Instituto Superior Técnico
Licenciatura em Engenharia Informática e Computadores - Alameda
14 Dezembro 2012

Grupo 47

João Neto	Mário Cardoso	Valmiky Arquissandas
64787	64814	64875

ACTIVIDADES@IST

Formulações

Para abordar o problema foram utilizadas duas formulações.

Formulação A:

Nesta formulação, cada estado é uma lista de alternativas. O índice da alternativa indica a que tarefa corresponde. O tamanho da lista também nos indica para quantas tarefas já foi escolhido um estado, e também que tarefa escolheremos a seguir.

A ordem das tarefas é a ordem pela qual aparecem na lista fornecida à formulação. Não chegámos a comparar de que forma a ordenação afectaria a velocidade das procuras.

O facto de a cada nível da árvore de procura apenas expandirmos uma tarefa faz com que esta formulação não permita estados repetidos - o facto de realizarmos uma procura em grafo não irá trazer qualquer benefício (pelo contrário - existe um overhead bastante significativo, o qual iremos discutir mais à frente).

Formulação B:

Nesta outra formulação, cada estado é uma lista de pares. Cada par contém uma alternativa, e o índice da tarefa da qual a alternativa pertence.

Nesta formulação é possível haver estados repetidos. Esta formulação, além de permitir estados repetidos, gera bastante mais estados: todos o da formulação A, mais todos os estados resultantes de aplicar a todas as tarefas ainda não presentes no estado as acções possíveis para as mesmas.

Considerações:

Para cada uma das formulações, a função ***acções*** apenas devolve as acções que não levem a estados que violem restrições (i.e. apresentem tarefas com alternativas com conflitos entre si).

Não chegámos a estudar como as diferentes ordenações das tarefas afectavam a performance das procuras (tempo e memória), da mesma forma que afectam as procuras de satisfação de restrições.

O custo de caminho corresponde ao número de subtarefas (não duplicadas) num estado.

Procuras

Para pesquisar nas árvores geradas a partir de cada uma das formulações, foram criadas as seguintes procuras (variantes árvore e grafo).

- Procura em Largura Primeiro
- Procura em Profundidade Primeiro
- Procura em Profundidade Limitada
- Procura em Profundidade Iterativa
- Procura de Custo Uniforme
- Procura Gananciosa
- Procura A*
- Procura Melhor Primeiro Recursiva

A complexidade algorítmica é parametrizada de seguinte modo:

- Profundidade da Solução Óptima = Profundidade Máxima = Número de Tarefas

- Factor de Ramificação = Para a formulação A, corresponde ao número de alternativas de uma tarefa; para a formulação B corresponde ao número de alternativas de todas as tarefas para as quais ainda não foi escolhida uma alternativa.

As procuras foram baseados no pseudocódigo de Artificial Intelligence: A Modern Approach. Estas apresentam uma boa eficiência e são relativamente simples de implementar em LISP.

Heurísticas

Para as formulações descritas acima foram implementadas várias heurísticas. Apenas falaremos em detalhe das heurísticas para a formulação A, que é a que apresenta melhor desempenho.

Heurística 2

Esta heurística simplesmente devolve o número de tarefas para as quais ainda não foi escolhida nenhuma alternativa.

Como o custo de transição pode ser igual a zero (e em cada transição é escolhida uma alternativa para uma tarefa), esta pode sobrestimar o custo da solução óptima. Como tal, não é nem admissível nem consistente.

Heurística 3

Para cada uma das tarefas as quais ainda não foram escolhidas uma alternativa, devolve a soma do custo das alternativas de menor custo para cada uma dessas tarefas.

Novamente, esta não é admissível - as alternativas podem partilhar tarefas (ver exemplos "htest").

Heurística 4

Esta surgiu como uma melhoria da heurística 3. Para cada uma das tarefas as quais ainda não foram escolhidas uma alternativa, tem em conta todas as alternativas de menor custo (e a repetição de subtarefas entre elas e o estado).

