Instituto Superior Técnico
Inteligência Artificial
1º Semestre 2012/2013

Planificação de um Período de Trabalho

Ricardo Leitão nº69632
Diogo Pinto nº 69905
Fábio Almeida nº70227

Introdução

Este relatório tem como objectivo registar o estudo realizado relativo à utilização de dois métodos de procura diferentes (procura em espaço de estados e a satisfação de restrições) aplicados à resolução de dois problemas diferentes da planificação de um período de trabalho. Os dois problemas a estudar consistem em escolher, de um conjunto de tarefas que se querem executar e do conjunto de alternativas para executar cada uma das tarefas, escolher um conjunto de alternativas que permitem realizar todas as tarefas pretendidas.

No primeiro problema procuramos as alternativas adequadas para conseguir executar o conjunto das tarefas, sem que tarefas distintas sejam executas simultaneamente. No segundo problema procuramos as alternativas adequadas para minimizar o tempo necessário para executar as tarefas, permitindo que tarefas distintas sejam executadas simultaneamente, no caso de existirem subtarefas em comum.

Para estudarmos a procura em espaço de estados vamos utilizar os algoritmos de Procura Não Informada (Procura em Profundidade Primeiro (PPP), Procura em profundidade Limitada (PPL), Procura em Profundidade Iterativa (PPI), Procura em Largura Primeiro (PLP) e Procura com Custo Uniforme (PCU), todas em árvore e em grafo) e procura informada (Procura Gananciosa (PG), Procura A* (PA*) em árvore e em grafo e a Procura RBFS em árvore) e para a satisfação de restrições vamos utilizar o algoritmo de procura com retrocesso.

Para simplificar iremos dividir o relatório em três secções, sendo estas correspondentes à abordagem utilizada na interpretação da tarefa a desempenhar e aos dois problemas a estudar. Nas duas secções correspondentes ao estudo dos dois problemas iremos abordar dois métodos de procura: começando pela procura em espaço de estados, passando depois à satisfação de restrições.

Abordagem do Problema

Visto que se trata de um projecto cujas entradas e saídas são listas, decidimos manter a representação de todos os dados necessários à resolução do problema guardados em listas ao invés da representação por nós. Assim sendo, o que identifica um estado, no nosso projecto, é uma lista que corresponde à concatenação de todas as acções realizadas até ao momento. Apesar de ser necessário a utilização de alguma abstracção (criação de funções auxiliares que manipulam as listas) e a introdução de alguns ciclos extra aquando o cálculo do custo de um caminho, pareceu-nos a maneira mais correcta e eficaz de implementar os nossos algoritmos. Assim não só mantemos os dados como nos são fornecidos, como também simplificamos o retorno dos resultados.

Formulações do problema utilizadas

Formulação Problema – Corresponde à segunda formulação apresentada no enunciado em que a cada passo do algoritmo se tenta executar apenas uma das tarefas, ou seja, qualquer que seja o estado em que o algoritmo se encontre, esta formulação devolve apenas as alternativas correspondentes à próxima tarefa.

Formulação Problema1 – Corresponde à primeira formulação apresentada no enunciado do projecto, em que a cada passo do algoritmo se tenta executar qualquer uma das tarefas com qualquer uma das alternativas, ou seja, qualquer que seja o estado em que o algoritmo se encontre, esta formulação devolve sempre as mesmas acções a executar.

Heurísticas utilizadas

Heurística Admissível – Esta heurística vai realizar uma previsão optimista relativamente ao custo necessário para executar todas as tarefas restantes, o que corresponde na prática a escolher a alternativa com menos subtarefas a realizar para cada tarefa, sendo o valor da heurística o número de subtarefas a realizar.

Heurística Não admissível – Esta heurística vai realizar uma previsão pessimista relativamente ao custo necessário para executar todas as tarefas restantes, o que corresponde na prática a escolher a alternativa com mais subtarefas a realizar para cada tarefa, sendo o valor da heurística o maior número de subtarefas a realizar.

Heurística Consistente – Esta heurística vai realizar uma previsão optimista relativamente ao custo necessário para executar as restantes tarefas, o que corresponde na prática a escolher a alternativa com menos subtarefas a realizar para cada tarefa, sendo o valor da heurística o número de subtarefas distintas a realizar. Esta heurística é assim simultaneamente admissível e consistente visto que além de realizar uma previsão optimista (admissível), vai também fornecer o "caminho" com menor custo até à solução.

