Inteligência Artificial

Actividades@IST

2/11/2012
(Act. 6/11/2012)

1 Introdução

Este trabalho consiste em fazer um estudo sobre métodos de procura aplicados ao problema de afectação de recursos necessários para planificar um período de trabalho.

Um exemplo deste problema é a decisão que cada aluno tem de tomar para planificar as tarefas a desempenhar semanalmente, que inclui, entre outras, a ida às aulas. Há um conjunto de disciplinas a que é suposto assistir às aulas e um conjunto de horários para cada disciplina com as alternativas dos turnos que os alunos podem escolher, em que são especificadas as horas das aulas. O problema que vai ser estudado é o de conseguir conjugar as tarefas que se pretende realizar com os horários disponíveis para a realização das tarefas.

2 O problema

Vamos considerar que se quer executar um determinado conjunto de tarefas, em que cada tarefa pode ser executada de várias formas alternativas.

Por exemplo, considerando que o conjunto de tarefas a executar inclui a tarefa de "assistir às aulas teóricas da disciplina X ", se houver dois turnos de teóricas, esta tarefa pode ser executada de duas maneiras alternativas: ou executando cada uma das subtarefas do primeiro turno (ir de casa para a universidade e assistir à aula teórica semanal de cada turno – está-se a supor que a disciplina X tem apenas uma aula teórica por semana) ou executando cada uma das subtarefas do segundo turno.

Cada subtarefa vai ser caracterizada pelo dia (inteiro positivo) do período em que vai ocorrer, a hora (inteiro entre 0 e 23) em que vai ocorrer (vai-se considerar que a subtarefa dura uma hora e que começa a horas certas) e por um nome (um símbolo).

Em termos de implementação, uma subtarefa define-se chamando a função `faz-subtarefa`, que recebe os argumentos `dia`, `hora`, `nome` passados por nome, como no exemplo abaixo.

```
(faz-subtarefa :dia 1 :hora 9 :id 't101)
```

O valor retornado pela chamada à função `faz-subtarefa` é uma lista com o dia, a hora e a identificação.

A execução de todas as subtarefas de uma alternativa corresponde à execução de uma das alternativas para a realização da tarefa. A especificação de todas as subtarefas correspondentes à alternativa faz-se construindo uma lista de subtarefas, como no exemplo abaixo, em que se considera que uma alternativa para a realização da tarefa consiste em executar a subtarefa de identificação `t101` no dia 1 do período às 9 horas e também executar a subtarefa de identificação `t102` no dia 1 às 10 horas.

```
(list
 (faz-subtarefa :dia 1 :hora 9 :id 't101)
 (faz-subtarefa :dia 1 :hora 10 :id 't102))
```

A especificação do conjunto de alternativas para a realização da tarefa faz-se identificando uma lista com cada uma das alternativas. Por exemplo, se considerarmos além da alternativa do exemplo anterior a alternativa da tarefa que consiste em executar a subtarefa de identificação `t103` no dia 1 do período às 9 horas e também executar a subtarefa de identificação `t104` no dia 1 às 11 horas, podemos construir uma tarefa com duas alternativas como a representada no exemplo abaixo.

```
(list
   (list
      (faz-subtarefa :dia 1 :hora 9 :id 't101)
      (faz-subtarefa :dia 1 :hora 10 :id 't102))
   (list
      (faz-subtarefa :dia 1 :hora 9 :id 't103)
      (faz-subtarefa :dia 1 :hora 11 :id 't104)))
```

O conjunto de tarefas a executar é especificado escrevendo-se uma lista de tarefas em que cada tarefa é descrita por uma lista de maneiras alternativas de executar a tarefa, em que cada alternativa corresponde a uma lista de subtarefas a executar. Por exemplo,

```
(list ; lista de tarefas
   (list ; uma tarefa é uma lista de alternativas
         ; de maneiras de realizar a tarefa
      (list ; uma maneira de realizar a tarefa é
            ; executar uma lista de subtarefas
         (faz-subtarefa :dia 1 :hora 9 :id 't101)
         (faz-subtarefa :dia 1 :hora 10 :id 't102))
      (list
         (faz-subtarefa :dia 1 :hora 9 :id 't103)
         (faz-subtarefa :dia 1 :hora 11 :id 't104)))
   (list
      (list
         (faz-subtarefa :dia 1 :hora 10 :id 't111)
         (faz-subtarefa :dia 1 :hora 12 :id 't112))
      (list
         (faz-subtarefa :dia 1 :hora 11 :id 't113)
         (faz-subtarefa :dia 1 :hora 12 :id 't114))))
```

descreve duas tarefas, uma das quais discutida anteriormente, com duas alternativas cada.

