MS877 - Projeto supervisionado 2

Estudo de modelos de sequenciamento da produção.

Letícia Satie Morimoto

1. Objetivo.

Neste projeto, pretendemos formular e resolver problemas de sequenciamento da produção utilizando um pacote comercial de modelagem matemática, o AIMMS, e uma rotina de solução de problemas de programação inteira mista no MATLAB.

2. Noções básicas de um problema de sequenciamento de produção.

Primeiramente, vamos definir a notação necessária para modelarmos um problema de sequênciamento de produção.

Definimos como *n* o número de tarefas a serem executadas e como *m* o número de máquinas disponíveis. Além disso, usamos a seguinte notação:

- p_{ij} é o tempo que cada tarefa j gasta para ser processado na máquina i;
- r_j é a hora que a tarefa j chega no sistema;
- d_j é o prazo de entrega da tarefa j;
- w_j é o fator de prioridade da tarefa j.

Para descrevermos o problema, usamos a representação $\alpha\,|\beta\,|\gamma$, onde α especifica o tipo de máquinas usadas, β fornece detalhes do processo e γ indica a função objetivo a ser minimizada.

Consideramos as seguintes possibilidades para α:

- 1: apenas uma máquina está envolvida no processo;
- P_m: temos m máquinas idênticas em paralelo, e cada tarefa precisa de apenas uma operação em uma das máquinas;
- Q_m: temos m máquinas em paralelo com diferentes velocidades de processamento (v_i);
- R_m: temos m máquinas diferentes em paralelo, e cada máquina i pode realizar uma tarefa j com velocidade v_{ij} no tempo $p_{ij;}$

- F_m: temos m máquinas em série, e cada tarefa j deve ser processada em cada uma das m máquinas, ou seja, todas as tarefas passam pelo mesmo caminho, seguindo o esquema FIFO (”First In, First Out”);
- FF_s: temos s estágios em série, com um número de máquinas em paralelo em cada estágio. Cada tarefa j deve passar por todos os estágios;
- O_m: cada tarefa j deve passar por uma das m máquinas. No entanto, alguns desses processos podem ser nulos;
- J_m: cada tarefa j tem seu próprio caminho a seguir;

Os valores possíveis para β são:

- r_j: indica que a tarefa j não deve começar antes da sua liberação (r_j). Se r_j não aparecer em β, então a tarefa j pode começar a qualquer hora;
- s_{jk}: indica que o tempo de preparação das máquinas depende da sequência de tarefas a ser executada. Se o tempo também depende da máquina, usamos a representação s_{ijk}.
- prmp: indica que não é necessário manter uma tarefa na máquina até o fim do seu processo, podendo-se trocar a máquina (em paralelo) ou devolver a tarefa;
- prec: indica que uma ou mais tarefas devem ser completadas antes de outras tarefas poderem ser feitas. Se as tarefas têm sucessores e predecessores, as restrições são denominadas *chains* (ou seja, são encadeadas). Se a tarefa tem apenas sucessores, as restrições são denominadas *intree*. Se a tarefa tem apenas predecessores, as restrições são denominadas *outtree*;
- brkdwn: indica que as máquinas não estão disponíveis continuamente;
- M_j: indica que nem todas as máquinas podem realizar a tarefa j, mas somente aquelas incluídas no conjunto M_j;
- prmu: indica que a ordem em que as tarefas passam pela primeira máquina é mantida dentro do sistema;
- block: é um fenômeno que pode ocorrer em *flow shops*. Ocorre quando uma tarefa bloqueia uma máquina quando a máquina para qual irá em seguida não está livre e não há espaço para armazenagem.
- nmt: indica que tarefas não podem esperar entre duas máquinas. Este é um fenômeno que pode ocorrer em *flow shops*;
- recrc: indica que uma tarefa pode visitar uma máquina mais de uma vez;
- p_j: indica que o tempo de processamento é igual para todas as tarefas;
- d_j: indica que o prazo de entrega é igual para todas as tarefas.