Testes em estudo

Teste1 – Este exercício corresponde ao exercício 5 sem comentários disponibilizado nos testes da página da cadeira, onde temos 12 tarefas a realizar, cada uma dela com um número variável de alternativas e em que a última tarefa a realizar condiciona muitas das primeiras tarefas, devido a potenciais sobreposições de horário. Assim, este teste representa um caso em que são necessárias muitas comparações entre alternativas de diferentes tarefas pois existem muitas sobreposições.

Teste2 – Este exercício corresponde ao exercício 6 disponibilizado nos testes da página da cadeira, onde temos as mesmas tarefas e alternativas do teste EX1 mas encontram-se apresentadas com uma ordem diferente, onde a maioria dos potenciais conflitos são

desencadeados pela primeira tarefa. Assim, nos algoritmos de procura, as sobreposições de horário são encontradas logo nas primeiras iterações.

Teste3 – Este exercício, da nossa autoria, corresponde a um conjunto de tarefas que representam um horário baseado no 3º ano do curso de LEIC. Assim, temos um exemplo de uma aplicação prática do projecto na planificação de um horário real. Este exercício combina a presença em aulas teóricas e práticas de 4 cadeiras diferentes, sem deixar de ter em conta que o aluno necessita de se deslocar de casa para a escola e vice-versa. O código correspondente a este exemplo encontra-se no final do ficheiro LISP submetido.

Primeiro Problema

Procuras não-informadas

Apresentam-se de seguida os resultados das procuras não-informadas, fazendo variar a formulação problema:

	TESTE1	
	FP	FP1
PPPA	Real Time=5.591s Run Time=5.584s	>1000s
PPPG	Real Time=5.592s Run Time=5.584s	>1000s
PPLA	Real Time=5.586s Run Time=5.580s	>1000s
PPLG	Real Time=5.594s Run Time=5.584s	>1000s
PPIA	Real Time=12.460s Run Time=12.445s	>1000s
PPIG	Real Time=12.469s Run Time=12.453s	>1000s
PLPA	Real Time=370.714s Run Time=370.271s	>1000s
PLPG	Real Time=367.081s Run Time=366.651s	>1000s
PCUA	Real Time=8.790s Run Time=8.781s	>1000s
PCUG	Real Time=6.651s Run Time=6.644s	>1000s

Tab.1 - Algoritmos de procura não informada aplicados ao teste1

	TESTE2	
	FP	FP1

PPPA	Real Time=0.001s Run Time=0.000s	>1000s
PPPG	Real Time=0.001s Run Time=0.000s	>1000s
PPLA	Real Time=0.001s Run Time=0.000s	>1000s
PPLG	Real Time=0.001s Run Time=0.000s	>1000s
PPIA	Real Time=0.496s Run Time=0.496s	>1000s
PPIG	Real Time=0.494s Run Time=0.492s	>1000s
PLPA	Real Time=5.524s Run Time=5.516s	>1000s
PLPG	Real Time=5.713s Run Time=5.704s	>1000s
PCUA	Real Time=0.008s Run Time=0.008s	>1000s
PCUG	Real Time=0.009s Run Time=0.008s	>1000s

Tab.2 - Algoritmos de procura não informada aplicados ao teste2

	TESTE3	
	FP	FP1
PPPA	Real Time=0.001s Run Time=0.000s	>1000s
PPPG	Real Time=0.001s Run Time=0.000s	>1000s
PPLA	Real Time=0.001s Run Time=0.000s	>1000s
PPLG	Real Time=0.001s Run Time=0.000s	>1000s
PPIA	Real Time=0.024s Run Time=0.024s	>1000s
PPIG	Real Time=0.024s Run Time=0.024s	>1000s
PLPA	Real Time=0.062s Run Time=0.060s	>1000s
PLPG	Real Time=0.062s Run Time=0.060s	>1000s
PCUA	Real Time=0.004s Run Time=0.004s	>1000s
PCUG	Real Time=0.004s Run Time=0.004s	>1000s

Tab.3 - Algoritmos de procura não informada aplicados ao teste3

Análise dos resultados obtidos

Os resultados obtidos foram os esperados, dadas as formulações-problema utilizadas. No caso da formulação-problema1, não se obteve nenhum resultado em tempo útil pelo facto de nesta formulação, estarmos a tentar realizar todas as acções em qualquer estado, o que corresponde a uma quantidade enorme de estados que se tentam expandir, levando a um elevado tempo de execução.

