

Implementação de geração de colunas usando CPLEX em linguagem C/C++

Ministrantes:

Dr. Landir Saviniec (UFPR)

Dr. Luiz Henrique Cherri (ODM/USP)

01 e 02 de fevereiro de 2018



Introdução

Neste minicurso, abordaremos a técnica de geração de colunas (GC) (DESAULNIERS; DESROSIERS; SOLOMON, 2005) dando enfoque na implementação do método. Mostraremos como implementar GC usando o pacote de otimização CPLEX (concert) com interface de programação na linguagem C++. O minicurso está estruturado da seguinte forma:

1. Apresentamos uma breve revisão do método de decomposição Dantzig-Wolfe (BAZARAA; JARVIS; SHERALI, 2010).
2. Propomos um framework genérico em C++ para implementação do método de GC.
3. Mostramos como usar o framework para implementar uma GC para o *problema de dimensionamento de lotes capacitado com múltiplos itens*.
4. Mostramos como usar o framework para implementar a GC para o *problema de corte unidimensional*.

Ao final do minicurso, espera-se que o participante esteja apto a reusar o framework para implementar GC para outros problemas.

Decomposição Dantzig-Wolfe

Método de resolução de problemas lineares;

Aproveita da estrutura especial de algumas formulações;

Definições:

- x : vetor de variáveis do PL.
- x_1, x_2, \dots, x_N : uma decomposição das variáveis de x em N subconjuntos disjuntos.

Minimizar

$$c_1x_1 + c_2x_2 + \dots + c_Nx_N$$

Sujeito a:

$$A_1x_1 + A_2x_2 + \dots + A_Nx_N = b_0 \quad (\text{restrições de ligação})$$

$$B_1x_1 \leq b_1 \quad (\text{bloco 1})$$

$$B_2x_2 \leq b_2 \quad (\text{bloco 2})$$

$$\ddots \quad \vdots$$

$$B_Nx_N \leq b_N \quad (\text{bloco N})$$

$$x_1, x_2, \dots, x_N \geq 0$$

Decomposição Dantzig-Wolfe

Ideia básica: **decomposição por bloco** $i = 1, \dots, N$

Seja:

- S_i : conjunto de todos os pontos extremos pertencentes ao poliedro

$$\Pi_i \equiv \{x_i : B_i x_i \leq b_i, x_i \geq 0\}$$

Supondo que Π_i é limitado, ver (BAZARAA; JARVIS; SHERALI, 2010) para o caso ilimitado.
Logo:

- $x_i \in \Pi_i$, se e somente se:

$$x_i = \sum_{j=1}^{|S_i|} \lambda_{ij} x_{ij}$$

$$\sum_{j=1}^{|S_i|} \lambda_{ij} = 1$$

$$\lambda_{ij} \geq 0 \quad j = 1, \dots, |S_i|$$

Isto é, x_i é combinação convexa dos pontos extremos de Π_i .

Problema mestre (PM)

Substituindo x_i no PL original, temos:

$$\text{Minimizar} \quad \sum_{i=1}^N \sum_{j=1}^{|S_i|} (c_i x_{ij}) \lambda_{ij} \quad (1)$$

Sujeito a:

$$(w) \quad \sum_{i=1}^N \sum_{j=1}^{|S_i|} (A_i x_{ij}) \lambda_{ij} = b_0 \quad (2)$$

$$(\alpha_i) \quad \sum_{j=1}^{|S_i|} \lambda_{ij} = 1 \quad i = 1, \dots, N \quad (3)$$

$$\lambda_{ij} \geq 0 \quad i = 1, \dots, N; j = 1, \dots, |S_i| \quad (4)$$

Em que:

- (3): são as restrições de convexidade.

Seja w o vetor de variáveis duais associadas às restrições (2) e α_i a variável dual associada a i -ésima restrição de (3). O subproblema Π_i pode ser escrito como segue:

$$\text{Minimizar } cr_i = c_i x_i - w A_i x_i - \alpha_i \quad (5)$$

Sujeito a:

$$B_i x_i \leq b_i \quad (6)$$

$$x_i \geq 0 \quad (7)$$

Em que:

- cr_i : custo reduzido associado a coluna gerada pelo subproblema Π_i .

