Florence Alyssa Sakuma Shibata, Shayenne da Luz Moura

Método Simplex
Fases I e II

São Paulo

2015

Sumário

Introdução

O método simplex é baseado em encontrar uma solução viável ótima para um problema de programação linear, caso seja viável, e realiza esta busca movendo-se de uma solução viável básica para outra, percorrendo os lados do poliedro que define a região viável, sempre numa direção onde o custo se reduz. Quando uma solução viável básica é alcançada e nenhuma das direções viáveis reduzem o custo temos então a solução ótima, e o algoritmo termina.

Para iniciar o método simplex, é necessário encontrar uma solução básica inicial não degenerada. Para isso, são introduzidas variáveis de folga, não negativas, no problema original, criando assim um problema auxiliar no qual se conhece uma solução viável básica inicial. Ao resolvê-lo, é possível decidir se o problema original é viável, e se for, também encontra uma solução viável básica inicial para ele.

1 Descrição do algoritmo

1.1 Fase I

O método simplex fase I utilizado como base para o desenvolvimento do algoritmo está descrito em Bertsimas e Tsitsiklis (1997, pág. 116-117).

Dadas uma matriz $A \in \Re^{m \times n}$, $b \in \Re^m$, o vetor de custos $c \in \Re^n$, o algoritmo realiza os seguintes passos:

1 Multiplica algumas restrições por -1, mudando o problema para que $b \geq 0$.

2 Introduz variáveis artificiais $y_1, \ldots, y_m$, se necessário, e aplica o método simplex ao problema auxiliar com função de custo $\Sigma_{i=1}^m y_i$.

3 Se o custo ótimo do problema auxiliar é positivo, o problema original é inviável e o algoritmo termina.

4 Se o custo ótimo é 0, uma solução viável para o problema original foi encontrada. Se nenhuma variável artificial está na base final, as variáveis artificiais são eliminadas, e uma base viável para o problema foi encontrada.

5 Se a l-ésima variável básica é artificial, examinamos a l-ésima entrada das colunas de $B{-}1A_j$, $j = 1, \ldots, n$. Se todas as entradas são zero, a l-ésima linha representa uma restrição redundante e é eliminada. Caso contrário, se a l-ésima entrada da j-ésima coluna é diferente de zero, aplica a mudança de base (com esta entrando e servindo de elemento pivô): a l-ésima variável básica sai e x_j entra na base. Repete essa operação até que todas as variáveis artificiais sejam tiradas da base.

1.2 Fase II

O método simplex fase II utilizado como base para o desenvolvimento do algoritmo está descrito em Bertsimas e Tsitsiklis (1997, pág. 90-91).

Dadas uma matriz $A \in \Re^{m \times n}$, uma solução viável básica $x \in \Re^n$, $b \in \Re^m$, o vetor de custos $c \in \Re^n$, o algoritmo realiza os seguintes passos:

1 Gera a matriz básica B associada a x.

2 Calcula o vetor de custos reduzidos para toda variável não básica. Se nenhuma componente é negativa, então a solução viável básica atual é ótima, acabou.

3 Caso contrário, armazena o índice da variável cujo custo reduzido foi menor. Calcula $u = B^{-1}A_j$. Se nenhum componente de u é positivo, então o custo ótimo é $-\infty$, acabou;

4 Caso contrário, toma $\theta^* = min_{(i=1,\ldots,m|u_i>0)}\{\frac{x_{B(i)}}{u_i}\}$. Seja l o índice onde o mínimo foi encontrado. Forma uma nova base substituindo $A_{B(l)}$ por A_j. Sendo y é a nova solução viável básica, os valores das novas variáveis básicas são $y_j = \theta^*$, $y_{B(i)} - \theta^* u_i$, $i \neq l$. Volta ao passo 1.

É necessário que o problema possua pelo menos uma solução viável básica e que todas as soluções viáveis básicas sejam não degeneradas.

2 Funcionamento do algoritmo

A seguir está a descrição do funcionamento do algoritmo quando o problema de programação linear dado é inviável, possui pelo menos uma solução ótima e quando o custo ótimo é ilimitado. Também é apresentado um exemplo de problema no qual o algoritmo falha por tentar o cálculo da fase 2 com uma solução viável básica degenerada, encontrada pela fase 1.

2.1 Problema inviável

O objetivo da fase 1 é decidir se existe ou não solução viável, e se existir, devolve a mesma para a fase 2.

