Escalonamento de projetos

MIMI – Programação, Lda.

Paulo Jorge de Faria dos Reis

Miguel José Camisão Rossi de Azevedo Seabra

FEUP-PLOG, Turma 3MIEIC1, Grupo 5

Abstract.

De forma a obter uma maximização de lucros desenvolveu-se um recurso adequado aos objetivos.

A nossa abordagem baseou-se no predicado *cumulative(+Tasks, +Options)* da

1 Introdução

A obtenção de planos de agendamento das tarefas com maximização de lucros

Neste documento será apresentada uma descrição mais detalhada do

1 Descrição do problema

De uma lista de encomendas de projetos de programação, pretende-se maximizar o lucro anual obtido.

Não existe a obrigatoriedade de aceitar todas as encomendas, mas uma vez

A quantidade de linhas estabelece também uma classificação dos projetos em

adfs.v.2.2011.

O preço de cada linha de código é dependente da complexidade do projeto, o

pagando previamente de forma global entre todos os projetos.

A equipa de programação inicial consistem 4 programadores Seniores,

projeto 11 dias de férias por cada 5 meses de trabalho.

Para fazer face às despesas fixas mensais e remunerações (variável), é

entregues.

Para a obtenção de soluções, utilizamos um conjunto de operações do cálculo

1 Ficheiro de dados

Os dados utilizados estão no formato de lista de listas, em que cada lista interna

1 Variáveis de Decisão

Os dados que caracterizam as encomendas: Data de entrega do projeto,

Custo_AtrasoDiario = Percentagem do total do projeto a pagar por cada dia de atraso (At).

Atraso_MaximoDias = Maximo de dias de atraso por projeto (Adias).

Atraso_MaximoCusto = Percentagem máxima do valor em falta do projeto que pode ser perdido por atrasos (Amax).

LimiteProjetoMedio = Projetos acima deste valor são médio (Nmed). O valor

LimiteProjetoComplexo = Projetos acima deste valor são sempre complexos (Ng).

Senior_ProjetoComplexo = Percentagem de dedicação do tempo de Senior necessária para cada projeto complexo (Dc).

Senior_Minimo = Percentagem de dedicação do tempo de Senior necessária para qualquer projeto (Dmin).

EntregaIntermédia_Min = Percentagem minima do n.º de linhas do projeto para entrega intermédia (Imin).

EntregaIntermedia_Max = Percentagem maxima do n.º de linhas do projeto para entrega intermédia (Imax).

ValorLinha_Complexo = 35 Preço das linhas de projetos complexos (PUc).

ValorLinha_Simples = Preço das linhas de projetos simples (PUs).

Seniores_Permanentes = Número de Seniores iniciais (P).

CustoContrato = Custo de novo contrato (C).

OrdenadoJunior = Remuneração mensal de junior (Jr).

OrdenadoSeniorPerm = Remunerção mensal de Senior permanente (Srp).

OrdenadoSeniorCont = Remunerção mensal de Senior contratado (Src).

Produtividade = Linhas de codigo por dia por programador (N).

Apredizagem = Periodo de produtividade nula após contrato.

Capital_Inicial = Capital inicial (E).

DespesasMensais = Outras despesas mensais.

1 Restrições

De forma a permitir que o motor de restrições tenha o maximo de flexibilidade para encontrar a melhor solução para o problema em questão, tentou-se que as regras permitissem várias soluções.

Sendo que existe a possibilidade de recusar projectos, tentou-se por criar uma lista de aceitação [illegible] recusados sejam ignorados definimos que os recursos usados no escalonamento são 0.

Para alocar os recursos aos projectos, primeiro, retiramos os custos fixos e os ordenados dos Programadores Senior permanente dos 12 meses do ano ao

capital inical, ficando assim com o capital que é possivel distribuir pelos projectos.

