UNIVERSIDADE FEDERAL DA PARAÍBA
ENGENHARIA DE COMPUTAÇÃO

Tassany Onofre de Oliveira

Thiago Fernandes Bonfim Sousa

RELATÓRIO DE PESQUISA OPERACIONAL: PLANO MESTRE DE PRODUÇÃO

João Pessoa, PB
2020

0.1 Introdução

Programação Linear é a área da computação onde tem sua aplicação no apoio de decisões a serem tomadas para atingir um objetivo, bastante usada na área de otimização de recursos. A programação Linear inteira é uma área mais restrita onde as variáveis dentro do problema são inteiras, quando todo o modelo possui variáveis inteiras é chamado de Programação Linear Inteira Pura, caso o contrário é denominado Programação Inteira mista

0.2 Modelagem

O modelo foi pensado focando nos estoque forçando com que a demanda seja sempre atendida e de acordo com a otimização saberíamos se seria rentável produzir a cada demanda ou se seria melhor manter em estoque, sempre forçando a produção do primeiro produto ser igual a demanda por meio de uma restrição e respeitando o máximo da produção de acordo com as horas de trabalho. Lembrando que como na primeira semana o estoque é 0 para todos os produtos, então temos que separar essas restrições das demais, fazendo apenas a subtração da demanda com o que é produzido. As restrições das demais semanas, levam em consideração o estoque da semana passada para atender a demanda, lembrando que nas restrições seguintes depois do produto 1 a demanda é representada pela proporção de quanto se utiliza para produzir o produto 1 no caso da segunda restrição, de quanto se utiliza para produzir o produto 2 no caso da última restrição, no caso no produto 3 como entra nos processo de produção de 1 e 2 é necessário que incluir o quanto é necessário para a produção de cada um. As restrições de compra na primeira semana como não há estoque é necessário que haja compra logo a variável booleano que regula o custo fixo da compra foi colocada em 1 na primeira semana, nas semanas seguintes foi colocada conforme houver a compra dos produto.

0.2.1 Variáveis

Cj: Boolean que indica se houve ou não compra na semana j

Cj $\in [0, 1], \forall j = 1, 2, 3, ...T$

Xij: Quantidade do produto i na semana j

X ij $\in [0; \infty+], \forall i \in [1, 2, 3, 4], j = 0, 1, 2, 3, ...T - 1$

Yij: Estoque do produto i na semana j,

Yij $\in [0; \infty+], \forall i \in [1, 2, 3, 4], j = 0, 1, 2, 3, ...T - 1$

Dados de entrada

Pj: Demanda da semana j

CC1: Custo de compra do produto X2j

CC2: Custo de compra do produto X3j

EP1: Custo de armazenamento do produto Y1j

EP2: Custo de armazenamento do produto Y2j

EC1: Custo de armazenamento do produto Y3j

EC2: Custo de armazenamento do produto Y4j

CF: Custo fixo de quando há compra

0.2.2 Função objetivo

Minimizar

$$\sum_{j=0}^{T-1} CC1*X3j+CC2*X4j+EP1*Y1j+EP2*Y2j+EC1*Y3j+EC2*Y4j+CF*Cj = Z$$

0.2.3 Restrições

Estoque na semana inicial

Y10=X10-P0

Y20 = X20 - 2X10

Y30 = X30 - 3X10 - X20

Y40 = X40 - 2X20

Estoque nas demais semanas

Y1j=Y1j-1 + X1j-Pj, $\forall j \in j = 0, 1, 2, 3, ...T - 1$

Y2j=Y2j-1 + X2j - 2X1j, $\forall j \in j = 0, 1, 2, 3, ...T - 1$

Y3j=Y3j-1 + X3j - 3X1j - X2j, $\forall j \in j = 0, 1, 2, 3, ...T - 1$

Y4j=Y4j-1 + X4j - 2X2j, $\forall j \in j = 0, 1, 2, 3, ...T - 1$

Compra da semana inicial

C0 =1

Compra das demais semanas

X3j +X4j <= 9PjCj, $\forall j \in j = 0, 1, 2, 3, ...T - 1$

Tempo de trabalho

X1j +X2j <=800, $\forall j \in j = 0, 1, 2, 3, ...T - 1$

Produção igual a demanda X1j = Pj, $\forall j \in j = 0, 1, 2, 3, ...T - 1$

0.3 Implementação

A implementação da modelagem foi eleborada utilizando a linguagem de programação C++ em conjunto com a biblioteca CPLEX, responsável por resolver as instruções e indicar qual sua melhor solução.