No caso da formulação-problema, os tempos de execução são tão baixos porque em cada estado se tenta apenas realizar as acções de uma tarefa ainda não realizada, o que baixa drasticamente a quantidade de estados que se tenta expandir.

Relativamente à formulação-problema, concluímos que as procuras em profundidade são as mais eficientes, encontrando mais rapidamente uma solução, embora esta possa não ser a solução óptima. Para obtermos a solução necessariamente óptima, é necessário recorrer ao algoritmo de procura com custo uniforme, visto esta expandir os nós por ordem crescente de custo.

Relativamente às procuras em grafo, no problema de planificação de um período de trabalho, estas não apresentam uma importância substancial visto que não existem ciclos no espaço de estados do problema. Além disso, pelo facto de em cada iteração dos algoritmos de procura em grafo ser necessário verificar se o estado a ser expandido já se encontra nos estados fechados, o tempo de execução vai aumentar.

Em relação aos diferentes tempos de execução verificados nos diferentes testes, estes podem ser explicados pela complexidade dos mesmos. No caso dos testes 1 e 2, apesar da complexidade ser semelhante, a ordem pela qual as tarefas são apresentadas influência drasticamente a eficiência do algoritmo, visto que num dos casos, a acção que gera maior número de conflitos é a primeira a ser tomada, o que vai evitar que ocorram tantos retrocessos na árvore de procura.

Em última análise, concluímos que, caso se queira uma resposta mais rapidamente possível, a melhor procura é a procura em profundidade primeiro. Caso se queira a resposta óptima, ou seja, a com menos subtarefas a desenvolver, a melhor procura é a procura com custo uniforme. A procura em largura é claramente inadequada para este tipo de problema porque além do elevado tempo de execução, não garante uma solução óptima.

Procuras informadas

Apresentam-se de seguida os resultados das procuras informadas, fazendo variar as funções heurísticas:

	TESTE1		
	Heurística Não Admissível	Heurística Admissível	Heurística Consistente
PGA	Real Time=10.443 Run Time=10.433	Real Time=9.361 Run Time=9.349	Real Time=9.643 Run Time=9.625
PGG	Real Time=7.718 Run Time=7.712	Real Time=7.480 Run Time=7.472	Real Time=8.076 Run Time=8.065
PA*A	Real Time=9.583 Run Time=9.525	Real Time=9.785 Run Time=9.773	Real Time=9.391 Run Time=9.381
PA*G	Real Time=7.661 Run Time=7.652	Real Time=7.685 Run Time=7.676	Real Time=7.723 Run Time=7.724
RBFS	Real Time=30.266 Run Time=30.230	Real Time=30.313 Run Time=30.274	Real Time=43.845 Run Time=43.791

Tab.4 - Algoritmos de procura informada aplicados ao teste1

	TESTE2		
	Heurística Não Admissível	Heurística Admissível	Heurística Consistente
PGA	Real Time=0.008 Run Time=0.008	Real Time=0.008 Run Time=0.008	Real Time=0.008 Run Time=0.008
PGG	Real Time=0.009 Run Time=0.008	Real Time=0.009 Run Time=0.008	Real Time=0.009 Run Time=0.008
PA*A	Real Time=0.008 Run Time=0.008	Real Time=0.008 Run Time=0.008	Real Time=0.008 Run Time=0.008
PA*G	Real Time=0.009 Run Time=0.008	Real Time=0.009 RunTime=0.008	Real Time=0.009 Run Time=0.008
RBFS	Real Time=0.011 Run Time=0.012	Real Time=0.011 RunTime=0.012	Real Time=0.013 Run Time=0.012

Tab.5 - Algoritmos de procura informada aplicados ao teste2

	TESTE3		
	Heurística Não Admissível	Heurística Admissível	Heurística Consistente
PGA	Real Time=0.004 Run Time=0.004	Real Time=0.004 Run Time=0.004	Real Time=0.004s Run Time=0.004s
PGG	Real Time=0.004 Run Time=0.004	Real Time=0.004 Run Time=0.004	Real Time=0.004s Run Time=0.004s
PA*A	Real Time=0.004 Run Time=0.004	Real Time=0.004 Run Time=0.004	Real Time=0.004s Run Time=0.004s
PA*G	Real Time=0.004 Run Time=0.004	Real Time=0.004 RunTime=0.004	Real Time=0.004s Run Time=0.004s
RBFS	Real Time=0.007 Run Time=0.008	Real Time=0.008 Run Time=0.007	Real Time=0.013s Run Time=0.012

Tab.6 - Algoritmos de procura informada aplicados ao teste3

Análise dos resultados obtidos

A formulação utilizada nos testes das procuras informadas foi a formulação-problema onde em cada estado se tenta apenas realizar as acções de uma tarefa ainda não realizada.

