Método Simplex

Alunos: Vitor, Renan e André, os guerreiros da

Introdução

- Criado por George Bernard Dantzig (1947)
- Um dos métodos mais utilizados para otimização de problemas

Introdução

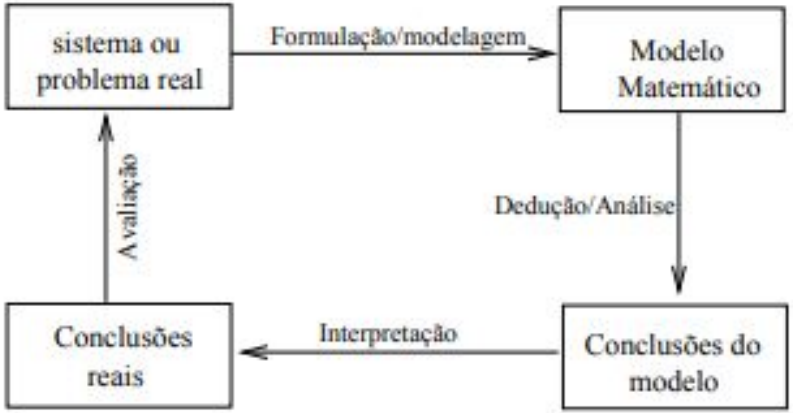
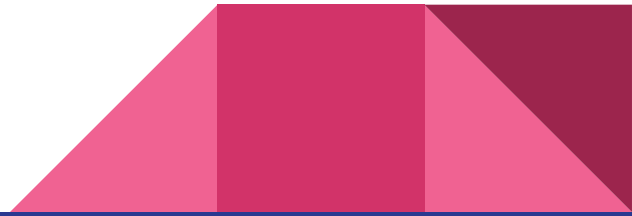


Figura 1.1: Processo de modelagem matemática
(Arenales, Armentano, Morabito e Yanasse 2007)

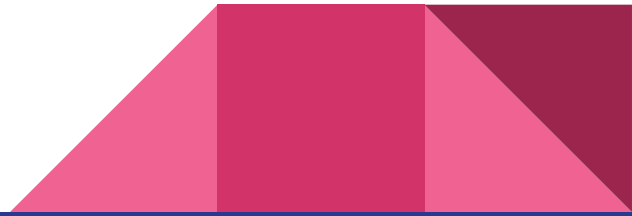
Desenvolvimento

- Utilização do que aprendemos na disciplina
- Notação Matricial



Desenvolvimento

- getInfoFromCplexFile():
  - Abre o arquivo CPLEX "teste.cplex.lp"
  - Obtém nomes de variáveis, número de variáveis, número de restrições, coeficientes, etc.
  - Configura matrizes e arrays para os dados do problema
  - Determina se o problema é de maximização ou minimização
  - Retorna os dados relevantes



Desenvolvimento

- Utilização da biblioteca cplex para retirar as informações do arquivo de entrada
- Variáveis retiradas do arquivo .cplex.lp:
  - matrizNaoBasica
  - matrizBasica
  - vetorB (vetor resultado)
  - vetorcn (custo nao-basicas)
  - vetorcb (custo basicas)
  - maxOrMin
  - nomeVariaveis
  - precisaFaseUm
  - variaveisArtificiais (número de variáveis artificiais)
  - variaveisBasicas, variaveisNaoBasicas (posição original delas)

Desenvolvimento

- simplex (matrizNaoBasica, matrizBasica, vetor_b, custo_n, custo_b, maxOrMin, nomeVariaveis, variaveisBasicas, variaveisNaoBasicas)
  - caso seja max, todo elemento custo_n recebe custo_n[i] = -custo_n[i]
  - solucaoOtima = falso
  - iteracao = 1
  - enquanto !solucaoOtima
    - inverte matriz basica **<-**
    - calcula vetor xb (multiplicando matriz basica invertida por vetor_b)
    - calcula vetor multiplicador(custo_b pela matriz invertida)
    - calcula vetor de custos relativos (cada elemento do vetor ĉn = ci - vetorMult * coluna indice)
    - pega o indice do menor custo relativo para entrar na base

