Programação em Lógica
Trabalho prático 2
Calendarização de Competição Desportiva

Turma 3MIEIC02 - Grupo 13:
Hugo Ari Rodrigues Drumond - 201102900
João Alexandre Gonçalinho Loureiro - 200806067

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, 4200-465 Porto, Portugal

Resumo O objetivo deste trabalho é a resolução de um problema de Calendarização usando restrições da linguagem PROLOG. O relatório está organizado da seguinte forma: Descrição do problema, onde se descreve o problema de forma estruturada; Ficheiros de Dados, descreve como se deve inserir os dados num ficheiro de modo a que o programa faço o que se pretende; Variáveis de Decisão, aqui podemos encontrar todas as variáveis e os respetivos domínios; Restrições, nome de algumas restrições e qual o seu objetivo; Estratégia de pesquisa, opções do labeling usadas e breve explicação do seu modo de funcionamento; Visualização da Solução, mostramos o nosso print simplificado(sem Home/away) e o completo e a cabeça da função que imprime a solução; Resultados, tempos de procura de solução por complexidade;e, Conclusões e Perspectivas de Desenvolvimento.

Keywords: Programação em Lógica com Restrições, PLR, Calendarização de Eventos Desportivos

1 Introdução

O trabalho visa usar os conceitos de restrições da Programação em Lógica, para fazer a Calendarização de uma Competição Desportiva. A abordagem PLR é muito mais eficiente porque segue o princípio de restringir e só depois gerar ao invés da abordagem "padrão"do prolog, gerar e testar. A forma de atacar o problema seguindo este novo princípio, é feita da seguinte maneira: 1º define-se um domínio para as variáveis do problema; 2º aplica-se restrições a essas variáveis que irão diminuir o domínio do problema, feito através de consistência dos arcos; 3º Pesquisa da solução de diferentes maneiras, por exemplo: instanciar o valor mais a esquerda e ir testando as restrições e selecionando novas instanciações e testando novamente,etc. A programação em lógica com restrições é do nosso agrado pois permite resolver problemas complexos de forma eficiente através de conceitos matemáticos: domínios, expressões sobre as variáveis restringem o domínio e procura da solução dado o domínio reduzido. Não nos foi possível dialogar com os outros grupos visto que tivemos de mudar de trabalho à última da hora. Isto deveu-se a algumas dúvidas de interpretação do enunciado de Gestão de Servidores Cloud, que nos impossibilitaram de progredir no trabalho. Mandámos dois emails ao professor Tiago Pinto Fernandes só que no primeiro(12/05) não ficámos esclarecidos e não recebemos resposta ao segundo(13/05).

2 Descrição do problema

O Calendarização tem as seguintes características:

Tipo de competição: Round Robin.

Jornadas: Deve-se evitar que uma equipa jogue em casa mais que uma vez numa dada jornada. Cada equipa deve alternar ao máximo os jogos em casa e fora, se joguei nesta jornada em casa na próxima devo jogar fora.

Voltas: Numa competição com duas voltas, o calendário da 2ª volta é idêntico ao da 1ª, mantendo-se a ordem dos jogos e alternando a situação casa/fora. Cada equipa joga contra cada uma das restantes equipas uma vez em cada uma das voltas da competição.

(Opcional) Distâncias: Se a competição for realizada em territórios vastos o critério distância deve prevalecer sobre o objetivo alternância casa/fora. Tal pode ser feito se minimizarmos as deslocações de uma equipa, juntando os jogos a realizar fora num dado local em jornadas consecutivas.

Balanceamento: não pode haver mais do que X clubes grandes a jogar em casa numa dada jornada.

Iremos ter de transformar estas condições para restrições de modo a construir este calendário.

