Procura e Planeamento

Campus Alameda

IST @ 2014/2015

23 de Abril de 2015

1. Introdução

Este projecto tem dois objectivos: (1) Desenvolver um programa em ANSI Common Lisp que resolva diferentes problemas de *calendarização da produção industrial por encomenda*[1]. Devem ser utilizadas diferentes estratégias de procura estudadas na cadeira, e desenvolvidas heurísticas que permitam resolver este tipo de problemas da forma mais eficaz possível; (2) Produzir um estudo que avalie as alternativas implementadas tanto do ponto de vista quantitativo como qualitativo.

[1] Na literatura de língua inglesa este problema é designado por Job-shop scheduling.

2. Descrição do problema de calendarização da produção industrial por encomenda

Um problema de calendarização da produção industrial por encomenda é constituído por um conjunto de encomendas. Cada encomenda é constituída por uma sequência ordenada de tarefas em que cada tarefa tem que ser realizada por uma determinada máquina.

O problema de calendarização da produção industrial por encomenda corresponde à atribuição de tempos de inicio para cada uma das tarefas respeitando as restrições do problema e minimizando o tempo necessário para completar todas as encomendas.

As restrições do problema são:

- as *restrições de capacidade* - uma máquina só pode realizar uma tarefa em cada momento;
- as *restrições de precedência* - a primeira tarefa de uma encomenda tem de preceder a segunda tarefa da mesma encomenda e assim por diante.

Este é um *problema de optimização* pois o objectivo é minimizar o tempo para completar todas as encomendas.

Uma tarefa é definida pelo tuplo:

```
(<n°encomenda>, <n°ordem>, <n°máquina>, <duração>, <tempo inicio>)
```

Na modelação do problema todas as durações devem ser apresentadas em minutos. Também os instantes devem ser apresentados em minutos, referindo-se o valor ao número de minutos que decorreram desde o início da contagem do tempo.

3. Resolução do problema

A obtenção de uma solução válida para o problema de calendarização da produção industrial por encomenda, tal como descrito na secção anterior, é trivial visto que não é imposto qualquer limite temporal à conclusão das encomendas. Já a obtenção de uma boa solução ou mesmo da solução óptima — o objectivo deste trabalho — não é de todo trivial.

Deve ser escolhida uma formulação adequada que permita resolver qualquer problema da forma mais eficiente possível, tendo em conta o objectivo de minimização do tempo necessário para completar todas as encomendas. Os problemas fornecidos na página da cadeira são apenas exemplos dos problemas que vão ser usados no teste das soluções produzidas[2].

4. Implementação

Parte da avaliação do trabalho desenvolvido vai decorrer automaticamente, pelo que é essencial que a especificação da interface seja seguida rigorosamente.

O conjunto de funções que implementa o *solucionador* deve ser definido num único ficheiro, este deve poder ser compilado sem erros nem avisos. Uma dos avisos produzidos pelos compiladores é a falta de uma declaração de package num ficheiro de código; de forma a evitar este aviso deve ser incluída a seguinte forma Lisp:

```
(in-package :user)
```

Esta forma deve ser a primeira forma do ficheiro.

Refira-se ainda que os testes a realizar automaticamente impõem alguns limites espaciais e temporais considerados razoáveis: (1) um limite à heap de 256 MBytes e (2) um tempo máximo por problema de afectação de 5 minutos num Core i5 a 2,6 GHz, ou seja a eficiência temporal e espacial das soluções apresentadas é relevante, tal como já tinha sido afirmado anteriormente.

[2] Os problemas fornecidos na página da cadeira foram obtidos da colecção de problemas mantida por J. E. Beasley na OR-Library. Outras instâncias podem ser obtidas em:

http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt

4.1. Interface

Deve ser implementada a função *calendarização*, que produz uma solução para um dado problema de calendarização da produção industrial por encomenda.

Esta função recebe como argumentos: um problema na sua representação externa definida em baixo e a estratégia de procura a usar. A função deve devolver a solução do problema usando a representação externa de uma solução, também definida em baixo.

As estratégias a suportar são:

```
"melhor.abordagem"
"a*.melhor.heuristica"
"a*.melhor.heuristica.alternativa"
"sondagem.iterativa"
"ILDS"
"abordagem.alternativa"
```

A representação externa de um problema de calendarização da produção industrial por encomenda é uma structure do tipo `job-shop-problem` em que o campo `name` é uma string que representa o nome do problema, o campo `n.jobs` é um inteiro que representa o número de encomendas, o campo `n.machines` é um inteiro que representa o número de máquinas e o campo `jobs` é uma lista com as encomendas:

```
(defstruct job-shop-problem
   name
   n.jobs
   n.machines
   jobs)
```

Cada encomenda é representada por uma structure do tipo `job-shop-job` em que o campo `job.nr` é um inteiro que representa o número da encomenda e o campo `tasks` é uma lista com as tarefas necessárias para completar a encomenda.

```
(defstruct job-shop-job
   job.nr
   tasks)
```