Desenvolvimento

- inverterMatriz(matrizOriginal)
  - numero de linhas = len(matrizOriginal)
  - identidade = gerarMatrizIdentidade(numero de linhas)
  - matrizExtendida = gerarMatrizExtendida(matrizOriginal, identidade)
  - método de gauss-jordan
    - com a matriz extendida, faz o pivoteamento completo para deixar a identidade do lado esquerdo
    - após o pivoteamento, o que tem a direita da matriz é a matriz inversa
  - return matrizInvertida

Desenvolvimento

- simplex (matrizNaoBasica, matrizBasica, vetor_b, custo_n, custo_b, maxOrMin, nomeVariaveis, variaveisBasicas, variaveisNaoBasicas)
  - caso seja max, todo elemento custo_n recebe custo_n[i] = -custo_n[i]
  - solucaoOtima = falso
  - iteracao = 1
  - enquanto !solucaoOtima
    - inverte matriz basica
    - calcula vetor xb (multiplicando matriz basica invertida por vetor_b) **<-**
    - calcula vetor multiplicador(custo_b pela matriz invertida)
    - calcula vetor de custos relativos (cada elemento do vetor ĉn = ci - vetorMult * coluna indice)
    - pega o indice do menor custo relativo para entrar na base

Desenvolvimento

- multiplicaMatrizPorVetor(matriz, vetor)
  - vetorResultante[] = 0 (tamanho da matriz)
  - for i in range (tamanho da matriz)
    - for j in range (tamanho do vetor)
      - vetorResultante[i] += matriz[i][j] * vetor[j]
  - return vetorResultante

$$\hat{x}_B \leftarrow B^{-1}b$$

Desenvolvimento

- simplex (matrizNaoBasica, matrizBasica, vetor_b, custo_n, custo_b, maxOrMin, nomeVariaveis, variaveisBasicas, variaveisNaoBasicas)
  - caso seja max, todo elemento custo_n recebe custo_n[i] = -custo_n[i]
  - solucaoOtima = falso
  - iteracao = 1
  - enquanto !solucaoOtima
    - inverte matriz basica
    - calcula vetor xb (multiplicando matriz basica invertida por vetor_b)
    - calcula vetor multiplicador(custo_b pela matriz invertida) **<-**
    - calcula vetor de custos relativos (cada elemento do vetor ĉn = ci - vetorMult * coluna indice)
    - pega o indice do menor custo relativo para entrar na base

Desenvolvimento

- calculaVetorMultiplicadorSimplex(vetorCustoB, matrizInvertida)
  - vetorMultiplicador[] = [0.0] * tamanho vetor custo B
  - for i in range (tamanho vetor B)
    - for j in range (tamanho vetor B)
      - vetorMultiplicador[i] = vetorCustoB[j] * matrizInvertida[j][i]
  - return vetorMultiplicador

$$\lambda^T \leftarrow c_B^T B^{-1}$$

Desenvolvimento

- simplex (matrizNaoBasica, matrizBasica, vetor_b, custo_n, custo_b, maxOrMin, nomeVariaveis, variaveisBasicas, variaveisNaoBasicas)
  - caso seja max, todo elemento custo_n recebe custo_n[i] = -custo_n[i]
  - solucaoOtima = falso
  - iteracao = 1
  - enquanto !solucaoOtima
    - inverte matriz basica
    - calcula vetor xb (multiplicando matriz basica invertida por vetor_b)
    - calcula vetor multiplicador(custo_b pela matriz invertida)
    - calcula vetor de custos relativos (cada elemento do vetor ĉn = ci - vetorMult * coluna indice) **<-**
    - pega o indice do menor custo relativo para entrar na base

Desenvolvimento

- calculaCustosRelativos(vetorMultiplicador, matrizNaoBasica, vetorCustoN)
  - nVariaveis = tamanho do vetorCustoN
  - nRestricoes = tamanho da matrizNaoBasica (quantas linhas)
  - custosRelativos[] = [0.0] * nVariaveis
  - for i in range nVariaveis
    - coluna i = primeira coluna da matriz nao basica
    - custosRelativos[i] = vetorCustoN[i] - (vetorMultiplicador[i] * coluna i [i])
  - return custosRelativos

$$\hat{c}_{N_j} \leftarrow c_{N_j} - \lambda^T a_{N_j} \quad j = 1, 2, \ldots, n - m$$