3 Ficheiros de Dados

Foram criados três ficheiros com os seguintes campos: numeroDeEquipas, numeroDeVoltas, escolha, equipa1, equipa2, jornadamin, jornadamax. Escolha pode ser *1* ou *0*, dependendo se se quer aplicar a restrição ou não. Isto possibilitou fazer testes mais rápidos e robustos, dado que não é necessário pedir ao utilizador pelos valores (modo interativo) nem depender de uma só fonte de testes. Outra vantagem é um utilizador poder inserir diferentes dados sem ter um compilador PROLOG, algo que não seria possível se (só) houvesse valores hard-coded.

4 Variáveis de Decisão

Foram criadas duas listas uma para guardar todos os jogos, Dist, e outra para a situação casa/fora, HomeAway. Dist e HomeAway tem tamanho NúmeroDeEquipas * NúmeroDeRondas. O domínio inicial de Dist vai de 1..4 e HomeAway de 0..1. As longo do código vão sendo introduzidas diversas restrições que fazem com que o domínio vá sendo reduzido. Por exemplo, temos regras que fazem com que numa jornada não posso haver mais que um jogo por equipa, uma equipa não pode jogar contra si mesma, numa volta uma equipa tem de jogar contra todas as outras, etc.

5 Restrições

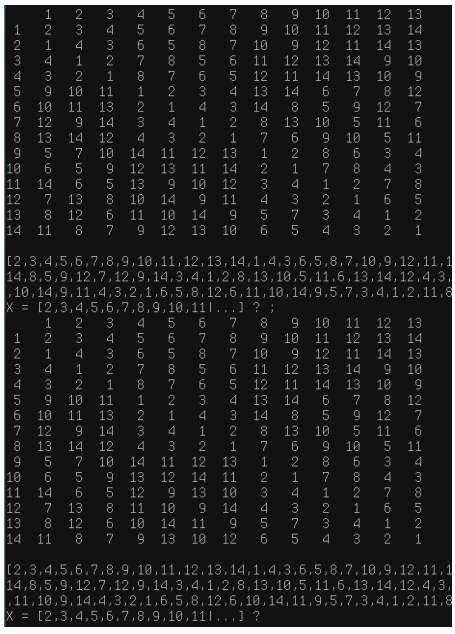
A única restrição flexível do nosso problema é a das jornadas mínima e máxima para a ocorrência de um jogo entre determinadas equipas, o valor dessas equipas deve estar entre 1 e o numero de equipas. Caso a escolha seja *0* então esta restrição não é imposta no nosso problema. No nosso problema temos diversas restrições fixas: restrições para os jogos das equipas, restrições para as jornadas, e para cada jogo. ConstraintForTeam, ConstraintForRound e ConstraintForGame. Foram criados vários predicados para ir buscar as variáveis que nos interessavam de Vars(que se passa para labeling) para depois se aplicarem as restrições necessárias. Por exemplo: temos um predicado chamado getTeamList, cujo objetivo é ir buscar a sublista das equipas que jogam contra uma dada equipa; getRoundList, vai buscar a lista das equipas que jogam na jornada n. Após irmos buscar essas listas aplicamos certas restrições. ConstraintForTeam, para uma equipa N, a lista das equipas com quem ela joga não pode ter equipas repetidas nem pode conter a equipa n. ConstraintForRound, todos os valores têm de ser diferentes visto que uma equipa não pode jogar mais que uma vez numa jornada. ConstraintForGame, numa dada jornada se uma equipa X joga contra uma equipa Y, Y também joga contra X.

6 Estratégia de Pesquisa

A técnica de labeling utilizada é a padrão. Instanciação lado esquerdo, atualização do arco, instanciação, etc.E caso haja falhanço é usado um mecanismo tipo Backtrack. Isto é feito até chegar-se a uma solução, o processo pode ser repetido se invocarmos um redo ; . É passado para o labeling uma lista chamada Vars, que é constituída por duas listas Dist e HomeAway. Na lista Dist os jogos são guardados da seguinte maneira, [JogosEquipa1,...,JogosEquipaN]. JogosEquipa1 possui as equipas com que a equipa1 vai jogar da ronda1 até à rondaN = Nequipas - 1. Possibilidades = NEquipas*(NEquipas - 1), Rondas = Possibilidades / Nequipas = (Nequipas - 1). Antes do labeling é feito append da lista Dist com a lista HomeAway. Dlist e HomeAway são duas listas simples, isto é, cada uma delas só possui inteiros.