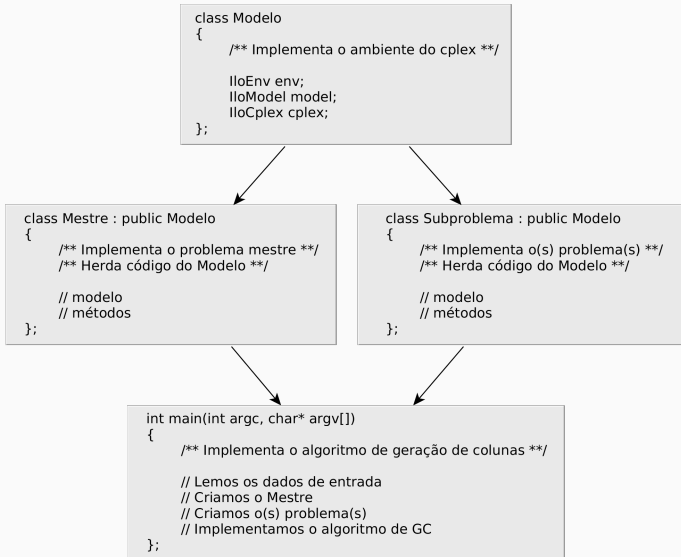
- O PMR é inicializado com um pequeno número de colunas artificiais penalizadas com um Big M.
- A cada iteração todos os subproblemas são resolvidos e novas colunas são geradas.
- As colunas com **custos relativos negativos** são adicionadas ao PMR.
- O algoritmo de GC termina quando não houver nenhuma coluna com custo relativo negativo.

Pseudo-código de um algoritmo de geração de colunas

GC()

```
1  Inicializar o PMR com variáveis artificiais
2  Inicializar um vetor de subproblemas  $\Pi$ 
3  do {
4      Resolver o PMR
5      novaColuna = 0
6      for each ( $\pi \in \Pi$ ) {
7          Atualizar a função objectivo do subproblema  $\pi$  usando as valores duais do PMR
8          Resolver o subproblema  $\pi$ 
9          if (custo reduzido  $cr < 0$ ) {
10             Inserir a coluna no PMR
11             novaColuna = novaColuna + 1
12         }
13     }
14 } while (novaColuna > 0)
15 return valor do PMR
```

Framework genérico em C++



```
/** -----  
Minicurso: Implementação de geração de colunas usando CPLEX em linguagem C/C++  
Descricao: Framework para implementacao de geracao de colunas  
Autores: Landir Saviniec e Luiz Henrique Cherri  
-----**/  
  
#include <iostream>  
#include <vector>  
#include <pthread.h>  
#include <ilcplex/ilocplex.h>  
  
//declaracao das estruturas para armazenar os dados de entrada  
  
//declaracao das classes Subproblema e Mestre  
class Subproblema;  
class Mestre;
```

```
class Modelo
{
    //esta classe implementa as ferramentas do cplex, modelo e resolvidor

public:

    IloEnv env; //ambiente do cplex
    IloModel model; //modelo
    IloCplex cplex; //cplex
    IloObjective objective; //expressao da funcao objetivo

    //construtores-----
    Modelo(){
        //chamado quando o modelo e criado
        env = IloEnv(); //cria o ambiente
        model = IloModel(env); //cria o modelo
        cplex = IloCplex(env); //cria o cplex
    }

    ~Modelo(){
        //chamado quando o modelo e destruido
        model.end(); //destroi o ambiente
        cplex.end(); //destroi o modelo
        env.end(); //destroi o cplex
    }

    //metodos-----
    void setStream(std::ostream& st){
        //habilita o log do cplex
        cplex.setOut(st);
    }
}
```

```
void setStreamOff(){  
    //desabilita o log do cplex  
    cplex.setOut(env.getNullStream());  
}  
  
void solve(){  
    //invoca o cplex  
    cplex.solve();  
}  
  
IloNum getObjective(){  
    //retorna o valor da funcao objetivo  
    return cplex.getObjValue();  
}  
  
IloCplex::CplexStatus getStatus(){  
    //retorna o estado do cplex  
    return this->cplex.getCplexStatus();  
}  
};
```



```
class Mestre : public Modelo //herda a implementacao da classe 'Modelo'
{
    //esta classe implementa o problema mestre

public:
    //declaracao das variaveis e ranges

    //construtores-----
    Mestre(){
        //chamado quando o PM e criado
    }

    ~Mestre(){
        //chamado quando o PM e destruido
    };

    //metodos-----
    void criar_modelo(){
        //criar array de variaveis e coeficientes
        //criar Ranges
    }

    //declaracao do metodo que adiciona uma coluna ao PM,
    //usando o subproblema passado como argumento
    void adicionar_coluna(Subproblema* sub);

    void criar_colunas_artificiais(){
        //metodo para inicializar o mestre
    }
};
```

