**Autores:**  
José Pedro Ribeiro Ferreira Pinto – 201603713

Luís Miguel dos Santos Magalhães - 201606761

**Relatório Do Trabalho Prático**

Satisfação de restrições em domínios finitos e problemas de otimização combinatória

Métodos de Apoio à Decisão 2018/2019

Índice de conteúdos

[Introdução 2](#_Toc9088034)

[Problema 1 3](#_Toc9088035)

[Pergunta 1 3](#_Toc9088036)

[Modelo 3](#_Toc9088037)

[Resolução do Problema 3](#_Toc9088038)

[Analise Comparativa de Performance 4](#_Toc9088039)

[Pergunta 2 4](#_Toc9088040)

[Resolução do Problema 5](#_Toc9088041)

[Analise Comparativa de Performance 5](#_Toc9088042)

[Pergunta 3 6](#_Toc9088043)

[Resolução do Problema 6](#_Toc9088044)

[Analise Comparativa de Performance 6](#_Toc9088045)

[Pergunta 4 7](#_Toc9088046)

[Modelo 7](#_Toc9088047)

[Resolução do Problema 8](#_Toc9088048)

[Analise Comparativa de Performance 9](#_Toc9088049)

[Conclusões 9](#_Toc9088050)

[Problema 2 10](#_Toc9088051)

[Modelo 10](#_Toc9088052)

[Resolução do Problema 13](#_Toc9088053)

# Introdução

O objetivo do trabalho é desenvolver programas que resolvem problemas de satisfação de restrições em domínios finitos, no caso particular de problemas de calendarização com partilha de recursos (scheduling problems). Os problemas devem ser resolvidos em eclipse prolog com o auxílio dos módulos “ic”.

No caso do primeiro problema também devem ser resolvidos numa outra linguagem (sem auxílio de ferramentas especificas) para se poder comparar a dificuldade de desenvolvimento e as diferenças em termos de performance.

# Problema 1

## Pergunta 1

Qual é a duração mínima do projeto?

O objetivo é calcular a duração mínima do projeto e as datas de início de cada tarefa, tendo em consideração as restrições de precedência entre tarefas.

Para responder a esta pergunta o número de trabalhadores de cada tarefa não é relevante.

### Modelo

**Objetivo**

Minimizar Concl (Data de conclusão do projeto)

**Dados**

Tarefas (conjuntos de tarefas)

di, com i ϵ Tarefas (duração tarefa i)

(i,j) ϵ Precedência (a tarefa i precede a tarefa j, i.e a tarefa j apenas pode começar quando a tarefa i terminar)

**Variáveis de Decisão**

Concl (data de conclusão do projeto)

Di, com i ϵ Tarefas (data de início da tarefa i)

**Sujeito a:**

Di + di ≤ Dj

Di + di ≤ Concl

### Resolução do Problema

**Em Prolog:**

Obter a lista de tarefas compilando um ficheiro que serve de base de dados.

Restringir Di de cada tarefa entre 0 e a soma das durações de todas as tarefas.

Fazer restrições de precedência, cada tarefa tem de começar apenas depois de uma tarefa que a precede acaba. Simultaneamente restringir o domínio da conclusão, a conclusão é maior ou igual ao máximo dos tempos em que cada tarefa acaba.

Apos ter as restrições nos tempos de início e no tempo de conclusão do projeto usar o predicado “minimize” para obter o tempo mínimo de conclusão.

**Em Java:**

Usar um simples algoritmo de ordenação topológica sobre o grafo de precedências, adicionando um nó final fictício para considerar as durações das tarefas finais.

