Ano Escolar

Resolução de Problemas de Otimização utilizando Programação em Lógica com Restrições

José Oliveira (up201406208) e Manuel Gomes (up201402679)

FEUP-PLOG, Turma 3MIEIC3, Grupo AnoEscolar3

Resumo No âmbito da unidade curricular de Programação em Lógica desenvolveu-se um programa capaz de gerar a calendarização das atividades avaliativas de uma turma, recorrendo-se à Programação em Lógica com Restrições com auxílio do software ***SICStus Prolog***. Com efeito, e tendo-se em conta as várias restrições impostas para se gerar uma possível solução do problema, implementaram-se vários predicados de forma a se construir um programa o mais eficiente possível. Através da manipulação dos vários predicados construidos, assim como os pertencentes ao ***SICStus Prolog***, conseguiu-se não só resolver o problema em específico, mas também de uma forma alargada, com calendarização de várias turmas em simultâneo.

Palavras-chave: Prolog, Restrições, Ano Escolar

1 Introdução

No âmbito da unidade curricular de Programação em Lógica pretende-se desenvolver um programa capaz de gerar a calendarização das atividades avaliativas de uma turma, recorrendo-se à Programação em Lógica com Restrições com auxílio do software ***SICStus Prolog***. Este problema consiste na calendarização de um período escolar escolar de uma turma mediante diversas restrições ao nível dos trabalhos de casa (TPCs) assim como dos testes. Por outro lado pretende-se que a solução seja eficiente, pelo que se pretende que a mesma seja extensível a mais do que uma turma, assim como não só a uma variada duração do período escolar, como também diferentes valores para as restrições pretendidas. O presente artigo expõe a estratégia utilizada para resolução do problema, assim como os resultados com ela obtidos.

2 Descrição do Problema

O problema em análise é referente à calendarização de um período escolar, com agendamento dos trabalhos de casa (TPC), assim como as datas dos testes. Para resolução do mesmo são implementadas as seguintes restrições:

1. Em pelo menos um dia por semana, que deve ser o mesmo ao longo do período, não pode haver TPC;

2. Cada dia não pode ter TPC em mais do que *NumMaxTPCDay* disciplinas;
3. Em cada disciplina só pode haver TPC no máximo em metade das aulas;
4. 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);
5. Os estudantes não podem ter mais do que *NumMaxTestsWeek* testes per semana);
6. Os estudantes não podem ter mais do que um teste em dias consecutivos;
7. Os testes realizados pelas diferentes turmas a uma mesma disciplina devem ser o mais próximos possível.
   Deve ser possível resolver o problema como um problema de satisfação de restrições, de modo a que seja possível resolver problemas desta classe com diferentes parâmetros, isto é, fazendo variar o número de turmas e disciplinas, horários, número máximo de TPC por disciplina e por dia, entre outros.

3 Abordagem

Por forma a se desenvolver o método o mais adequado à resolução do problema, começou-se por tentar modelar o calendário de um ano escolar como um problema de restrições. Findo este processo, procurou-se entender as variáveis de decisão a utilizar, assim como a forma mais correta de restringir as mesmas, de modo a tornar a solução o mais eficiente quanto possível.

3.1 Variáveis de Decisão

Para estruturação do problema em análise, e assim para uma mais fácil legibilidade do mesmo, optou-se por construir para cada turma uma lista de listas, sendo que cada lista corresponde a uma diferente disciplina. Cada lista de uma disciplina possui um identificador da disciplina, assim como uma lista com os TPC's e outra com os Testes, de tamanho igual ao número de dias do do período. As variáveis de decisão presentes em cada uma das listas podem assumir o valor de 0 (*falso*), ou de 1 (*verdadeiro*), caso se encontre uma ocorrência de um TPC ou teste.

3.2 Restrições

1. **Em pelo menos um dia por semana, que deve ser o mesmo ao longo do período, não pode haver TPC:**

   De modo a se implementar esta restrição, foi desenvolvido o predicado *freeTPC(+FreeTPCDay, +Class)*, que recebe o dia em que não deve haver TPC, podendo o mesmo oscilar entre 1 e 5 (representativo de um dia da semana), percorrendo a lista de TPC's de cada uma das disciplinas e instanciando as variáveis de decisão correspondentes com o valor de 0 (*falso*).

