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 se obter uma maximização de lucros, desenvolve-se com recurso a Programação em Lógica um programa que processa um conjunto de encomendas de projetos de programação e apresenta a solução que melhor se adequa aos objetivos.

A nossa abordagem baseou-se no predicado *cumulative(+Tasks,+Options)* da biblioteca *clpfd* do *Sicstus [1]*. Como retorno recebemos os dados suficientes para a elaboração de uma calendarização dos projetos: duração, data de inicio, data final e eventualmente data de entrega intermédia.

1. Introdução

A obtenção de planos de agendamento das tarefas com maximização de lucros anuais levou-nos a desenvolver este trabalho. A empresa Mimi tem uma equipa permanente de programadores seniores e a sua carteira de clientes permite-lhe ter uma lista de encomendas no início de cada ano, sendo o principal objetivo obter o máximo lucro possível em cada um dos anos civis, existe a necessidade de ter dados de apoio a decisões de gestão em termos de contratação de programadores e quais as encomendas que devem ser aceites e passar a projetos de programação.

Neste documento será apresentada uma descrição mais detalhada do problema, a solução que foi desenvolvida e uma análise da mesma.

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 aceites devem ser terminadas. Uma encomenda de projeto é caracterizada pela quantidade de linhas de código, complexidade (Simples ou Complexo), data de entrega final contratada e opcionalmente data para entrega intermédia.

A quantidade de linhas estabelece também uma classificação dos projetos em pequenos, médios e grandes, cada um destes com características próprias e que obrigam a alocar programadores especializados em maior quantidade consoante aumenta a dimensão do projeto.

O preço de cada linha de código é dependente da complexidade do projeto, o pagamento é sempre realizado contra entrega de linhas de código terminadas em datas previamente contratadas. Existe no entanto a possibilidade de atrasar a entrega incorrendo com isso num custo por cada dia de atraso, até um máximo em termos de dias e de valor do projeto. As penalizações pelos atrasos são sempre descontadas do último pagamento do projeto, o qual nunca pode ser negativo. As entregas intermédias devem ter linhas de código num intervalo definido previamente de forma global entre todos os projetos.

A equipa de programação inicial consiste em 4 programadores Seniores, mediante o pagamento de valor fixo de contratação é possível angariar mais programadores contratados Juniores ou Seniores, estes contratos são realizados por períodos de 6 meses, no caso de renovação não existe pagamento do custo de contratação. Estes programadores têm um período de adaptação durante o qual não produzem linhas de código. A remuneração mensal de cada tipo de programador é diferente, mas todos têm a mesma produtividade diária e têm direito a 11 dias de férias por cada 6 meses de trabalho.

Para fazer face às despesas fixas mensais e remunerações (variáveis), a empresa conta com um capital inicial e o valor cobrados por linhas de código entregues.

Para a obtenção de soluções, utilizamos um conjunto de operações de cálculo sobre os dados do problema de forma a obter valores que são posteriormente utilizados para aplicar restrições sobre os domínios das variáveis do problema, de forma a reduzir o espaço de pesquisa de soluções e assim obter em tempo útil uma solução para o problema.

1. Ficheiro de dados

Os dados utilizados estão no formato de lista de listas, em que cada lista interna descreve uma encomenda em termos de linhas de código, complexidade, data de entrega intermédia e data de entrega final contratada.

1. Variáveis de Decisão

Os dados que caracterizam as encomendas: Data de entrega do projeto, complexidade e quantidade de linhas são os pincipais elementos deste trabalho, mas sem a lista abaixo não teriam grane valor.

CustoAtrasoDiario = Percentagem do total do projeto a pagar por cada dia de atraso (A).

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 não está definido no texto, foi escolhido pelo grupo.

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).

EntregaIntermedia\_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

<Colar aqui a lista de variáveis do programa>

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 predicado dedicado à apresentação destes. Utilizou-se apenas o predicado interno write para apresentar as listas com as datas de inicio, intermédia e final, opcionalmente são incluídos outros valores com utilidade para a descrição da solução e/ou análise do resultado.