0.3.1 Data.hpp

A primeira parte do código está localizada no Data.hpp, onde se iniciará as variáveis necessárias e, além disso, os "Get's"para conseguir interagir com as variáveis sem risco de compromote-las. Como é um problema linear inteiro, todas elas foram atribuidas como "int", para garantir que haja a solução ótima. A demanda da semana está como um vetor, ou seja, um array dinâmico pois sua quantidade depende do número de semans que será inseridas em "numSemanas"quando for feita a leitura das instâncias.

```
1     private:
2         int numSemanas; //j
3         int numProdutos; //i
4         int cc1; // custo c1
5         int cc2; // custo c2
6         int cf;// custo fixo
7         int ep1; // custo estoque de p1
8         int ep2; // custo estoque de p2
9         int ec1; // custo estoque de c1
10        int ec2; // custo estoque de c2
11
12        std::vector<int>demandaSem; //demanda da semana
```

0.3.2 Data.cpp

Essa página é onde o programa irá ler o arquivo que contém as intâncias do problema. O trecho de código abaixo mostra que o arquivo é aberto para a leitura e a partir do fscanf e caso ele retornar uma número diferente de 1 significará que houve um erro na leitura. Logo após a inserção de todas as variáveis, o programa pode ser fechado com o "fclose(f)".

```
1 FILE* f = fopen(filePath,"r");
2
3     //lendo o número de semanas
4     if(fscanf(f, "%d", &numSemanas) != 1)
```

```
    {
        printf("Problem while reading instance.1\n");

    }
fclose(f);
```

Essa parte ddo Data.cpp mostra como é feita as funções "Get's"do programa, com funcionalidade de apenas retornar a variável.

```
int Data::getNumSemanas()
{
    return numSemanas;
}
```

0.3.3 Main.cpp

A main é responsável por realmente envolver a biblioteca que resolve a modelagem, nela será iniciado a s variáveis que não veem nas instâncias e onde será implementado a função objetivo e as restrições.

```
#include "../include/Data.hpp"
#include <stdio.h>
#include <iostream>
#include <ilcplex/ilocplex.h> // biblioteca resolvedora


void solve(Data& data);

int main(int argc, char** argv)
{
    /* precisa de duas coisas para compilar o programa,
    o nome do arquivo e as instâncias utilizadas*/
    if(argc != 2)
    {
        printf("Usage:\n./bin instance\n");
        return 0;
    }

```

```
19      Data data(argv[1]);
20      solve(data);
21
22      return 0;
23  }
```

Na main estará a função "solve()"que lê o arquivo das instâncias e os coloca de forma a atender o modelo feito. Nele, há inicialmente, adição das variáveis X(quantidade do produto i na semana j), Y(quantidade em estoque do produto i na semana j) e a C (booleano que indica se houve ou não compra na semana j). Definimos C como "IloBoolVarArray"pois ele somente tem dois valores, ou zero ou um, e haverá um C para cada semana, por tal motivo, é colocado o "data.getNumSemana()", que indica quantas semanas o problema específico contém. O "for"serve para adicionar o C ao modelo. As variáveis X e Y são semelhantes na inicialização, porém elas são do tipo "IloNumVarArray"e precisam de outro "for", porque dependem tanto do número de semanas, quanto na quantidade de produtos.

```
1       IloEnv env;
2       IloModel modelo(env);
3
4       //variavel  booleana c_j {0,1} que indica
5       //se houve compra ou não na semana especifica
6       IloBoolVarArray c(env, data.getNumSemanas());
7
8       //adiciona a variavel c_j ao modelo
9       for(int j = 0; j < data.getNumSemanas(); j++)
10      {
11          char name[100];
12          sprintf(name, "C(%d)", j+1);
13          c[j].setName(name);
14          modelo.add(c[j]);
15      }
16
17          IloArray <IloNumVarArray> x(env, data.getNumProdutos());
18      for(int i = 0; i < data.getNumProdutos(); i++)
19      {
20          IloNumVarArray vetor(env, data.getNumSemanas(), 0, IloInfinity, ILOINT);
21          x[i] = vetor;
22      }
```