Contudo, não estamos ainda a analisar alternativas suficientes. Esta pode sobrestimar o custo da solução óptima caso a alternativa que leva à solução de menor custo não seja uma das alternativas de menor custo.

Heurística 5

Esta surge como uma melhoria da heurística 4. Para cada uma das tarefas as quais ainda não foram escolhidas uma alternativa, tem em conta todas as alternativas dessa tarefa. Iterativamente vai a cada uma das tarefas restantes, fazendo uniões de conjuntos de subtarefas (entre as alternativas e superconjuntos das subtarefas presentes no estado e de todas as subtarefas de todas as alternativas de iterações anteriores).

Esta heurística revela-se admissível e consistente, como iremos provar em seguida.

Heurística 6

Esta é uma simples modificação à heurística 5 que detecta conflitos antecipadamente.

Iremos então provar a admissibilidade e consistência da heurística 6 (e heurística 5):

A função gera-heurística6 para a formulação A recebe uma lista de tarefas e devolve uma função que devolve o custo heurístico para um determinado estado de um nó da árvore gerada pelas procuras.

Na primeira iteração da função heurística, o custo mínimo será o custo da alternativa de menor custo da primeira tarefa para a qual não foi escolhida uma alternativa, tendo em conta as subtarefas presentes no estado inicial.

Para iterações subsequentes, o custo mínimo será o custo da alternativa da próxima tarefa para a qual não foi escolhida uma alternativa, tendo em conta as subtarefas presentes no estado inicial e para todas as subtarefas de todas as alternativas consideradas em recursões anteriores da função.

Considere-se o problema "test-tarefas-1". Vamos então estimar o custo restante para o estado inicial.

Iteração para a Tarefa#1:
SubtarefasEstado = (); // estado inicial
CustoMínimo <-- + infinito;
- Alternativa #1: ((1 9 T1 2 1) (1 10 T1 2 2)) tem custo 2
Como 2<CustoMínimo, então CustoMínimo<--2;
- Alternativa #2: ((2 10 T1 2 3) (2 11 T1 2 4)) tem custo 2.
O CustoMínimo não é alterado.

Iteração para a Tarefa#2:
SubtarefasEstado = ((1 9 T1 2 1) (1 10 T1 2 2) (2 10 T1 2 3) (2 11 T1 2 4));
CustoMínimo <-- +infinito;
- Alternativa #1: ((1 9 T1 2 1) (1 11 T1 3 2)) tem custo 1, pois a subtarefa (1 9 T1 2 1) já se encontra no conjunto de subtarefas do estado inicial unindo com o conjunto das subtarefas de todas as alternativas da Tarefa#1.
CustoMínimo <-- 1;

- Alternativa #2: ((2 11 T133) (2 12 T134)) tem custo 2.
O CustoMínimo não é alterado.

O custo estimado pela função heurística para o estado inicial neste problema é 3, o que coincide com o custo da solução óptima.

É impossível esta heurística sobrestimar o custo da solução óptima - está a considerar repetições de subtarefas no superconjunto de todas as subtarefas do estado inicial e de todas as alternativas de todas as tarefas de iterações anteriores. Como tal, a heurística é admissível.

Esta heurística é também consistente, assumindo que para uma dada formulação um estado que possa ter uma ordem diferente para as atribuições de alternativas a tarefas não tenha repetição de atribuições.

Detecção prematura de conflitos:

A heurística detecta ainda se a partir do estado para a qual foi chamada não existem soluções. Se as subtarefas desse estado estiverem em conflito com todas as alternativas de qualquer uma das tarefas para as quais ainda não foram escolhidas alternativas, verifica-se que é impossível chegar a um estado objectivo - todas as alternativas dessa tarefa entrarão em conflito com o estado.

Neste caso, a heurística irá devolver um custo infinito para o nó, assinalando às procuras que o devem descartar de imediato.

Para problemas suficientemente grandes com alguns conflitos entre tarefas constatou-se que esta verificação reduzia de forma significativa os tempos de procura (e memória utilizada), comparando com a heurística 5 (cuja única diferença era a ausência de detecção de colisões).

