Ano Escolar

Resolução de Problemas de Otimização utilizando Programação em Lógica com Restrições

João Barbosa and José Martins

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

Resumo Resolução de um problema de otimização referente a calendarização de testes e trabalhos de casa numa escola utilizando programação em lógica com restrições em SICStus Prolog , permitindo variar o numero de turmas, disciplinas, trabalhos de casa por dia e por disciplina assim como os horários de cada turma. Foi implementado um conjunto de predicados de modo a aplicar todas as restrições pertinentes para uma boa resolução do problema em questão.

Deve contextualizar e resumir o trabalho, salientando o objetivo, o método utilizado e fazendo referência aos principais resultados e à principal conclusão que esses resultados permitem obter.

Keywords: computational geometry, graph theory, Hamilton cycles

1 Introdução

Este projeto esta a ser realizado no âmbito da unidade curricular Programação em lógica do Mestrado Integrado em Engenharia Informática e Computação. O objetivo do projeto prende-se com a resolução de um problema de otimização utilizando programação em lógica com restrições. O problema consiste na calendarização de testes e trabalhos de casa de um periodo letivo, com vários parametros váriaveis como o numero de turmas, disciplinas, trabalhos de casa (por dia e por disciplina) assim como a existencia de diferentes horários para cada turma, existem também diversas restrições que devem ser respeitadas. Ao longo deste artigo será descrito o respetivo problema com um nivel de detalhe mais elevado, a abordagem levada a cabo pelo grupo para a implementação de uma possivel solução assim como as conclusões retiradas do solução implementada.

Descrição dos objetivos e motivação do trabalho, referência sucinta ao problema em análise (idealmente, referência a outros trabalhos sobre o mesmo problema e sua abordagem), e descrição sucinta da estrutura do resto do artigo.

2 Descrição do Problema

O Problema "Ano Escolar" é referente á maracação de testes e tpc ao longo do periodo sendo que as seguintes restrições devem ser tidas em conta na resolução apresentada:

- Cada disciplina tem 2 testes por período de aulas, que decorrem num conjunto de semanas específico (mais ou menos a meio e no fim do período).
- Os alunos não podem ter mais do que 2 testes na mesma semana de aulas, nem testes em dias consecutivos.
- Os testes realizados pelas diferentes turmas a uma mesma disciplina devem ser o mais próximos possível.
- Em cada dia, não pode haver TPC em mais do que 2 disciplinas.
- Em pelo menos um dia por semana (que deve ser sempre o mesmo ao longo do período), não pode haver TPC.
- Em cada disciplina, só pode haver TPC em metade das aulas.

O resultado deve incluir as datas dos testes de cada turma/disciplina, bem como os dias em que o professor de cada disciplina pode mandar trabalho para casa. Deve ser possível resolver problemas desta classe com diferentes parâmetros, como por exemplo variando o número de turmas e disciplinas, horários, número máximo de TPC por disciplina e por dia entre outros.

Descrever com detalhe o problema de otimização ou decisão em análise.

3 Abordagem

A abordagem para o problema em questão consistiu inicialmente na avaliação das variáveis de decisão necessárias para a resolução do problema e por fim na elaboração de uma estrutura que permitisse a imposição de restrições de um modo simples e eficiente. Depois decidiu-se de forma a separar conceitos elaborar predicados que resolvem o problema apenas para uma turma que depois são utilizados para resolver o problema geral.

3.1 Variáveis de Decisão

A estrutura adoptada pelo grupo para resolver o problema consiste numa lista para cada turma com tamanho igual ao numero de disciplinas da turma, sendo que cada elemento dessa mesma lista é outra lista com 3 variaveis:

- Identificador da disciplina - Variavel já instanciada(conhecida pelo problema);
- Lista de testes - Lista com o tamanho igual ao numero de dias uteis do periodo, em que cada elemento da lista se trata de uma variavel de decisão com dominio de 0 a 1 simbolizando a existencia(1) ou não(0) de teste a disciplina em questão.
- Lista de trabalhos de casa - Lista com o tamanho igual ao numero de dias uteis do periodo, em que cada elemento da lista se trata de uma variavel de decisão com dominio de 0 a 1 simbolizando a existencia(1) ou não(0) de tabalhos de casa a disciplina em questão.