Em seguida, temos a criação das Função Objetivo, iniciáda como "IloExpr". Como é um somatório contendo todas as variáveis das semanas, ela estará em um "for"que vai de zero até "data.getNumSemanas()". Quando ela sair do laço de repetição é adicionada ao modelo com o "modelo.add(IloMinimize(env, obj)"que estará minimizando.

```
1     IloExpr obj(env);
2     for(int j = 0; j < data.getNumSemanas(); j++)
3     {
4         obj += ((data.getCC1())*x[2][j] + (data.getCC2())*x[3][j] + (data.getEP1())
5         (data.getEP2())*y[1][j] + (data.getEC1())*y[2][j] +
6         (data.getEC2())*y[3][j] + data.getCF()*c[j]);
7     }
8     modelo.add(IloMinimize(env, obj));
```

As retrições foram atribuídas como "IloRange"que é uma maneira conveniente de expressar uma restrição de alcance. Depois disso, é colocado o nome da restrição com a função "r.setName("name")"com o intuito apenas de uma melhor visualização e organização do problema. Finalmente, ela é colocado no modelo com o "modelo.add(r"). É possível perceber que no programa algumas restrições foram colocas em um laço que roda apenas uma vez, isso serviu apenas para não usar inúmeras variáveis no programa, assim, o melhorando, porque após o final da repetição o "IloRange r"some.

```
1     for(int i = 0; i < 1; i++)
2     {
3         IloRange r = (y[0][0] - x[0][0] + data.getDemandaSemana(0) == 0);
4         char name[100];
5         sprintf(name, "Estoque_semana_1_produto_1");
6         r.setName(name);
7         modelo.add(r);
8
9         //forma antiga como estáva sendo feita
10        //modelo.add(y[0][0] == x[0][0] - data.getDemandaSemana(0));
11    }
```

O trecho de código em seguida tem apenas o intuito de definir o modelo, seus parâmetros (quantidade de threads usadas e tempo máximo) e onde o modelo será escrito. Além disso, contém a chamada para solucionar o problema dentro de um "try/catch"caso haja algum erro na compilação expressar o que está errado.

```
1    IloCplex tas(modelo);
2     tas.setParam(IloCplex::TiLim, 2*60*60);
3     tas.setParam(IloCplex::Threads, 1);
4     tas.exportModel("modelo.lp");
5
6     try
7     {
8         tas.solve();
9     }
10    catch(IloException& e)
11    {
12        std::cout << e;
13    }
```

0.4 Compilação e Resultados

As intâncias para a compilação estão no arquivo de texto "1.txt". Ele possui a quantidade de semanas, custo de C1, custo de C2, custo fixo, custo de estocagem de P1, custo de estocagem de P2, custo de estocagem de C1,custo de estocagem de C2 e as demandas das semanas, lido respectivamente assim pelo programa.

O código deverá ser executado no terminal a partir do comando "make rebuild". Caso não aconteceça nenhum erro, as instâncias podem ser lidas com o comando "./tas 1.txt". Se toda as instâncias escritas pelo usuário estiverem corretas o programa mostrará o valor do custo mínimo obtido, a quantidade de produzida de P1 e P2, a quantidade compradada de C1 e C2 e se houve compra a cada semana.

REFERÊNCIAS