Para problemas pequenos, ou problemas de grande dimensão sem conflitos entre tarefas, constatou-se que havia um aumento inferior a 1% da memória e tempo utilizado, o qual considerámos insignificante. Ficámos contentes com este resultado obtido.

Problema de Satisfação de Restrições

Considerámos que a cada variável correspondia uma tarefa, e a cada valor uma das suas alternativas. Considerámos ainda que o único tipo de restrições existentes eram restrições binárias, as quais representavam um conflito entre duas atribuições (escolha de alternativas para duas tarefas distintas).

Para o problema de satisfação de restrições considerámos as seguintes estruturas (a notação utilizada para descrever procuras é NomeTipo(Atributo1, ..., AtributoN).

CSP(Variáveis, Domínios, Restrições)
- Variáveis é uma lista de estruturas do tipo Variável;
- Domínios é uma lista de estruturas do tipo Domínio;
- Restrições é uma lista de estruturas do tipo Restrição.

Variável(Identificador, Valor, Domínio, Restrições)
- Identificador é o índice da tarefa na lista de tarefas (a começar em zero);
- Valor é uma referência para uma estrutura do tipo Valor, representando um valor do domínio da variável atribuído à mesma.

- Domínio é uma referência para uma estrutura do tipo domínio, a qual representa o domínio da variável;
- Restrições é uma lista de referências para estruturas do tipo Restrição, representando as restrições nas quais a variável está envolvida.

Valor(Valor, Restrições)
- Valor é uma alternativa existente na lista de tarefas;
- Restrições é uma lista de restrições, nas quais a atribuição deste Valor à sua variável está envolvido.

Atribuição(Variável, Valor)
- Variável: uma referência para uma estrutura do tipo Variável;
- Valor: uma referência para uma estrutura do tipo Valor.

Representa a atribuição de um valor a uma variável.

Domínio(Variável, Valores)
- Variável é uma referência para uma estrutura do tipo Variável, representando a variável à qual está associada este domínio;
- Valores é uma referência para uma lista de estruturas do tipo Valor, representando os valores possíveis de atribuir à variável.

Considerámos ainda uma estrutura adicional, no contexto de uma variável, para facilitar a procura de restrições ao atribuir um valor a uma variável.

RestriçãoVariável(Valor, AtribuiçãoOutraVariável)
- Valor é uma referência para uma estrutura do tipo Valor (um valor do domínio desta variável);
- AtribuiçãoOutraVariável é uma referência para uma atribuição a uma outra variável.

Esta estrutura representa uma restrição indicando que caso seja atribuído um valor à variável (à qual a estrutura está associada), a atribuição à outra variável não pode assistir.

Procura de Satisfação de Restrições

Relativamente à Procura de Satisfação de Restrições foram implementadas heurísticas genéricas que visam em diminuir drasticamente o tempo de procura e a memória usada:
- Heurística do Maior Grau;
- Heurística do Valor Menos Restritivo;

Não foi implementada a Heurística dos Valores Remanescentes Mínimos por falta de tempo - acreditamos que iria ter um impacto bastante positivo para problemas com um número elevado de tarefas, com poucas alternativas cada um.

Também por falta de tempo não nos foi possível implementar qualquer tipo de inferência, o que diminuiria radicalmente o tempo de procura, o qual já é bastante diminuto relativamente às outras procuras.

Devido às múltiplas referências entre variáveis, valores e restrições, a implementação do algoritmo MAC (Maintaining Arc Consistency) seria bastante eficiente e não muito complicada de implementar.

