A constraint programming approach to a construction team scheduling problem

David Luís Dias da Silva and Luís Pedro Pereira Lopes Mascarenhas Cunha

FEUP-PLOG, Turma 3MIEIC04. Grupo Equipas de Construção_1
Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias,
4200-465 Porto, Portugal
{up201705373,up201706736}@fe.up.pt

Abstract. Neste projeto propomos utilzar programação por restrições, utilizando a biblioteca *clfpd* do Prolog, para resolver um problema de escalonamento aplicado a uma equipa de construção. O objetivo é o de encontrar uma distribuição temporal de operações e de trabalhadores alocados a essas operações que maximize o lucro da equipa. Neste trabalho descrevemos o problema em detalhe e apresentamos a nossa solução para esse problema utilizando restrições. Nas partes finais do relatório é discutida a viabilidade da solução encontrada e sobre os métodos utilizados para otimizar a mesma.

Keywords: Programação por restrições · Programação em lógica · Escalonamento de construções · Problema de otimização · Programação por restrições sobre domínios finitos.

1 Introdução

Na indústria e na sociedade em geral é frequente o aparecimento de problemas de escalonamento. Estes problemas são , tipicamente, **NP-completos** pelo que a sua resolução não é trivial e de elevado custo computacional. Neste sentido a programação por restrições surge como uma alternativa dado à sua aplicabilidade em problemas de optimização combinatória.

Neste projecto pretendemos aplicar a programação por restrições, utilizando a biblioteca de programação com restrições em domínios finitos do *Prolog* (*clfpd*) à resolução do problema de alocação de trabalhadores e escalonamento de tarefas de uma equipa de construção. Esta aplicação é feita com o sentido de maximizar o lucro obtido nessa alocação e escalonamento.

Inicialmente, iremos descrever o problema e explicar quais os domínios e restrições associadas às variáveis de decisão. De seguida iremos tratar da faceta de optimização de problema abordando a função de avaliação e iremos discutir quais os resultados obtidos, bem como tecer conclusões sobre esses resultados e sobre as configurações possíveis para a pesquisa de soluções

2 Descrição do Problema

O problema abordado neste projecto é o de uma equipa de construção que pretende determinar qual a alocação dos seus trabalhadores, às **obras** com que está comprometida, que levará à obtenção de um **maior lucro**.

Cada obra possui um **preço contratado** e um **prazo limite**. Cada obra possui também um valor de **bónus** (por dia) que é acrescentado, ou subtraído, ao seu preço contratado caso a obra seja terminada antes ou após o prazo estabelecido. É de notar que existem duas abordagens para prazo estabelecido, sendo que podemos considerar que este prazo representa um número fixo de dias desde um instante inicial igual para todas as obras ou que é independente para cada obra, mais à frente esta dualidade será explorada. Uma obra possui um **conjunto de operações de especialidade** que têm um **custo de materiais e equipamentos** associado. Estas operações podem ter precedências entre si, ou seja, poderá ser necessário completar as tarefas de uma ada especialidade de uma dada obra antes de efectuar as operações de outra especialidade. Uma operação de especialidade tem também uma **duração prevista** (duração, em dias, esperada para apenas um trabalhador) e são-lhe **alocados um ou mais trabalhadores**, no entanto, **pelo menos um deles terá de ter a mesma especialidade da operação**. A **duração** de uma tarefa **diminui proporcionalmente com número de trabalhadores** a ela alocados (com mínimo de um dia).

A equipa de construção possui um **conjunto de trabalhadores** que se caracterizam pela sua **especialidade** e pelo seu **salário**. Um trabalhador poderá ter **uma ou mais especialidades**, ou ser **indiferenciado**, caso em que é esperado que o seu salário seja menor.

Neste projecto considerou-se que os **recursos** humanos, materiais e de equipamentos estão **sempre disponíveis** a serem utilizados. Para além disso ao invés de definir um custo para cada material/equipamento e colocar nas tarefas a lista dos recursos necessários optamos por colocar o custo de cada operação já calculado pois consideramos que o simples cálculo do custo dos recursos não trás uma utilidade relevante ao problema modelado.