7 Visualização da Solução

Cada solução possível é imprimida para o ecrã, após o labeling. Isto é feito através da regra, print(Vars,NTeams,NRounds,1). Aqui se encontram dois exemplos de como é feita a impressão na tela. A primeira linha vai de NEquipas até NEquipas-1, cada valor corresponde a uma jornada. Todos os valores a baixo desta linha são as equipas. Se pretendermos ver quem joga contra quem na 1ª jornada, basta olhar para a coluna com o número 1(jornada 1) e comparar os valores dessa coluna com a 1ª coluna. Podemos observar que se a equipa X joga contra Y o contrário também sucede.



```
| ?- start(12,2,X).
    1   2   3   4   5   6   7   8   9  10  11
 1   2   3   4   5   6   7   8   9  10  11  12
 2   1   4   3   6   5   8   7  10   9  12  11
 3   4   1   2   7   8   5   6  11  12   9  10
 4   3   2   1   8   7   9  10  12  11   5   6
 5   9  10  11   1   2   3  12   6   7   4   8
 6  10   9  12   2   1  11   3   5   8   7   4
 7  11  12  10   3   4   1   2   8   5   6   9
 8  12  11   9   4   3   2   1   7   6  10   5
 9   5   6   8  10  12   4  11   1   2   3   7
10   6   5   7   9  11  12   4   2   1   8   3
11   7   8   5  12  10   6   9   3   4   1   2
12   8   7   6  11   9  10   5   4   3   2   1

1 round
    1   2   3   4   5   6   7   8   9  10  11
 1   H   H   H   H   H   H   H   H   H   H   H
 2   A   H   H   H   H   H   H   H   H   H   H
 3   H   A   A   H   H   H   H   H   H   H   H
 4   A   A   A   H   H   H   H   H   H   H   H
 5   H   H   H   A   A   A   H   H   H   A   H
 6   H   H   H   A   A   H   A   A   H   H   A
 7   H   H   H   A   A   A   A   H   A   A   H
 8   H   H   H   A   A   A   A   A   A   H   A
 9   A   A   A   H   H   A   H   A   A   A   A
10   A   A   A   A   H   H   A   A   A   A   A
11   A   A   A   H   A   A   A   A   A   A   A
12   A   A   A   A   A   A   A   A   A   A   A

2 round
    1   2   3   4   5   6   7   8   9  10  11
 1   A   A   A   A   A   A   A   A   A   A   A
 2   H   A   A   A   A   A   A   A   A   A   A
 3   A   H   H   A   A   A   A   A   A   A   A
 4   H   H   H   A   A   A   A   A   A   A   A
 5   A   A   A   H   H   H   A   A   A   H   A
 6   A   A   A   H   H   A   H   H   A   A   H
 7   A   A   A   H   H   H   H   A   H   H   A
 8   A   A   A   H   H   H   H   H   H   A   H
 9   H   H   H   A   A   H   A   H   H   H   H
10   H   H   H   H   A   A   H   H   H   H   H
11   H   H   H   A   H   H   H   H   H   H   H
12   H   H   H   H   H   H   H   H   H   H   H
```

8 Resultados

A procura de solução até 20 equipas é bastante rápida. No entanto, para valores superiores o tempo de espera é enorme. 4 - 8.4 centésimas de segundo; 8 - 8.5 centésimas de segundo; 16 - 2.48 décimas de sugundo. O tempo foi contado através do comando time do linux, time /opt/sicstus4.2.3/bin/sicstus -l sport.pdb.

9 Conclusões e Perspectivas de Desenvolvimento