Cada projecto recebe um orçamento para poder contratar programadores Junior

A contratação de programadores junior faz-se pela necessidade de produzir um

As despesas do projecto são o custo de contratação dos programadores junior

As receitas são o numero de linhas do projecto a multiplicar pelo presso de cada uma.

1 Estratégia de Pesquisa

A estratégia foi a de obtenção do valor máximo de lucro, com a opções por defeito do labeling.

1 Visualização da Solução

Porque os resultados são listas de valores numéricos, não foi criado nenhum

1 Resultados

```
Datas de entrega:[12,2,4,7]

Datas de inicio:[11,1,3,6]

Datas intecalares:[3,1,2,3]

DuraÃ§Ã£o dos projetos:[1,1,1,1]
```

1 Conclusões e Perspetivas de Desenvolvimento

A aplicação de técnicas de programação em lógica com restrições à resolução

Já perto do final do trabalho e depois de ganhar uma importante experiência

A utilização de relações matemáticas para estabelecer restrições entre as

No entanto também aqui tivemos dificuldade na criação de um algoritmo de exclusão de encomendas, por não ser possível a sua concretização.

Em função das anteriores conclusões, a maior lição foi a de que na programação em lógica os algoritmos utilizados para o estabelecimento das restrições.

1 Bibliografia

1Página do Sicstus - http://www.sics.se/isl/sicstuswww/site/index.html

1 Anexos

1.1 Anexo 1- Código fonte elaborado.

```
:- use_module(library(clpfd)).

:- use_module(library(lists)).



mimi :-

       % Variáveis globais para teste.


       CustoAtrasoDiario is 1,          % Percentagem do
total do projeto a pagar por cada dia de atraso (A).

       Atraso_MaximoDias is 60,         % Maximo de dias
de atraso por projeto (Adias).

       Atraso_MaximoCusto is 50,        % Percentagem
máxima do valor em falta do projeto que pode ser perdido
por atrasos (Amax).


       LimiteProjetoMedio is 50000,     % Projetos acima
deste valor são médio (Nmed). O valor não está definido no
texto, foi escolhido pelo grupo.
```

```
       LimiteProjetoComplexo is 100000, % Projetos acima
deste valor são sempre complexos (Ng).

       Senior_ProjetoComplexo is 50,    % Percentagem de
dedicação do tempo de Senior necessária para cada projeto
complexo (Dc).

       Senior_Minimo is 33,             % Percentagem de
dedicação do tempo de Senior necessária para qualquer
projeto (Dmin).

       EntregaIntermedia_Min is 10,     % Percentagem
minima do n.º de linhas do projeto para entrega intermédia
(Imin).

       EntregaIntermedia_Max is 90,     % Percentagem
maxima do n.º de linhas do projeto para entrega intermédia
(Imax).


       ValorLinha_Complexo is 35,       % Preço das linhas
de projetos complexos (PUc).

       ValorLinha_Simples is 25,        % Preço das linhas
de projetos simples (PUs).


       Seniores_Permanentes is 4,       % Número de
Seniores iniciais (P).

       CustoContrato is 500,            % Custo de novo
contrato (C).

       OrdenadoJunior is 1000,          % Remuneração
mensal de junior (Jr).

       OrdenadoSeniorPerm is 4000,      % Remunerção
mensal de Senior permanente (Srp).

       OrdenadoSeniorCont is 3000,      % Remunerção
mensal de Senior contratado (Src).

       Produtividade is 35,             % Linhas de codigo
por dia por programador (N).

       Apredizagem is 15,               % Periodo de
produtividade nula após contrato.


       Capital_Inicial is 200000,        % Capital inicial
(E),
```

```
       DespesasMensais is 5000,         % Outras despesas
mensais fixas.

       Ferias is 11,                    % periodo de