3 Abordagem

3.1 Input do problema

Como input do problema são dados um conjunto de especialidades (*Especialidades*) que poderão ser utilizadas nas operações (exemplo: Carpintaria, Pichelaria, Canalização) , uma lista de trabalhadores (*Trabalhadores*) no formato *[salario, especialidades]* onde *especialidades* é vector com o tamanho da lista de especialidades cujos elementos têm o valor de 1 se o trabalhador possui a respectiva especialidade ou 0 caso contrário. As operações são fornecidas no formato *[id, idObra, especialidade, tempo, custoEquipamentos]*, onde *id* é o seu identificador único, *idObra* é o identificador da obra à qual pertence a operação, *especialidade* é o nome da especialidade da operação, *tempo* é a duração esperada para

essa operação e *custoEquipamentos* é o custo total de equipamentos e materiais. AS precedências são fornecidas numa lista com o formato *[especialidadeA-especialidadeB,...]* o que significa que a *especialideA* tem precedência sobre a *especialidadeB*. As obras são fornecidas em listas com o formato *[id, preco, duracao, bonus]* onde *id* é o identificado único da obra, *preco* é o pagamento acordado para a realização da operação, *duracao* é o número de dias acordado entre o inicio e o fim das operações da obra e *bonus* é o valor do bónus recebido ou descontado em casos em que a obra termina antes ou depois do tempo acordado.

3.2 Variáveis de decisão

Para encontrar a solução para o problema existe um conjunto de variáveis cujos valores é necessário conhecer. Sobre cada operação (**Opi**) é necessário saber o dia em que começa (variável **Oi**), o dia em que termina (variável **Ei**), duração (variável **Di**) e número de trabalhadores alocados (variável **Hi**). Para além disso necessitamos saber que trabalhadores estão alocados a uma dada operação, para tal, utilizamos uma matriz (variável **WorkersMatrix**) com dimensões $NTrabalhadores \times$NOperações em que uma dada posição (i,j) tem o valor a *1* se o trabalhador j participou em *Opi* e *0* caso contrário.

As variáveis *Oi* e *Ei* têm o seu domínio compreendido entre 0 (correspondente ao instante inicial) e o valor da soma das durações base das operações, o que corresponde a todas as tarefas serem feitas sequencialmente utilizando um trabalhador para cada. A variável *Di* tem um domínio entre 1 e a duração base de *Opi* e a variável *Hi* possui um domínio entre 1 e o número total de trabalhadores, no entanto, mais à frente vamos descrever outras restrições que são aplicadas a estas variáveis em particular.

Conhecendo estes valores é possível calcular qual o lucro associado às obras (**Profit**).

3.3 Restrições

Tendo as variáveis descritas anteriormente existe um conjunto de restrições que foram impostas. Definiu-se que o valor de fim de uma tarefa (Ei) tem de estar compreendido entre o valor de início da mesma tarefa (Oi) e o valor da soma entre Oi e a duração base dessa tarefa. Para além disso, o valor da duração de uma operação (Di) é definido como o valor da duração base a dividir pelo número de trabalhadores que a ela estão alocados (Hi), de maneira que o valor de Di diminui com o aumento de Hi. Como o valor mínimo de Di é 1, Hi é restringido como sendo inferior a BaseDuration, valor máximo de Di.

Outra restrição aplicada é feita através da restrição global *cumulative* à qual são passados os valores das operações na forma *task(Oi,Di,Ei,Hi,Opi)* e onde o recurso associado a esta restrição é o número de trabalhadores, neste sentido o valor de *limit* é colocado como o número total de trabalhadores. Também é passada uma lista de precedências que são construídas a partir das precedências obtidas como input e que garântem que se existe uma precedência entre uma especialidade A e uma especialidade B então, para uma dada obra, nenhuma

tarefa do tipo B pode começar depois do instante correspondente ao fim das tarefas de tipo A.