Relativamente à heurística não admissível testada, esta só nos garante que não estamos a ser optimistas na procura da solução. Assim, a procura não será óptima para nenhum dos algoritmos de procura pois não encontra a melhor solução, realizando sempre estimativas pessimistas do custo até chegar ao estado objectivo, visto que estima que se realiza sempre a alternativa com mais subtarefas. Em todos os testes foram obtidos valores satisfatórios de complexidade temporal, embora as soluções a que os mesmos chegam não sejam óptimas.

Para a heurística admissível, esperamos encontrar soluções necessariamente óptimas na procura A* em árvore e na procura RBFS. Efectivamente, encontraram-se soluções óptimas utilizando a procura A* em árvore em todos os testes bem como para a procura RBFS, embora normalmente a complexidade temporal da procura A* seja menor que a da procura RBFS. Esta diferença de complexidade temporal também se deve à implementação de cada um dos algoritmos, em que no caso da procura RBFS é necessário uma maior manipulação de dados. Na procura gananciosa também se encontra sempre solução em tempo útil embora esta solução possa não ser a óptima.

Para a heurística consistente, vamos encontrar as soluções óptimas tanto nas procuras A* em árvore como em grafo, bem como na procura RBFS. A procura gananciosa também encontra sempre solução em tempo útil embora não seja uma solução óptima.

Deste conjunto de testes utilizando a procura informada, concluímos que podemos obter resultados óptimos através da procura A* em árvore usando uma heurística admissível e através da procura A* em árvore e em grafo e da RBFS usando uma heurística consistente. Concluímos ainda que mesmo variando as heurísticas, não temos a garantia de que a procura gananciosa devolva uma solução óptima.

À semelhança do que ocorreu nas procuras não informadas, o teste mais complexo de resolução foi o teste1, embora tenhamos obtido solução em todas as procuras (árvores e grafos). Todos os outros testes têm uma complexidade temporal mais baixa, situada nas milésimas de segundo.

Satisfação de Restrições

Para implementarmos a resolução do problema de satisfação de restrições, utilizámos o método de procura com retrocesso recursivo, em que as variáveis são tantos valores zero como as tarefas na lista de tarefas a realizar. A cada chamada ao algoritmo é atribuída à

primeira variável a zero a alternativa escolhida no domínio da variável, que corresponde ao conjunto de alternativas da tarefa a realizar. É de seguida verificado se as seguintes restrições são satisfeitas: i) ALLDIFF(V1,…,Vn) em que V1,…,Vn correspondem às variáveis atribuídas do problema; ii) Tarefas iguais têm necessariamente os mesmos valores nos seus campos;iii)Não existem sobreposições nas tarefas. Apresenta-se de seguida os resultados da aplicação do problema de satisfação de restrições aos nossos testes:

Problema de satisfação de restrições	
Teste1	>1000s
Teste2	Real Time=0.001 Run Time=0.000
Teste3	Real Time=0.002 Run Time=0.000

Tab.7 – Problema de satisfação de restrições aplicado aos 3 testes

Análise dos resultados obtidos

Para o Teste1, não obtemos resposta em tempo útil. Isto é devido à elevada complexidade do problema teste1, que possui muitas tarefas para realizar e que a tarefa que desencadeia um maior número de conflitos é a última tarefa a ser analisada pelo algoritmo, o que faz com que hajam grandes retrocessos, ficando a procura muito pesada em termos de complexidade temporal. Além disso, temos ainda que quanto maior o número de tarefas, maior o número de variáveis a atribuir e por conseguinte maior o tempo de execução do algoritmo.

Para o Teste2 temos a situação contrária à reportada no teste1. Obtemos resposta em 0.001 segundos pois a primeira variável a ser atribuída corresponde à primeira tarefa, que como explicado anteriormente, desencadeia o maior número de conflitos. Assim, grande parte dos conflitos de atribuição de variáveis vai ocorrer no início do algoritmo, não havendo grandes retrocessos no algoritmo, o que vai fazer diminuir a sua complexidade temporal.