Um dos problemas que vai ser estudado é o problema que consiste em, a partir de um conjunto de tarefas que se quer executar e do conjunto de alternativas para executar cada uma das tarefas, escolher as alternativas adequadas para conseguir executar o conjunto das tarefas, tendo em atenção que tarefas distintas não podem ser executadas simultaneamente.

Vai-se ainda considerar que pode haver subtarefas partilhadas por várias tarefas. Por exemplo, e de forma informal, se considerarmos as subtarefas "apanhar boleia de casa para a Alameda", essa subtarefa pode ser partilhada pela tarefa que tem as duas subtarefas "apanhar boleia de casa para a Alameda" e "assistir à teórica de X" e pela tarefa que tem as duas subtarefas "apanhar boleia de casa para a Alameda" e "assistir à teórica de Y". O mecanismo para identificar a partilha de tarefas é a utilização da mesma descrição para as tarefas partilhadas. No exemplo abaixo, a subtarefa com o identificador `t121` é partilhada por duas alternativas, uma de cada tarefa, o que permite reduzir o número de horas a gastar para executar as tarefas.

```
(list
   (list
      (list
         (faz-subtarefa :dia 1 :hora 9 :id 't121)
         (faz-subtarefa :dia 1 :hora 10 :id 't122))
      (list
```

```
        (faz-subtarefa :dia 2 :hora 10 :id 't123)
        (faz-subtarefa :dia 2 :hora 11 :id 't124)))
(list
   (list
      (faz-subtarefa :dia 1 :hora 9 :id 't121)
      (faz-subtarefa :dia 1 :hora 11 :id 't132))
   (list
      (faz-subtarefa :dia 2 :hora 11 :id 't133)
      (faz-subtarefa :dia 2 :hora 12 :id 't134))))
```

O segundo problema a ser estudado é o problema de, a partir de um conjunto de tarefas que se quer executar e do conjunto de alternativas para executar cada uma das tarefas, escolher as alternativas adequadas para **minimizar** o tempo necessário para desenvolver um determinado conjunto de tarefas, tendo em atenção que subtarefas distintas não podem ser executadas simultaneamente.

3 O estudo a fazer

Pretende-se estudar a adequação de alguns métodos, em particular de procura em espaço de estados e satisfação de restrições, para resolver os dois problemas referidos acima. No que se refere aos métodos de procura de espaço de estados, quere-se estudar o efeito de se variar formulações do problema e o efeito de variar as funções heurísticas.

Por um lado, no que diz respeito à variação da formulação do problema, deve ser considerada uma primeira formulação em que a cada passo se tenta executar *qualquer* uma das tarefas com qualquer das formas alternativas de a realizar. Posteriormente, deve-se igualmente considerar uma outra formulação em que a cada passo se tenta executar *apenas* uma das tarefa e estudar o efeito da variação das formulações. Estas variações, tal como as seguintes, devem ser testadas para um número significativo de exemplos, suficiente para as conclusões pretendidas.

Por outro lado, deve-se experimentar variar as funções heurísticas para testar os vários algoritmos de procura heurística. Em particular, devem ser consideradas funções heurísticas consistentes, heurísticas simultaneamente admissíveis e não consistentes, e heurísticas não admissíveis e estudar o desempenho das procuras. Ainda por outro lado, quando apropriado, devem-se considerar as variantes da procura em árvore e procura em grafo.

Em termos de procura não-informada, devem ser considerados os algoritmos de procura em profundidade primeiro (e as suas variantes que forem consideradas apropriadas), procura em largura e procura de custo uniforme. Em termos de procura informada, devem ser considerados os algoritmos de procura gananciosa, A* e RBFS.