Para criar a matriz de alocação de trabalhadores (**WorkersMatrix**), é imposto que a soma dos elementos de cada fila da matriz, a qual representa a lista de trabalhadores que participam numa dada tarefa, tem de ser igual ao valor de Hi, de modo a restringir o número de trabalhadores alocados à tarefa. Também é necessário que pelo menos um dos trabalhadores possua a especialidade da tarefa, isto é conseguido através da criação de um vector com o tamanho igual ao número de especialidades, que tem a 1 a posição correspondente à especialidade da tarefa e a 0 as restantes (**VSpecialtyTask**). Também é criado um vector com o tamanho igual ao número total de trabalhadores que tem a 1 as posições dos trabalhadores que têm a especialidade da tarefa em questão e a zero as restantes (**VSpecialtyWorkers**), o que é feito através da reificação do produto escalar entre o vector de especialidades da tarefa e o vector de especialidades de cada trabalhador . Para garantir a restrição é necessário restringir que o produto escalar de **VSpecialtyWorkers** e a linha da matriz é superior a 0, ou seja, que existe pelo menos um trabalhador alocado que possui a especialidade da tarefa.

A última restrição imposta garante que , num dado instante, não existe nenhum trabalhador alocado a duas tarefas. Esta restrição é garantida com o uso à restrição global *disjoint1* que é aplicada uma vez para cada trabalhador e onde *Lines* contém pares *line(Oi,Di)* com os valores de início das tarefas (Oi) e o valores das durações (Di).

3.4 Função de Avaliação

A função de avaliação corresponde ao cálculo do lucro obtido com a configuração escolhida. Assim a variável *Profit* é definida da seguinte forma:

$$Profit = \sum_{i}^{Obra} (Pagamento_i - CustoRecursos_i) - CustoSalarios \quad (1)$$

Em que *CustoSalarios* é a soma total dos salários dos trabalhadores. O salário total de um trabalhador pode ser obtido somando, para todas as operações nas quais este trabalhou, o valor da duração dessa operação multiplicado pelo seu salário diário. Para cada obra, o valor de *Pagamento* é o custo base dessa obra ajustado consoante o bónus estipulado.

$$Pagamento_i = CustoBase_i + Bonus_i \times (DurEsperada_i - DurEfetiva_i) \quad (2)$$

A duração efectiva de uma obra é o número de dias entre o primeiro dia de uma operação dessa obra e o último. Assim, se a diferença entre a duração efectiva e a duração esperada for negativa o bónus será negativo e , caso contrário, será positivo:

$$DuracaoEfetiva_i = max(Ej, TarefasObra_i) - min(Oj, TarefasObra_i) \quad (3)$$

O objectivo então, é o de encontrar uma configuração para as variáveis que leve a um lucro máximo.

4 Visualização da Solução

De modo a apresentar a solução de forma legível em texto, foram feitos predicados que imprimem o lucro obtido com a solução e as variáveis usados no *labeling* e, para cada obra, informação relativa à mesma e às operações que lhe pertencem em formato tabular. É visualizado, para cada operação, o seu ID, especialidade, duração com 1 trabalhador, custo dos materiais necessários, dias de fim e início, duração, número de trabalhadores que farão parte da obra e os números de identificação de cada trabalhador.
Para além disto, é ainda apresentado para cada trabalhador o seu salário e as operações nas quais ele terá que participar.
Assim, são apresentados todos os valores necessários para escalonar as tarefas pelos trabalhadores, bem como para reconstruir os cálculos que levam ao lucro obtido.

5 Resultados

5.1 Exemplo de pequenas dimensões

A solução apresentada foi testada num problema de pequenas dimensões (ficheiro *data.pl*), o que permite verificar que leva a resultados que satisfazem a descrição do problema.
Neste problema existem 5 especialidades, carpintaria, pichelaria, canalização, jardinagem e telhados, havendo a restrição que operações de carpintaria devem ser feitas antes de operações de pichelaria. A empresa tem 4 trabalhadores, um com especialidade de canalização e salário de 20 (ID 1), um com especialidades de carpintaria e pichelaria e salário de 50 (ID 2), um com especialidade de canalização e salário de 60 (ID 3) e um com especialidades de jardinagem e telhados e salário de 30 (ID 4).
A empresa tem ao seu encargo realizar 2 obras. A primeira, com preço de 300 para uma duração de 10 dias, e bónus de 10 por cada dia de diferença em relação à duração acordada. Esta obra tem duas operações, uma com especialidade pichelaria, custo de equipamentos de 25 e que demora 10 dias a ser feita por 1 trabalhador (ID 1), e outra com especialidade carpintaria, custo de equipamentos de 25 e que demora 1 dia a ser feita por 1 trabalhador (ID 2). A segunda obra tem preço de 200 para uma duração de 15 dias e bonus de 20. Esta obra também tem duas operações, uma com especialidade canalização, custo de equipamentos 25 e