Vamos considerar, agora, que C_{ij} representa o tempo de processamento da tarefa j na máquina i; T_j: corresponde ao atraso na entrega do produto j, sendo dado por $T_j = \max \{C_j - d_j, 0\}$; e U_j é uma penalidade por atraso, valendo 1 se $T_j > 0$ e 0, caso contário. Neste caso, os valores possíveis para γ são:

- C_{max}: minimizamos o tempo necessário para completar a última tarefa que deixa o sistema (*makespan*);
- L_{max}: minimizamos a pior violação do prazo de entrega;
- $\sum [w_j C_j)]$: minimizamos a soma ponderada dos tempos de conclusão das tarefas;
- $\sum [w_j(1-e^{-rC_j})]$: minimizamos a soma ponderada dos tempos de conclusão das tarefas, com um descondo dado pelo parâmetro r.
- $\sum (w_j T_j)$: minimizamos o atraso ponderado;

- $\sum (w_j U_j)$: minimizamos o número de tarefas atrasadas, considerando seu fator de prioridade.

Um sequenciamento factível é chamado *sem atraso* se nenhuma máquina é mantida inativa quando há uma operação para ser realizada. Por outro lado, um *schedule* factível é chamado *ativo* se nenhuma operação pode ser completada mais cedo alterando a ordem de processamento nas máquinas e sem atrasar outra operação.

3. **Resolvendo problemas no AIMMS e no MATLAB.**

Selecionamos alguns problemas para que fosse possível modelá-los e resolvê-los usando os programas AIMMS e MATLAB. Começamos por um problema simples de programação linear, para aprender os comandos básicos e a sintaxe dos programas. Em seguida, resolvemos problemas mais complexos de roteamento.

3.1. **Problema de Programação Linear.**

O primeiro problema que resolvemos envolve apenas programação linear. O objetivo é a maximização do lucro de uma fábrica, considerando-se a disponibilidade de matéria-prima, e a capacidade da fábrica.

A formulação do problema foi feita da seguinte maneira:

Variáveis:

x_p : quantidade de produtos p a serem produzidos, onde p =1 , 2.

Função Objetivo:

Max $\sum_p lucro(p) * x(p)$

Restrições:

$$\sum_p capacidade(m,p) * x(p) \leq disponibilidade(m),$$

onde m é o tipo de matéria-prima, m=1,2,3.

Parâmetros:

Disponibilidade (m):

Mat_1	6000
Mat_2	6000
Mat_3	2500

Capacidade (m,p):

	$Produto_1$	$Produto_2$
Mat_1	1	1
Mat_2	2	2
Mat_3	0	1

Lucro(p):

$Produto_1$	22 $
$Produto_2$	20 $

a) Programa feito no AIMMS.

```
Set: Produtos
Index: p
Data { prod1, prod2}

Set: Matprima
Index: m
Data { mat1, mat2, mat3 }

Parameter: lucro (p)
Data { prod1 : 22,  prod2 : 20 }

Parameter: disponibilidade (m)
Data { mat1 : 6000,  mat2 : 6000,  mat3 : 2500 }

Parameter: matriz (m,p)
Data { ( mat1, prod1 ) : 1,  ( mat1, prod2 ) : 1,  ( mat2, prod1 ) :
2,  ( mat2, prod2 ) : 2,   ( mat3, prod2 ) : 1 }

Variable: x(p)
Range: nonnegaive

Variable: objetivo(a)
Definition: sum(p, lucro(p)*x(p))

Constraint: dispon.
Definition: sum(p, matriz(m,p)* x(p))<= disponibilidade(m)

Mathematical Programming
Objetive: objetivo
Direction: Maximize
Variables: All Variables
Constraints: All Constraints
Type: LP (Linear Programing)
```

Resultados:

A solução ótima encontrada foi:

x(1) = 3000 produtos; x(2) = 0 produtos.

Valor da F.O.: $66.000.

b) Programa feito no MATLAB.

No MATLAB, é preciso definir dos dados do problema diretamente na forma matricial. De uma forma geral, dado um problema de programação linear na forma

$$\begin{array}{ll} \min & f^T x \\ \text{sujeito a} & A\,x \leq b \\ & A_{eq}\,x = b_{eq} \\ & l_b \leq x \leq u_b \end{array}$$

devemos fornecer os vetores f, b, b_{eq}, l_b e u_b, além das matrizes A e A_{eq}. Observe que o MATLAB trabalha separadamente com restrições de igualdade e desigualdade.