### Analise Comparativa de Performance

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Prolog** | 1ª medição | 2ª medição | 3ª medição | Média | Resultados |
| SmallDB1(10 tarefas) | 0ms | 0ms | 0ms | 0ms | 29 |
| SmallDB2(10 tarefas) | 0ms | 0ms | 0ms | 0ms | 24 |
| SmallDB3(10 tarefas) | 0ms | 0ms | 0ms | 0ms | 29 |
| MediumDB1(75 tarefas) | 20ms | 20ms | 20ms | 20ms | 234 |
| MediumDB2(75 tarefas) | 20ms | 20ms | 20ms | 20ms | 155 |
| MediumDB3(75 tarefas) | 0ms | 20ms | 20ms | 13ms | 195 |
| LargeDB1(500 tarefas) | 1.22s | 1.22s | 1.22s | 1.22s | 1257 |
| LargeDB2(500 tarefas) | 1.13s | 1.13s | 1.11s | 1.13s | 1275 |
| LargeDB3(500 tarefas) | 1.27s | 1.28s | 1.28s | 1.28s | 1421 |
| Ex1(13 tarefas) | 0ms | 0ms | 0ms | 0ms | 21 |
| Ex2\_1(9 tarefas) | 0ms | 0ms | 0ms | 0ms | 21 |
| Ex2\_v(9 tarefas) | 0ms | 0ms | 0ms | 0ms | 21 |
| Ex3(8 tarefas) | 0ms | 0ms | 0ms | 0ms | 22 |
| Ex7(257 tarefas) | 20ms | 20ms | 30ms | 23ms | 173 |
| largeA\_Prolog(1432 tarefas) | 380ms | 410ms | 410ms | 400ms | 468 |
| largeB\_Prolog(1432 tarefas) | 270ms | 380ms | 360ms | 336ms | 468 |
| **Java** | 1ª medição | 2ª medição | 3ª medição | Média | Resultados |
| SmallDB1(10 tarefas) | 11ms | 1ms | 0ms | 4ms | 29 |
| SmallDB2(10 tarefas) | 1ms | 0ms | 1ms | 1ms | 24 |
| SmallDB3(10 tarefas) | 0ms | 1ms | 0ms | 1ms | 29 |
| MediumDB1(75 tarefas) | 1ms | 1ms | 0ms | 1ms | 234 |
| MediumDB2(75 tarefas) | 2ms | 2ms | 1ms | 2ms | 155 |
| MediumDB3(75 tarefas) | 1ms | 4ms | 2ms | 2ms | 195 |
| LargeDB1(500 tarefas) | 3ms | 3ms | 2ms | 3ms | 1257 |
| LargeDB2(500 tarefas) | 2ms | 3ms | 4ms | 3ms | 1275 |
| LargeDB3(500 tarefas) | 2ms | 3ms | 3ms | 3ms | 1421 |
| Ex1(13 tarefas) | 1ms | 0ms | 0ms | 1ms | 21 |
| Ex2\_1(9 tarefas) | 1ms | 1ms | 0ms | 1ms | 21 |
| Ex2\_v(9 tarefas) | 0ms | 0ms | 0ms | 0ms | 21 |
| Ex3(8 tarefas) | 0ms | 0ms | 1ms | 1ms | 22 |
| Ex7(257 tarefas) | 1ms | 1ms | 2ms | 1ms | 173 |
| largeA\_Prolog(1432 tarefas) | 4ms | 4ms | 5ms | 4ms | 468 |
| largeB\_Prolog(1432 tarefas) | 5ms | 4ms | 5ms | 5ms | 468 |

## Pergunta 2

Quantos trabalhadores são necessários se todas as tarefas tiverem início na data de início mais próxima?

Para esta pergunta não é necessário um modelo matemático, uma vez que se reduz a um cálculo simples, mas ao contrário de na primeira pergunta, obviamente, se vai utilizar o número de trabalhadores de cada tarefa.

### Resolução do Problema

**Em Prolog:**

Usando os resultados da primeira pergunta (datas de início e conclusão), utilizasse o predicado “minimize” nas datas de início para obter as datas de início mais próximas.