ANEXO I
TESTES UTILIZADOS

TESTE TESTE-TAREFAS-1

Este é o problema descrito no enunciado do projecto.

```
(list
        (list ;TAREFA #1
                (list ; - Tarefa #1 - alternativa #1
                        (faz-subtarefa :dia 1 :hora 9  :id 't121) ;optimizacao--1
                        (faz-subtarefa :dia 1 :hora 10 :id 't122))
                (list ; - Tarefa #1 - alternativa #2
                        (faz-subtarefa :dia 2 :hora 10 :id 't123)
                        (faz-subtarefa :dia 2 :hora 11 :id 't124))) ;conflito--1
        (list ;TAREFA #2
                (list ; - Tarefa #2 - alternativa #1
                        (faz-subtarefa :dia 1 :hora 9  :id 't121) ;optimizacao--1
                        (faz-subtarefa :dia 1 :hora 11 :id 't132))
                (list ; - Tarefa #2 - alternativa #2
                        (faz-subtarefa :dia 2 :hora 11 :id 't133)
                        (faz-subtarefa :dia 2 :hora 12 :id 't134))))) ;conflito--1
```

TESTE HTEST1

```
;;
;; Teste heuristico #1
;; Utilizado para calcular erros nas heuristicas
;;
;; Solucao optima:
;; (((2 1 TX) (2 2 TY) (2 3 TZ)) ((2 1 TX) (2 2 TY) (2 3 TZ)) ((2 1 TX) (2 2 TY) (2 3 TZ)))
;;
(defparameter *htest1*
        (list
                (list ; Tarefa 1
                        (list ; Alternativa 1
                                (faz-subtarefa :dia 1 :hora 1 :id 'tA)
                                (faz-subtarefa :dia 1 :hora 2 :id 'tB))
                        (list ; Alternativa 2
                                (faz-subtarefa :dia 2 :hora 1 :id 'tX)
                                (faz-subtarefa :dia 2 :hora 2 :id 'tY)
                                (faz-subtarefa :dia 2 :hora 3 :id 'tZ)))
                (list ; Tarefa 2
                        (list ; Alternativa 1
                                (faz-subtarefa :dia 1 :hora 3 :id 'tC)
                                (faz-subtarefa :dia 1 :hora 4 :id 'tD))
                        (list ; Alternativa 2
                                (faz-subtarefa :dia 2 :hora 1 :id 'tX)
                                (faz-subtarefa :dia 2 :hora 2 :id 'tY)
                                (faz-subtarefa :dia 2 :hora 3 :id 'tZ)))
                (list ; Tarefa 3
                        (list ; Alternativa 1
                                (faz-subtarefa :dia 1 :hora 5 :id 'tE)
                                (faz-subtarefa :dia 1 :hora 6 :id 'tF))
                        (list ; Alternativa 2
                                (faz-subtarefa :dia 2 :hora 1 :id 'tX)
                                (faz-subtarefa :dia 2 :hora 2 :id 'tY)
                                (faz-subtarefa :dia 2 :hora 3 :id 'tZ)))))
(defvar htest1 (formulacao-problema *htest1*))
```

TESTE HTEST2

```
;;
;; Teste heuristico #2
;; Utilizado para calcular erros nas heuristicas
;;
(defparameter *htest2*
        (list
                (list
                        (list
                                (faz-subtarefa :dia 0 :hora 0 :id 'tSTART)))
                (list ; Tarefa 1
                        (list ; Alternativa 1
                                (faz-subtarefa :dia 1 :hora 1 :id 'tA)
                                (faz-subtarefa :dia 1 :hora 2 :id 'tB))
                        (list ; Alternativa 2
                                (faz-subtarefa :dia 2 :hora 1 :id 'tX)
                                (faz-subtarefa :dia 2 :hora 2 :id 'tY)
                                (faz-subtarefa :dia 2 :hora 3 :id 'tZ)))
                (list ; Tarefa 2
                        (list ; Alternativa 1
                                (faz-subtarefa :dia 1 :hora 3 :id 'tC)
                                (faz-subtarefa :dia 1 :hora 4 :id 'tD))
                        (list ; Alternativa 2
                                (faz-subtarefa :dia 2 :hora 1 :id 'tX)
                                (faz-subtarefa :dia 2 :hora 2 :id 'tY)
                                (faz-subtarefa :dia 2 :hora 3 :id 'tZ)))
                (list ; Tarefa 3
                        (list ; Alternativa 1
                                (faz-subtarefa :dia 1 :hora 5 :id 'tE)
                                (faz-subtarefa :dia 1 :hora 6 :id 'tF))
                        (list ; Alternativa 2
                                (faz-subtarefa :dia 2 :hora 1 :id 'tX)
                                (faz-subtarefa :dia 2 :hora 2 :id 'tY)
                                (faz-subtarefa :dia 2 :hora 3 :id 'tZ)))
                (list
                        (list
                                (faz-subtarefa :dia 0 :hora 1 :id 'tEND)))))
(defvar htest2 (formulacao-problema *htest2*))
```