Para resolver nosso pequeno problema de produção, os comando usados foram

```
f  = [-22 -20]';
A  = [1 1; 2 2; 0 1];
b  = [6000; 6000; 2500];
lb = [0 0];
ub = [inf inf];
x  = linprog(f,A,b,[ ],[ ],lb,ub);
```

Resultados:

A solução ótima encontrada foi a mesma do AIMMS, ou seja

x(1)=3000 produtos; x(2)= 0 produtos.

Valor da F.O. = $66.000.

3.2-**Problema de sequenciamento com uma única máquina.**

Entrando mais especificamente no assunto a ser estudado, modelamos e resolvemos um problema se seqüenciamento com uma única máquina, cujo objetivo é minimizar o *makespan*, ou seja, a última tarefa que deixa o sistema ($1 \mid\mid C_{max}$).

A tabela abaixo fornece as tarefas e seus respectivos pesos e tempos de processamento.

Tarefas	1	2	3	4	5	6	7
Peso	0	18	12	8	8	17	16
Tempo de proc	3	6	6	5	4	8	9

O modelo matemático associado a esse problema é dado a seguir.

Variáveis:

$$x(j,i) = \begin{cases} 1, se\ tarefa\ j\ é\ executada\ na\ posição\ i \\ 0, caso\ contrário. \end{cases}$$

Função objetivo:

$$\min \sum_i \sum_j x(i,j) * \frac{peso(j)}{tempo\ proc(j)} * ord(i)$$

Restrições:

Em cada posição só pode haver uma tarefa:

$$\sum_j x(i,j) = 1,\ \ i = 1, \dots, 7$$

Cada tarefa só pode estar em uma posição:

$$\sum_i x(i,j) = 1\,,\ \ j = 1, \dots, 7$$

a) Programa feito no AIMMS.

```
Set: Jobs
Index: j
data { job1, job2, job3, job4, job5, job6, job7 }

Set: Maquina
Index: m
data { maq1 }

Set: Posição
Index: i
data { pos1, pos2, pos3, pos4, pos5, pos6, pos7 }

Parameter: peso (j)
data { job2 : 18,  job3 : 12,  job4 : 8,  job5 : 8,  job6 : 17,  job7
: 16 }

Parameter: TempoProc (j)
data { job1 : 3,  job2 : 6,  job3 : 6,  job4 : 5,  job5 : 4,    job6
: 8,  job7 : 9 }

Variable: x(j,i)
Range: binary

Variable: objetivo(a)
```

```
Definition: sum(i,sum (j, x(j,i)*peso(j)/tempoProc(j))*(ord(i)))

Constraint: Tarefa. (em cada posição só pode haver uma tarefa)
Definition: sum(i,x(j,i))=1

Constraint: Pos. (cada tarefa só pode estar em uma  posição)
Definition: sum(j,x(j,i))=1


Mathematical Programming
Objetive: objetivo
Direction: Minimize
Variables: All Variables
Constraints: All Constraints
Type: MIP (Mixed Integer Programing)
```

Resultados:

x (j,i):

0	0	0	0	0	0	1
1	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	1	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	1	0	0

Essa solução indica que as tarefas devem ser executadas na seguinte sequencia:

2 - 6 - 3 - 5 - 7 - 4 - 1

O valor da F.O. foi de 39,7.

b) Programa feito no MATLAB.

No matlab, definimos os parâmetros

```
w = [0; 18; 12; 8; 8; 17; 16];
p = [3; 6; 6; 5; 4; 8; 9];
n = length(p);
I = eye(n,n);
ord = (1:n);
```

Com base nesses dados, definimos os vatores e matrizes do problema:

```
c = (w./p)*ord;
c = c(:);

A = zeros(2*n, n^2);
um = ones(1,n);
for i=1:n,
```

```
  A(i,(i-1)*n+1:i*n) = um;
end
for i=1:n,
  A(n+1:2*n,(i-1)*n+1:i*n) = I;
end

b=ones(2*n,1);
lb=zeros(n^2,1);
ub=ones(n^2,1);
```

Para resolver esse problema de programação binária, utilizamos a função IP, criada por Sherif Tawfik e disponível na URL http://www.mathworks.com/matlabcentral/fileexchange/6990-mixed-integer-lp. Essa função destina-se à solução de problemas de programação inteira mista, e requer a definição de dois parâmetros adicionais (além daqueles exigidos pela função linprog do MATLAB).