ferias em cada 6 meses




       GLOBAIS = [
            CustoAtrasoDiario,       %  1
            Atraso_MaximoDias,       %  2
            Atraso_MaximoCusto,      %  3
            LimiteProjetoMedio,      %  4
            LimiteProjetoComplexo,   %  5
            Senior_ProjetoComplexo,  %  6
            Senior_Minimo,           %  7
            EntregaIntermedia_Min,   %  8
            EntregaIntermedia_Max,   %  9
            ValorLinha_Complexo,     % 10
            ValorLinha_Simples,      % 11
            Seniores_Permanentes,    % 12
            CustoContrato,           % 13
            OrdenadoJunior,          % 14
            OrdenadoSeniorPerm,      % 15
            OrdenadoSeniorCont,      % 16
            Produtividade,           % 17
            Apredizagem,             % 18
            Capital_Inicial,         % 19
            DespesasMensais,         % 20
            Ferias                   % 21
       ],
```

```
       % Lista de encomendas para testes.

          % [ [Nlinhas, Complex, Data_Intermedia,
Data_Final],..]


      Encomendas = [

             [10000, 1, 3, 12 ],

             [1550,   1, 1, 2 ],

             [2400,   1, 2, 4 ],

             [100,    1, 3, 7 ]

       ],


       % Criação das listas com os vários parâmetros de
trabalho.

       length(Encomendas, QtdEncomendas),             %
Verificar quantos projetos são para escalonar.

%        length(DatasInicio, QtdEncomendas),           %
Lista com as datas de inicio dos projetos.

       length(DatasFinaisContratadas, QtdEncomendas),  %
Lista com as datas de fim dos projetos.

       length(SenioresAAtribuir, QtdEncomendas),       %
Lista com a alocação de Séniores aos Projetos (Recurso
limitado).

       length(Complexidade, QtdEncomendas),           %
Lista com a complexidade de acordo com a encomenda.

       length(LinhasDeCodigo, QtdEncomendas),          %
Linhas de código do projeto.

       length(DatasMeioContratadas, QtdEncomendas),    %
Lista com as datas intercalares dos projetos.
```

```
        % Ler as encomendas.
        (
          foreach([LC, CP, DM, DF], Encomendas),
          foreach(LC, LinhasDeCodigo),
          foreach(CP1, Complexidade),
          foreach(DM, DatasMeioContratadas),
          foreach(DF, DatasFinaisContratadas),
          foreach(SA, SenioresAAtribuir),
          foreach([LC, SA, DM, DF], ProjetosAceites) do
% Criar a lista de tarefas.
                (LC >= LimiteProjetoComplexo -> CP1 = 1;
CP1 = CP),     % Garante que o requisito de projetos acima
de certa dimensão é sempre complexo.
                (LC >= LimiteProjetoComplexo -> SA = 100;
% Atribuir Seniores de acordo com complexidade e número de
linhas de código.
                 CP =:= 1 -> SA = 100; SA = 100)
        ),



        % Seniores disponiveis.
        Seniores #= Seniores_Permanentes * 100,



        escalonar(1000, Seniores, ProjetosAceites,
DatasFim, DatasInicio, DatasMeio,  [
                  ProjetosAEscalonar,
                  MaximoDeProjetos,
                  Aceitacoes,
                  DuracaoProjetos,
```

```
                Escal_Vars
            ]),

        append(Aceitacoes, DuracaoProjetos, VV1),
        append(VV1, Escal_Vars,VV2),

        date_to_duration(ProjetosAceites, DatasInicio,
Complexidade, ENC_aloc),

        alocate(GLOBAIS,ENC_aloc,Aceitacoes,Lucro,VARS),

        append(VV2,[],VVV),

        cumulative(ProjetosAEscalonar, [limit
(MaximoDeProjetos)]),
        labeling([],VVV),

        write('Datas de entrega': DatasFim), nl,
        write('Datas de inicio': DatasInicio), nl,