TESTE EX5

```
; Problemas do anuncio de dia 22/novembro
(defparameter *ex5*
  (list ;de tarefas
   (list (list (faz-subtarefa :dia 1 :hora 9 :id 't1))
               (list (faz-subtarefa :dia 2 :hora 9 :id 't2))
               (list (faz-subtarefa :dia 3 :hora 9 :id 't3))
               (list (faz-subtarefa :dia 1 :hora 10 :id 't4))
               (list (faz-subtarefa :dia 2 :hora 10 :id 't5))
               (list (faz-subtarefa :dia 3 :hora 10 :id 't6))
               (list (faz-subtarefa :dia 1 :hora 11 :id 't7))
               (list (faz-subtarefa :dia 2 :hora 11 :id 't8))
               (list (faz-subtarefa :dia 3 :hora 11 :id 't9)))
   (list (list (faz-subtarefa :dia 1 :hora 19 :id 't01))
               (list (faz-subtarefa :dia 2 :hora 19 :id 't02))
               (list (faz-subtarefa :dia 3 :hora 19 :id 't03)))
   (list (list (faz-subtarefa :dia 3 :hora 10 :id 't11))
               (list (faz-subtarefa :dia 4 :hora 10 :id 't12))
               (list (faz-subtarefa :dia 5 :hora 10 :id 't13)))
   (list (list (faz-subtarefa :dia 3 :hora 11 :id 't21))
               (list (faz-subtarefa :dia 4 :hora 11 :id 't22))
               (list (faz-subtarefa :dia 5 :hora 11 :id 't23)))
   (list (list (faz-subtarefa :dia 3 :hora 12 :id 't31))
               (list (faz-subtarefa :dia 4 :hora 12 :id 't32))
               (list (faz-subtarefa :dia 5 :hora 12 :id 't33)))
   (list (list (faz-subtarefa :dia 3 :hora 13 :id 't41))
               (list (faz-subtarefa :dia 4 :hora 13 :id 't42))
               (list (faz-subtarefa :dia 5 :hora 13 :id 't43)))
   (list (list (faz-subtarefa :dia 3 :hora 14 :id 't51))
               (list (faz-subtarefa :dia 4 :hora 14 :id 't52))
               (list (faz-subtarefa :dia 5 :hora 14 :id 't53)))
   (list (list (faz-subtarefa :dia 3 :hora 15 :id 't61))
               (list (faz-subtarefa :dia 4 :hora 15 :id 't62))
               (list (faz-subtarefa :dia 5 :hora 15 :id 't63)))
 ;  (list (list (faz-subtarefa :dia 3 :hora 16 :id 't71))
;              (list (faz-subtarefa :dia 4 :hora 16 :id 't72))
;              (list (faz-subtarefa :dia 5 :hora 16 :id 't73)))
 ;  (list (list (faz-subtarefa :dia 3 :hora 17 :id 't81))
;              (list (faz-subtarefa :dia 4 :hora 17 :id 't82))
;              (list (faz-subtarefa :dia 5 :hora 17 :id 't83)))
;   (list (list (faz-subtarefa :dia 3 :hora 18 :id 't91))
;              (list (faz-subtarefa :dia 4 :hora 18 :id 't92))
;              (list (faz-subtarefa :dia 5 :hora 18 :id 't93)))
   (list  (list (faz-subtarefa :dia 1 :hora 9 :id 't101)
                        (faz-subtarefa :dia 2 :hora 9 :id 't102)
                        (faz-subtarefa :dia 3 :hora 9 :id 't103)
                        (faz-subtarefa :dia 1 :hora 10 :id 't104)
                        ;(faz-subtarefa :dia 2 :hora 10 :id 't105)
                        (faz-subtarefa :dia 3 :hora 10 :id 't106)
                        (faz-subtarefa :dia 1 :hora 11 :id 't107)
                        (faz-subtarefa :dia 2 :hora 11 :id 't108)
                        (faz-subtarefa :dia 3 :hora 11 :id 't109)
                        ))))
```