O parâmetro "M" indica quais são as variáveis inteiras do problema. Já o parâmetro "e" deve fornecer uma tolerância da integralidade da solução. Esses parâmetros, assim como os comandos de execução da função são definidos abaixo.

```
M=(1:n^2);
e=2^-24;
[x,v,s] = IP(c,[],[],A,b,lb,ub,M,e);
x = reshape(x,n,n)
```

Resultados:

x =

0	0	0	0	0	0	1
1	0	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	1	0
0	0	0	1	0	0	0
0	1	0	0	0	0	0
0	0	0	0	1	0	0

Assim, a sequência encontrada foi a mesma do AIMMS, isto é

2 - 6 - 3 - 5 - 7 - 4 - 1

O valor encontrado para a função objetivo foi v = 39,7389.

3.3 **Problema de sequenciamento com várias máquinas, sem precedência.**

Analisaremos, agora, um problema com várias máquinas em paralelo, com a mesma velocidade, cujo objetivo é minimizar o *makespan*. Na tabela abaixo, temos as tarefas com seus respectivos pesos.

Tarefas	1	2	3	4	5	6
p_j	6	6	7	7	9	10

O problema considerado tem 6 tarefas a serem distribuídas por 3 máquinas. Consideramos que cada máquina pode executar, no máximo, três tarefas, de modo que há três posições possíveis para a execução de uma tarefa em uma máquina.

O modelo matemático desse problema é dado abaixo.

Variáveis:

$$x(j,i,m) = \begin{cases} 1, se\ tarefa\ j\ é\ executada\ na\ posição\ i\ da\ máquina\ m \\ 0, caso\ contrário. \end{cases}$$

$$t(m) = \sum_j tempoproc(j) * \sum_i x(j,i,m)$$

objetivo

Observa-se que j = 1, ..., 6; i = 1,...,3; m = 1,...,3.

Função objetivo:

min objetivo

Restrições:

Em cada posição só pode haver, no máximo, uma tarefa:

$$\sum_j x(j,i,m) \leq 1,\ \ i = 1,\dots,3;\ \ m = 1,\dots,3.$$

Cada tarefa só pode estar em uma posição:

$$\sum_m \sum_i x(j,i,m) = 1,\ \ j = 1,\dots,6.$$

O tempo de cada máquina deve ser menor ou igual ao *makespan*.

$$objetivo \geq t(m),\ \ m = 1,\dots,3.$$

a) Programa feito no AIMMS.

```
Set: Jobs
Index: j
data { job1, job2, job3, job4, job5, job6 }

Set: Maquinas
```

```
Index: m
data { maq1, maq2,maq3 }

Set: Posição
Index: i
data { pos1, pos2, pos3}

Paramenter: TempoProc (j)
data{ job1 : 6, job2 : 6, job3 : 7, job4 : 7, job5 : 9,
job6 : 10}

Variable: x(j,i,m)
Range: binary

Variable: objetivo(a)

Variable: t(m)
Definition: sum(j,tempoProc(j)*sum(i,x(j,i,m)))


Constraint: Tarefa. (em cada posição só pode haver uma tarefa)
Definition: sum(m,sum(i,x(j,i,m)))=1

Constraint: Pos. (cada tarefa só pode estar em uma posição)
Definition: sum(j,x(j,i,m))<=1

Constraint: Makespan
Definition: objetivo>=t(m)

Mathematical Programming
Objetive: objetivo
Direction: Minimize
Variables: All Variables
Constraints: All Constraints
Type: MIP (Mixed Integer Programing)
```

Resultados:

A solução obtida foi

x(1,1,3) = x(2,1,2) = x(3,2,2) = x(4,1,1) = x(5,2,1) = x(6,2,3) = 1
x(j,i,m) = 0, caso contrário.

Ou seja, a sequência encontrada para cada máquina foi:

Máquina	Posição 1	Posição 2	Posição 3
1	Tarefa 4	Tarefa 5	-
2	Tarefa 2	Tarefa 3	-
3	Tarefa 1	Tarefa 6	-

O valor do *makespan* encontrado foi 16.

b) Programa feito no MATLAB.