2. **Cada dia não pode ter TPC em mais do que *NumMaxTPCDay* disciplinas:**

   Para esta restrição foi implementado o predicao *maxNumTPCDay(+Day, +Days, +Class, +NumMaxTPCDay)*, que recebe o número máximo de TPC's para cada dia representado por *NumMaxTPCDay*. São percorridas as listas de TPC's de cada disciplina e criadas uma lista para cada dia com as variáveis de decisão correspondentes. Por fim, é colocada a restrição de que o somatório de cada uma dessas listas deve ser menor ou igual a *NumMaxTPCDay*.

3. **Em cada disciplina só pode haver TPC no máximo em metade das aulas:**

   Nesta restrição foi implementado o predicado *maxTPCPeriod(+Days, +RatioTPC, +Schedule, +Class)*, o qual recebe a percentagem de máxima das aulas em que pode haver TPC. De modo a implementar esta restrição é determinado primeiramente o número de aulas de cada disciplina ao longo do período escolar, a partir do qual, e utilizando-se *RatioTPC*, é calculado o número de dias máximo para cada disciplina em que pode haver TPC. Por fim é restringindo-se o somatório da lista de TPC's de cada disciplina, de modo a que este seja menor ou igual ao número máximo de TPC's por período respetivo.

4. **Cada disciplina tem dois testes por período de aulas, que decorrem num conjunto de semanas específico (mais ou menos a meio e no fim do período):**

   Para resolução desta restrição foram utilizados dois predicados. Para a restrição de haverem dois testes por período de aulas foi utilizado o predicado *twoTestsDisciplinePeriod(+Class)*, o qual para cada lista de testes de cada disciplina, restringe o valor da soma dos elementos para que seja igual a 2. Por outro lado, e de modo a solucionar a segunda parte da restrição, foi implementada a restrição *twoDifferentTestMoments(+StartFirstTestMoment, +TestMomentDuration, +StartSecondTestMoment, +Class*, a qual começa primeiro por determinar o começo e duração de cada um dos dois momentos de avaliação. Para tal, é obtido o quociente entre o número de dias correspondente à duração do período escolar e o valor de 4. Este valor de quociente não só corresponderá à duração de cada momento de avaliação, assim como também será utilizado para o cálculo do começo de cada avaliação, sendo que o primeiro período de avaliação ocorrerá durante o segundo quarto do período e o segundo momento de avaliação o último quarto do período.

5. **Os estudantes não podem ter mais do que *NumMaxTestsWeek* testes per semana):**

   De modo a se implementar esta restrição foi desenvolvido o predicado *max-*

TestsPerWeek(+Day, +Days, +NumMaxTestsWeek, +Class), o qual recebe o número máximo possível de testes por semana *NumMaxTestsWeek*. Neste predicado é gerada uma lista para cada semana, com as variáveis de decisão representativas dos dias, a partir de cada uma das listas de testes de cada disciplina. Por fim, restringe-se a que o somatório de cada uma destas listas (semanas) seja menor ou igual a *NumMaxTestsWeek*.

6. **Os estudantes não podem ter mais do que um teste em dias consecutivos:**

   Para a implementação deste predicado foi desenvolvido o predicado *testsConsecutiveDays(+Day, +Days, +Class)*, o qual em cada iteração cria duas listas, cada uma das quais com as variáveis de decisão correspondentes a dois dias diferentes e consecutivos, até que todos os dias do período sejam percorridos. Por fim é restringido o somatório das duas para que seja menor ou igual a um. Note-se que caso os dois dias consecutivos se tratem de Sexta-feira e Segunda-feira, será possível ter teste nestes dois dias, uma vez que existe o fim-de-semana entre os mesmos.

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

   Para esta restrição é usado o predicado *testsCloseDays(+Classes, +Test1, +Test2)*, o qual para cada diferente disciplina, tenta restringir a diferença de dias de um teste de uma mesma disciplina, para turmas diferentes, sendo esse valor representado por *Test1* e *Test2*. Estes valores serão depois minimizados através da estratégia de etiquetagem utilizada com o uso da opção *minimize*.

3.3 Função de Avaliação

Para testar e avaliar a solução obtida é usado o predicado *run(+Days, +FreeTPCDay, +NumMaxTPCDay, +RatioTPC, +NumMaxTestsWeek, +ClassResult)*, o qual verifica se o tempo de execução do programa. Por outro lado é neste predicado que é efetuada a restrição que minimiza o tempo entre testes de várias turmas de uma mesma disciplina.