Para o Teste3 temos a representação de um caso real da construção de um horário e à semelhança do que foi estudado nas aulas teóricas da disciplina, os problemas de satisfação de restrições que envolvem planificações de períodos de trabalho são facilmente satisfeitos através de atribuições de variáveis e procuras com retrocesso. Assim, o Teste3, que corresponde à calendarização de um horário prático de um aluno do 3º ano do curso de LEIC do IST, tem uma complexidade temporal baixa quando utilizamos a satisfação de restrições para o resolver (na ordem dos milésimos de segundo).

Segundo Problema

No estudo do segundo problema, pretende-se 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.

Assim, pretende-se, de forma simples, encontrar o conjunto de tarefas que permita ao utilizador passar menos tempo a executar tarefas. Existem duas formas de abordar este problema. A primeira forma consiste em minimizar ao máximo o intervalo entre tarefas, ou seja, no caso de um horário escolar, pretender-se-ia ter o mínimo de "furos" no horário. A segunda forma consiste em escolher as subtarefas comuns entre várias tarefas, ou seja, para duas tarefas (no caso do nosso problema representadas pelo mesmo dia, hora e identificador) diferentes, se existe a mesma subtarefa em cada uma delas, o utilizador só a vai executar uma vez. No nosso caso, vamos apenas estudar a segunda forma e considerar o custo de caminho com o número de subtarefas distintas a desenvolver em cada tarefa.

Como seria de esperar, as procuras em profundidade (profundidade primeiro, profundidade limitada, profundidade iterativa) e largura (largura primeiro) não são capazes de resolver este tipo de problema, visto se tratarem de procuras cegas que não têm qualquer tipo de noção de custo de caminho ou do número de tarefas a desenvolver. Assim, estas procuras cegas vão apenas basear-se na árvore de procura que lhes é fornecida para obter resultados.

A primeira procura que consegue, de forma relativamente satisfatória, resolver este problema é a procura de custo uniforme, já que expande os nós por ordem crescente de custo e embora seja uma procura cega, já tem uma mínima noção de qual o nó com menor custo que pode expandir. Porém, esta procura é ainda algo limitada visto se tratar de uma procura cega e não seguir sempre o caminho de menor custo.

As procuras que efectivamente resolvem este problema são as procuras informadas, que, com acesso a uma boa heurística, conseguem minimizar o problema ao máximo, podendo escolher sempre o nó com o menor custo.

Relativamente à procura gananciosa, esta procura tem apenas a noção de uma estimativa do custo do estado actual até ao objectivo, expandindo sempre o nó que parece mais perto do objectivo, o que pode fazer com que caminhe para uma situação de custo elevado sem se aperceber disso.

Considerando agora a procura A*, temos que esta procura é óptima quando se utiliza a sua variação em árvore e a heurística utilizada é uma heurística admissível, isto é, que nunca sobrestima o custo. Assim, esta procura vai conseguir minimizar o problema quando associada a uma heurística admissível.

Porém, o problema de minimização é verdadeiramente resolvido quando são utilizadas heurísticas consistentes, ou seja, uma heurística que garante que se existem dois caminhos para chegar ao mesmo objectivo óptimo, então o caminho de menor custo é sempre seguido em primeiro lugar. Utilizando esta heurística, temos que as procuras A* em árvore e em grafo e a RBFS são necessariamente óptimas, conseguindo assim minimizar o problema da melhor maneira.

Note-se assim que é possível resolver este problema de minimização variando somente a noção de custo caminho, ou seja, no caso de considerarmos o custo caminho como o número de subtarefas distintas a realizar, esta noção de custo já minimiza por si mesmo o problema. Assim, todas as procuras que utilizam a noção de custo, ou seja, a procura com custo uniforme e as procuras informadas são capazes de minimizar o problema, embora sejam as informadas que, variando a heurística, conseguem minimizar o problema da melhor maneira. As melhores minimizações correspondem à utilização de uma heurística consistente nas procuras A* e RBFS, que as tornam necessariamente óptimas (encontram a solução de melhor custo).

Conclusão

Depois do estudo realizado, vamos apresentar de seguida as principais conclusões retiradas relativamente aos dois problemas estudados.