Descrever as variáveis de decisão e os seus domínios.

3.2 Restrições

Neste tópico serão abordadas as restrições do problema de otimização já mencionadas no ponto 2 com detalhes sobre a sua implementação.

1. **Garantir dois testes por período para cada disciplina, mais ou menos a meio e no fim do período.**

   Para garantir o que o numero de testes por periodo para cada disciplina fosse igual a 2 e que cada teste estivesse numa época diferente (meio do periodo ou fim do periodo), foi utilizado o predicado *twoTestsPerPeriod(+Class)* que recebe uma turma com a estrutura especificada no ponto **3.1** e para cada disciplina dessa turma utiliza a lista de variaveis de decisão que contem a informação dos testes, obtendo depois duas sublistas a partir desta uma de 1/6 a 3/6 o numero de dias e outra de 4/6 a 6/6, que representam então as duas épocas de testes. Sendo depois garantida que em cada uma das épocas a soma dos elementos de cada lista é igual a 1 desta forma, exige-se a existencia de um teste em cada época de testes num total de dois testes por periodo.

2. **Máximo de 2 testes na mesma semana de aulas, e não permitir testes em dias consecutivos.**

   Para solucionar esta restrição foi utilizado o predicado *testPlacementRestrictions(+Days,+Class)* que recebe mais uma vez uma turma e numero de dias do periodo. Este predicado obtém numa primeira instancia uma lista que contém o numero total de testes em cada dia do periodo para a turma em questão (tendo em consideração todas as disciplinas), de seguida, verifica com ajuda de predicados auxiliares que a soma dos testes em 2 dias seguidos tem que ser inferior ou igual a 1, impedindo assim a existencia de testes em dias seguidos e de vários testes no mesmo dia(foi tambem tido em consideração os fins-de-semanas, sendo então possivel a existencia de testes na sexta-feira e na segunda-feira).

3. **Os testes realizados pelas diferentes turmas a uma mesma disciplina devem ser o mais próximos possível.**

   De modo a permitir que os testes da mesma disciplina fossem o mais proximos possiveis em turmas diferentes foi utilizado o predicado *testsCloseBetweenClasses(+Classes, +DisciplineIds, -Sum1, -Sum2)* , que recebe todas as turmas e os identificadores de todas as disciplinas lecionadas nas turmas em questão, retornando o somatório da diferença entre os testes da mesma disciplina em turmas diferentes para a primeira época de testes(*Sum1*) e para a segunda (*Sum2*), depois nas opções do labeling é feita a minimização destas variaveis utilizando *minimize(Sum1) , minimize(Sum2)* na lista de opções do labeling.

4. **Em cada dia, não pode haver TPC em mais do que N disciplinas.**

   A fim de permitir ao utilizador especificar o numero máximo de trabalhos de casa por dia, foi implementado o predicado *maxNumberTpcPerDay(+Class,+Days,+N)* que em semelhança á restrição numero 2, obtem uma lista com o numero total de tpc's em cada dia do periodo para a turma em questão (tendo em consideração todas as disciplinas), sucedendo-se a seguir para cada elemento desta lista, uma comparação de modo a garantir que o número de tpc's será inferior a N uma vez instanciadas as variaveis de decisão.

5. **Em pelo menos um dia por semana (que deve ser sempre o mesmo ao longo do período), não pode haver TPC.**

   No sentido da implementação desta restrição foi criado o predicado *clearTpcDay(Class,NoTpcDay)* que recebe o dia em que não existem trabalhos de casa e depois percorre a lista de disciplinas da turma de forma a que na lista com as variáveis de decisão que dizem respeito aos tpc's, sejam instanciadas(com 0) de forma a indicar a não existencia de tpc's todas as variaveis de decisão

que dizem respeito ao dia da semana especificado nos parâmetros do predicado.

6. **Em cada disciplina, só pode haver TPC numa percentagem das aulas.**

   Em relação a esta restrição foi concebido o predicado *limitNumberOfTpcPerPeriod(+Class,+Ratio,+Schedule,+Days)* que recebe a razão especificada pelo utilizador (ex: 2 - metade, 3 - um terço, 4 - um quarto) e numa primeira fase é obtido o numero total de aulas de uma disciplina ao longo do período sendo que depois a soma das variaveis de decisão referentes aos trabalhos de casa é limitada de modo a ser inferior ou igual a $1/Ratio$, este processo é de seguida repetido para todas as disciplinas da turma.

