1.88518476237	2.58862624532	37.31	0.748	67111	419
500-10-1.00-1	1.15940469939	2.22899594721	92.25	0.734	68845	397
600-10-0.75-1	0.758537341152	1.47159276379	94	0.751	42901	245
600-10-1.00-1	1.4498727059	1.99181885086	37.37	0.712	43772	100
700-12-0.75-1	1.10541580693	2.24735368114	103.3	0.763	35813	155
700-12-1.00-1	0.646235634068	1.72911027509	167.56	0.859	40360	217

Tabela 1: Tabela com os resultados obtidos pela execução do SA

Fiquei extremamente surpreso com o resultado da meta-heurística, que obteve soluções muito melhores que os BKV para todas as instâncias. Como não fui capaz de obter soluções via GLPK, por questões de hardware, não sou capaz de determinar exatamente o que houve.

Que questões de hardw

4 Conclusões (com bug)

Foi uma grande supresa ver que a meta-heurística obteve resultados muito melhores que o BKV para todas as instâncias, inclusive nas soluções iniciais aleatórias. Minha hipótese é de que meu valor de α seja muito maior do que o utilizado para obter os BKV, dessa forma o meu espaço de soluções é muito maior.

Além disso, analisando todas as soluções obtidas, pude ver que o valor final dependia quase que unicamente da solução inicial. Isso indica pouca variabilidade no espaço de soluções durante a execução, o que deveria ser mudado ao modificar T e r, mas isso não foi observado também. Portanto, o que deve estar pesando é o fator da aleatoriedade ou a dificuldade de achar uma solução factível. Para identificar qual o real problema teriam de ser feitos maiores testes e obter melhores soluções pelo GLPK.

5 Resultados

Havia um bug no algoritmo executado. Ao calcular a distâncias de cada coordenada a função estava somando as mesmas, ao invés de subtrair. Isso levou a resultados muito errados e, portanto, a uma análise equivocada.

Foi então corrigido o bug e as instâncias foram executadas novamente. Os parâmetros foram mantidos os mesmos, já que a temperatura inicial possibilitava uma probabilidade ainda maior que 80% no caso médio agora que as distâncias são menores e achei que isso seria razoável. Executei alguns testes e constatei que o sistema estava resfriando muito rapidamente e o valor final obtido dependia quase que exclusivamente da solução inicial, então fiz com que o sistema esfriasse somente uma vez a cada 10 trocas. Isso eliminou este problema.

Porém, me surpreendeu que o número de trocas feitas eram baixíssimos. Quase todas instâncias realizaram cerca de 10 mil trocas, somente uma apresentou um número quase cinco vezes maior em duas de suas execuções, mas com um número semelhante de melhoras.

Para testar o motivo disso, executei o algoritmo para uma das instâncias que realizaram pouquíssimas trocas com $T = 1000$, porém o número de trocas permaneceu quase o mesmo. Não consegui realizar maiores testes. Teria de ser analisado mais a fundo como a probabilidade de aceite estáva se comportando durante a execução para estabelecer uma estratégia melhor de como atacar esse problema e melhorar a solução.

Instância	Solução Inicial	Solução Obtida	Otimalidade (%)	BKV	Nº Trocas	Nº Melhoras
300-5-0.75-1	0.0660136295156	0.155741460153	31.08	0.501	10503	131
300-5-1.00-1	0.0580505605609	0.136585398461	25.77	0.530	49578	146
400-11-0.75-1	0.0605888046031	0.163105476245	14.69	1.110	11292	113
400-11-1.00-1	0.0787513229477	0.165382568319	16.7	0.985	10548	107
500-10-0.75-1	0.0274848569491	0.143635526288	19.20	0.748	10535	105
500-10-1.00-1	0.0411223331156	0.127896034861	17.42	0.734	11075	102
600-10-0.75-1	0.0507292065809	0.115805784266	15.42	0.751	11156	81
600-10-1.00-1	0.0429877343876	0.108679849135	15.26	0.712	11513	73
700-12-0.75-1	0.0218696278815	0.1019195749	13.35	0.763	11366	60
700-12-1.00-1	0.0519530093692	0.117799485471	13.71	0.859	11439	66

Tabela 2: Tabela com os melhores resultados obtidos para cada instância.

6 Conclusões

O baixíssimo número de trocas e o menor ainda índice de melhoras (cerca de 1%) foi um grande problema na execução do algoritmo. Além de ser difícil manter a factibilidade, é muito baixo o número de sequências de trocas que levam a uma melhora. Portando, teriam de ser realizados novos testes alternando entre outros seeds randômicos, para garantir uma maior cobertura do espaço de soluções, além de uma investigação mais profunda do comportamento da função de probabilidade de aceite.

Para realizar estes testes, seria interessante implementar o algoritmo em outra linguagem, como C, além do Python. Além disso, seria ideal executar as instâncias por períodos maiores de tempo e com decaimento de temperatura ainda mais lento, para explorar mais o espaço de soluções e analisar como isso impacta no resultado final.

Meu maior erro foi usar parâmetros fixos para todas as instâncias, ao invés de obtê-los através de uma função sobre o número de vértices, por exemplo. Fazendo isso eu teria mais informações sobre como a solução se comporta em casos mais diversos e, até mesmo, poderia ter aumentado a qualidade das soluções.

Existe um problema de

Referência

[1] S. Kirkpatrick; C. D. Gelatt; M. P. Vecchi. *Optimization by Simulated Annealing*. Science, New Series, Vol. 220, No. 4598. (13 de Maio, 1983), pg. 671-680.

Relatório: bem escrito, de modo geral, com alguns erros e faltando alguns dados (resultados GLPK, tempos da expe