TESTE EX6

```
; PROBLEMAS DO ANUNCIO DE DIA 22/NOVEMBRO
(DEFPARAMETER *EX6*
  (LIST ;DE TAREFAS
   (LIST  (LIST (FAZ-SUBTAREFA :DIA 1 :HORA 9 :ID 'T101)
                (FAZ-SUBTAREFA :DIA 2 :HORA 9 :ID 'T102)
                (FAZ-SUBTAREFA :DIA 3 :HORA 9 :ID 'T103)
                (FAZ-SUBTAREFA :DIA 1 :HORA 10 :ID 'T104)
                ;(FAZ-SUBTAREFA :DIA 2 :HORA 10 :ID 'T105)
                (FAZ-SUBTAREFA :DIA 3 :HORA 10 :ID 'T106)
                (FAZ-SUBTAREFA :DIA 1 :HORA 11 :ID 'T107)
                (FAZ-SUBTAREFA :DIA 2 :HORA 11 :ID 'T108)
                (FAZ-SUBTAREFA :DIA 3 :HORA 11 :ID 'T109)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 1 :HORA 19 :ID 'T01))
         (LIST (FAZ-SUBTAREFA :DIA 2 :HORA 19 :ID 'T02))
         (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 19 :ID 'T03)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 10 :ID 'T11))
         (LIST (FAZ-SUBTAREFA :DIA 4 :HORA 10 :ID 'T12))
         (LIST (FAZ-SUBTAREFA :DIA 5 :HORA 10 :ID 'T13)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 11 :ID 'T21))
         (LIST (FAZ-SUBTAREFA :DIA 4 :HORA 11 :ID 'T22))
         (LIST (FAZ-SUBTAREFA :DIA 5 :HORA 11 :ID 'T23)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 12 :ID 'T31))
         (LIST (FAZ-SUBTAREFA :DIA 4 :HORA 12 :ID 'T32))
         (LIST (FAZ-SUBTAREFA :DIA 5 :HORA 12 :ID 'T33)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 13 :ID 'T41))
         (LIST (FAZ-SUBTAREFA :DIA 4 :HORA 13 :ID 'T42))
         (LIST (FAZ-SUBTAREFA :DIA 5 :HORA 13 :ID 'T43)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 14 :ID 'T51))
         (LIST (FAZ-SUBTAREFA :DIA 4 :HORA 14 :ID 'T52))
         (LIST (FAZ-SUBTAREFA :DIA 5 :HORA 14 :ID 'T53)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 15 :ID 'T61))
         (LIST (FAZ-SUBTAREFA :DIA 4 :HORA 15 :ID 'T62))
         (LIST (FAZ-SUBTAREFA :DIA 5 :HORA 15 :ID 'T63)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 16 :ID 'T71))
         (LIST (FAZ-SUBTAREFA :DIA 4 :HORA 16 :ID 'T72))
         (LIST (FAZ-SUBTAREFA :DIA 5 :HORA 16 :ID 'T73)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 17 :ID 'T81))
         (LIST (FAZ-SUBTAREFA :DIA 4 :HORA 17 :ID 'T82))
         (LIST (FAZ-SUBTAREFA :DIA 5 :HORA 17 :ID 'T83)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 18 :ID 'T91))
         (LIST (FAZ-SUBTAREFA :DIA 4 :HORA 18 :ID 'T92))
         (LIST (FAZ-SUBTAREFA :DIA 5 :HORA 18 :ID 'T93)))
   (LIST (LIST (FAZ-SUBTAREFA :DIA 1 :HORA 9 :ID 'T1))
         (LIST (FAZ-SUBTAREFA :DIA 2 :HORA 9 :ID 'T2))
         (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 9 :ID 'T3))
         (LIST (FAZ-SUBTAREFA :DIA 1 :HORA 10 :ID 'T4))
         (LIST (FAZ-SUBTAREFA :DIA 2 :HORA 10 :ID 'T5))
         (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 10 :ID 'T6))
         (LIST (FAZ-SUBTAREFA :DIA 1 :HORA 11 :ID 'T7))
         (LIST (FAZ-SUBTAREFA :DIA 2 :HORA 11 :ID 'T8))
         (LIST (FAZ-SUBTAREFA :DIA 3 :HORA 11 :ID 'T9)))
))
```