1. Resultados

Datas de inicio dos projetos :[99,199,149,359,299,199,211,149,359,299,269,199,149,359,299,359,199,149,59,59,34,39,33,54,46,51,52,33,6]

Datas de entrega intercalar dos projetos :[80,170,145,340,280,140,186,135,300,290,230,150,135,300,220,320,170,135,40,40,23,20,22,45,35,43,35,22,3]

Datas de finalização dos projetos :[100,200,150,360,300,200,212,150,360,300,270,200,150,360,300,360,200,150,60,60,35,40,34,55,47,52,53,34,7]

Duração dos projetos:[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]

Aceitações:[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]

Linhas de código dos projetos:[100000,150000,50000,3000,1000,4000,1500,2900,5400,2100,3000,2300,1300,2700,7200,2500,1700,3800,950,1450,1600,1500,1000,1550,1850,1900,2400,500,100]

Utilização máximo de Séniores:[100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100]

Máximo de projetos simultâneos :400

Recursos utilizados:[100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100]

Projetos escalonados:[task(99,1,100,100,0),task(199,1,200,100,0),task(149,1,150,100,0),task(359,1,360,100,0),task(299,1,300,100,0),task(199,1,200,100,0),task(211,1,212,100,0),task(149,1,150,100,0),task(359,1,360,100,0),task(299,1,300,100,0),task(269,1,270,100,0),task(199,1,200,100,0),task(149,1,150,100,0),task(359,1,360,100,0),task(299,1,300,100,0),task(359,1,360,100,0),task(199,1,200,100,0),task(149,1,150,100,0),task(59,1,60,100,0),task(59,1,60,100,0),task(34,1,35,100,0),task(39,1,40,100,0),task(33,1,34,100,0),task(54,1,55,100,0),task(46,1,47,100,0),task(51,1,52,100,0),task(52,1,53,100,0),task(33,1,34,100,0),task(6,1,7,100,0)]

Projetos aceites:29

Inicio:6

Resumptions: 468

Entailments: 90

Prunings: 553

Backtracks: 0

Constraints created: 91

1. Conclusões e Perspetivas de Desenvolvimento

A aplicação de técnicas de programação em lógica com restrições à resolução de cenários de escalonamento de tarefas com otimização de resultados tem a vantagem de se produzirem programas de pequena dimensão, pois apenas é necessário fazer a descrição das variáveis envolvidas e respetivas restrições, deixando para o interpretador de Prolog o trabalho de obter as soluções.

Já perto do final do trabalho e depois de ganhar uma importante experiência com o trabalho já realizado, mas tarde demais para começar do inicio tivemos a ideia de realizar o trabalho recorrendo a uma representação dos dados em termos de listas de listas de listas. Listas de anos, cada contendo listas que representariam as variáveis de domínio do problema (ver ponto 4 deste artigo), cada uma destas listas teria uma dimensão de 12 por forma a representarem os meses. As encomendas seriam mantidas numa lista separada, composta de listas que individualmente representariam uma encomenda individual.

A utilização de relações matemáticas para estabelecer restrições entre as variáveis de domínio iriam modelar o problema (ver anexo com exemplos).

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 com Restrições um dos pontos mais importantes é a modelação dos dados e os algoritmos utilizados para o estabelecimento das restrições.

1. Bibliografia
2. Página do Sicstus - <http://www.sics.se/isl/sicstuswww/site/index.html>
3. Anexos
   1. Anexo 1- Código fonte elaborado.
   2. Predicado para escalonar as tarefas.

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

*% 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

DF #= DI + DP,

Res #= SA \* AC

),

*% Restrições ao problema.*

*% maximum(Final, DatasFim),*

MaximoDeProjetos in *1*..SenioresDisponiveis,

sum(Aceitacoes, #>=, *5*),

count(*1*, Aceitacoes, #=, Aceites),

minimum(Inicio, DatasInicio),

*% Preparação do labeling.*

append(DatasInicio, DatasFim, Vars1),

append(Vars1, [MaximoDeProjetos], Vars),

*% obtenção de soluções*

cumulative(ProjetosAEscalonar, [limit(MaximoDeProjetos)]), *% Processamento para obtenção do escalonamento de tarefas.*

labeling([down], Aceitacoes),

labeling([up], DuracaoProjetos),

labeling([], Vars), *% Atribuição de valores concretos Ã s variáveis (labeling).*

*% Apresentação de resultados.*

write('Datas de inicio dos projetos ': DatasInicio),nl,

write('Datas de entrega intercalar dos projetos ': DatasMeio),nl,

write('Datas de finalização dos projetos ': DatasFim),nl,

write('Duração dos projetos': DuracaoProjetos),nl,

write('Aceitações': Aceitacoes), nl,

write('Linhas de código dos projetos': LinhasDeCodigo), nl,

write('Utilização máximo de Séniores': SenioresAtribuidos),nl,

write('Máximo de projetos simultâneos ': MaximoDeProjetos),nl,

write('Recursos utilizados': SenioresAtribuidos), nl,

write('Projetos escalonados': ProjetosAEscalonar), nl,

write('Projetos aceites': Aceites), nl,

write('Inicio': Inicio), nl,

fd\_statistics.

* 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.