PCC104 - Projeto e Análise de Algoritmos

Marco Antonio M. Carvalho

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto

Conteúdo

Fonte

Este material é baseado nos livros

- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.
- S. Halim. *Competitive Programming*. 3rd Edition, 2013.
- Ian Parberry and William Gasarch. *Problems on Algorithms*. Second Edition, 2002.
- Ian Parberry *Lecture Notes on Algorithm Analysis and Complexity Theory*. Fourth Edition, 2001.

Licença

Citação

Cormen, Leiserson, Rivest & Stein - Algorithms. 3rd Edition

"Na prática, quase ótimo muitas vezes é suficientemente bom."

Introdução

Muitos problemas de significado prático são NP-completos, mas, apesar disto, são muito importantes para serem deixados de lado porque não sabemos como determinar uma solução ótima em tempo polinomial.

Podemos explorar possibilidades menos ambiciosas do que resolver o problema eficientemente.

Caso o tamanho da entrada seja pequeno, podemos empregar algoritmos exponenciais, ou ainda, isolar casos especiais que possam ser resolvidos em tempo polinomial.

Introdução

Em alguns casos, algumas abordagens podem determinar soluções "quase ótimas" em tempo polinomial.

Um algoritmo polinomial que retorna soluções aproximadas é chamado de **Algoritmo Aproximado** ou **Algoritmo de Aproximação**.

Algoritmos Aproximados vs. Heurísticas

Heurísticas são algoritmos polinomiais que não possuem nenhuma garantia quanto a qualidade da solução gerada: podem produzir resultados ótimos, soluções próximas do ótimo e também podem falhar miseravelmente.

Normalmente são baseadas em algoritmos gulosos ou adaptações de métodos exatos.

Algoritmos Aproximados são algoritmos polinomiais que geram soluções aproximadas dentro de um limite, ou seja, possuem garantia de qualidade.

Podem também ser baseados em algoritmos gulosos e adaptações de métodos exatos.

Relação de Aproximação

Algoritmos aproximados possuem uma **Relação de Aproximação** para qualquer entrada, que indica que a solução obtida estará dentro de um fator definido do custo da solução ótima.

Suponhamos um problema de otimização em que cada solução potencial tem custo positivo, para o qual desejamos obter uma solução aproximada.

A relação de aproximação $\rho(n)$ é definida em termos do custo C^* de uma solução ótima:

$$max\left\{\frac{C}{C^*}, \frac{C^*}{C}\right\} \leq \rho(n)$$

Para qualquer entrada, de tamanho n, o custo da solução C produzida pelo algoritmo de aproximação está dentro de um fator $\rho(n)$ da solução ótima.

Relação de Aproximação

Observações sobre relações de aproximação:

- A relação de aproximação nunca é menor que 1;
- Uma relação de aproximação 1 é ótima;
- Uma relação maior que 1 pode retornar uma solução muito pior que a solução ótima;
- Estas definições se aplicam tanto a problemas de maximização quanto a problemas de minimização:
  - Para problemas de maximização, $0 < C \leq C^*$ e a razão C^*/C nos fornece o fator pelo qual o valor da solução ótima é maior do que a solução aproximada.
  - Para problemas de minimização, $0 < C^* \leq C$ e a razão C/C^* nos fornece o fator pelo qual o valor da solução aproximada é maior do que a solução ótima.

Relação de Aproximação

Algumas relações de aproximação são constantes, enquanto outras variam de acordo com o tamanho da entrada.

Quando a relação de aproximação é independente de n, a denotamos apenas por ρ.

Para alguns problemas, existe uma relação entre tempo de execução e qualidade da aproximação, sendo possível alcançar relações de aproximação cada vez melhores, mas ao custo de mais tempo de computação.

Relação de Aproximação

Normalmente, não sabemos exatamente o valor da solução ótima. Para efeito da relação de aproximação, utilizamos limitantes no valor da solução ótima.

Através de uma análise do problema, temos uma idéia do valor mínimo ou máximo que uma solução ótima pode atingir – geralmente, a estrutura do problema nos dá alguma dica.

A precisão da aproximação depende da qualidade do limitante utilizado.

Em alguns casos é possível utilizar programação linear para obter limitantes: a solução do PL é polinomial (geralmente, consiste na relaxação da integralidade de variáveis).

Exercício

Quais seriam limitantes superiores para o problema da mochila 0-1?

Introdução

Nem sempre é possível criar aproximações.

