Algoritmo Guloso

Wladimir Araujo Tavares [1]

[1]Universidae Federal do Ceará - Campus de Quixadá

16 de agosto de 2018

Argumento de Troca

1. Rotule a solução encontrada pelo seu algoritmo e uma solução ótima. Por exemplo, seja $A = \{a_1, a_2, \ldots, a_k\}$ uma solução gulosa, e seja $O = \{o_1, o_2, \ldots, o_m\}$ uma solução ótima.

Argumento de Troca

1. Rotule a solução encontrada pelo seu algoritmo e uma solução ótima. Por exemplo, seja $A = \{a_1, a_2, \ldots, a_k\}$ uma solução gulosa, e seja $O = \{o_1, o_2, \ldots, o_m\}$ uma solução ótima.
2. Compare a solução gulosa com a solução ótima. Se a solução gulosa é diferente da solução ótima então
   - existe um elemento de O que não está em A e um elemento de A que não está em O;
   - existem dois elementos consecutivos em O que estão em uma ordem diferente da ordem em A, ou seja, existe uma inversão.

Argumento de Troca

1. Rotule a solução encontrada pelo seu algoritmo e uma solução ótima. Por exemplo, seja $A = \{a_1, a_2, \ldots, a_k\}$ uma solução gulosa, e seja $O = \{o_1, o_2, \ldots, o_m\}$ uma solução ótima.
2. Compare a solução gulosa com a solução ótima. Se a solução gulosa é diferente da solução ótima então
   - existe um elemento de O que não está em A e um elemento de A que não está em O;
   - existem dois elementos consecutivos em O que estão em uma ordem diferente da ordem em A, ou seja, existe uma inversão.
3. Troque os elementos em O e argumente que você obtém uma solução modificada que não é pior que a anterior. Então, argumente que se você pode continuar trocando até eliminar todas as diferenças entre O e A em um número finito de passos. Assim, a solução gulosa é tão boa quanto uma solução ótima qualquer.

Árvore Geradora Mínima

- Problema: Dado um grafo $G = (V, E)$ com $custo : e \rightarrow \mathbb{Z}$ encontre uma árvore geradora com o custo mínimo.
- **Algoritmo Guloso:** Ordene as arestas em ordem crescente de pesos e construa uma árvore incluindo as arestas seguindo essa ordem, ou seja, exclua as arestas se elas formam um ciclo com as outras arestas escolhidas previamente.

Árvore Geradora Mínima

- Seja $T = (V, A)$ a árvore geradora produzida pelo algoritmo guloso e $T^* = (V, O)$ uma geradora mínima.

Árvore Geradora Mínima

- Seja $T = (V, A)$ a árvore geradora produzida pelo algoritmo guloso e $T^* = (V, O)$ uma geradora mínima.
- Se T não é mínima então existe uma aresta $e \in O$ tal que $e \notin A$, $T + e$ tem um ciclo C e existe pelo menos uma aresta f cruza o corte definido por $T^* - e$ tal que $f \in A$

Árvore Geradora Mínima

- Seja $T = (V, A)$ a árvore geradora produzida pelo algoritmo guloso e $T^* = (V, O)$ uma geradora mínima.
- Se T não é mínima então existe uma aresta $e \in O$ tal que $e \notin A$, $T + e$ tem um ciclo C e existe pelo menos uma aresta f cruza o corte definido por $T^* - e$ tal que $f \in A$
- A aresta e não foi incluída pelo algoritmo guloso porque $C - e \subseteq A$. Como o algoritmo analisa em ordem crescente de custo, temos que $custo(f) \leq custo(e)$. $T^* - e \cup \{f\}$ é uma árvore gerado tal que $custo(T^* - e \cup \{f\}) = custo(T^*) - custo(e) + custo(f) \leq custo(T^*)$

Seleção de tarefas

- Problema: Dado um conjunto de n tarefas tal que o tempo de início da tarefa i é $s(i)$ e o tempo de final é $f(i)$. Um subconjunto de tarefas é compatível se não existe duas tarefas com sobreposição no tempo. Queremos encontrar o maior subconjunto de tarefas compatível possível.
- **Algoritmo Guloso:** Ordene as tarefas em ordem crescente do tempo de final da tarefa. Escolha as tarefas seguindo a ordem inicial mantendo a compatilidade com as tarefas já escolhidas.

Seleção de tarefas

- Seja um subconjunto $A = \{a_1, a_2, \ldots, a_k\}$ encontrado pelo algoritmo guloso e um subconjunto $O = \{o_1, \ldots, o_s\}$ um conjunto de tarefas ótimo, ambos seguindo a ordem crescente do tempo final.

Seleção de tarefas

- Seja um subconjunto $A = \{a_1, a_2, \ldots, a_k\}$ encontrado pelo algoritmo guloso e um subconjunto $O = \{o_1, \ldots, o_s\}$ um conjunto de tarefas ótimo, ambos seguindo a ordem crescente do tempo final.
- Se A não é ótimo então existe um k tal que $a_k \neq o_k$.