No estudo do 1º problema, em que procuramos apenas uma solução para executar um conjunto de tarefas, sem que existam sobreposições de horário em tarefas, verificou-se que todos os algoritmos de procura encontravam soluções em tempo útil. Nas procuras não informadas, as melhores procuras são em profundidade primeiro. Porém, a procura em profundidade primeiro e as suas variações não possuem qualquer tipo noção do número de subtarefas a realizar, não encontrando uma boa solução. A procura de custo uniforme já permite obter soluções óptimas pois possui uma noção de custo do caminho. Verificou-se ainda a extrema importância da formulação do problema, tendo-se verificado que para uma formulação problema que procura expandir muitos estados simultaneamente, não se vai obter soluções em tempo útil.

Na utilização de procuras informadas, conclui-se a extrema importância das heurísticas utilizadas, sendo estas que determinam se chegamos a uma solução necessariamente óptima ou não. Assim, concluímos que os melhores algoritmos de procura informada são o A* e RBFS quando associados a heurísticas consistentes.

No problema de satisfação de restrições, podemos obter também soluções para este problema, embora não se obtenha resposta em tempo útil quando os problemas são muito complexos, tendo muitos conflitos. Pode-se ainda associar algoritmos que permitam melhorar a procura com retrocesso, como o forward-checking ou o AC-3.

Relativamente ao estudo do 2ª problema, em que procuramos minimizar o problema, ou seja, encontrar a solução que corresponde a realizar o menor número de subtarefas em cada tarefa, as únicas procuras que permitem resolver este problema são a procura com custo uniforme e as informadas, visto terem em conta o custo do caminho. Embora estas procuras resolvam o problema, algumas são mais eficientes, nomeadamente as procuras A* e RBFS com heurísticas consistentes, em que se garante que encontram a solução óptima.

Foi ainda concluído, na realização deste estudo, que a variação das procuras em grafo não é muito relevante visto não possuirmos ciclos no nosso espaço de estados e embora diminuamos a complexidade temporal e espacial por expandirmos menos nós, estamos também a aumentar a complexidade espacial visto ser necessário manter o conjunto de nós fechados em memória e percorrê-lo sempre que avaliamos um novo estado.