Existem diferentes possibilidades em relação a problemas de otimização NP-Difíceis:

- Alguns permitem boas aproximações (e.g., Problema do Caixeiro Viajante com Desigualdade Triangular);
- Outros sequer permitem aproximações (e.g., Clique Máximo).

Esquemas de Aproximação

Um **Esquema de Aproximação** é um algoritmo de aproximação que toma como entrada além da instância para o problema, um valor ϵ>0.

Para qualquer valor de ϵ fixo, o esquema é um algoritmo de aproximação (1+ϵ).

Se o esquema de aproximação é executado em tempo polinomial para qualquer $\epsilon > 0$ fixo, o chamamos de **Esquema de Aproximação de Tempo Polinomial**.

O tempo de execução de um esquema de aproximação em tempo polinomial pode aumentar muito rapidamente à medida em que ϵ diminui.

Por exemplo, o tempo de execução de um esquema de aproximação em tempo polinomial pode ser $O(n^{2/\epsilon})$.

Esquemas de Aproximação

Idealmente, se ϵ diminui por um fator constante, então o tempo de execução não deve aumentar por mais do que um fator constante – embora este fator seja diferente daquele em que ϵ decresce.

Se o tempo de execução é polinomial tanto em $1/\epsilon$ quanto no tamanho da entrada n, o chamamos de **Esquema de Aproximação de Tempo Completamente Polinomial**.

Por exemplo, considerando um tempo de execução $O((1/\epsilon)^2 n^3)$, qualquer diminuição por um fator constante em ϵ possui um aumento por um fator constante no tempo de execução.

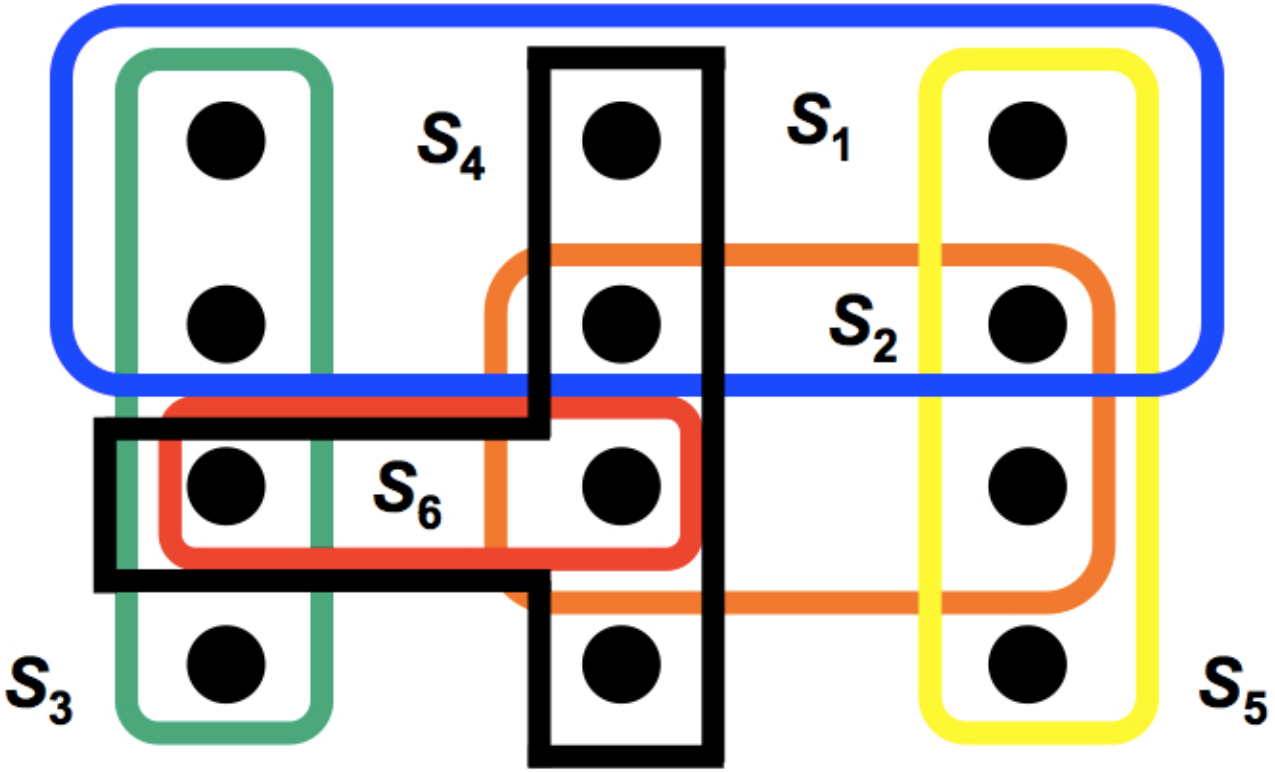
Problema de Cobertura de Conjuntos

Uma instância do problema de cobertura de conjuntos consiste de um conjunto finito X e uma família F de subconjuntos de X, tal que cada elemento de X pertence a pelo menos um subconjunto em F. Há também um custo $custo(S)$ associado a cada subconjunto $S \in F$.