Seleção de tarefas

- Seja um subconjunto $A = \{a_1, a_2, \ldots, a_k\}$ encontrado pelo algoritmo guloso e um subconjunto $O = \{o_1, \ldots, o_s\}$ um conjunto de tarefas ótimo, ambos seguindo a ordem crescente do tempo final.
- Se A não é ótimo então existe um k tal que $a_k \neq o_k$.
- Por hipótese, a_k é compatível com todas as tarefas $[o_1, \ldots, o_{k-1}]$ e $f(a_k) \leq f(o_k)$ e $s(o_{k+1}) > f(o_k) > f(a_k)$. Logo, podemos substituir o_k por a_k. Além disso, $r = s$, caso contrário, o algoritmo guloso estaria deixando de escolher uma tarefa compatível com as já escolhidas.

Escalonamento de tarefas com minimização do tempo de espera

- Problema: Dado um conjunto de n tarefas $J_1, \ldots, J_n$. Cada tarefa J_i tem um tempo de processamento não negativo $p(J_i)$. Existe apenas uma pessoa para processar todas as tarefas. O tempo de espera de uma tarefa J_i é a soma dos tempo de processamento de todas as tarefas agendadas antes da tarefa J_i. O tempo de espera total

	J_1	J_2	J_3	J_4	J_5	J_6
tempo	3	4	1	8	2	6

- O tempo de espera total do escalonamento $(J_1, J_2, J_3, J_4, J_5, J_6)$ é 0 + 3 + (3+4) + (3+4+1) + (3+4+1+8) + (3+4+1+8+2) = 55
- **Algoritmo Guloso:** Programe as tarefas seguindo a ordem crescente de tempo de processamento.

Escalonamento de tarefas

- Seja $O = (J_{o_1}, J_{o_2}, \ldots, J_{o_n})$ uma programação das tarefas ótima e $A = (J_1, J_2, \ldots, J_n)$ um programação de tarefas encontrada pelo algoritmo guloso.

Escalonamento de tarefas

- Seja $O = (J_{o_1}, J_{o_2}, \ldots, J_{o_n})$ uma programação das tarefas ótima e $A = (J_1, J_2, \ldots, J_n)$ um programação de tarefas encontrada pelo algoritmo guloso.
- Se A não é ótimo então existe um par de tarefas consecutivas na solução ótima J_{o_i} e J_{o_j} tal que J_{o_j} foi programada para depois de J_{o_i} e $p(J_{o_j}) \leq p(J_{o_i})$

Escalonamento de tarefas

- Seja $O = (J_{o_1}, J_{o_2}, \ldots, J_{o_n})$ uma programação das tarefas ótima e $A = (J_1, J_2, \ldots, J_n)$ um programação de tarefas encontrada pelo algoritmo guloso.
- Se A não é ótimo então existe um par de tarefas consecutivas na solução ótima J_{o_i} e J_{o_j} tal que J_{o_j} foi programada para depois de J_{o_i} e $p(J_{o_j}) \leq p(J_{o_i})$
- Vamos mostrar que o tempo de espera total no escalonamento $O' = (J_{o_1}, \ldots, J_{o_j}, J_{o_i}, J_{o_{i+2}}, \ldots, J_{o_n})$, não é maior que o tempo de espera total em O. A realização dessa troca afeta apenas o tempo de espera das tarefas J_{o_i} e J_{o_j}. Seja t o tempo de espera da tarefa J_{o_i} no escalonamento O. A soma do tempo de espera de J_{o_i} e J_{o_j} em O é $2t + p(J_{o_i})$ e em O' é $2t + p(J_{o_j})$. Como $p(J_{o_j}) \leq p(J_{o_i})$, então $2t + p(J_{o_j}) \leq 2t + p(J_{o_i})$.

Escalonamento de tarefas com minimização do maior atraso

- Problema: Dado um conjunto de n tarefas $J_1, \ldots, J_n$.
- Cada tarefa J_i tem um tempo de processamento não negativo $p(J_i)$ e um deadline $d(J_i)$.
- Existe apenas uma pessoa para processar todas as tarefas.
- Se uma tarefa começa no tempo $s(J_i)$, o tempo de termino da tarefa é $f(J_i) = s(J_i) + p(J_i)$.
- O atraso de uma tarefa é $l(J_i) = max(0, f(J_i) - d(J_i))$.
- O objetivo é $\min \{\max l(J) | J \in \{J_1, \ldots, J_n\}\}$

Escalonamento de tarefas com minimização do maior atraso

- Problema: Dado um conjunto de n tarefas $J_1, \ldots, J_n$.
- Cada tarefa J_i tem um tempo de processamento não negativo $p(J_i)$ e um deadline $d(J_i)$.
- Existe apenas uma pessoa para processar todas as tarefas.
- Se uma tarefa começa no tempo $s(J_i)$, o tempo de termino da tarefa é $f(J_i) = s(J_i) + p(J_i)$.
- O atraso de uma tarefa é $l(J_i) = max(0, f(J_i) - d(J_i))$.
- O objetivo é $\min \{\max l(J) | J \in \{J_1, \ldots, J_n\}\}$
- **Algoritmo Guloso:** Programe as tarefas seguindo a ordem crescente de tempo de deadline.