Cada tarefa é representada por uma *structure* do tipo `job-shop-task` em que o campo `job.nr` é um inteiro que representa o número da encomenda, o campo `task.nr` é um inteiro que representa a ordem pela qual a tarefa deve ser realizada, o campo `machine.nr`

é um inteiro que representa a máquina em que a tarefa deve ser realizada, o campo `duration` é um inteiro que representa a duração da tarefa em minutos e o campo `start.time` é um inteiro que representa o momento em que se deve iniciar.

```
(defstruct job-shop-task
   job.nr
   task.nr
   machine.nr
   duration
   start.time)
```

Na *Figura 1* representa-se um problema com duas encomendas e duas máquinas.

```
#S(JOB-SHOP-PROBLEM
    :NAME "Teste"
    :N.JOBS 2
    :N.MACHINES 2
    :JOBS (#S(JOB-SHOP-JOB
                :JOB.NR 0
                :TASKS (
#S(JOB-SHOP-TASK :JOB.NR 0 :TASK.NR 0 :MACHINE.NR 1 :DURATION 12 :START.TIME NIL)
#S(JOB-SHOP-TASK :JOB.NR 0 :TASK.NR 1 :MACHINE.NR 0 :DURATION 68 :START.TIME NIL)))
            #S(JOB-SHOP-JOB
                :JOB.NR 1
                :TASKS (
#S(JOB-SHOP-TASK :JOB.NR 1 :TASK.NR 0 :MACHINE.NR 0 :DURATION 5 :START.TIME NIL)
#S(JOB-SHOP-TASK :JOB.NR 1 :TASK.NR 1 :MACHINE.NR 1 :DURATION 5 :START.TIME NIL)))))
```

Figura 1: Exemplo de um problema.

Na *Figura 2* representa-se a invocação a calendarizacao, com o problema da *Figura 1* que foi previamente atribuído à variável *prob01* para o qual se quer obter um solução usando a estratégia de procura "a*". Na *Figura 3* representa-se a solução obtida.

```
(calendarizacao prob01 "a*")
```

Figura 2: Invocação do programa.

A representação externa de uma solução é uma lista com todas as tarefas necessárias para completar as encomendas do problema, cada tarefa tem que ter o momento em que se deve iniciar e todas as restrições do problema têm que ser respeitadas.

```
(#S(JOB-SHOP-TASK :JOB.NR 0 :TASK.NR 0 :MACHINE.NR 1 :DURATION 12 :START.TIME 0)
 #S(JOB-SHOP-TASK :JOB.NR 0 :TASK.NR 1 :MACHINE.NR 0 :DURATION 68 :START.TIME 12)
 #S(JOB-SHOP-TASK :JOB.NR 1 :TASK.NR 0 :MACHINE.NR 0 :DURATION 5 :START.TIME 0)
 #S(JOB-SHOP-TASK :JOB.NR 1 :TASK.NR 1 :MACHINE.NR 1 :DURATION 5 :START.TIME 12))
```

Figura 3: Solução do problema da *Figura 1*.

A estratégia `"melhor.abordagem"` deve usar a modelação, estratégia de procura, heurística e estratégia de corte de sucessores que produz os melhores resultados de entre tudo o que foi implementado.

4.2. Código Lisp fornecido

Para a realização deste projecto devem ser utilizados os algoritmos de procura, desenvolvidos em ANSI Common Lisp, disponíveis no *site* da cadeira. O código fornecido encontra-se no ficheiro '*procura.lisp*'. Este contém a implementação de vários algoritmos de procura. O funcionamento dos mesmos encontra-se descrito em '*procura.txt*'. O mais importante é compreender para que servem, e não perceber exactamente como é que as funções estão definidas. Este ficheiro não deve ser alterado; se houver necessidade de alterar definições incluídas neste ficheiro estas devem ser feitas no ficheiro de código desenvolvido que contém a implementação realizada pelo grupo.

É ainda fornecido código que implementa os modelos descritos na secção anterior no ficheiro '*job-shop-problemas-modelos.lisp'*.

5. Estudo a desenvolver

A abordagem a desenvolver deve modelar o problema como uma procura num espaço de estados usando uma *abordagem incremental* e uma *perspectiva de optimização*. Ou seja deve devolver a melhor solução possível de acordo com o definido anteriormente.

- Devem ser testadas todas as técnicas de procura fornecidas no ficheiro "*procura.lisp*". Note-se que é necessário alterar as estratégias de forma a que terminem dentro do tempo limite devolvendo a melhor solução encontrada até esse momento.
- Deve ser implementada e testada a *estratégia de sondagem iterativa*.
- Deve ser implementada e testada a *estratégia de discrepância melhorada ILDS (Improved Limited-Discrepancy Search)* com as necessárias adaptações.