O problema a seguir não possui região viável. Ao calcular o problema auxiliar, o algoritmo encontra uma solução ótima maior que zero. Assim, temos que o problema de programação linear original é inviável.

```
[ind, x, d] = simplex(A, b, c, m, n)
#====================================
          Simplex Fase 1
#====================================
           Iteracao: 0
#====================================
Variaveis basicas:
5:  12.000000
6:  20.000000

Valor funcao objetivo: 32.000000

Custos reduzidos:
1: -2.000000

Entra na base: 1

Direcao
5: 1.000000
6: 1.000000
```

```
Theta*: 12.000000

Sai da base: 5

#=====================================
           Iteracao: 1
#=====================================
Variaveis basicas:
1:  12.000000
6:  8.000000

Valor funcao objetivo: 8.000000

Custos reduzidos:
5: 2.000000
2: 0.000000
3: 1.000000
4: 1.000000
O problema é inviável
```

2.2 Problema com solução ótima

Fornecendo a seguinte entrada, temos abaixo descrito o resultado do algoritmo:

```
A = [1 2 3 0; -1 2 6 0; 0 4 9 0; 0 0 3 1]
b = [3;2;5;1]
c = [1;1;1;0]
m = 4
n = 4
```

Na fase 1, O algoritmo irá acrescentar as variáveis artificiais e resolver o problema auxiliar, afim de decidir se o problema é inviável ou encontrar uma solução viável básica para o problema original.

```
> [ind, x, d] = simplex(A, b, c, m, n)
#=====================================
          Simplex Fase 1
#=====================================
           Iteracao: 0
```

```
#====================================
Variaveis basicas:
5:  3.000000
6:  2.000000
7:  5.000000
8:  1.000000

Valor funcao objetivo: 11.000000

Custos reduzidos:
1: 0.000000
2: -8.000000

Entra na base: 2

Direcao
5: 2.000000
6: 2.000000
7: 4.000000
8: 0.000000

Theta*: 1.000000

Sai da base: 6

#====================================
           Iteracao: 1
#====================================
Variaveis basicas:
5:  1.000000
2:  1.000000
7:  1.000000
8:  1.000000

Valor funcao objetivo: 3.000000

Custos reduzidos:
1: -4.000000
```

```
Entra na base: 1

Direcao
5: 2.000000
2: -0.500000
7: 2.000000
8: 0.000000

Theta*: 0.500000

Sai da base: 5

#=====================================
           Iteracao: 2
#=====================================
Variaveis basicas:
1:  0.500000
2:  1.250000
7:  0.000000
8:  1.000000

Valor funcao objetivo: 1.000000

Custos reduzidos:
5: 2.000000
6: 2.000000
3: -3.000000

Entra na base: 3

Direcao
1: -1.500000
2: 2.250000
7: 0.000000
8: 3.000000

Theta*: 0.333333

Sai da base: 8
```

```
#====================================
           Iteracao: 3
#====================================
Variaveis basicas:
1:  1.000000
2:  0.500000
7:  0.000000
3:  0.333333

Valor funcao objetivo: 0.000000

Custos reduzidos:
5: 2.000000
6: 2.000000
8: 1.000000
4: 0.000000
```

Nesse momento, todos os custos reduzidos são não negativos, logo foi encontrada uma solução ótima para o problema auxiliar.

O algoritmo verifica que o custo ótimo desta solução é igual a zero. Isso quer dizer que o problema original é viável e na solução encontrada, podemos encontrar uma solução viável báśica para o original.

Como ainda sobraram variáveis artificiais na base da solução temos que as restrições associadas a essas váriáveis são redundantes no problema original. Logo, são removidas e assim é possível resolver o problema original com a solução encontrada.

A partir de agora, inicia-se a fase 2.

```
#====================================
          Simplex Fase 2
#====================================
           Iteracao: 0
#====================================
Variaveis basicas:
1:  1.000000
2:  0.500000
3:  0.333333

Valor funcao objetivo: 1.833333
```

```
Custos reduzidos:
4: -0.083333

Entra na base: 4

Direcao
1: 0.500000
2: -0.750000
3: 0.333333

Theta*: 1.000000

Sai da base: 3

#=====================================
           Iteracao: 1
#=====================================
Variaveis basicas:
1:  0.500000
2:  1.250000
4:  1.000000

Valor funcao objetivo: 1.750000

Custos reduzidos:
3: 0.250000

Solucao e otima
ind = 0
x =

   0.50000
   1.25000
   0.00000
   1.00000

d = 0
```

A solução devolvida em x é a solução ótima para o problema original.