Uma vez que o número de trabalhadores apenas muda quando uma tarefa acaba ou termina procede-se á analise desses momentos. Para cada tarefa obtém-se o seu tempo de início, e para cada outra tarefa se verifica se está a decorrer (começa antes ou ao mesmo tempo e acaba depois) e se tal for o caso procedesse á soma de todos os trabalhadores.

Apos ter o número de trabalhadores nesses momentos críticos, o número de trabalhadores necessários é o valor máximo.

Após testes de performance foi verificado que o tempo de execução utilizando o predicado “cumulative” é bastante menor, como tal, foi alterado para usar este predicado.

**Em Java:**

Criar uma espécie de diagrama de eventos, eventos estes podem ser início ou fim de tarefa, o diagrama temporal começa no instante 0 e termina na duração mínima do projeto. Em cada instante calcular o balanço de trabalhadores necessários (se ocorrer um fim de tarefa subtrai-se ao balanço o número de trabalhadores necessários nessa tarefa, se ocorrer um início de tarefa adiciona-se ao balanço o número de trabalhadores necessários nessa tarefa). O número de trabalhadores necessários no instante atual é igual à soma do número de utilizadores necessários no instante anterior com o balanço de utilizadores do instante atual.

O número mínimo de trabalhadores é o número máximo de trabalhadores necessários nos instantes do diagrama de eventos.

### Analise Comparativa de Performance

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Prolog** | 1ª medição | 2ª medição | 3ª medição | Média | Resultados |
| SmallDB1(10 tarefas) | 0ms | 0ms | 0ms | 0ms | 10 |
| SmallDB2(10 tarefas) | 0ms | 0ms | 0ms | 0ms | 16 |
| SmallDB3(10 tarefas) | 0ms | 0ms | 0ms | 0ms | 10 |
| MediumDB1(75 tarefas) | 50ms | 50ms | 50ms | 50ms | 12 |
| MediumDB2(75 tarefas) | 50ms | 60ms | 60ms | 56ms | 25 |
| MediumDB3(75 tarefas) | 50ms | 50ms | 60ms | 53ms | 18 |
| LargeDB1(500 tarefas) | 1.63s | 1.64s | 1.69s | 1.65s | 23 |
| LargeDB2(500 tarefas) | 1.58s | 1.56s | 1.56 | 1.57s | 20 |
| LargeDB3(500 tarefas) | 1.66s | 1.67s | 1.67s | 1.67s | 20 |
| Ex1(13 tarefas) | 0ms | 0ms | 0ms | 0ms | 6 |
| Ex2\_1(9 tarefas) | 0ms | 0ms | 0ms | 0ms | 4 |
| Ex2\_v(9 tarefas) | 0ms | 0ms | 0ms | 0ms | 9 |
| Ex3(8 tarefas) | 0ms | 0ms | 0ms | 0ms | 5 |
| Ex7(257 tarefas) | 200ms | 200ms | 200ms | 200ms | 22 |
| largeA\_Prolog(1432 tarefas) | 14.02s | 13.83s | 13.91s | 13.91s | 87 |
| largeB\_Prolog(1432 tarefas) | 6.44s | 6.64s | 6.81s | 6.63s | 264 |
| **Java** | 1ª medição | 2ª medição | 3ª medição | Média | Resultados |
| SmallDB1(10 tarefas) | 12ms | 2ms | 2ms | 5ms | 10 |
| SmallDB2(10 tarefas) | 2ms | 2ms | 2ms | 2ms | 16 |
| SmallDB3(10 tarefas) | 2ms | 2ms | 2ms | 2ms | 10 |
| MediumDB1(75 tarefas) | 2ms | 4ms | 3ms | 3ms | 12 |
| MediumDB2(75 tarefas) | 4ms | 7ms | 5ms | 5ms | 25 |
| MediumDB3(75 tarefas) | 5ms | 7ms | 5ms | 5ms | 18 |
| LargeDB1(500 tarefas) | 7ms | 8ms | 5ms | 7ms | 23 |
| LargeDB2(500 tarefas) | 6ms | 7ms | 7ms | 7ms | 20 |
| LargeDB3(500 tarefas) | 7ms | 5ms | 6ms | 6ms | 20 |
| Ex1(13 tarefas) | 2ms | 2ms | 3ms | 2ms | 6 |
| Ex2\_1(9 tarefas) | 2ms | 2ms | 2ms | 2ms | 4 |
| Ex2\_v(9 tarefas) | 3ms | 3ms | 3ms | 3ms | 9 |
| Ex3(8 tarefas) | 2ms | 2ms | 2ms | 2ms | 5 |
| Ex7(257 tarefas) | 3ms | 4ms | 4ms | 4ms | 22 |
| largeA\_Prolog(1432 tarefas) | 6ms | 5ms | 7ms | 6ms | 87 |
| largeB\_Prolog(1432 tarefas) | 7ms | 6ms | 6ms | 6ms | 264 |