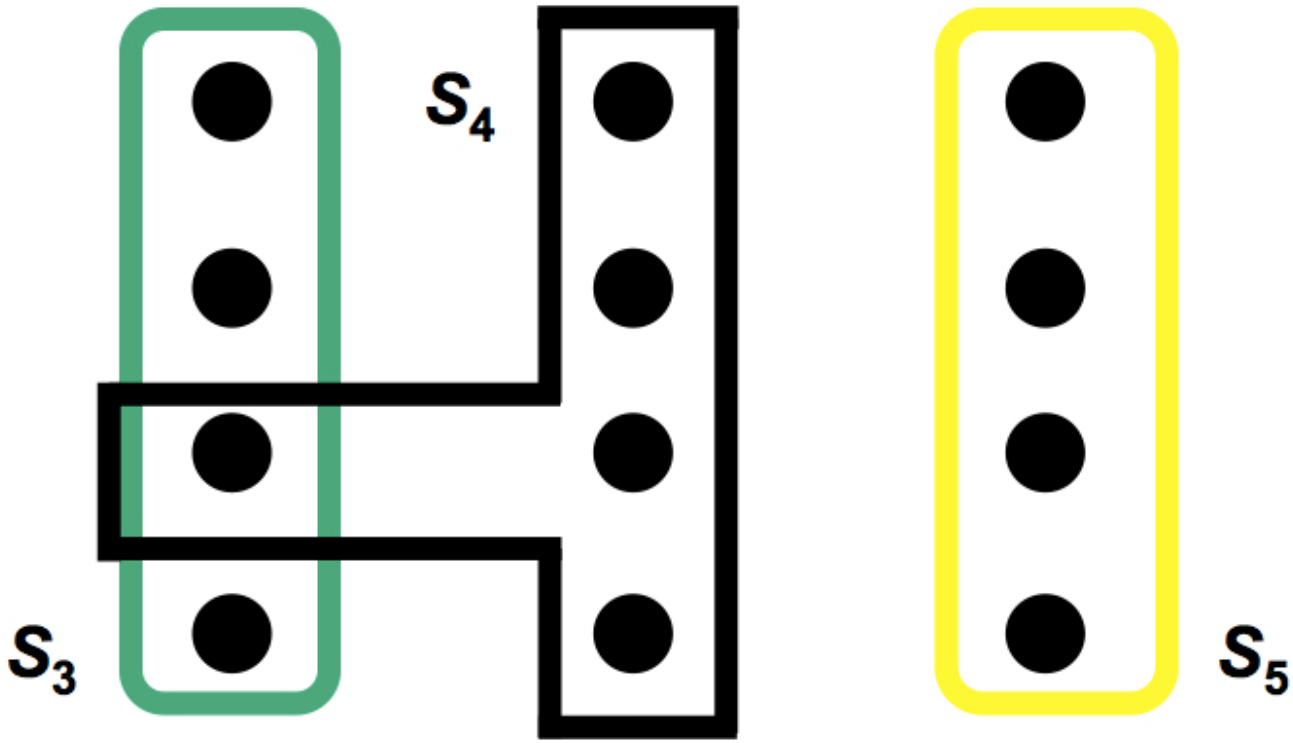
Elementos podem pertencer a mais de um subconjunto. Dizemos que um subconjunto S pertencente a F é uma cobertura de seus próprios elementos.

O objetivo do problema é encontrar o subconjunto $C \subseteq F$ cujos membros sejam uma cobertura de X de menor custo. O tamanho de C é definido pela quantidade de conjuntos que contém.

Este é um problema NP-Difícil, que abstrai muitos problemas combinatórios que surgem comumente.



Instância do Problema de Cobertura de Conjuntos.