Subproblema(s)

```
class Subproblema : public Modelo //herda a implementacao da classe 'Modelo'
{
    //esta classe implementa o(s) problema(s)

public:

    //declaracao das variaveis
    int i; //variavel para identificar o indice do subproblema

    //construtores-----
    Subproblema(int index){
        //chamado quando o subproblema e criado
    }

    ~Subproblema(){
        //chamado quando o subproblema e destruido
    };

    //metodos-----
    void criar_modelo(){
        //definir as variaveis do modelo
        //definir as restricoes do modelo
    }

    void update_objective(Mestre* mestre){
        //atualiza a funcao objetivo do subproblema
    }

    double get_custo_coluna(){
        //calcula o custo da coluna
    }

};
```

```
void Mestre::adicionar_coluna(Subproblema* sub){  
    //implementacao do metodo que adiciona uma coluna ao PM  
}  
  
void ler_dados(std::string arquivo){  
    std::ifstream in(arquivo.c_str());  
    //codigo para ler os dados de entrada  
}
```

Main (implementação do algoritmo de GC)

```
int main(int argc, char* argv){  
    //parametros: nome do arquivo de dados  
  
    if(argc != 2){  
        std::cout << "Argumento invalido." << std::endl;  
        return 0;  
    }  
  
    ler_dados(argv[1]); //leitura de dados  
  
    //criar os subproblemas  
  
    //criar o mestre  
  
    //algoritmo de geracao de colunas  
  
    return 0;  
}
```

Problema de dimensionamento de lotes capacitado com múltiplos itens (PDL)

O PDL consiste no planejamento da produção de múltiplos itens cuja demanda é conhecida.

A produção dos itens deve ser realizada durante um horizonte de planejamento, no qual cada período possui recursos limitados.

Esses recursos são consumidos pela produção de cada item e pelo tempo de setup dos mesmo.

Na produção de um item, são considerados custos de preparação de máquina e de estoque dos produtos.

O objetivo é encontrar uma solução que minimize esses custos.

Conjuntos:

- T : conjunto de períodos do horizonte de planejamento.
- I : conjunto de itens a serem produzidos.

Parâmetros:

- d_{it} : demanda do item $i \in I$ no período $t \in T$.
- b_i : tempo para produzir uma unidade do item i .
- sp_i : tempo de setup para processar o item i .
- s_i : custo de setup para processar o item i .
- h_i : custo unitário de estocagem do item i .
- C_t : capacidade de produção (tempo) no período t .
- $M_{it} = \min\left\{\frac{C_t - sp_i}{b_i}, \sum_{j \in T: j \geq t} d_{ij}\right\}$: limite de produção do item i no período t .

Variáveis:

- x_{it} : quantidade do item i produzida no período t . (tamanho do lote)
- e_{it} : estoque do item i no final do período t .
- $y_{it} \in \{0, 1\}$: indica se o item i é produzido no período t .

$$\text{Minimizar} \quad \sum_{i \in I} \sum_{t \in T} (s_i y_{it} + h_i e_{it}) \quad (8)$$

Sujeito a:

$$e_{it} = e_{i,t-1} + x_{it} - d_{it} \quad \forall i \in I; t \in T \quad (9)$$

$$\sum_{i \in I} (sp_i y_{it} + b_i x_{it}) \leq C_t \quad \forall t \in T \quad (10)$$

$$x_{it} \leq M_{it} y_{it} \quad \forall i \in I; t \in T \quad (11)$$

$$x_{it} \geq 0, e_{it} \geq 0, y_{it} \in \{0, 1\} \quad \forall i \in I; t \in T \quad (12)$$

Em que:

- (8): minimiza o custo total de preparação e estoque.
- (9): garante a conservação de estoque.
- (10): respeita o limite de capacidade.
- (11): o lote é limitado pela capacidade ou demanda dos períodos restantes.