## Pergunta 3

Quantos trabalhadores são necessários no mínimo para realizar as tarefas críticas?

Para se responder a esta pergunta será necessário obter as tarefas críticas e depois proceder de forma semelhante á pergunta 2 para se obter o número de trabalhadores, tal como na pergunta 2 não há um modelo matemático.

### Resolução do Problema

**Em Prolog:**

Inicialmente procedeu-se á obtenção de uma lista com tarefas críticas.

Para se obter tal lista é necessário decidir para cada tarefa se ela é critica ou não.

Uma tarefa critica é tal que o seu tempo de início mais próximo (ES) é igual ao seu tempo de início mais afastado (LS).

Apos se ter a lista com as tarefas críticas utilizou-se o predicado da pergunta 2(com a nova lista de tarefas) para se obter o número de trabalhadores necessários.

Também foi testado usando o predicado “cumulative”, mas ao contrário de na pergunta 2, uma vez que o número de datas a analisar é muito inferior ao total, o *hoverhead*, de obter as listas de recursos não compensa.

**Em Java:**

Como as tarefas críticas são executadas umas a seguir às outras de forma linear, para calcular o número mínimo de trabalhadores necessários é equivalente a calcular máximo dos números de trabalhadores necessários para as tarefas críticas.

### Analise Comparativa de Performance

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Prolog** | 1ª medição | 2ª medição | 3ª medição | Média | Resultados |
| SmallDB1(10 tarefas) | 0ms | 0ms | 0ms | 0ms | 4 |
| SmallDB2(10 tarefas) | 0ms | 0ms | 0ms | 0ms | 5 |
| SmallDB3(10 tarefas) | 0ms | 0ms | 0ms | 0ms | 5 |
| MediumDB1(75 tarefas) | 20ms | 0ms | 20ms | 6ms | 5 |
| MediumDB2(75 tarefas) | 20ms | 20ms | 20ms | 20ms | 6 |
| MediumDB3(75 tarefas) | 20ms | 20ms | 0ms | 6ms | 5 |
| LargeDB1(500 tarefas) | 80ms | 60ms | 90ms | 83ms | 8 |
| LargeDB2(500 tarefas) | 910ms | 920ms | 910ms | 913ms | 10 |
| LargeDB3(500 tarefas) | 0ms | 20ms | 20ms | 13ms | 7 |
| Ex1(13 tarefas) | 0ms | 0ms | 0ms | 0ms | 2 |
| Ex2\_1(9 tarefas) | 0ms | 0ms | 0ms | 0ms | 2 |
| Ex2\_v(9 tarefas) | 0ms | 0ms | 0ms | 0ms | 5 |
| Ex3(8 tarefas) | 0ms | 0ms | 0ms | 0ms | 2 |
| Ex7(257 tarefas) | 60ms | 60ms | 60ms | 60ms | 13 |
| largeA\_Prolog(1432 tarefas) | 12.20s | 12.28s | 12.39s | 12.29s | 77 |
| largeB\_Prolog(1432 tarefas) | 12.89s | 13.16s | 12.95s | 13.00s | 228 |
| **Java** | 1ª medição | 2ª medição | 3ª medição | Média | Resultados |
| SmallDB1(10 tarefas) | 0ms | 0ms | 1ms | 1ms | 4 |
| SmallDB2(10 tarefas) | 0ms | 1ms | 0ms | 1ms | 5 |
| SmallDB3(10 tarefas) | 1ms | 0ms | 1ms | 1ms | 5 |
| MediumDB1(75 tarefas) | 1ms | 0ms | 0ms | 1ms | 5 |