- Deve ser implementada pelo menos uma abordagem alternativa diferente das listadas acima.
- Adicionalmente, novas estratégias que facilitem a resolução dos problemas devem ser desenvolvidas. Estas podem incluir variações aos algoritmos básicos de procura, novos algoritmos de procura, estratégias híbridas, macro-operadores, sub-objectivos, etc. A análise do problema e dos primeiros resultados obtidos deverá fornecer indicadores quanto a abordagens adequadas.

Para as técnicas de procura informadas, são necessárias heurísticas.

- Devem ser apresentadas pelo menos duas heurísticas de desempenho relevante. Essas heurísticas devem, tanto quanto possível, basear-se em ideias distintas (heurísticas que variam apenas em constantes, por exemplo, são consideradas a mesma heurística).

O desempenho das várias estratégias de procura utilizadas, bem como das heurísticas e das estratégias de corte desenvolvidas deve ser discutido no relatório, fazendo uma comparação dos resultados obtidos para alguns dos problemas usados para teste. Toda esta avaliação deve ser baseada em resultados quantitativos nomeadamente: custo das soluções, tempo da procura, nós gerados, nós expandidos, factor médio de ramificação, profundidade atingida, etc.

A discussão deve limitar-se a problemas que salientem características ou limitações relevantes do trabalho desenvolvido. Os alunos são encorajados a criar novos problemas, para além dos fornecidos como exemplo, que sejam adequados para ilustrar aspectos interessantes da discussão.

Todo o processo de decisão deve ser documentado no relatório a elaborar.

O relatório do projecto deve incluir ainda:

- A descrição da(s) modelação(ões) do problema e as razões subjacentes às decisões tomadas;
- A descrição das estruturas de dados desenvolvidas e as razões subjacentes ao seu desenvolvimento;

- A descrição das heurísticas desenvolvidas e as razões subjacentes ao seu desenvolvimento;
- A descrição das estratégias de corte implementadas e as razões subjacentes ao seu desenvolvimento;
- As opções tomadas durante o desenvolvimento do projecto;
- Uma discussão sobre a forma de conseguir obter melhores resultados, indicando genericamente o tipo de heurísticas/estratégias a desenvolver para isso ou alterações à modelação.

Ainda quanto ao relatório, convém salientar que a legibilidade do mesmo (para a qual contribuem não só a organização dos capítulos/secções como a correcção ortográfica a construção das frases e a fluidez do discurso) é de importância capital para a correcta compreensão do seu conteúdo. Devem evitar-se frases complexas (a utilização da voz passiva, de duplas negativas, etc.). Todas as figuras e gráficos apresentados devem ser legíveis e estar adequadamente legendados.

6. Critérios de avaliação

Os factores mais importantes para a avaliação do projecto são (não necessariamente por esta ordem):

- Execução correcta e elegância do programa;
- Qualidade da(s) modelação(ões) do problema;
- Qualidade das heurísticas e estratégias desenvolvidas;
- Eficiência das estruturas de dados escolhidas para a representação do problema;
- Qualidade do estudo desenvolvido (inclui a descrição do trabalho desenvolvido, os resultados obtidos e conclusões);
- Apreciação global.

7. Inscrições e entrega do projecto

Os elementos de cada grupo, que deverão ser no máximo 2, devem inscrever-se no Fénix.

A entrega será dividida em duas fases, a primeira relativa ao código desenvolvido e a segunda relativa ao relatório.

A entrega do código desenvolvido deve ser feita até às 23h59 do dia 25 de Maio de 2015 da seguinte forma:

- o ficheiro com o programa deve ser entregue de forma electrónica usando o sistema Fénix. O nome do ficheiro Lisp a entregar deve ser "GXXX.lisp", em que "XXX" é o número do grupo, por exemplo "G007.lisp";

A entrega do relatório deve ser feita até às 23h59 do dia 29 de Maio de 2015 da seguinte forma:

- o ficheiro com o relatório em formato electrónico (ficheiro em formato “pdf” ou alternativamente em formato “word") deve ser entregue de forma electrónica usando o sistema Fénix. O nome do ficheiro a entregar deve ser "GXXX.pdf" ou "GXXX.doc", em que "XXX" é o número do grupo, por exemplo "G007.pdf";
- os relatórios em papel (incluindo a listagem do programa devidamente paragrafada e utilizando um tipo de letra mono espaçado) devem ser entregues na aula prática a seguir.

Todo o material entregue pelos alunos (documentação e ficheiro com o código) deve conter a identificação do grupo (número do grupo, fornecido durante a inscrição do grupo) e o número e nome de cada elemento do grupo.

Atenção: Não são aceites entregas fora do prazo!

8. Novidades do projecto

No caso de haver novidades relativas ao projecto, estas serão afixadas na página da cadeira pelo que esta página deve ser visitada diariamente.

Procura e Planeamento

Campus Alameda

Projecto (2014/2015)

Número do grupo (obtido a partir do sistema Fénix):

Nome:	Número:
Nome:	Número:

Classificação:

Soma das horas gastas exclusivamente para fazer este trabalho:

Limite para entrega: dia 23h59 do dia 29 de Maio de 2015.