Desenvolvimento

- simplex (matrizNaoBasica, matrizBasica, vetor_b, custo_n, custo_b, maxOrMin, nomeVariaveis, variaveisBasicas, variaveisNaoBasicas)
  - caso seja max, todo elemento custo_n recebe custo_n[i] = -custo_n[i]
  - solucaoOtima = falso
  - iteracao = 1
  - enquanto !solucaoOtima
    - inverte matriz basica
    - calcula vetor xb (multiplicando matriz basica invertida por vetor_b)
    - calcula vetor multiplicador(custo_b pela matriz invertida)
    - calcula vetor de custos relativos (cada elemento do vetor ĉn = ci - vetorMult * coluna indice)
    - pega o indice do menor custo relativo para entrar na base **<-**

Desenvolvimento

- indiceDoValorMinimo(vetorCustosRelativos)
  - indiceMinimo = vetor.index(min(vetor))
  - return indiceMinimo

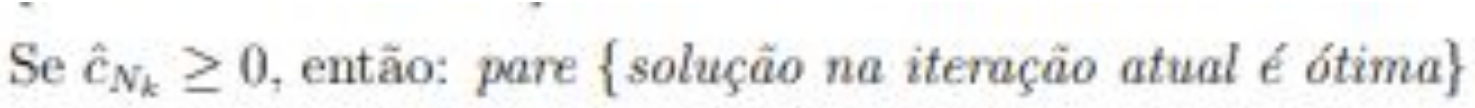
$$\hat{c}_{N_k} \leftarrow \min. \left\{ \hat{c}_{N_j}, j = 1, 2, ..., n - m \right\}$$

Desenvolvimento

- simplex (matrizNaoBasica, matrizBasica, vetor_b, custo_n, custo_b, maxOrMin, nomeVariaveis, variaveisBasicas, variaveisNaoBasicas)
  - cont. enquanto !solucaoOtima
    - check se solução é ótima pelo vetor de custos relativos (todos maior que 0) **<-**
    - caso sim, a solução é ótima, printa os resultados **<-**
    - caso nao, continua **<-**
    - calcula a direção simplex (y = B^{-1} * a[menorIndice])
    - calcula o theta, escolhe o menor valor positivo para sair da base
    - troca de posição as colunas

Desenvolvimento

- checarSolucaoOtima(custosRelativos)
  - for i in range(tamanho custosRelativos)
    - se custosRelativos [i] < 0
      - return false
  - return true



Desenvolvimento

- simplex (matrizNaoBasica, matrizBasica, vetor_b, custo_n, custo_b, maxOrMin, nomeVariaveis, variaveisBasicas, variaveisNaoBasicas)
  - cont. enquanto !solucaoOtima
    - check se solução é ótima pelo vetor de custos relativos (todos maior que 0)
    - caso sim, a solução é ótima, printa os resultados
    - caso nao, continua
    - calcula a direção simplex (y = B^{-1} * a[menorIndice]) **<-**
    - calcula o theta, escolhe o menor valor positivo para sair da base
    - troca de posição as colunas

Desenvolvimento

- calculoDirecaoSimplex(matrizInvertida, matrizNaoBasica, indicePraEntrarBase)
  - coluna = coluna da matriz nao basica que tem o indice para entrar
  - direcao = multiplicaMatrizPorVetor(matrizInvertida, coluna)
  - return direcao

$$y \leftarrow B^{-1}a_{N_k}$$

Desenvolvimento

- simplex (matrizNaoBasica, matrizBasica, vetor_b, custo_n, custo_b, maxOrMin, nomeVariaveis, variaveisBasicas, variaveisNaoBasicas)
  - cont. enquanto !solucaoOtima
    - check se solução é ótima pelo vetor de custos relativos (todos maior que 0)
    - caso sim, a solução é ótima, printa os resultados
    - caso nao, continua
    - calcula a direção simplex (y = B^{-1} * a[menorIndice])
    - calcula o theta, escolhe o menor valor positivo para sair da base
    - troca de posição as colunas **<-**