| MediumDB2(75 tarefas) | 1ms | 3ms | 1ms | 2ms | 6 |
| MediumDB3(75 tarefas) | 1ms | 1ms | 1ms | 1ms | 5 |
| LargeDB1(500 tarefas) | 3ms | 4ms | 2ms | 3ms | 8 |
| LargeDB2(500 tarefas) | 3ms | 4ms | 3ms | 3ms | 10 |
| LargeDB3(500 tarefas) | 3ms | 5ms | 5ms | 4ms | 7 |
| Ex1(13 tarefas) | 0ms | 1ms | 0ms | 1ms | 2 |
| Ex2\_1(9 tarefas) | 1ms | 0ms | 1ms | 1ms | 2 |
| Ex2\_v(9 tarefas) | 0ms | 1ms | 0ms | 1ms | 5 |
| Ex3(8 tarefas) | 0ms | 0ms | 1ms | 1ms | 2 |
| Ex7(257 tarefas) | 1ms | 2ms | 2ms | 2ms | 13 |
| largeA\_Prolog(1432 tarefas) | 5ms | 7ms | 7ms | 6ms | 77 |
| largeB\_Prolog(1432 tarefas) | 6ms | 5ms | 5ms | 5ms | 228 |

## Pergunta 4

Qual o número mínimo de trabalhadores a contratar?

Quando é que cada tarefa tem início?

Existem soluções ótimas alternativas?

Apesar de esta pergunta se dividir em 3 questões devido á sua natureza as se responder á primeira obtém-se os dados para a resposta das outras duas.

Na primeira questão, ao contrário de na pergunta 2 as datas de início não estão fixas no tempo de início mais próximo, no entanto, o projeto tem de ser na mesma acabado no tempo mínimo. Como tal as datas de início das tarefas estarão algures entre a sua data de início mais próxima e a data de início mais afastada.

### Modelo

**Objetivo:**

Minimizar Trabs (Número de trabalhadores necessário)

**Dados:**

Tarefas (índices das tarefas)

n (número de tarefas)

Trab(i), com iϵ Tarefas (número de trabalhadores necessários para a tarefa i)

Dres(i), com iϵ Tarefas (Data de início da tarefa i, com as restrições de precedência)

**Variáveis de decisão:**

Trabs (Número de trabalhadores necessário)

Df(i), com iϵ Tarefas (Data início da tarefa i)

**Sujeito a:**

Sendo S={(i,j)|Df(i)≤Df(j)<Df(i)+di}

i ϵ Tarefas

### Resolução do Problema

**Em Prolog:**

Usar o predicado da pergunta 1 para obter as datas com as restrições.

Obter o número máximo de trabalhadores (a soma dos trabalhadores de todas as tarefas).

Utilizar o predicado da pergunta 2 para obter o número mínimo de trabalhadores (número de trabalhadores para as tarefas críticas).

Utilizar o predicado da pergunta 3 para obter o número máximo de trabalhadores (número de trabalhadores se as tarefas começarem no ES).

Para obter melhor eficiência foi implementado um algoritmo de pesquisa binaria, que utiliza os valores mínimos e máximos obtidos anteriormente para começar a pesquisa.

Durante a pesquisa binaria é utilizado o predicado “cumulative” para encontrar as soluções.

Apos a pesquisa terminar o predicado “labeling” é usado para obter as datas de início e outras soluções alternativas.