3.3 Função de Avaliação

Na solução encontrada para o problema, um dos factores a ter em conta é ser necessário que os testes entre turmas, para uma dada disciplina, sejam o mais próximos entre si o quanto possível.

Para garantir esta condição, são calculadas as diferenças das datas de teste entre cada par de turmas, sendo estes valores também somados no final. Como há varios testes de disciplina, estas somas também terão de ser somadas - estamos, portanto, perante uma soma de somas de somas. Este valor deve ser minimizado, quando a solução está a ser procurada.

Para testar a eficiência e os resultados do programa, foi criado um predicado *run*, que calcula o tempo de execução do programa.

3.4 Estratégia de Pesquisa

A estratégia de etiquetagem passa por utilizar o predicado de *labeling* com a seguinte estrutura:

labeling([ff, down, minimize(Sum), time_out([90000, _])], R)

Em que

- Sum - Soma das somas das somas entre a diferença de dias de teste para cada disciplina, entre cada turma.
- R - lista de variáveis de domínio, a serem instanciadas.

A opção ***ff*** melhora a eficiência global do programa, utilizando a estratégia de *First-Fail*.

Utilizando ***down***, a instanciação das variáveis ocorre por ordem descendente do seu intervalo de domínio. Esta opção permite, de forma simples e eficiente, garantir que são colocados o número máximo possível de TPCs para as turmas,

que obedeçam às restrições dadas. Note-se que outra solução seria usar a opção *maximize*, para a soma de todos os TPCs, mas essa estratégia seria mais lenta, oferecendo os mesmos resultados.

O ***minimize***, para a variável *Sum*, garante que os testes, entre turmas, estão o mais próximos entre si, o quanto possível.

Finalmente, o ***time_out*** funciona como uma salvaguarda, no sentido em que para o programa, quando 1 minuto e 30 segundos estiverem decorridos, desde o início da execução. Caso decorra o tempo estipulado, então será retornada a melhor solução encontrada até então.

4 Visualização da Solução

Explicar os predicados que permitem visualizar a solução em modo de texto

5 Resultados

Demonstrar exemplos de aplicação em instâncias do problema com diferentes complexidades e analisar os resultados obtidos. Devem ser utilizadas formas convenientes para apresentação dos resultados (tabelas e/ou gráficos).

6 Conclusões e Trabalho Futuro

Que conclusões retira deste projeto? O que mostram os resultados obtidos? Quais as vantagens e limitações da solução proposta? Como poderia melhorar o trabalho desenvolvido?

Figura 1. This is the caption of the figure displaying a white eagle and a white horse on a snow field

$$\begin{aligned} \dot{x} &= JH'(x) \\ x(0) &= x(T) \end{aligned} \tag{1}$$

Proposition 1. *Assume* $H'(0) = 0$ *and* $H(0) = 0$. *Set:*

$$\delta := \liminf_{x \to 0} 2N(x) \left\| x \right\|^{-2} \ . \tag{2}$$

If $\gamma < -\lambda < \delta$, *the solution* $\overline{u}$ *is non-zero:*

$$\overline{x}(t) \neq 0 \quad \forall t \ . \tag{3}$$

Tabela 1. This is the example table taken out of *The TEXbook,* p. 246

Year	World population
8000 B.C.	5,000,000
50 A.D.	200,000,000
1650 A.D.	500,000,000
1945 A.D.	2,300,000,000
1980 A.D.	4,400,000,000

Notes and Comments. The results in this section are a refined version of [**?**]; the minimality result of Proposition 14 was the first of its kind.

To understand the nontriviality conditions, such as the one in formula (**??**), one may think of a one-parameter family x_T, $T \in \left(2\pi\omega^{-1}, 2\pi b_\infty^{-1}\right)$ of periodic solutions, $x_T(0) = x_T(T)$, with x_T going away to infinity when $T \to 2\pi\omega^{-1}$, which is the period of the linearized system at 0.

Referências

1. SWI-Prolog, http://www.swi-prolog.org
2. SICStus-Prolog, https://sicstus.sics.se

Anexo

Código fonte

Bla Bla