0/1 Knapsack Problem

Universidade Federal do Rio Grande do Sul

Instituto de Informática

INF05515 - Complexidade de Algoritmos - B
Profa. Dra. Mariana Luderitz Kolberg

Turma A:
Jonathas Gabriel Dipp Harb - 193027

Turma B:
Raphael de Leon Ferreira Lupchinski - 191942

{ jgdharb , rlflupchinski }@inf.ufrgs.br

ABSTRACT

This paper aims to describe the Binary Knapsack Problem (0/1 Knapsack Problem) through a formal characterization of the decision problem and the proof that this is an NP-complete Problem.

We will use methods to classify the problem in this class that include a verification of a certificate in polynomial time (proofing that belongs to the NP class) and a reduction of the Subset Sum Problem into the Binary Knapsack Problem in polynomial time (proofing that belongs to the NP-Complete class).

RESUMO

Este artigo tem finalidade de descrever o problema da Mochila Binária (0/1 Knapsack Problem) através de uma caracterização formal de seu problema de decisão e da prova de que se trata de um problema NP-Completo.

Utilizaremos métodos para enquadrar o problema na classe supra-citada que incluem a verificação de um certificado em tempo polinomial (provando a pertinência em NP) e também a redução do Subset Sum Problem para o problema da Mochila Binária em tempo polinomial (provando a pertinência em NP-Completo).

1 INTRODUÇÃO

Um problema é dito ser “tratável” se puder ser resolvido por um algoritmo em tempo polinomial com respeito ao tamanho do problema (Classe P de problemas), caso contrário, é dito ser um problema intratável. Algoritmos de tempo polinomial são considerados “bons” algoritmos, uma vez que retornam uma solução num período de tempo razoável; adicionalmente, um problema não é considerado “bem-resolvido” até que um algoritmo de tempo polinomial seja descoberto para ele.

Neste artigo trataremos especificamente do problema da Mochila Binária (0/ 1 Knapsack Problem - KP), sobre o KP, provaremos que se trata de um problema NP-Completo. Para tanto, é importante saber primeiramente que, segundo [6]:

- A classe dos problemas NP engloba problemas de decisão para os quais são conhecidos algoritmos não-determinísticos polinomiais (problemas de decisão para os quais qualquer certificado pode ser verificado em tempo polinomial para uma resposta “sim” ou “não”);
- Um problema NP é dito ser NP-Completo se, além de pertencer a classe NP, outro problema já provado ser NP-Completo puder ser reduzido(transformado) ao problema em questão em tempo polinomial;
- Um problema de decisão A se transforma (reduz) polinomialmente em outro problema de decisão B se, dada qualquer entrada IA para o problema A, pode-se construir uma entrada IB para o problema B em tempo polinomial no tamanho L(IA) da entrada IA, tal que IA é uma instância “sim” para A se e somente se IB é uma instância “sim” para B.

Feita tal contextualização, fica fácil ver que a prova é baseada nas afirmativas acima. Um maior detalhamento será dado na prova em si. Por fim, uma característica importante na classe dos problemas NP-Completo é a de que se existir um algoritmo (determinístico) polinomial para a resolução de algum problema NP-Completo, então todos os problemas da classe NP também poderão ser resolvidos em tempo polinomial.

2 CARACTERIZAÇÃO DO PROBLEMA

O problema da Mochila Binária pertence a grande classe dos problemas conhecidos como problemas de otimização combinatória, onde tenta-se maximizar ou minimizar alguma “quantidade” satisfazendo-se determinadas condições; no caso do problema da Mochila, deseja-se maximizar o lucro obtido (ou seja, a soma total de custos de uma coleção de itens) sem exceder o peso máximo suportado pela mochila.

O problema da mochila pode ser melhor entendido se considerarmos e compreendermos o seguinte problema: “*Um caroneiro deseja encher sua mochila selecionando determinados objetos dentre um conjunto de objetos, sabe-se que cada objeto tem um determinado peso e gera um determinado lucro. A mochila do caroneiro tem um determinado limite de peso. Como os objetos podem ser selecionados para preencher a mochila de maneira que o lucro gerado seja o máximo respeitando-se o peso máximo da mochila?”*

O problema de otimização diz respeito a achar, dados um conjunto S de objetos e o peso W da mochila, a configuração que gere mais lucro, respeitando o limite de peso, tal problema não possui algoritmo polinomial que o resolva (é intuitivo que devem ser testadas todas combinações possíveis, o que não é feito em tempo polinomial). Por outro lado, o problema de decisão leva em consideração um novo parâmetro V (o lucro desejado) e consiste em decidir se existe ou não uma “solução” com um lucro não menor do que V.