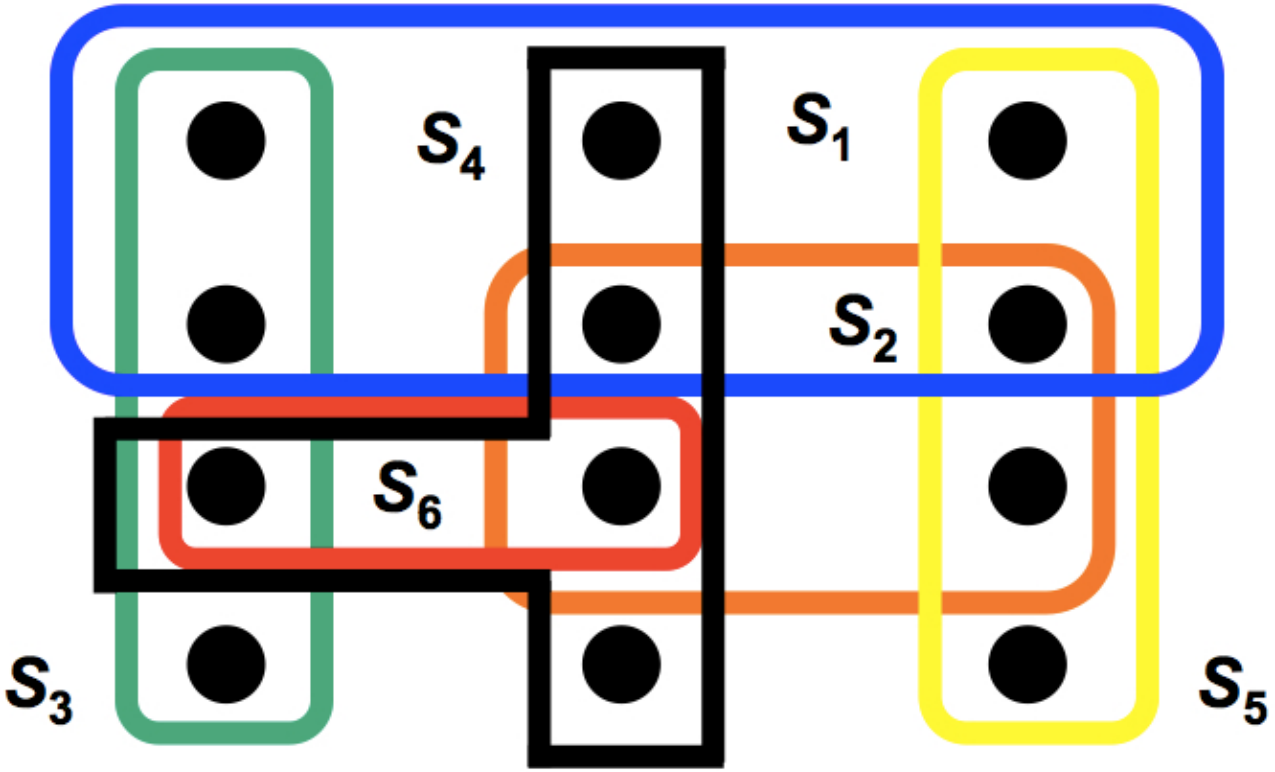


Possível solução, em que C=3 (conjuntos S_3, S_4, S_5).

Algoritmo Guloso 1

"Iterativamente, selecione o conjunto que cubra o maior número de elementos até então não cobertos e os remova do problema, até que todos os elementos estejam cobertos. Resolva empates arbitrariamente."

Este algoritmo guloso pode facilmente ser implementado em tempo $O(|X||F|\min(|X|,|F|))$.



Qual a solução obtida pelo algoritmo guloso 1?

Algoritmo Guloso 2

"Iterativamente, selecione o conjunto que possua a melhor relação custo/benefício e remova do problema os elementos por ele cobertos, até que todos os elementos estejam cobertos

Resolva empates arbitrariamente."

Princípio e Terminologia

Seja C o conjunto de elementos já cobertos no início de uma iteração.

Durante esta iteração, a **relação custo/benefício** de um conjunto S é definida pelo custo médio pelo qual ele cobre elementos ainda não cobertos, ou seja, $\frac{custo(S)}{|S-C|}$.

Definimos o **preço** de um elemento como a média dos custos de sua cobertura.

De maneira equivalente, o custo de S é distribuído igualmente entre os novos elementos cobertos por ele, para definir seus preços.