2.3 Problema com custo ótimo ilimitado

Como exemplo do funcionamento do algoritmo para um problema de programação linear que contém solução ilimitada segue os dados de entrada.

```
> A = [1 -1 1 0; 2 -1 0 1];
> b = [10; 40]
 c = [-2; -1; 0; 0];
 m = 2;
 n = 4;
```

A execução do algoritmo devolve as iterações a seguir.

```
> [ind, x, d] = simplex(A, b, c, m, n)
#=====================================
          Simplex Fase 1
#=====================================
           Iteracao: 0
#=====================================
Variaveis basicas:
5:  10.000000
6:  40.000000

Valor funcao objetivo: 50.000000

Custos reduzidos:
1: -3.000000

Entra na base: 1

Direcao
5: 1.000000
6: 2.000000

Theta*: 10.000000

Sai da base: 5
```

```
#=====================================
           Iteracao: 1
#=====================================
Variaveis basicas:
1:  10.000000
6:  20.000000

Valor funcao objetivo: 20.000000

Custos reduzidos:
5: 3.000000
2: -1.000000

Entra na base: 2

Direcao
1: -1.000000
6: 1.000000

Theta*: 20.000000

Sai da base: 6

#=====================================
           Iteracao: 2
#=====================================
Variaveis basicas:
1:  30.000000
2:  20.000000

Valor funcao objetivo: 0.000000

Custos reduzidos:
5: 1.000000
6: 1.000000
3: 0.000000
4: 0.000000
#=====================================
```

```
          Simplex Fase 2
#=====================================
           Iteracao: 0
#=====================================
Variaveis basicas:
1:  30.000000
2:  20.000000

Valor funcao objetivo: -80.000000

Custos reduzidos:
3: -4.000000

O valor da funcao objetivo vai para -Inf

Solucao possui custo -Inf
ind = -1
x = 0
d =

  -1
  -2
   0
   0
```

Em d, é devolvida a direção viável para a qual o custo é ilimitado.

2.4 Solução inicial degenerada

Caso a fase 1 encontre uma solução viável básica degenerada, o problema original não pode ser resolvido por meio do simplex, uma vez que é necessário fornecer uma solução viável básica não degenerada para prosseguir o método simplex na fase 2.

Abaixo está transcrito o resultado do funcionamento do algoritmo e do erro ocasionado ao tentar iniciar o simplex fase 2 com uma solução viável básica degenerada.

A seguinte entrada:

```
> A = [4 4 1 0 0 0; 2 0 0 1 0 0;-1 3 0 0 1 0;0 1 0 0 0 1]
> b = [28;10;9;4]
> c = [0;-1;0;0;0;0]
```

```
> m=4
> n=6
```

Tem o seguinte resultado:

```
> [ind, x, d] = simplex(A, b, c, m, n)
#=====================================
          Simplex Fase 1
#=====================================
           Iteracao: 0
#=====================================
Variaveis basicas:
7:  28.000000
8:  10.000000
9:  9.000000
10:  4.000000

Valor funcao objetivo: 51.000000

Custos reduzidos:
1: -5.000000

Entra na base: 1

Direcao
7: 4.000000
8: 2.000000
9: -1.000000
10: 0.000000

Theta*: 5.000000

Sai da base: 8

#=====================================
           Iteracao: 1
#=====================================
Variaveis basicas:
7:  8.000000
1:  5.000000
```

```
9:  14.000000
10:  4.000000

Valor funcao objetivo: 26.000000

Custos reduzidos:
8: 2.500000
2: -8.000000

Entra na base: 2

Direcao
7: 4.000000
1: 0.000000
9: 3.000000
10: 1.000000

Theta*: 2.000000

Sai da base: 7

#====================================
           Iteracao: 2
#====================================
Variaveis basicas:
2:  2.000000
1:  5.000000
9:  8.000000
10:  2.000000

Valor funcao objetivo: 10.000000

Custos reduzidos:
8: -1.500000

Entra na base: 8

Direcao
2: -0.500000
```

```
1: 0.500000
9: 2.000000
10: 0.500000

Theta*: 4.000000

Sai da base: 9

#=====================================
           Iteracao: 3
#=====================================
Variaveis basicas:
2:  4.000000
1:  3.000000
8:  4.000000
10:  0.000000

Valor funcao objetivo: 4.000000

Custos reduzidos:
9: 0.750000
7: 1.437500
3: 0.437500
4: -1.000000

Entra na base: 4

Direcao
2: 0.000000
1: 0.000000
8: 1.000000
10: 0.000000

Theta*: 4.000000

Sai da base: 8

#=====================================
           Iteracao: 4
```

```
#=====================================
Variaveis basicas:
2:  4.000000
1:  3.000000
4:  4.000000
10:  0.000000

Valor funcao objetivo: 0.000000

Custos reduzidos:
9: 1.250000
7: 1.062500
3: 0.062500
8: 1.000000
5: 0.250000
6: -1.000000

Entra na base: 6

Direcao
2: 0.000000
1: 0.000000
4: 0.000000
10: 1.000000

Theta*: 0.000000

Sai da base: 10

#=====================================
           Iteracao: 5
#=====================================
Variaveis basicas:
2:  4.000000
1:  3.000000
4:  4.000000
6:  0.000000

Valor funcao objetivo: 0.000000
```

```
Custos reduzidos:
9: 1.000000
7: 1.000000
3: 0.000000
8: 1.000000
5: 0.000000
10: 1.000000
#=====================================
          Simplex Fase 2
error: inverse: argument must be a square matrix
error: called from:
error:   matrizB at line 172, column 8
error:   subsimplex at line 108, column 15
error:   simplex at line 85, column 13
```