A definição desse problema é muito mais complexa se usamos o MATLAB, pois é preciso definir os dados do problema supondo que todas as variáveis são armazenadas em um único vetor.

Para criar um vetor solução único, y, utilizamos a notação

$$y = [x^T, t^T, \text{objetivo}]^T.$$

Além disso, o vetor x é definido de modo que

$$x(i,j,m) = y(nj*ni*(m-1)+ni*(j-1)+i).$$

Os parâmetros do MATLAB são:

```
ni=3;
nj=6;
nm=3;

i = (1:ni);
j = (1:nj);
m = (1:nm);
p = [6; 6; 7; 7; 9; 10;];
```

De posse desse valores, as matrizes e vetores das restrições do problema são dadas por:

```
% montando a matriz A

A1 = sparse(ni*nm,ni*nj*nm+1);
for m=1:nm,
  for i=1:ni,
    for j=1:nj,
      A1(ni*(m-1)+i, nj*ni*(m-1)+ni*(j-1)+i)=1;
    end
  end
end

A2 = sparse(nm,ni*nj*nm);
for m=1:nm,
  for i=1:ni,
    for j=1:nj,
      A2(m, nj*ni*(m-1)+ni*(j-1)+i) = p(j);
    end
  end
end
A2=[A2,-ones(m,1)];

A = [A1;A2];

% montando o vetor b.

b1=ones(ni*nm,1);
b2=zeros(nm,1);
b = [b1;b2];

% montando a matriz Aeq
```

```
Aeq = sparse(nj,ni*nj*nm+1);
for m=1:nm,
  for i=1:ni,
    for j=1:nj,
      Aeq(j,nj*ni*(m-1)+ni*(j-1)+i)=1;
    end
  end
end

% montando o vetor beq.

beq=ones(nj,1);
```

O vetor de custos e os limites das variáveis são dados abaixo.

```
% montando o vetor c.

c = [zeros(nm*ni*nj,1);1];

% montando o vetor l.

l = zeros(nm*ni*nj+1,1);
```

Finalmente, os comandos para a resolução do problema usando a função IP são:

```
M=(1:nm*ni*nj);
e=2^(-24);
[y,v,s] = IP(c,A,b,Aeq,beq,l,[],M,e)
objetivo = x(end);
x = reshape(y,nm,nj,ni)
```

Resultados:

A solução obtida foi

y(4) = y(14) = y(25) = y(30) = y(39) = y(53) = 1.
y(k) = 0, caso contrário.

Essa solução corresponde a

x(2,1,1) = x(5,2,1) = x(3,1,2) = x(4,3,2) = x(1,3,3) = x(6,2,3) = 1.
x(j,i,m) = 0, caso contrário.

Ou seja, a sequência encontrada para cada máquina foi:

Máquina	Posição 1	Posição 2	Posição 3
1	Tarefa 2	Tarefa 5	-
2	Tarefa 3	-	Tarefa 4
3	-	Tarefa 6	Tarefa 1

O valor do *makespan* foi 16,0, o mesmo obtido pelo AIMMS.

3.4 **Problema de sequenciamento com várias máquinas, com precedência.**

Nosso último problema inclui várias máquinas, em paralelo, com a mesma velocidade, porém incluindo relações de precedência entre as tarefas.

Supomos que uma empresa deseja montar um esquema de produção para as próximas 9 semanas. As tarefas duram várias semanas e, uma vez iniciadas, não podem ser interrompidas. Durante cada semana um certo número de trabalhadores especializados são necessários para executar cada tarefa. Assim, se a tarefa *i* dura p_i semanas então $l_{[i,u]}$ trabalhadores são necessários na semana u= 1,...,p_i. O número total de trabalhadores disponíveis na semana t é $L_{[t]}$. A tabela abaixo mostra os dados no formato (i,p_i, $l_{[i,1]}$,$l_{[i,1]}$,...,$l_{[i,pi]}$):

Tarefa	Duração	Semana 1	Semana 2	Semana 3	Semana 4
1	3	2	3	1	-
2	2	4	5	-	-
3	4	2	4	1	5
4	4	3	4	2	2
5	3	9	2	3	-

O objetivo é minimizar o número máximo de trabalhadores utilizados em qualquer uma das 9 semanas, levando em consideração as restrições abaixo:

- A tarefa 1 deve começar no mínimo 2 semanas antes da tarefa 3;
- A tarefa 4 deve começar no máximo uma semana depois da tarefa 5;
- A tarefa 1 e 2 necessitam da mesma máquina, de modo que não podem ser feitas simultaneamente.