Problema de Cobertura de Conjuntos

APPROX-SET-COVER(X, F)
Entrada: Conjuntos X e F
$C \leftarrow \emptyset$;
enquanto $C \neq X$ **faça**
    Selecione $S \in F$, o conjunto de melhor relação $\frac{custo(S)}{|S-C|}$;
    para cada $e \in S - C$ **faça**
        preço$(e) \leftarrow \frac{custo(S)}{|S-C|}$;
    fim
    $C \leftarrow C \cup S$;
fim
retorna *conjuntos S selecionados;*

Análise de Complexidade

Este algoritmo guloso pode facilmente ser implementado em tempo $O(|X||F|min(|X|,|F|))$:

- O laço de repetição externo é executado em tempo $min(|X|,|F|)$;
- O conteúdo do laço é executado em tempo $O(|X||F|)$.

Há também uma versão linear deste algoritmo.

Relação de Aproximação

Para a análise da relação de aproximação a seguir, suponhamos que o conjunto X contenha n elementos, enumerados na ordem em que foram cobertos. Empates são resolvidos arbitrariamente.

Suponhamos também que a solução ótima tenha custo OPT.

Lema 1

Para cada elemento e_k, $k \in \{1, \ldots, n\}$, preço$(e_k) \leq$ OPT$/(n - k + 1)$.

Prova

Em qualquer iteração, os conjuntos restantes da solução ótima cobrem os elementos restantes a um custo no máximo OPT.

Portanto, dentre estes conjuntos, deve haver um cuja relação custo/benefício seja no máximo $\frac{\text{OPT}}{|X-C|}$.

Na iteração em que o elemento e_k foi coberto, $X - C$ continha pelo menos $n - k + 1$ elementos.

Uma vez que e_k foi coberto pelo conjunto de maior relação custo/benefício nesta iteração, segue que:

$$\text{preço}(k) \leq \frac{\text{OPT}}{|X - C|} \leq \frac{\text{OPT}}{n - k + 1}$$

Teorema 1

O algoritmo guloso apresentado é uma H_n-aproximação para o problema de cobertura de conjuntos, em que $H_n = 1 + \frac{1}{2} + \ldots + \frac{1}{n}$.

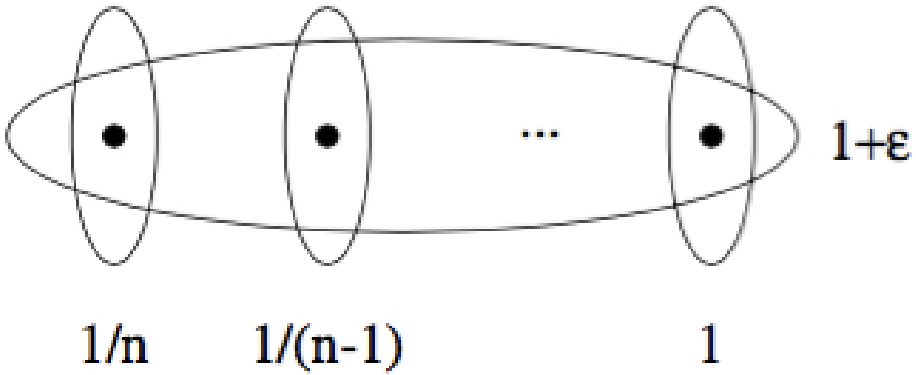
Prova

Uma vez que o custo do conjunto selecionado é dividido entre os elementos por ele cobertos, o custo total da solução é $\sum_{k=1}^{n} \text{preço}(e_k)$.

Pelo Lema 1, este valor é, no máximo, $\left(\frac{1}{n} + \frac{1}{n-1} + \ldots + 1\right) \times \text{OPT}$.

Finalização

Temos que $\sum_{k=1}^{n} \frac{1}{k} \leq ln\ n + 1$. Desta maneira, temos que a solução obtida pelo algoritmo apresentado é limitada por OPT $\times\ ln\ n$, ou seja, é uma aproximação $O(ln\ n)$.



Exemplo de instância com $m = n$ conjuntos unitários, cuja solução tem custo $\frac{1}{n} + \frac{1}{n-1} + \ldots + 1 = H_n$.

Algoritmos Aleatorizados

Em se tratando de algoritmos aproximados, frequentemente lidamos com algoritmos aleatorizados: algoritmos que empregam escolhas arbitrárias em sua lógica.

Como os algoritmos anteriores, estes obedecem uma relação de aproximação.

São semelhantes aos algoritmos de aproximação determinísticos, exceto pelo fato de que a relação de aproximação se refere a um valor esperado.