Formalmente, podemos definir o problema de decisão da Mochila Binária da seguinte maneira:
Problema: Mochila Binária
Instância:

- Um conjunto finito de objetos S;
- Um inteiro *n* representando a quantidade de objetos;
- Para cada objeto *i* em {1, … , n}:
  - Um valor $v_i \geq 0$ representando o valor do objeto;
  - Um número $w_i \geq 0$ representando o peso do objeto;
- Um inteiro positivo W representando a capacidade da mochila;
- Um inteiro positivo V representando o lucro desejado.

Questão:

Existe um subconjunto S' de S tal que:

- $\sum_{i \in S'} w_i \leq W$ e $\sum_{i \in S'} v_i \geq V$?

Portanto, o objetivo é decidir (sim ou não) se existe um conjunto de elementos cuja soma dos pesos não ultrapasse o peso máximo suportado pela mochila e a soma de seus lucros (valores) seja maior ou igual do que um determinado lucro determinado inicialmente.

3 PROVA QUE PERTENCE A NP

Para provarmos que o problema da Mochila Binária pertence a classe NP necessitaremos provar, segundo [2], que:

- *"O problema da mochila possui verificação em tempo polinomial, apresentando um algoritmo não-determinista em tempo polinomial ou mostrando que uma dada solução pode ser verificada em tempo polinomial."*

A seguir será apresentado o algoritmo de verificação, que comprovará que o problema pertence à classe NP.

3.1 ALGORITMO DE VERIFICAÇÃO

Para provarmos que o problema da Mochila Binária pertence à classe NP iremos utilizar um algoritmo que verifica uma entrada em tempo polinomial.

```
  NONDET_KNAPSACK( cBP, vBP, n, w[1,...,n], v[1,...,n], x[1,...,n] )
1  if (∑1ton w[i]*x[i] > cBP) or (∑1ton v[i]*x[i] < vBP) then
2      return FAIL
3  else
4      return SUCCESS
5  endif
```

Na linha 4 é feita a verificação de que se uma configuração de seleção de objetos faz com que o peso acumulado extrapole a capacidade da mochila (cBP) ou se o valor da configuração é inferior ao valor estipulado de lucro (vBP), o problema não seja aceito como solução.

Caso contrário a configuração de seleção é condizente e portanto a configuração dada é uma possível solução.

3.2 ANÁLISE DA COMPLEXIDADE

$$cp[1\text{-}5] = (\sum_{1ton} (1)) + 1 = n+1 = O(n)$$

A complexidade é condizente com a esperada uma vez que para verificar se uma configuração da mochila é correta basta verificar se para o conjunto de n objetos, os selecionados satisfazem os valores de capacidade da mochila e valor de lucro estipulado. Portanto, como cp[*NONDET_KNAPSACK*] = O(n), está provado que o problema da Mochila

Binária pertence a classe NP.

4 PROVA QUE PERCENTE A NP-COMPLETO

Para provarmos que o problema da Mochila Binária pertence a classe NP-Completo necessitaremos provar, segundo [2], que:

- O problema da mochila pertence a classe NP, apresentando um algoritmo não-determinista em tempo polinomial ou mostrando que uma dada solução pode ser verificada em tempo polinomial;
- Apresentar um redução em tempo polinomial de um problema já provado ser NP-Completo para o problema da mochila.

Conseguimos mostrar no item anterior que existe um algoritmo capaz de verificar se uma dada configuração de seleção de objetos é sim ou não uma possível solução, resolvendo assim a primeira parte da prova (apresentar um algoritmo que verifique o problema da Mochila Binária em tempo polinomial).

Para a prova de NP-completude, iremos fazer uma redução, em tempo polinomial, do problema já provado ser NP-Completo denominado *SUBSET SUM* (Soma dos Subconjuntos) para o problema da Mochila Binária.

4.1 CARACTERIZAÇÃO DO SUBSET SUM PROBLEM

O problema da Soma dos Subconjuntos consiste em verificar se existe um subconjunto não-vazio de números inteiros de um conjunto *S* cuja soma seja igual a um determinado valor *n*. É sabido, segundo [5], que esse problema é NP-Completo.