Desenvolvimento

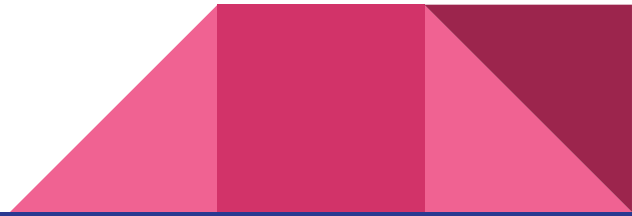
- se indiceVariavelParaSair == -1, unfeasiable
- senão, troca as colunas entre as matrizes básicas e não-básicas, entre os vetores custo_b e custo_n, e entre o vetor que guarda as variáveis
- iteração += 1
- volta para o início do loop

Desenvolvimento

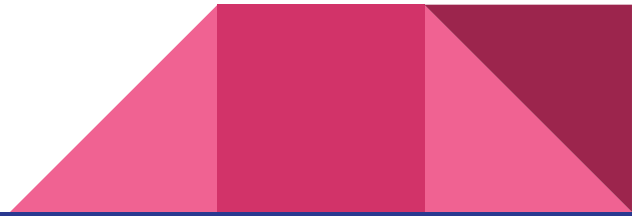
- calculoTheta(xb, y)
  - theta = [0.0] * tamanho de xb
  - for i in range(tamanho de xb)
    - se y[i] > 0:
      - theta[i] = xb[i] / y[i]
  - return theta
- indiceVariavelSair(theta):
  - indice = -1
  - thetaMinimo = infinito
  - for i in range(tamanho theta)
    - se theta[i] > 0 e theta[i] < thetaMinimo
      - thetaMinimo = theta[i]
      - indice = i
  - return indice

- Pré-processamento
  - Receber arquivo
  - Matriz não-básica
  - Matriz Básica
    - Criação das variáveis de folga ou excesso
  - Tratamento sinal maior ou igual
    - Se sim, precisa da fase 1
  - Vetor B
    - Tratamento valor menor que zero
  - Custo N
  - Custo B
  - Tratamento Max ou Min
  - Tratamento bounds

- Fase 1
  - Criação problema artificial
  - Cálculo custos relativos
    - Vetor multiplicador simplex
    - Custos relativos
    - Variável a entrar na base
  - Teste otimalidade
  - Cálculo direção simplex
  - Determinação variável entrar e sair base
  - Atualização partição básica
  - Verificação variáveis artificiais na base

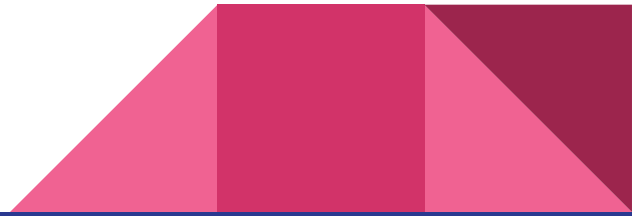


- Fase 2
  - Cálculo da solução básica
    - Inversa
  - Cálculo dos custos relativos
    - Vetor multiplicador simplex
    - Custos relativos
    - Determinação variável a entrar na base
  - Teste de otimalidade
  - Cálculo direção simplex
  - Determinação do passo e variável a sair da base
  - Atualização partição básica
- Cálculo solução ótima
  - variáveis em ordem



Considerações

- Funciona para minimizações e maximizações
- Funciona para problemas que não precisam a fase 1 e que não tenham restrições maiores que zero
- Poderia ter ficado melhor com uso da biblioteca NumPy para manipulação de matrizes
- Criado primeiramente para resolver apenas o algoritmo simplex (sem a primeira fase)



Referências

http://www.phpsimplex.com/pt/historia.htm

ARENALES, M.; ARMENTANO, V.; MORABITO, R. & YANASSE, H. Pesquisa Operacional. Rio de Janeiro: Elsevier/Campus, 2007.

Apostila (do conteúdo básico)

https://docs.python.org/3.11/

https://www.ibm.com/docs/api/v1/content/SSSA5P_22.1.1/ilog.odms.cplex.help/refpythoncplex/html/cplex-module.html