ANEXO II
FUNÇÃO HEURÍSTICA #6

PSEUDOCÓDIGO DA HEURÍSTICA 6

```
function Heuristica6(Estado) {
  SubtarefasEstadoInicial <- RemoveSubtarefasDuplicadas(SubtarefasEstado(Estado));
  TarefasRestantes <- Tarefas para as quais ainda nao foram atribuidas alternativas;
  Heuristica6-Aux(SubtarefasEstadoInicial, SubtarefasEstadoInicial, TarefasRestantes);
}

function Heuristica6-Aux(SubtarefasEstadoInicial, SubtarefasEstado, TarefasRestantes) {
  CustoMínimo <- +inf;
  SubtarefasTudo <- SubtarefasEstado;
  TarefaActual <- PrimeiraTarefa(TarefasRestantes);
  AlternativasSemConflito <- NumAlternativas(TarefaActual);

  // condição de paragem - nao faltam tarefas
  if NumTarefas(TarefasRestantes) == 0 {
    return 0;
  }

  for each Alternativa in (PrimeiraTarefa(TarefasRestantes))
    SubtarefasAlternativa <- Subtarefas(Alternativa);
    SubtarefasAlternativaComEstado <- SubtarefasAlternativa U SubtarefasEstado
    SubtarefasTudo <- SubtarefasTudo U SubtarefasAlternativa;
    CustoAlternativa <- (NumSubtarefas(SubtarefasAlternativaComEstado) -
_____________________NumSubtarefas(SubtarefasEstado));
    if (CustoAlternativa < CustoMínimo) {
      CustoMínimo <- CustoAlternativa;
    }
    // vamos detectar se esta alternativa entra em conflito com o estado inicial
    if (HáConflitoEntreSubtarefas(SubtarefasEstadoInicial, SubtarefasAlternativa)) {
      Decrementa(AlternativasSemConflito);
      if (AlternativasSemConflito == 0) {
        // A partir daquele estado inicial nao ha solucoes!!!
        return +infinito;
      }
    }
  }

  ProxTarefasRestantes <- RemovePrimeiraTarefa(TarefasRestantes);
  return CustoMinimo + Heuristica6Aux(SubtarefasEstadoInicial, SubtarefasTudo, ProxTarefasRestantes);
}
```