Sendo a solução viável básica fornecida degenerada, a matriz básica B associada a solução não é quadrada, não sendo possível calcular sua inversa. Era esperado que o programa devolvesse erro porque o código do simplex foi reaproveitado do EP2 e este supõe que a solução viável básica inicial é não degenerada.

2.5 Problema com soluções viáveis básicas degeneradas

```
> A = [1/4 -8 -1 9 1 0 0 ; 1/2 -12 -1/2 3 0 1 0; 0 0 1 0 0 0 1]
> b = [1;2;3]
> c = [0;0;0;0;0;0;1]
> m = 3
> n = 7

> [ind, x, d] = simplex(A, b, c, m, n)
#=====================================
          Simplex Fase 1
#=====================================
           Iteracao: 0
#=====================================
Variaveis basicas:
8:  1.000000
9:  2.000000
10:  3.000000
```

```
Valor funcao objetivo: 6.000000

Custos reduzidos:
1: -0.750000

Entra na base: 1

Direcao
8: 0.250000
9: 0.500000
10: 0.000000

Theta*: 4.000000

Sai da base: 8

#=====================================
           Iteracao: 1
#=====================================
Variaveis basicas:
1:  4.000000
9:  0.000000
10:  3.000000

Valor funcao objetivo: 3.000000

Custos reduzidos:
8: 3.000000
2: -4.000000

Entra na base: 2

Direcao
1: -32.000000
9: 4.000000
10: 0.000000

Theta*: 0.000000
```

```
Sai da base: 9

#=====================================
           Iteracao: 2
#=====================================
Variaveis basicas:
1:  4.000000
2:  0.000000
10:  3.000000

Valor funcao objetivo: 3.000000

Custos reduzidos:
8: 1.000000
9: 1.000000
3: -1.000000

Entra na base: 3

Direcao
1: 8.000000
2: 0.375000
10: 1.000000

Theta*: 0.000000

Sai da base: 2

#=====================================
           Iteracao: 3
#=====================================
Variaveis basicas:
1:  4.000000
3:  0.000000
10:  3.000000

Valor funcao objetivo: 3.000000
```

```
Custos reduzidos:
8: -0.333333

Entra na base: 8

Direcao
1: -1.333333
3: -1.333333
10: 1.333333

Theta*: 2.250000

Sai da base: 10

#=====================================
           Iteracao: 4
#=====================================
Variaveis basicas:
1:  7.000000
3:  3.000000
8:  2.250000

Valor funcao objetivo: 2.250000

Custos reduzidos:
10: 0.250000
9: 1.500000
2: 2.000000
4: -7.500000

Entra na base: 4

Direcao
1: 6.000000
3: 0.000000
8: 7.500000

Theta*: 0.300000
```

```
Sai da base: 8

#=====================================
           Iteracao: 5
#=====================================
Variaveis basicas:
1:  5.200000
3:  3.000000
4:  0.300000

Valor funcao objetivo: 0.000000

Custos reduzidos:
10: 1.000000
9: 1.000000
2: 0.000000
8: 1.000000
5: 0.000000
6: 0.000000
7: 0.000000
#=====================================
          Simplex Fase 2
#=====================================
           Iteracao: 0
#=====================================
Variaveis basicas:
1:  5.200000
3:  3.000000
4:  0.300000

Valor funcao objetivo: 0.000000

Custos reduzidos:
2: 0.000000
5: 0.000000
6: 0.000000
7: 1.000000

Solucao e otima
```

```
ind = 0
x =

   5.20000
   0.00000
   3.00000
   0.30000
   0.00000
   0.00000
   0.00000

d = 0
```

Conclusão

O algoritmo simplex implementado para resolver problemas de programação linear com soluções viáveis básicas não degeneradas e com pelo menos uma ótima é correto.

Mantendo as hipóteses em todos os casos, obtêm-se os resultados esperados em problemas cuja solução ótima é dada, quando o custo ótimo é ilimitado, ou seja necessário encontrar uma direção viável cujo custo seja menor, encontrando a solução ótima.

Referências

BERTSIMAS, D.; TSITSIKLIS, J. *Introduction to Linear Optimization*. [S.l.]: Athena Scientific, 1997. (Athena Scientific series in optimization and neural computation). ISBN 9781886529199. Citado na página 3.