3.4 Estratégia de Pesquisa

Em termos de processo de etiquetagem foi usado o predicado *labeling([ffc, down, minimize(Test1), minimize(Test2), step, time_out(120000, _], Result)*, em que *Result* corresponde a todas as varáveis a serem instanciadas. É utilizada a estratégia de *first-fail* através de *ffc* em que as variáveis que participam em mais restrições são etiquetadas primeiro. Por outro lado, através de *down* os elementos de domínio são etiquetados por ordem descendente do seu intervalo de domínio. Em relação aos *minimize(Test1)* e *minimize(Test2)*, os mesmos são usados de

modo a que a solução obtida para as datas dos testes, seja a mais próxima possível para testes de uma mesma disciplina. Por fim, a estratégia de *branching* utilizada foi a estratégia padrão do predicado *labeling*, ou seja, foi utilizada a opção *step*.

4 Visualização da Solução

Para se efetuar a visualização do resultado é utilizado o predicado *displayResults(+Days, +Schedules, +ClassResults)*, o qual recebe as listas com os diferentes horários, assim como os resultados para as distribuições dos TPC's e testes para cada turma.
Este predicado começa por imprimir em modo de texto o horário da turma respetiva através do predicado *displaySchedule(+Day, +Schedule)*, o qual utiliza a variável *Day* instanciada com o valor zero, de modo a ser usada para que se possa imprimir o nome do dia da semana corretamente. Assim, e de forma recursiva, até a lista *Schedule* estar vazia, uma nova variável é instanciada sendo igual ao valor de *Day*, acrescido de uma unidade.

Por fim, é usado o predicado *displayTPCsAndTests(+StatingDay, +EndDay, +ClassResult)*, o qual recebendo os dias inicial e final do período, imprime a calendarização de TPC's e testes para cada turma. Neste predicado, para cada dia do período, é percorrida a lista de disciplinas, sendo usado o predicado *nth0/3* de modo a que, usando o dia do período em questão como índice para as respetivas listas de TPC's e testes, se verifica se para aquele determinado dia existe algum destas ocorrências. Em caso afirmativo é então imprimido o nome da disciplina seguido do tipo de ocorrência.

5 Resultados

Para a obtenção dos resultados, foram efetuadas diversas simulações, com diferentes valores de turmas, disciplinas, assim como duração do período escolar. Deste modo apresentam-se os resultados obtido para 3 e 6 turmas com 3 disciplinas, e fazendo-se variar a duração do período escolar.

Como seria de esperar, com o aumento da duração do período escolar, assim como do número de turmas o processamento do resultado é mais demorado. No entanto, verifica-se que com o aumento do número de turmas, o aumento do tempo de processamento é mais rápido do que com o aumento do número de turmas.
Por outro lado, testou-se também com um número variável de disciplinas, tendo-se experimentado o valor de 10 disciplinas. No entanto, não foi possível obter qualquer solulão dentro do tempo de *time_out* de 120 segundos, pelo que assim se pode verificar que esta variável, a par do número de turmas, são as que mais influênciam a rapidez de obtenção de uma solução para o programa desenvolvido.

```
######################## Horario ########################

Segunda: Ed.Fisica, Portugues
Terca: Ed.Fisica, Portugues, Tecnologias
Quarta: Ed.Fisica, Portugues, Tecnologias
Quinta: Portugues, Tecnologias
Sexta: Ed.Fisica, Tecnologias

######################## TPCs e Testes ########################

Dia - 1 - Segunda:
Dia - 2 - Terca: TPC-Ed.Fisica  TPC-Tecnologias
Dia - 3 - Quarta: TPC-Ed.Fisica  TPC-Tecnologias
Dia - 4 - Quinta: TPC-Portugues  TPC-Tecnologias
Dia - 5 - Sexta: TPC-Ed.Fisica  TPC-Tecnologias
Dia - 6 - Segunda:
Dia - 7 - Terca: TPC-Ed.Fisica  TPC-Tecnologias
Dia - 8 - Quarta: TPC-Ed.Fisica  TPC-Tecnologias  TESTE-Tecnologias
Dia - 9 - Quinta: TPC-Portugues  TPC-Tecnologias
Dia - 10 - Sexta: TPC-Ed.Fisica  TPC-Tecnologias
Dia - 11 - Segunda:
Dia - 12 - Terca: TPC-Ed.Fisica  TESTE-Ed.Fisica  TPC-Tecnologias
Dia - 13 - Quarta: TPC-Ed.Fisica  TPC-Tecnologias
Dia - 14 - Quinta: TPC-Portugues  TESTE-Portugues  TPC-Tecnologias
Dia - 15 - Sexta: TPC-Ed.Fisica  TPC-Tecnologias
Dia - 16 - Segunda:
Dia - 17 - Terca: TPC-Portugues  TPC-Ed.Fisica
Dia - 18 - Quarta: TPC-Portugues  TPC-Ed.Fisica
Dia - 19 - Quinta: TPC-Portugues
Dia - 20 - Sexta: TPC-Ed.Fisica
Dia - 21 - Segunda:
Dia - 22 - Terca: TPC-Portugues  TESTE-Tecnologias
Dia - 23 - Quarta: TPC-Portugues
Dia - 24 - Quinta: TPC-Portugues  TESTE-Portugues
Dia - 25 - Sexta:
Dia - 26 - Segunda:
Dia - 27 - Terca: TPC-Portugues  TESTE-Ed.Fisica
Dia - 28 - Quarta: TPC-Portugues
Dia - 29 - Quinta: TPC-Portugues
Dia - 30 - Sexta:
```