        write('Datas intecalares': DatasMeio), nl,
        write('Duração dos projetos': DuracaoProjetos), nl,
        write('ALocation vars ':VARS),nl.


%ENC = [[+Nlinhas,+NSeniors, +Data_It, +Data_F_proj,
+Complexidade],..]
```

```
alocate( Globals, ENC, Enc_aceites, Lucro, VARS ) :-

 nth1(19,Globals,E) ,    % capital inicial

 nth1(13,Globals,C),     % custo de novo contracto

 nth1(15,Globals,Srp),   % ordenado senior permanente

 nth1(14,Globals,Jr),    % ordenado Jr

 nth1(1, Globals,A),     % percentagem a decontar no lucro
por dia de atraso

 nth1(3, Globals,Amax),  % percentagem max de desconto

 nth1(2, Globals,Adias), % dias max de atraso

       nth1(17,Globals,N),     % producao de linhas de
codigo num dia por programador

 nth1(20,Globals,M),     % despesas mensais

 nth1(10,Globals,PUc),   % preco por linha de codigo num
projecto complexo

       nth1(11,Globals,PUs),   % preco por linha de codigo
simples

 nth1(18,Globals,Z),     % dias apos contratacao em que n
produz

       nth1(21,Globals,F),     % ferias por cada 6 meses

       nth1(12,Globals,NSr),   % numero de seniors
existentes




       % despesas mensais durante um ano e ordenados dos
Srs

       % inclui-se aqqui pos os Nseniors nas enc estao em
percentagem

       Desp_Fixas_ano is 12*(M + NSr*Srp),

       Cap_projectos is E - Desp_Fixas_ano,


       % os orcamentos podem ir de 1e ate ao capital
```

```
       % mas a soma dos orcamentos dos projectos nao
ultrapassa o capital inicial

       (

           foreach(Aceite, Enc_aceites),

           foreach( OT , OrcT),

           foreach( O , Orcamento)

           do

              OT in 1..Cap_projectos,

              O #= OT*Aceite

       ),

       sum(Orcamento, #=< , Cap_projectos),




       (

           foreach( Orc, Orcamento),

           foreach( [Nlinhas, Nseniors, Data_it, Data_F,
Complex], ENC),

           foreach( Aceite, Enc_aceites),

           foreach( [Contr,Nlp,Nlps], Contractos),

           foreach( [Njuniors, Despesas,Receitas,
Lucro_it, Lucro_F], Projectos)

           do

               %quantidade de contratos por junior

               Contr #= Aceite*(Data_F/6),


               %numero de linhas que cada programador Jr
produz durante o projecto

               Nlp #= Aceite*( (4*5*Data_F - Z -
F*Contr)*N  ),
```

```
                %numero de linhas que cada programador sr
produz durante o projecto

                % nsenior e' a percentagem que o Sr esta no
proj

                Nlps #= Aceite*( (4*5*Data_F - F*Contr)*
(Nseniors*N/100) ),



                %numero de juniores e' o numero de
programadores necessarios para

                % programar as linhas q o  senior nao
consegue produzir

                Njuniors #= Aceite*((Nlinhas-Nlps)/Nlp),



               %despesas sao os ordenados + custos de
contrato

        Despesas #= Aceite*(C*Njuniors +
Jr*Contr*6*Njuniors),



                %as despesas nao podem ser mais q o
orcamento

        Despesas #=< Orc,

                Receitas #= Aceite*(PUc*Nlinhas),




                %TODO

                Lucro_it #= 0,


        Lucro_F #= Aceite*(Receitas - Despesas),

                Lucro_F #>= 0

     ),
```

```
     % juntar vars de forma a fazer o labeling
     % append/2 == flatten
     append(Contractos, FContractos),
     append(Projectos, FProjectos),
     append( FContractos, FProjectos, VARS3),
     append( OrcT, Orcamento, VARS1),
     append(VARS1,VARS3,VARS).




% funcao com o objectivo de manipular os dados recebidos
do escalonamento