**Em Java:**

Utilizar a mesma noção de diagrama de eventos como no problema 2, só que neste momento é necessário ter em atenção que existem tarefas que se podem deslocar no tempo e não prejudicam a solução ótima. Na implementação foi usada pesquisa em profundidade recursiva(foi usada este tipo de pesquisa porque através desta pesquisa temos acesso a backtracking), em que em cada instancia da recursividade é selecionada uma tarefa (que ainda não tenha sido selecionada anteriormente na pesquisa e se esta tarefa tiver ES != LS significa que se pode deslocar no máximo LS-ES posições no diagrama temporal), cria-se um novo diagrama temporal que suporte a alteração na posição da tarefa selecionada, invoca-se a recursividade sobre o novo diagrama temporal. Caso não exista nenhuma tarefa com ES! =LS calculamos o número mínimo de trabalhadores necessários se as tarefas começarem nos inícios determinados pelo diagrama temporal nessa instância da pesquisa, retornando o número mínimo de trabalhadores e tempos de início de tarefas.

### Analise Comparativa de Performance

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Prolog** | 1ª medição | 2ª medição | 3ª medição | Média | Resultados |
| SmallDB1(10 tarefas) | 0ms | 0ms | 20ms | 6ms | 9 |
| SmallDB2(10 tarefas) | 0ms | 20ms | 0ms | 6ms | 11 |
| SmallDB3(10 tarefas) | 0ms | 0ms | 0ms | 0ms | 8 |
| MediumDB1(75 tarefas) | 130ms | 130ms | 110ms | 123ms | 11 |
| MediumDB2(75 tarefas) | 170ms | 170ms | 170ms | 170ms | 11 |
| MediumDB3(75 tarefas) | 130ms | 130ms | 110ms | 123ms | 11 |
| LargeDB1(500 tarefas) | 7.69s | 7.86s | 7.78s | 7.78s | 13 |
| LargeDB2(500 tarefas) | 7.48s | 7.63s | 7.38s | 7.50s | 16 |
| LargeDB3(500 tarefas) | 8.02s | 7.95s | 7.89s | 7.95s | 12 |
| Ex1(13 tarefas) | 20ms | 0ms | 0ms | 6ms | 4 |
| Ex2\_1(9 tarefas) | 0ms | 0ms | 0ms | 0ms | 3 |
| Ex2\_v(9 tarefas) | 0ms | 0ms | 0ms | 0ms | 6 |
| Ex3(8 tarefas) | 0ms | 0ms | 0ms | 0ms | 4 |
| Ex7(257 tarefas) | 1.61s | 1.61s | 1.57s | 1.60s | 15 |
| largeA\_Prolog (1432 tarefas) | 62.52s | 59.20s | 56.67s | 63.46s | 78 |
| largeB\_Prolog (1432 tarefas) | 74.38s | 77.86s | 76.08s | 76.11s | 228 |
| **Java** | 1ª medição | 2ª medição | 3ª medição | Média | Resultados |
| SmallDB1(10 tarefas) | 274ms | 187ms | 151ms | 204ms | 9 |
| SmallDB2(10 tarefas) | 315.64s | 383.95s | 340.14s | 346.58s | 11 |
| SmallDB3(10 tarefas) | 474ms | 465ms | 512ms | 484ms | 8 |
| MediumDB1(75 tarefas) | N/A | N/A | N/A | N/A | N/A |
| MediumDB2(75 tarefas) | N/A | N/A | N/A | N/A | N/A |
| MediumDB3(75 tarefas) | N/A | N/A | N/A | N/A | N/A |
| LargeDB1(500 tarefas) | N/A | N/A | N/A | N/A | N/A |
| LargeDB2(500 tarefas) | N/A | N/A | N/A | N/A | N/A |
| LargeDB3(500 tarefas) | N/A | N/A | N/A | N/A | N/A |
| Ex1(13 tarefas) | 1180.54s | 1118.13s | 1068.38s | 1122.35 | 4 |
| Ex2\_1(9 tarefas) | 1.41s | 1.51s | 1.44s | 1.45s | 3 |
| Ex2\_v(9 tarefas) | 1.59s | 1.42s | 1.68s | 1.56s | 6 |
| Ex3(8 tarefas) | 811ms | 706ms | 813ms | 777ms | 3 |
| Ex7(257 tarefas) | N/A | N/A | N/A | N/A | N/A |
| largeA\_Prolog(1432 tarefas) | N/A | N/A | N/A | N/A | N/A |
| largeB\_Prolog(1432 tarefas) | N/A | N/A | N/A | N/A | N/A |