ANEXO III
EXECUÇÃO

Formulação A, ex5

```
* (analisa-procuras *ex5*)

"PLPA"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PLPG"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PPPA"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PPPG"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PPIA"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PPIG"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PCUA"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PCUG"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PGA"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PGG"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PA*A"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"PA*G"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"RBFS"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
"CSP"
(((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
  (2 11 T108) (3 11 T109))
measuring PROFILE overhead..done
 ((3 12 T31)) ((4 11 T22)) ((4 10 T12)) ((1 19 T01)) ((2 10 T5)))
  seconds  |     gc     |   consed   | calls |  sec/call  |  name
-----------------------------------------------------------
     0.024 |      0.000 |  6,635,040 |     1 |   0.024000 | PCUG
     0.020 |      0.000 |  6,553,520 |     1 |   0.020000 | PCUA
     0.008 |      0.004 |    327,680 |     1 |   0.008000 | PA*A
     0.008 |      0.000 |  2,251,744 |     1 |   0.008000 | PLPG
     0.004 |      0.000 |  2,195,456 |     1 |   0.004000 | PLPA
     0.004 |      0.000 |     65,536 |     1 |   0.004000 | PPPA
     0.004 |      0.000 |     65,536 |     1 |   0.004000 | RBFS
     0.004 |      0.000 |    128,624 |     7 |   0.000571 | PPLG
     0.000 |      0.000 |     32,768 |     1 |   0.000000 | PGA
     0.000 |      0.000 |    131,072 |     1 |   0.000000 | PGG
     0.000 |      0.000 |    458,096 |     1 |   0.000000 | PA*G
     0.000 |      0.000 |    128,768 |     1 |   0.000000 | PPPG
     0.000 |      0.000 |     65,536 |     7 |   0.000000 | PPLA
     0.000 |      0.000 |          0 |     1 |   0.000000 | PPIA
     0.000 |      0.000 |          0 |     1 |   0.000000 | PPIG
     0.000 |      0.000 |          0 |     1 |   0.000000 | PSR
-----------------------------------------------------------
     0.076 |      0.004 | 19,039,376 |    28 |            | Total
```

Formulação B, ex5:

```
* (analisa-procuras-cegas *ex5*)

"PLPA"
((5
  ((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
   (2 11 T108) (3 11 T109)))
 (4 ((3 12 T31))) (3 ((4 11 T22))) (2 ((4 10 T12))) (1 ((1 19 T01)))
 (0 ((2 10 T5))))
"PLPG"
((5
  ((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
   (2 11 T108) (3 11 T109)))
 (4 ((3 12 T31))) (3 ((4 11 T22))) (2 ((4 10 T12))) (1 ((1 19 T01)))
 (0 ((2 10 T5))))
"PPPA"
((5
  ((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
   (2 11 T108) (3 11 T109)))
 (4 ((3 12 T31))) (3 ((4 11 T22))) (2 ((4 10 T12))) (1 ((1 19 T01)))
 (0 ((2 10 T5))))
"PPPG"
((5
  ((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
   (2 11 T108) (3 11 T109)))
 (4 ((3 12 T31))) (3 ((4 11 T22))) (2 ((4 10 T12))) (1 ((1 19 T01)))
 (0 ((2 10 T5))))
"PPIA"
((5
  ((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
   (2 11 T108) (3 11 T109)))
 (4 ((3 12 T31))) (3 ((4 11 T22))) (2 ((4 10 T12))) (1 ((1 19 T01)))
 (0 ((2 10 T5))))
"PPIG"
((5
  ((1 9 T101) (2 9 T102) (3 9 T103) (1 10 T104) (3 10 T106) (1 11 T107)
   (2 11 T108) (3 11 T109)))
 (4 ((3 12 T31))) (3 ((4 11 T22))) (2 ((4 10 T12))) (1 ((1 19 T01)))
 (0 ((2 10 T5))))
  seconds  |     gc     |     consed     | calls |  sec/call  |  name
-----------------------------------------------------------
    30.189 |      1.556 | 22,524,259,616 |     1 |  30.189000 | PLPG
     8.893 |      1.324 | 22,513,759,424 |     1 |   8.893000 | PLPA
     0.825 |      0.008 |    104,547,040 |     1 |   0.825000 | PPPG
     0.804 |      0.008 |    104,618,560 |     7 |   0.114857 | PPLG
     0.088 |      0.004 |    103,307,200 |     7 |   0.012571 | PPLA
     0.084 |      0.004 |    103,211,120 |     1 |   0.084000 | PPPA
     0.000 |      0.000 |              0 |     1 |   0.000000 | PPIG
     0.000 |      0.000 |              0 |     1 |   0.000000 | PPIA
-----------------------------------------------------------
    40.883 |      2.904 | 45,453,702,960 |    20 |            | Total
```