No que diz respeito à utilização das técnicas de satisfação de restrições, deve-se fazer a implementação do algoritmo descrito na 3ª Edição do livro de texto e, tal como referido anteriormente, explorar a adequação do método para a resolução dos dois problemas referidos acima.

Deve-se construir exemplos significativos que permitam tirar conclusões relativas a cada um dos algoritmos utilizados, incluindo conclusões relativas à análise de complexidade dos vários algoritmos.

4 Avaliação

A avaliação vai ser feita tanto quanto possível de forma automática. Neste sentido, é essencial que sejam *escrupulosamente* observadas as interfaces pedidas.

Assim, deve-se usar as funções com os nomes `pppa, pppg, ppla, pplg, ppia, ppig, plpa, plpg, pcua, pcug, pga, pgg, pA*a, pA*g, RBFS` e `psr`, que implementam respectivamente a procura em profundidade (em árvore e grafo), em profundidade limitada (em árvore e em grafo), em profundidade iterativa (em árvore e grafo), em largura primeiro (em árvore e grafo), de custo uniforme (em árvore e grafo), pelo melhor primeiro (em árvore e grafo), RBFS e satisfação de restrições, que recebem como argumentos o resultado de produzir a formulação do problema a partir de uma lista de tarefas em que cada tarefa é descrita por uma lista de maneiras alternativas de executar a tarefa, em que cada alternativa corresponde a uma lista de subtarefas a executar (a função `ppla` e `pplg` recebem um segundo argumento que corresponde ao limite da procura em profundidade a considerar e as funções da procura heurística recebem também um segundo argumento que é a função heurística). Todas estas funções retornam uma lista de alternativas.

Por exemplo, poderiamos ter a interacção que se segue.

```
> (pppa
    (formulacao-problema
       (list
          (list
             (list
                (faz-subtarefa :dia 1 :hora 9 :id 't121)
                (faz-subtarefa :dia 1 :hora 10 :id 't122))
             (list
                (faz-subtarefa :dia 2 :hora 10 :id 't123)
                (faz-subtarefa :dia 2 :hora 11 :id 't124)))
          (list
             (list
```

```
                    (faz-subtarefa :dia 1 :hora 9 :id 't121)
                    (faz-subtarefa :dia 1 :hora 11 :id 't132))
                (list
                    (faz-subtarefa :dia 2 :hora 11 :id 't133)
                    (faz-subtarefa :dia 2 :hora 12 :id 't134))))))

(((1 9 T121) (1 11 T132)) ((1 9 T121) (1 10 T122)))
```

O código desenvolvido deve compilar em SBCL sem qualquer "warning" ou erro.

A avaliação vai ser feita com base nos seguintes parâmetros:

- Estudo desenvolvido sobre o problema (dimensão máxima do estudo é a menor de 10 páginas A4 ou 30 mil caracteres), valendo 50% da nota total;
- Qualidade da implementação, medida em termos da velocidade dos algoritmos desenvolvidos, valendo 30% da nota total;
- Execução correcta, velendo 10% da nota total;
- Apreciação global, valendo 10% da nota total.

5 Outras informações

Os alunos devem-se inscrever em grupos de até 3 pessoas no Fénix.

Vai decorrer um campeonato para testar as implementações dos algoritmos A*, RBFS e das restrições de cada grupo, sendo atribuído 1 valor suplementar à nota do projecto do grupo (entre os grupos do Tagus e Alameda) que desenvolver o código mais eficiente (isto é, que produza a solução mais rápida) para um determinado conjunto de testes, para cada um dos 3 algoritmos referidos.

O código desenvolvido para as procuras cegas deve ser entregue por via electrónica de acordo com instruções na página da disciplina e deve ser entregue até ao dia **23/11** pelas 23:59 horas.

O código desenvolvido para as procuras heurísticas deve ser entregue por via electrónica de acordo com instruções na página da disciplina e deve ser entregue até ao dia 3/12 pelas 23:59 horas.

As novidades relativas ao projecto serão descritas na página da disciplina, na entrada Projecto, pelo que se recomenda visita diária à referida página.