O modemlo matemático desse problema é definido a seguir.

<u>Dados do problema:</u>

- l_{iu}: número de trabalhadores necessários para a tarefa i na u-ésima semana a partir da qual iniciou-se a tarefa i
- L_t: número de trabalhadores disponíveis na semana t

<u>Variáveis:</u>

- x_{ij}: $\begin{cases} 1, \text{se a tarefa i começou a ser executada na semana j} \\ 0, \text{caso contrário,} \end{cases}$
- w_t: número de trabalhadores na semana $t \in T$
- z_{iu}=$\begin{cases} 1, \text{se a tarefa i ainda está sendo executada na u} - \text{ésima semana} \\ 0, \text{caso contrário} \end{cases}$

Para todas as variáveis, i = 1, ...,5 (número de tarefas a serem realizadas), e j = 1,..., 9 (semanas disponíveis).

Restrições:

- Toda tarefa deve ser executada apenas uma vez:

$$\sum_{t \in Yi} x_{it} = 1 \quad \forall\, i$$

onde Y_i: {t: t + p_i − 1 ∈ T}.

- Número de trabalhadores necessários na semana t:

$$w_t = \sum_{i \in I} \sum_{s \in Yi} l_{it}\, x_{is} \quad \forall\, s \le t \le s + p_i$$

- Número máximo de trabalhadores por semana:

$$w_t \le L_t \qquad \forall \boldsymbol{t}$$

- A tarefa 1 deve começar no mínimo 2 semanas antes da tarefa 3:

$x_{1,t-2} \geq x_{3,t}$

- A tarefa 4 deve começar no máximo uma semana depois da tarefa 5:

$x_{4,t} \geq x_{5,t+1}$

- As tarefas 1 e 2 usam a mesma máquina e não podem ser feitas simultaneamente:

$$\sum_{i=1}^{2} \sum_{s \in Yi} z_{it} x_{is} = 1 \quad \forall\, s \le t \le s + p_i$$

- Restrição auxiliar que define o número máximo de trabalhadores:

Objetivo ≤ w_t $\forall t$

Função objetivo:

min objetivo

Resultados

Uma vez que esse problema apresenta uma restrição não-linear, não foi possível resolvê-lo neste momento. Entretanto, esse projeto de iniciação científica deve continuar no próximo semestre, de modo que esperamos, ser capazes de resolvê-lo futuramente.

4. **Conclusão.**

A aluna familiarizou-se com problemas reais da área de produção, bem como com a integração de um problema prático às técnicas de modelagem e otimização vistas no curso de matemática aplicada da UNICAMP. Além disso, aprendeu a programar tais modelos no pacote AIMMS e no MATLAB.

Os problemas resolvidos em ambos os pacotes tiveram resultados equivalentes. Entretanto, o AIMMS obteve respostas em menor tempo em comparação com o MATLAB. Isso, naturalmmente, é consequencia do fato do AIMMS utilizar internamente o programa CPLEX, enquanto a rotina IP do MATLAB usa uma versão interpretada do algoritmo *branch-and-bound*, que chama a função linprog, disponível na *toolbox* de otimização do programa. No futoro, pretendemos explorar a biblioteca matlog, que também resolve problemas de programação inteira mista no MATLAB. Essa função está disponível em http://www.ise.ncsu.edu/kay/matlog/.

5. **Bibliografia.**

1. BISSCHOP, J. – *AIMMS: optimization modeling*. Haarlem, Paragon Decision Technology, 2001.
2. BISSCHOP, J. & ROELOFS, M. – *AIMMS: the language reference*. Haarlem, Paragon Decision Technology, 2001.
3. PINEDO, M. – *Scheduling: theory, algorithms and systems*. Englewood Cliffs, Prentice-Hall, 1995.
4. http://www.ime.unicamp.br/~moretti/ms901_ms915/ms901_ms915.html