MAC315 - EP2: Implementação da Fase 2 do Método Simplex

Guilherme Schützer - NUSP 8658544
Tomás Paim - NUSP 7157602

17/05/2015

1 Instruções de execução

2 Comentários Iniciais

Para realizar este exercício, passamos a tratar todos os vetores que vimos em aula como vetores colunas como vetores linha para que o Octave realizasse as operações com naturalidade. Por este motivo, sempre que mencionarmos, por exemplo, $c'x$, estamos implementando cx'.

3 Algoritmo ingênuo

A partir dos argumentos da função `simplex`, começamos a função copiando a matriz A para a B e em seguida varremos o vetor x de trás para frente, para evitar que ao eliminar colunas de B acabemos acessando a posição errada na próxima iteração desse loop, e, para cada elemento i, adicionamos ao vetor `bind` seu índice caso o valor de x_i seja diferente de zero (o que implica que aquela é uma variável básica), e ao mesmo tempo construímos o vetor `cB`, que representa os custos associados às variáveis básicas. Caso contrário, eliminamos a coluna i da matriz B, pois a variável i é uma das variáveis não-básicas, já que não há soluções básicas degeneradas. Em seguida, invertemos os vetores `bind` e `cB` para que eles estejam de acordo com x.

No próximo passo do algoritmo, vamos chamar a função `simplex_rec`, que é recursiva e recebe como parâmetro a matriz B, o vetor `bind`, o vetor c_B e o número da iteração atual, além das variáveis que foram passadas para a função `simplex`. Dentro da recursão, primeiro devemos calcular os custos reduzidos associados a cada uma das variáveis não-básicas. Para isso, usaremos a fórmula $\bar{c}_j = c_j - c'_B B^{-1} A_j$, e para que isso seja possível, usaremos a decomposição LU da matriz B, tomando muito cuidado com as transposições. Assim que encontrarmos pela primeira vez um custo reduzido negativo, tomaremos a variável correspondente a esse custo como l, que será a variável que entrará na base. Com l já definida, calculamos o vetor $u = -d_B$ e calcularemos θ^* que corresponde ao máximo que podemos andar na direção d sem violar as restrições, através do método visto em aula que consiste em verificar se $u_i > 0$ e $x_{B(i)}/u_i < \theta^*$, com

θ^* começando em ∞, para todo $i \in \{B(1), B(2), ..., B(m)\}$. Sempre que isso acontecer, também atualizaremos a variável `imin` que indica o índice da variável que violará as restrições primeiro, que será a variável a deixar a base.

Uma vez terminado esse processo, terminamos de calcular os $\bar{c}_j$ restantes para que possamos imprimi-los.

Em seguida, imprimimos as variáveis básicas, valor da função objetivo e os custos reduzidos associados às variáveis não básicas relativos a essa iteração da função. Então verificamos se o programa chegou ao fim, isto é, se não há mais nenhuma direção associada a uma variável não-básica na qual o custo diminui, o que significa que encontramos o custo ótimo da função, ou se $\theta^* = \infty$, o que significa que ainda há uma direção na qual o custo diminúi porém podemos "andar" infinitamente nesta direção, o que implica que o custo ótimo é $-\infty$.

Caso não tenhamos chegado no final da recursão continuaremos a imprimir os dados relativos à iteração atual, que são a variável que entra na base, as coordenadas básicas da direção d relativa à variável l, θ^* e a variável que sai da base. Então atualizamos as variáveis que serão mandadas para o próximo passo da recursão: atualizamos o vetor x com o quando "andamos" na direção d (ou seja, para cada variável básica i, fazemos $x_{B(i)} = x_{B(i)} + \theta^* d_{(B(i))}$) e mudamos x_l para θ^*), atualizamos também o vetor `bind` com a variável que entra na base substituindo a variável que sai, substituimos a coluna associada à variável que saiu da base em B por A_j, substituimos a componente associada à variável que saiu por $c[l]$ em c_B e atualizamos o contador de iterações.

Por fim, chamamos a função recursiva novamente, com os dados atualizados.