## Conclusões

Todos os testes foram realizados num único computador com o CPU Intel i7-4710MQ.

O java utiliza os dados no mesmo formato que o prolog.

Para criar mais facilmente problemas de maiores dimensões desenvolvemos um simples gerador de projetos aleatórios em python.

Todos os projetos começados por “Ex” ou acabados por ”prolog” são os exemplos de teste fornecidos pela professora, enquanto que todos os outros apresentados foram criados utilizando o gerador.

O gerador e todos os ficheiros de teste encontram-se nos ficheiros enviados.

Na quarta pergunta é possível observar muitos valores a N/A, tal não acontece devido a um erro, mas sim devido ao tempo excessivo que demora a calcular a resposta (para todos >3horas), o que impossibilita a obtenção de resultados em tempo útil. Mesmo em problemas de menores dimensões é possível observar que a performance é bastante baixa, como no SmallDB2, que demora 6 minutos. Foi possível perceber que o tempo de execução é exponencial á folga dos tempos de início.

Nas primeiras três perguntas o tempo de execução do java é bastante inferior ao do prolog, tal é provavelmente causado pelo facto de algoritmos eficientes e específicos terem sido implementados para estes problemas. Guardar os valores de cálculos intermédios poderá aumentar bastante a performance do prolog.

Para as segunda e terceira perguntas foram tentadas duas alternativas para o cálculo do uso de trabalhadores utilizados, o uso do predicado “cumulative” e a criação de um predicado específico para o problema. Na segunda pergunta “cumulative” foi mais eficiente, mas na quarta pergunta o predicado especifico foi mais eficiente, provavelmente devido ao facto de muito menos datas serem utilizadas, e a restrição das listas a apenas estas datas ter um custo relativamente elevado(como já mencionado na descrição de implementação).

Para a quarta pergunta em prolog foi feita uma análise entre duas opções, uma em que se restringiu mais o domínio das variáveis, utilizando o número de trabalhadores necessários para o ES, e outra em que o valor é a soma do que cada tarefa usa. Foi possível constatar que o tempo para calcular os trabalhadores das tarefas para ES é inferior ao que se ganha na procura. Como tal foi possível concluir que restringir o domínio muitas a melhor opção, mesmo no caso em que esta restrição usa recursos computacionais.

Em geral devido á existência de mais desenvolvimento em linguagens imperativas e á consequente existência de mais e mais detalhados algoritmos é possível obter uma maior performance nessas linguagens. Apesar de ser possível implementar os mesmos algoritmos em linguagens logicas, o tempo de desenvolvimento será superior e a performance inferior. No caso em não existam algoritmos bem documentados para o problema a ser tratado (ou que se desconheça a sua existência), prolog pode ser usado para rapidamente se obter uma solução eficiente.

# Problema 2

Tal como no problema 1 tem de se tratar um problema de escalonamento de tarefas, mas neste caso existem mais dados e restrições.

Existe um conjunto de tarefas, com as precedências conhecidas, algumas das quais tem de ser acabadas no dia em que começam e outras podem ser feitas em vários dias.

Algumas das tarefas também tem de começar um intervalo de tempo depois de outras tarefas acabarem.

As tarefas para serem realizadas necessitam de trabalhadores de certas especialidades, como tal também existe uma lista de trabalhadores e as respetivas especialidades.