Figura 1. Exemplo de um resultado de visualização para uma turma

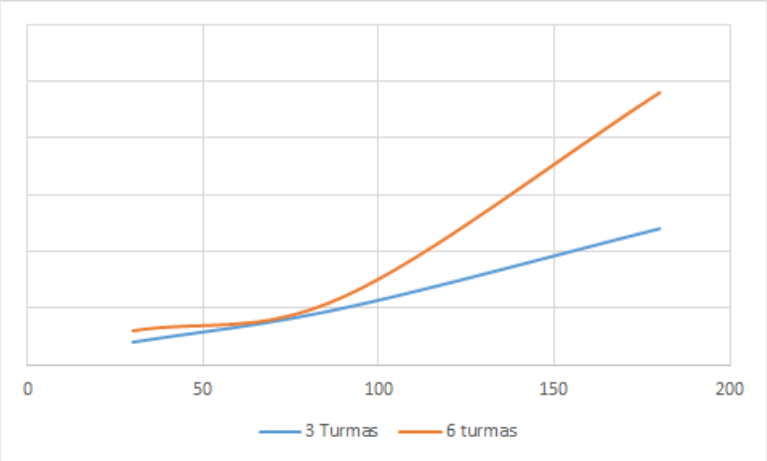

Figura 2. Resultados obtidos para 3 turmas

6 Conclusões e Trabalho Futuro

Com a realização deste trabalho verifica-se que a linguagem *Prolog*, mais especificamente os seus módulos de resolução com restrições, é bastante poderosa para resolver uma ampla variedade de problemas de decisão e/ou otimização, tal como fui possível constatar com o seu predicado *labeling*.
Em relação à solução obtida para o problema proposto, consideramos que a mesma atingiu de forma satisfatória os objetivos propostos, implementando não só todas as restrições pedidas, como também apresentando os resultados de forma eficiente.
No que diz aos resultados obtidos, verifica-se que o principal fator que leva a um maior tempo de processamento de uma solução é o aumento do número de disciplinas e/ou turmas, pelo que seria neste âmbito que se poderia tentar tornar a solução ainda mais eficiente. Por outro lado, a implementação de novas restrições de forma a tornar o programa desenvolvido ainda mais completo, seria um outro ponto poder ser desenvolvido em iterações futuras.

Referências

1. Vários autores, SWI-Prolog,
   http://www.swi-prolog.org

2. Vários autores, The Prolog Library,
   https://sicstus.sics.se/sicstus/docs/4.3.0/html/sicstus/The-Prolog-Library.html

3. Pierre Deransart, AbdelAli Ed-Dbali, Laurent Cervoni, Prolog The Standard: Reference Manual, Springer, 2007

Anexo

A totalidade do código desenvolvido encontra-se na pasta *codigo*.