Também nos referimos a estes algoritmos como **Algoritmos Aleatórios de Aproximação**.

Recapitulando...

Entre os algoritmos aleatórios, existem aqueles que sempre dão a resposta ótima, os chamados de algoritmos **Las Vegas**.

Também existem aqueles que eventualmente podem dar uma resposta que não seja a ótima, mas uma aproximada, os chamados de algoritmos **Monte Carlo**.

SAT

O Problema de Satisfabilidade Booleana (SAT) foi o primeiro problema caracterizado como NP-Completo.

Dada uma expressão lógica com n variáveis booleanas e m conectivos lógicos NOT ($\neg$), AND ($\wedge$) e OR ($\vee$), é necessário determinar se há uma atribuição satisfatória de valores às variáveis, ou seja, que resulte em valor 1 (ou verdadeiro).

3-SAT

O 3-SAT é um caso especial do SAT, em que cada cláusula contém exatamente 3 literais.

Adicionalmente, a expressão está na Forma Normal Conjuntiva (CNF): grupamentos AND de cláusulas, cada uma sendo um OR de um ou mais literais.

Esta variante do SAT também é um problema NP-Completo.

Exemplo 3-SAT

$$E = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

A resposta para esta instância é SIM, pois se atribuírmos o valor 1 (ou verdadeiro) a todas as variáveis teremos resultado 1 (ou verdadeiro).

MAX-3-CNF *Satisfiability*

O 3-SAT é claramente um problema de decisão.

Entretanto, caso uma instância não seja satisfazível, podemos estar interessados em determinar qual atribuição às variáveis satisfaz o maior número de cláusulas individuais.

Este problema de otimização NP-Difícil é conhecido como MAX-3-CNF *Satisfiability* (ou MAX-3SAT).

Exemplo MAX-3-CNF *Satisfiability*

$$E = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (x_1 \vee \neg x_3 \vee x_4)$$

Solução Subótima: $x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1$ (3 cláusulas satisfeitas).
Solução Ótima: $x_1 = x_2 = x_3 = 0, x_4 = 1$ (todas as cláusulas satisfeitas).

MAX-3-CNF *Satisfiability*

APROX-MAX-3SAT(E, n)
Entrada: expressão E, inteiro n
para $i \leftarrow 0$ ***até*** n **faça**
    Gere aleatoriamente um número c no intervalo [0..1);
    se $c \leq 0,49$ **então**
        $x_i \leftarrow 1$;
    senão
        $x_i \leftarrow 0$;
    fim
fim
retorna E;

Análise

Este algoritmo é executado em tempo $O(n)$.

Variável Aleatória Indicadora

Uma variável aleatória X é uma função que mapeia eventos a números reais.

Uma variável aleatória indicadora I é uma variável aleatória que mapeia resultados para os valores do conjunto $\{0, 1\}$.

Variáveis aleatórias indicadoras fornecem um método conveniente para convertermos probabilidades em valores esperados.

Por exemplo, suponha um espaço amostral S e um evento A. Então, a variável aleatória indicadora $I\{A\}$, associada ao evento A é definida como $X_a = I\{A\}$, então $E[A] = P[A]$

MAX-3-CNF *Satisfiability*

O algoritmo aleatorizado atribui o valor 1 para cada variável com probabilidade 1/2.

Consequentemente, também atribui o valor 0 com probabilidade 1/2.

Definimos a variável aleatória indicadora I_k, de forma que I_k=1 desde que pelo menos um literal na k-ésima cláusula tenha sido definido como 1.

Uma cláusula não é satisfeita somente se todos os seus literais forem definidos como 0, logo, P(cláusula k não satisfeita) = $(1/2)^3$ =1/8.

Consequentemente, P(cláusula k satisfeita) = 1-(1/8)=7/8.

O valor esperado da variável aleatória indicadora é então definido como E[I_k]=7/8, ou $\approx 0,88$.

MAX-3-CNF *Satisfiability*

Para que a expressão tenha uma atribuição satisfatória, todas as suas m cláusulas devem ser satisfeitas, logo, temos que o somatório dos indicadores de variáveis aleatórias é $7m/8$.

Um limitante superior para o número de cláusulas satisfeitas é m, logo, $m/(7m/8)$=8/7.

O algoritmo aleatório apresentado é uma aproximação cuja solução esperada é no máximo 8/7 ($\approx$1,142857143) vezes a solução ótima.

Em outras palvras, temos que o erro esperado do algoritmo aproximado é de no máximo $\approx$ 14%.

Dúvidas?