duração 10 dias quando apenas 1 trabalhador lhe é alocado (ID 3), e outra com especialidade jardinagem, custo de equipamentos 25 e duração 2 dias para 1 trabalhador (ID 4).
O escalonamento ótimo obtido tem um lucro de -10 para a equipa. A tarefa 1 é realizada nos dias 1, 2 e 3 pelos trabalhadores 1, 3 e 4. A tarefa 2 é realizada entre no dia 0 pelo trabalhador 2. A tarefa 3 é realizada nos dias 4, 5, 6, 7 e 8 pelos trabalhadores 1 e 4. A tarefa 4 é realizada no dia 9 pelos trabalhadores 1 e 4. Todas as tarefas têm pelo menos um trabalhador da especialidade. As tarefas da primeira obra são realizadas de seguida, com uma duração de 4 dias. A com ID 1 é efetuada por apenas um trabalhador, pois ter mais trabalhadores não diminui a sua duração, que é 1 para apenas um trabalhador. A com ID 2, que requer obrigatoriamente o trabalhador com ID 2 devido a ser o único da sua especialidade, é efetuada com ajuda de mais 2 trabalhadores (IDs 1 e 4, que são os que têm salário mais baixo), pois a diminuição de tempo (que faz com que seja recebido um bonus da obra e diminui o salário total a pagar ao trabalhador com ID 2, que é relativamente mais alto que dos restantes) é favorável. A com ID 2 é realizada antes da com ID 1, devido à restrição relativa às precedências. A segunda obra é realizada em 6 dias, sendo ambas as tarefas realizadas pelos trabalhadores 1 e 4, pois são os que possuem os salários mais baixos e têm as especialidades necessárias, levando à conclusão da obra em menos tempo do que o contratado, aumentando o lucro recebido por ela.

5.2 Exemplo grandes dimensões e estratégias de labeling

Um problema encontrado com a solução proposta é a sua **escalabilidade**. Foi possível observar que o tempo de resolução do programa aumenta **exponencialmente** com a adição de novos trabalhadores e/ou operações de especialidade, de tal modo que no exemplo de grandes dimensões (11 trabalhadores, 20 operações e 4 obras, ficheiro *bigdata.pl*) o programa não consegue resolver o problema em tempo útil (superior a 4 horas). Neste sentido, questionou-se se a equipa de construção apenas tivesse um determinado tempo para poder ter uma solução para o seu problema, qual seriam as opções de *labeling* que levariam a um lucro maior. Para estudar tal questão, o programa foi corrido , sucessivas vezes, utilizando o banco de dados "grande" com valores diferentes para as flags de labeling, para valores de timeout de 5 e 10 minutos. Foram testadas todas as combinações das flags de ordenação de valores (*min*, *max*, *ff*, *occcurrence*, *ffc* e *max_regret*), com as flags de selecção de valores (*step*, *enum*, *bisect* e *median*) e as flags de ordenação de valores (*up* e *down*). É de notar que não foram testadas as flags "*leftmost*" (dado ao facto de actuar com base na posição das variáveis), "*anti_first_fail*"/ (devido a testes preliminares mostrarem que esta flag não traria qualquer vantagem) e as flags "*variable/value*" (dado ao facto de o seu uso implicar outro grau de variabilidade que seria necessário estudar).

Na figura 1 apresentam-se os resultados das experiências efectuadas. É possível observar que a maior parte dos resultados obtidos não foram muito favoráveis à equipa de construção. De facto flags como *min*, *max* e *occurrence* não obtiveram

uma solução no tempo testado. O conjunto de flags que obteve o melhor resultado, no geral, foram os conjuntos [*max_regret*, *enum*, *up*], [*max_regret*, *bisect*, *up*], [*max_regret*, *step*, *up*], [*max_regret*, *median*, *up*], de onde se conclui que há vantagem em utilizar as flags *max_regret* e *up* em conjunto. No entanto é de notar que os maiores crescimentos entre os dois instantes de tempo vieram dos conjuntos [*ffc*, *bisect*,*down*] e [*ffc*,*enum*,*up*], enquanto que os conjuntos de maior lucro mantiveram os seus resultados entre os dois testes.