O problema da Soma dos Subconjuntos pode ser enunciado da seguinte maneira: *"Dados um conjunto de inteiros T de tamanho n (onde cada elemento de T possui um valor associado ti e um inteiro U , existe um subconjunto não-vazio de T cuja soma vale U?"*. Pode-se perceber que tal problema é um caso especial do problema da Mochila Binária [5].

4.2 REDUÇÃO

Para o Problema da Mochila (INST B) temos um conjunto *S* de *n* itens, cada um associado a um valor v_i e a um peso w_i. A mochila tem capacidade máxima de peso *W* e temos um número *V* (positivo) que representa o lucro (valor) mínimo que deve haver na mochila, ou seja, o problema se dá da seguinte maneira:

Existe um subconjunto *S* em *n* tal que:

$$\sum_{i \in S} w_i \leq W$$

sendo que

$$\sum_{i \in S} v_i \geq V \ ?$$

Para o Subset Sum Problem (INST A) temos um conjunto *T* de tamanho *n*, um tamanho t_i para cada um de seus elementos e um número natural *U*. Queremos saber se existe um subconjunto *T'* em *T* que satisfaça:

$$\sum_{i \in T'} t_i = U$$

Partindo dos problemas de decisão descritos anteriormente, podemos fazer a redução das instâncias do problema A (SUBSET SUM) em instâncias do problema B (0/1

KNAPSACK PROBLEM). Podemos começar reduzindo:

- Os conjuntos: S = T
- Os pesos, valores e tamanhos: $w_i = v_i = t_i$
- Os números delimitadores: W = V = U

Dar uma resposta “sim” para B implica que existe um conjunto S que satisfaz tais regras. Com isso, se tivermos “sim” para o problema B (0/1 Knapsack Problem), teremos uma resposta “sim” para o problema A (Subset Sum).

4.3 ALGORITMO DE REDUÇÃO

```
  REDUÇÃO_KNAPSACK( T.t[1,...,n], n, W, V, U )
1  for i = 1 to n do
2      S.w[i] = S.v[i] = T.t[i]
3  endfor
4  W = V = U
```

O algoritmo faz a atribuição dos valores de tamanho t_i do conjunto T’ tanto para os valores v_i do conjunto S quanto os pesos w_i, também de S. Após isso, é feita a atualização dos limites W e V para o valor U.

4.4 ANÁLISE DA COMPLEXIDADE

$$cp[1\text{-}4] = cp[1\text{-}3] + cp[4]$$
$$cp[1\text{-}3] = \sum_{1 \text{ton}} (1+1) = 2n$$
$$cp[4] = 1$$
$$cp[1\text{-}8] = 2n + 1 = O(n)$$

Portanto, como cp[*REDUÇÃO_KNAPSACK*] = O(n), está provado que o problema da *Mochila Binária* pertence a classe NP-Completo.

CONCLUSÕES

Este artigo apresentou, mesmo que brevemente, o problema da Mochila Binária (0/1 Knapsack Problem). A caracterização do problema e a definição formal do problema de decisão serviram como base para que então fosse apresentada uma prova de que se trata de um problema NP-Completo. Adicionalmente, foi apresentada a definição do problema da Soma dos Subconjuntos (Subset Sum Problem), utilizado na prova da NP-Completude do problema da Mochila Binária.

REFERÊNCIAS

[1] CALDAS, Ruiter Braga. Projeto de Análise de Algoritmos. **Universidade Ferederal de Minas Gerais.** Disponível em <http://homepages.dcc.ufmg.br/~nivio/cursos/pa04/tp2/tp22/tp22.pdf>. Acesso em 29 de Maio de 2012.

[2] ZIVIANE, Nivio. Projeto de Algoritmos: com implementações em Pascal e C. **Pioneira Thomson Learning, 2ed., 2004**.

[3] NP-Complete. Disponível em <http://en.wikipedia.org/wiki/NP-complete>. Acesso em 31 de Maio de 2012.

[4] Knapsack Problem. Disponível em <http://en.wikipedia.org/wiki/Knapsack_problem>. Acesso em 31 de Maio de 2012.

[5] Subset Sum Problem. Disponível em <http://en.wikipedia.org/wiki/Subset_sum_problem>. Acesso em 31 de Maio de 2012.

[6] RIBEIRO, Celso Carneiro. Metaheurísticas e Aplicações. **Departamento de Ciência da Computação (UFF), 2007.** Disponivel em <http://www.ic.uff.br/~celso/disciplinas/metah1.pdf>. Acesso em 1 de Junho de 2012.