Escalonamento de tarefas

- Seja $O = (J_{o_1}, J_{o_2}, \ldots, J_{o_n})$ uma programação das tarefas ótima e $A = (J_1, J_2, \ldots, J_n)$ um programação de tarefas encontrada pelo algoritmo guloso.

Escalonamento de tarefas

- Seja $O = (J_{o_1}, J_{o_2}, \ldots, J_{o_n})$ uma programação das tarefas ótima e $A = (J_1, J_2, \ldots, J_n)$ um programação de tarefas encontrada pelo algoritmo guloso.
- Se A não é ótimo então existe um par de tarefas consecutivas na solução ótima J_{o_i} e J_{o_j} tal que J_{o_j} foi programada para depois de J_{o_i} e $d(J_{o_j}) \leq d(J_{o_i})$

Escalonamento de tarefas

- Seja $O = (J_{o_1}, J_{o_2}, \ldots, J_{o_n})$ uma programação das tarefas ótima e $A = (J_1, J_2, \ldots, J_n)$ um programação de tarefas encontrada pelo algoritmo guloso.
- Se A não é ótimo então existe um par de tarefas consecutivas na solução ótima J_{o_i} e J_{o_j} tal que J_{o_j} foi programada para depois de J_{o_i} e $d(J_{o_j}) \leq d(J_{o_i})$
- Vamos mostrar que o menor maior tempo de atraso em $O' = (J_{o_1}, \ldots, J_{o_j}, J_{o_i}, J_{o_{i+2}}, \ldots, J_{o_n})$, não supera o menor maior de atraso em O. Note que o atraso da tarefa J_{o_j} é sempre menor em O'. Seja t o tempo de espera da tarefa J_{o_i} no escalonamento O. O atraso da tarefa J_{o_i} em O' é

$$t + p(J_{o_i}) + p(J_{o_j}) - d(i) \leq t + p(J_{o_i}) + p(J_{o_j}) - d(j) \qquad (1)$$

Merge de arquivos

- O custo do merge de dois arquivos com tamanho n e m, respectivamente é O(n+m).
- Dado n arquivos, qual é o custo mínimo para realizar o merge de todos os n arquivos.
  $(F_1, F_2, F_3, F_4, F_5) = (20, 30, 10, 5, 30)$

Merge de arquivos

- O custo do merge de dois arquivos com tamanho n e m, respectivamente é O(n+m).
- Dado n arquivos, qual é o custo mínimo para realizar o merge de todos os n arquivos.
  $(F_1, F_2, F_3, F_4, F_5) = (20, 30, 10, 5, 30)$
- Encontre os dois menores arquivos e faça o merge desses dois arquivos.
- As operações realizadas pelo algoritmo pode ser representada por uma árvore e o custo da árvore T será $cost(T) = \sum_{i=1}^{n} depth(F_i) * F_i$

Merge ótimo

- Seja T uma árvore binária ótima para a realização do merge. Seja u um nó interno de T com a maior profundidade. Seja x e y os dois arquivos com os menores tamanho. Se x e y são filhos de u então passe para um outro merge. os dois filhos de u. Caso contrário, a e b são filhos de u. Troque x com a e y com b.

Problema da Mochila Modificada

- Problema: Dado um conjunto de n objetos.
- Cada objeto i tem um valor v_i e um peso w_i.
- Encontre um subconjunto de objetos tal que a soma dos pesos os objetos é menor ou igual W.
- Sabendo que $w_1 \leq w_2 \leq \ldots \leq w_n$ e $v_1 \geq v_2 \geq \ldots \geq v_n$
- Complexidade: $O(n)$.

Reparação do celeiro

- O número total de baias é S.
- As portas de C baias foram perdidas.
- Você vai utilizar placas de madeiras para bloquear as baias com as portas perdidas.
- O fornecedor só pode entregar no máximo M placas de madeiras.
- Dado a lista de baias com a porta quebrada, minimize o comprimento total de placas que ele deve comprar.

Escalonamento de tarefas com tempo unitário com deadline e penalidades

- Um conjunto $S = \{a_1, a_2, \ldots, a_n\}$ de n tarefas com tempo unitário.
- Um conjunto de n inteiros $d_1, \ldots, d_n$ chamado de deadlines tal que cada $1 \leq d_i \leq n$ e a tarefa a_i deve terminar até o tempo d_i.
- Um conjunto de inteiros positivos $w_1, \ldots, w_n$ chamado de penalidades que representando a tarefa a_i incorre na penalidade w_i se ela não termina até o d_i.
- Encontre um escalonamento da tarefas que minimize a penalidade das tarefas atrasadas.

a_i	1	2	3	4	5	6	7
d_i	4	2	4	3	1	4	6
w_i	70	60	50	40	30	20	10

- Encontre um conjunto de tarefas que podem ser resolvidas com a penalidade máxima.