Anexos

Apresentamos agora os testes 1, 2 e 3 realizados para a elaboração deste relatório:

```
(defvar *teste1*  (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)))))


(defvar *teste2*  (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)))))
```

```
(defvar *teste3* (list
        (list ;Aulas Teóricas IA
                (list
                        (faz-subtarefa :dia 2 :hora 8 :id 'IRIST2)
                        (faz-subtarefa :dia 2 :hora 10 :id 'TIA1)
                        (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                        (faz-subtarefa :dia 4 :hora 10 :id 'TIA2))
                (list
                        (faz-subtarefa :dia 2 :hora 8 :id 'IRIST)
                        (faz-subtarefa :dia 2 :hora 11 :id 'TIA3)
                        (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                        (faz-subtarefa :dia 4 :hora 11 :id 'TIA4)))
        (list ;Aulas Práticas IA
                (list
                        (faz-subtarefa :dia 2 :hora 8 :id 'IRIST)
                        (faz-subtarefa :dia 2 :hora 10 :id 'PIA1))
                (list
                        (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                        (faz-subtarefa :dia 3 :hora 9 :id 'PIA2))
                (list
                        (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                        (faz-subtarefa :dia 3 :hora 12 :id 'PIA3))
                (list
                        (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                        (faz-subtarefa :dia 4 :hora 10 :id 'PIA4))
                (list
                        (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                        (faz-subtarefa :dia 4 :hora 11 :id 'PIA5))
                (list
                        (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                        (faz-subtarefa :dia 4 :hora 12 :id 'PIA6)))
        (list ;Aulas Teóricas BD
                (list
                        (faz-subtarefa :dia 2 :hora 8 :id 'IRIST)
                        (faz-subtarefa :dia 2 :hora 10 :id 'TBD1)
                        (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                        (faz-subtarefa :dia 4 :hora 10 :id 'TBD2))
                (list
                        (faz-subtarefa :dia 2 :hora 8 :id 'IRIST)
```

```
                (faz-subtarefa :dia 2 :hora 11 :id 'TBD3)
                (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                (faz-subtarefa :dia 4 :hora 11 :id 'TBD4)))
(list ;Aulas Práticas BD
        (list
                (faz-subtarefa :dia 1 :hora 8 :id 'IRIST1)
                (faz-subtarefa :dia 1 :hora 9 :id 'PBD1))
        (list
                (faz-subtarefa :dia 1 :hora 8 :id 'IRIST1)
                (faz-subtarefa :dia 1 :hora 12 :id 'PBD2))
        (list
                (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                (faz-subtarefa :dia 3 :hora 9 :id 'PBD3))
        (list
                (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                (faz-subtarefa :dia 3 :hora 12 :id 'PBD4))
        (list
                (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                (faz-subtarefa :dia 4 :hora 9 :id 'PBD5))
        (list
                (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                (faz-subtarefa :dia 4 :hora 11 :id 'PBD6))
        (list
                (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                (faz-subtarefa :dia 4 :hora 12 :id 'PBD7))
        (list
                (faz-subtarefa :dia 5 :hora 8 :id 'IRIST5)
                (faz-subtarefa :dia 5 :hora 10 :id 'PBD8)))
(list ;Aulas Teóricas IPM
        (list
                (faz-subtarefa :dia 1 :hora 8 :id 'IRIST1)
                (faz-subtarefa :dia 1 :hora 10 :id 'TIPM1)
                (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                (faz-subtarefa :dia 3 :hora 10 :id 'TIPM2))
        (list
                (faz-subtarefa :dia 1 :hora 8 :id 'IRIST1)
                (faz-subtarefa :dia 1 :hora 11 :id 'TIPM3)
                (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                (faz-subtarefa :dia 3 :hora 11 :id 'TIPM4)))
(list ;Aulas Práticas IPM
        (list
                (faz-subtarefa :dia 1 :hora 8 :id 'IRIST1)
                (faz-subtarefa :dia 1 :hora 12 :id 'PIPM1))
        (list
                (faz-subtarefa :dia 2 :hora 8 :id 'IRIST)
                (faz-subtarefa :dia 2 :hora 12 :id 'PIPM2))
        (list
                (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                (faz-subtarefa :dia 3 :hora 9 :id 'PIPM3))
        (list
```

```
                (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                (faz-subtarefa :dia 3 :hora 12 :id 'PIPM4))
        (list
                (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                (faz-subtarefa :dia 4 :hora 10 :id 'PIPM5))
        (list
                (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                (faz-subtarefa :dia 4 :hora 12 :id 'PIPM6)))
(list ;Aulas Teóricas RC
        (list
                (faz-subtarefa :dia 1 :hora 8 :id 'IRIST1)
                (faz-subtarefa :dia 1 :hora 10 :id 'TRC1)
                (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                (faz-subtarefa :dia 3 :hora 10 :id 'TRC2))
        (list
                (faz-subtarefa :dia 1 :hora 8 :id 'IRIST1)
                (faz-subtarefa :dia 1 :hora 11 :id 'TRC3)
                (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                (faz-subtarefa :dia 3 :hora 11 :id 'TRC4)))
(list ;Aulas Práticas RC
        (list
                (faz-subtarefa :dia 1 :hora 8 :id 'IRIST1)
                (faz-subtarefa :dia 1 :hora 12 :id 'PRC1))
        (list
                (faz-subtarefa :dia 2 :hora 8 :id 'IRIST)
                (faz-subtarefa :dia 2 :hora 9 :id 'PRC2))
        (list
                (faz-subtarefa :dia 2 :hora 8 :id 'IRIST)
                (faz-subtarefa :dia 2 :hora 10 :id 'PRC3))
        (list
                (faz-subtarefa :dia 2 :hora 8 :id 'IRIST)
                (faz-subtarefa :dia 2 :hora 12 :id 'PRC4))
        (list
                (faz-subtarefa :dia 2 :hora 8 :id 'IRIST)
                (faz-subtarefa :dia 2 :hora 13 :id 'PRC5))
        (list
                (faz-subtarefa :dia 3 :hora 8 :id 'IRIST3)
                (faz-subtarefa :dia 3 :hora 12 :id 'PRC6))
        (list
                (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                (faz-subtarefa :dia 4 :hora 10 :id 'PRC7))
        (list
                (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                (faz-subtarefa :dia 4 :hora 11 :id 'PRC8))
        (list
                (faz-subtarefa :dia 4 :hora 8 :id 'IRIST4)
                (faz-subtarefa :dia 4 :hora 12 :id 'PRC9)))))
```

Características do sistema em que os testes foram corridos:

- INTEL Core 2 Duo T7500 @ 2.20GHZ
- 2 GB RAM
- Sistema operativo LINUX MINT