Flags	Lucro após 5 minutos	Lucro após 10 minutos	Crescimento
ff step up	-1135	-425	710
ff step down	-3300	-3300	0
ff enum up	-1135	-1135	0
ff enum down	-3300	-3300	0
ff bisect up	-820	-425	395
ff bisect down	-3300	-3300	0
ff median up	-1135	-425	710
ff median down	-3300	-3300	0
ffc step up	-1060	-935	125
ffc step down	-3050	-2950	100
ffc enum up	-1060	-1060	0
ffc enum down	-2750	-1750	1000
ffc bisect up	-935	-935	0
ffc bisect down	-2250	530	2780
ffc median up	-1060	-935	125
ffc median down	-3100	-2950	150
max_regret step up	1935	1935	0
max_regret step down	-3645	-3635	10
max_regret enum up	1935	1935	0
max_regret enum down	-3645	-3635	10
max_regret bisect up	1935	1935	0
max_regret bisect down	-3645	-3635	10
max_regret median up	1935	1935	0
max_regret median down	-3645	-3635	10

Fig. 1: Resultados de efetuar labeling com vários sets de flags no exemplo de grandes dimensões

5.3 Utilização de heurísticas

Para além do uso de flags, tal como foi visto na secção anterior, existe a hipótese de utilizar **heurísticas** para o método usado na ordem de *labeling* das variáveis

bem como o valor a lhes atribuir. Como exemplo desta técnica concebemos a heurística de que seria vantajoso averiguar primeiro quais as durações finais das tarefas (Di), pois estas variáveis estão intrinsecamente ligadas à quantidade de trabalhadores alocada à tarefa (Hi = $DuraçãoBase$ / Di). Assumindo que também ficam definidos os valores de Di consideramos que saber os valores da matriz de trabalhadores permitiria ao programa propagar restrições para as variáveis seguintes bem como simplificar o cálculo do lucro (devido aos elementos de valor 0). Por fim supusemos que saber os valores do inicio das operações(Oi) iria propagar as restantes restrições, pois através de Oi e Di é possível descobrir os valores de Ei. O predicado de pesquisa utilizado é o demonstrado no código fonte 1.1. Neste caso o uso da heurística foi vantajoso pois permitiu superar os resultados das experiências com flags da secção 5.2, tendo obtido um lucro de 2375 em 10 minutos.

```
1     % Variable grouping for labeling
2     flattenMatrix(WorkersMatrix,FlatMatrix,[]),
3     splitTaskVars(Tasks,Origins,Dur,Ends,WorkerCounts),
4     append(Dur,FlatMatrix,VarsSet1),
5     append(Origins,Ends,VarsSet2),
6     append(WorkerCounts,[Profit],VarsSet3),
7
8     % Solution searching
9     solve( [maximize(Profit),time\_out(600000,\_)],
10        [
11            labeling([leftmost,up],VarsSet1),
12            labeling([occurrence,enum,up],VarsSet2),
13            labeling([],VarsSet3)
14        ]
15    ),
```

Código fonte 1.1: Pesquisa utilizada.

6 Conclusões e Trabalho Futuro

Com este trabalho foi possível concluir que existe uma grande utilidade no paradigma de programação por restrições e também no seu uso aliado à declaratividade do *Prolog*. No entanto, também foi possível observar que embora este paradigma seja aplicável à resolução de problemas de optimização e de escalonamento, o seu uso não dispensa dos problemas de complexidade temporal que são associados à resolução de problemas NP-completos. Como trabalho futuro gostaríamos de salientar duas alternativas. A primeira seria a de continuar o

estudo da solução implementada, através do estudo estatístico mais aprofundado de técnicas de pesquisa de soluções. Este estudo passaria por colocar mais níveis temporais à técnica apresentada na secção 5.2, bem como trabalhar sob outras heurísticas tal como na secção 5.3. A segunda alternativa, seria reformular a solução implementada de modo a reduzir o número de variáveis que são utilizadas na pesquisa, nomeadamente encontrando uma forma de substituir a matriz de alocação de trabalhadores.