Geração de colunas para o PDL

Ideia básica: decomposição por item

Conjuntos:

- S_i : conjunto de todos os planos de produção possíveis para o item $i \in I$ sobre o horizonte de planejamento.

Parâmetros:

- f_{ij} : custo do plano de produção $j \in S_i$ do item i .
- g_{ijt} : tempo de (setup + produção) do item i gasto pelo plano de produção $j \in S_i$ no período t .

Variáveis:

- $\lambda_{ij} \in \{0, 1\}$: indica se o item i segue o plano de produção $j \in S_i$.

$$\text{Minimizar} \quad \sum_{i \in I} \sum_{j \in S_i} f_{ij} \lambda_{ij} \quad (13)$$

Sujeito a:

$$\sum_{j \in S_i} \lambda_{ij} = 1 \quad \forall i \in I \quad (14)$$

$$\sum_{i \in I} \sum_{j \in S_i} g_{ijt} \lambda_{ij} \leq C_t \quad \forall t \in T \quad (15)$$

$$\lambda_{ij} \in \{0, 1\} \quad \forall i \in I; j \in S_i \quad (16)$$

Em que:

- (13): minimiza o custo total dos planos de produção.
- (14): **restrições de convexidade**: somente um plano de produção deve ser selecionado para cada item.
- (15): respeita o limite de capacidade.

Observação: relaxamos o modelo estendido (13)–(16)

$$\text{Minimizar} \quad \sum_{i \in I} \sum_{j \in S_i} f_{ij} \lambda_{ij} \quad (17)$$

Sujeito a:

$$(\pi_i^1) \quad \sum_{j \in S_i} \lambda_{ij} = 1 \quad \forall i \in I \quad (18)$$

$$(\pi_t^2) \quad \sum_{i \in I} \sum_{j \in S_i} g_{ijt} \lambda_{ij} \leq C_t \quad \forall t \in T \quad (19)$$

$$0 \leq \lambda_{ij} \leq 1 \quad \forall i \in I; j \in S_i \quad (20)$$

Em que:

- π^1 : variáveis duais associadas às restrições (18).
- π^2 : variáveis duais associadas às restrições (19).

Subproblema \mathcal{P}_i (gerador de colunas)

$$\text{Minimizar } cr_i = \sum_{t \in T} (s_i y_t + h_i e_t) - \pi_i^1 - \sum_{t \in T} \pi_t^2 (sp_i y_t + b_i x_t) \quad (21)$$

Sujeito a:

$$e_t = e_{t-1} + x_t - d_{it} \quad \forall t \in T \quad (22)$$

$$x_t \leq M_{it} y_t \quad \forall t \in T \quad (23)$$

$$x_t \geq 0, e_t \geq 0, y_t \in \{0, 1\} \quad \forall t \in T \quad (24)$$

Em que:

- cr_i : custo reduzido associado aos planos de produção (colunas) do item i .

Supondo que (x^*, e^*, y^*) é uma solução factível do subproblema (21)–(24). Isto é, um plano de produção $j' \in S_i$. Então temos:

- $f_{ij'} = \sum_{t \in T} (s_i y_t^* + h_i e_t^*)$

- $g_{ij't} = sp_i y_t^* + b_i x_t^*$

- Quando criamos o PM (17)–(20) no CPLEX, ele possui a seguinte cara:

$$\text{Minimizar} \quad (25)$$

Sujeito a:

$$= 1 \quad \forall i \in I \quad (26)$$

$$\leq C_t \quad \forall t \in T \quad (27)$$

- Logo é infactível. Para evitar tal problema, inicializamos o PM com colunas artificiais $\bar{\lambda}_i$ e custos elevados Q_i ($i \in I$).

$$\text{Minimizar} \quad \sum_{i \in I} Q_i \bar{\lambda}_i \quad (28)$$

Sujeito a:

$$\bar{\lambda}_i = 1 \quad \forall i \in I \quad (29)$$

$$\leq C_t \quad \forall t \in T \quad (30)$$

$$0 \leq \bar{\lambda}_i \leq 1 \quad \forall i \in I \quad (31)$$

- Implementando o método de geração de colunas para o PDL utilizando o framework proposto ...