% de forma a que a funcao de alocamento consiga usa-los.


date_to_duration(Proj, Data_I, Complexidade, ENC_aloc) :-


    (
         foreach( [ NLinhas, NSr, Data_m, Data_f] , Proj),
         foreach( DI, Data_I),
         foreach( C , Complexidade),
         foreach( [ NLinhas, NSr, Dur_m, Dur_f, C],
ENC_aloc)
         do
              Dur_m #= Data_m - DI,
              Dur_f #= Data_f - DI
    ),
```

```
    true.


% Escalonamento (scheduling) dos projetos.

% DataTermino = Data limite para terminar todos os
projetos.

% SenioresDisponiveis = Quantidade de Séniores disponíveis
a cada momento (se for utilizada percentagem, sendo que 1
Senior = 100, é possÃvel atribuir tempo parcial de Senior
a projetos).

% Projetos = Lista com os projetos aceites.

   %projectos = [ [NLinhas, Senior_atr, Data_M, Data_F],..
]



%        OUT_VARS = [

%             ProjetosAEscalonar,

%             MaximoDeProjetos,

%             Aceitacoes,

%             DuracaoProjetos,

%             Vars

%        ],


escalonar(DataTermino, SenioresDisponiveis, Projetos,
DatasFim, DatasInicio, DatasMeio,OUT_VARS) :-


       % Criação das listas com os vários parâmetros.

       length(Projetos, QtdProjetos),          % Verificar
quantos projetos são para escalonar.
```

```
       length(DatasInicio, QtdProjetos),       % Lista com
as datas de inicio dos projetos.

       length(DatasFim, QtdProjetos),          % Lista com
as datas de fim dos projetos.

       length(DuracaoProjetos, QtdProjetos),   % Lista com
a duração dos projetos.

       length(SenioresAtribuidos, QtdProjetos),% Lista com
a alocação de Séniores aos Projetos (Recurso limitado).

       length(LinhasDeCodigo, QtdProjetos),    % Linhas de
código do projeto.

       length(DatasMeio, QtdProjetos),         % Lista com
as datas intercalares dos projetos.

       length(Aceitacoes, QtdProjetos),


       % Definição das variáveis de dominio do problema.


       domain(DatasInicio, 1, DataTermino),

       domain(DatasFim, 1, DataTermino),

       domain(DuracaoProjetos, 1, DataTermino),

       domain(Aceitacoes, 0, 1),




       % leitura dos dados dos projetos.

       (

         foreach([LC, SA, DM, DF], Projetos),

         foreach(LC, LinhasDeCodigo),

         foreach(SA, SenioresAtribuidos),

         foreach(DM, DatasMeio),

         foreach(DF, DatasFim),

         foreach(DP, DuracaoProjetos),

         foreach(DI, DatasInicio),

         foreach(AC, Aceitacoes),
```

```
          foreach(SA, SenioresAtribuidos),
          foreach(task(DI, DP, DF, Res, 0),
ProjetosAEscalonar) do
                DP #= DF - DI,
                Res #= SA * AC
        ),


        % Restrições ao problema.

        MaximoDeProjetos in 1..SenioresDisponiveis,
        sum(Aceitacoes, #>=, 1),
        count(1, Aceitacoes, #=, Aceites),
        minimum(Inicio, DatasInicio),

        % Preparação do labeling.
        append(DatasInicio, DatasFim, Vars1),
        append(Vars1, [MaximoDeProjetos], Vars),


        OUT_VARS = [
             ProjetosAEscalonar,
             MaximoDeProjetos,
             Aceitacoes,
             DuracaoProjetos,
             Vars
        ].
```

1 Anexo 2 – Exemplo de restrições da perspectiva de desenvolvimento

```
ValorProjeto = Linhas * ValorLinha,

DataFim #>= DataContrato,

DataFim #< DataContrato + Adias,

DiasCustoAtraso #= ValorProjeto

DataFim #< DataContrato + DiasCustoAtraso,

DiasAtraso #= DataFim – DataContrato,

Ganho #= ValorProjeto – (DiasAtraso * Amax *
ValorProjeto / 100),

Duracao #= DataFim – DataInicio.
```