Problema de corte unidimensional (PCU)

O PCU consiste em cortar um conjunto de barras maiores em um conjunto de itens.

A demanda por esses itens é conhecida a priori.

O objetivo é cortar todos os itens demandados utilizando o menor número de barras possível.

Conjuntos:

- B : conjunto de tipo de barras menores a serem cortados a partir de barras maiores com comprimento fixo.
- P : conjunto de todos os padrões de corte possíveis.

Parâmetros:

- L : comprimento das barras maiores.
- l_i : comprimento das barras do tipo $i \in B$.
- a_{ij} : número de barras do tipo $i \in B$ no padrão $j \in P$.
- d_i : demanda de barras do tipo $i \in B$.

Variáveis:

- x_j : número de barras maiores cortadas segundo o padrão $j \in P$.

$$\text{Minimizar } \sum_{j \in P} x_j \quad (32)$$

Sujeito a:

$$\sum_{j \in P} a_{ij} x_j \geq d_i \quad \forall i \in B \quad (33)$$

$$x_j \in \mathbb{Z} \quad \forall j \in P \quad (34)$$

Em que:

- (32): minimiza o número de barras cortadas.
- (33): garante o atendimento das demandas.

Dificuldade: Enumerar todos os padrões de corte $j \in P$ é impraticável.

Geração de colunas para o PCU

Observação: relaxamos o modelo (32)–(34)

$$\text{Minimizar } \sum_{j \in P} x_j \quad (35)$$

Sujeito a:

$$(\pi_i) \quad \sum_{j \in P} a_{ij} x_j \geq d_i \quad \forall i \in B \quad (36)$$

$$x_j \geq 0 \quad \forall j \in P \quad (37)$$

Em que:

- π : variáveis duais associadas às restrições (36).

Subproblema \mathcal{P} (gerador de padrões ou colunas)

$$\text{Minimizar } cr = 1 - \sum_{i \in B} \pi_i y_i \quad (38)$$

Sujeito a:

$$\sum_{i \in B} l_i y_i \leq L \quad (39)$$

$$y_i \in \mathbb{Z} \quad \forall i \in B \quad (40)$$

Em que:

- cr : custo reduzido associado ao padrão de corte (coluna) gerado.
- y_i : número de barras do tipo $i \in B$ no padrão de corte gerado.

Supondo que y^* é uma solução factível do subproblema (38)–(40). Isto é, um padrão de corte $j' \in P$. Então temos:

- $a_{ij'} = y_i^*$

- Quando criamos o PM (35)–(37) no CPLEX, ele possui a seguinte cara:

$$\text{Minimizar} \quad (41)$$

Sujeito a:

$$\geq d_i \quad \forall i \in B \quad (42)$$

$$(43)$$

- Logo é infactível. Para evitar tal problema, inicializamos o PM com colunas artificiais \bar{x}_i e custos elevados Q_i ($i \in B$).

$$\text{Minimizar} \quad \sum_{i \in B} Q_i \bar{x}_i \quad (44)$$

Sujeito a:

$$\bar{x}_i \geq d_i \quad \forall i \in B \quad (45)$$

$$\bar{x}_i \geq 0 \quad \forall i \in B \quad (46)$$

- Implementando o método de geração de colunas para o PCU utilizando o framework proposto ...



BAZARAA, M. S.; JARVIS, J. J.; SHERALI, H. D. *Linear Programming and Network Flows*. 4th. ed. [S.l.]: Wiley-Interscience, 2010. ISBN 9780470462720.



DESAULNIERS, G.; DESROSIERS, J.; SOLOMON, M. M. *Column generation*. 1st. ed. [S.l.]: Springer Science & Business Media, 2005. v. 5.