O projeto tem de ser concluído até uma certa data, e se com os trabalhadores iniciais não permitirem a conclusão mais trabalhadores, apenas com uma especialidade devem ser contratados.

O objetivo é contratar o mínimo de trabalhadores extra possível, concluindo o projeto antes do tempo máximo de conclusão, e se possível usando esse mesmo número de trabalhadores concluir o projeto ainda mais cedo.

## Modelo

**Objetivo:**

Minimizar NTrabs,

Minimizar Concl mantendo o mínimo de NTrabs

**Dados:**

n (número de tarefas)

m (número de trabalhadores)

Tarefas (Ids das tarefas)

Trabs (Ids dos trabalhadores)

Specs (Lista de especialidades)

di (duração da tarefa i)

(i,i’) Precedência (a tarefa i precede a tarefa i’, i.e a tarefa i’ apenas pode começar quando a tarefa i acabar)

eij (o número de trabalhadores de especialidade j que a tarefa i precisa)

hk (lista de especialidades que o trabalhador k tem)

Ii (se a tarefa i pode ser interrompida)

IMINii’ (valor mínimo apos a tarefa i’ terminar que a tarefa i pode começar)

IMAXii’ (valor máximo apos a tarefa i’ terminar que a tarefa i tem de começar)

Duteis (a lista com os dias uteis)

MaxConcl (data de conclusão máxima do projeto)

Huteis (lista com as horas de trabalho)

Nhoras = 8 (horas de trabalho por dia)

**Variáveis de decisão:**

Di (índice da data de início da tarefa i na lista Duteis)

Hi (índice da hora de início da tarefa i na lista Huteis)

Tarik (qual a especialização que o trabalhador k faz na tarefa i, 0 significa que não faz a tarefa)

Tarei (se o trabalhador extra e faz a tarefa i)

Concl (índice do dia em que o projeto é concluído)

Extrae (a especialidade do trabalhador extra e)

NTrabs (número extra de trabalhadores a contratar)

Sii’ (se as tarefas i e i’ ocorrem em simultâneo)

**Sujeito a:**

Se nada for dito em contrário os índices correspondem ao seguinte.



Se a tarefa i precede a tarefa i’ então a data/hora de início de i’ é superior á data/hora de finalização de i

Di\*Nhoras+Hi+di ≤ Di'\*Nhoras + Hi' ∀(i,i') ∈Precedencia

A data de conclusão do projeto é maior ou igual ao dia em que qualquer tarefa termina.

Duteis(Di+Mod(di,Nhoras)) ≤Concl

Se a tarefa não pode ser interrompida tem de ser concluída até ao fim do dia em que começa.

∀i Ii=0 ⇒Hi+di≤Nhoras

Os trabalhadores apenas realizam trefas para as quais tem especialidade.

∀i,k Tarik∈hk

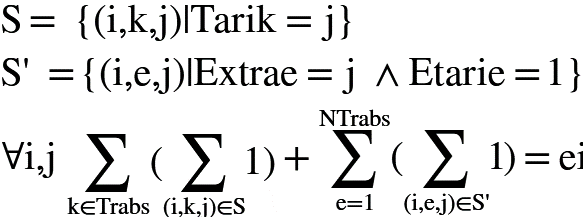
Duas tarefas ocorrem em simultâneo se uma começa antes ou quando a outra começa e acaba depois da outra começar.

∀i,i' Sii'=1⇔Si'i⇔Di\*Nhoras+Hi≤Di'\*Nhoras+Hi'<Di\*Nhoras+Hi+di

Se duas tarefas ocorrem em simultâneo os trabalhadores apenas trabalham numa delas.

∀i,i',k Tarik≠0∧ i≠i' ∧ Sii'=1⇒Tari'k=0

Todas as tarefas têm o número necessário de trabalhadores para cada especialidade.



O projeto tem de ser terminado até uma data limite.

Concl ≤ MaxConcl

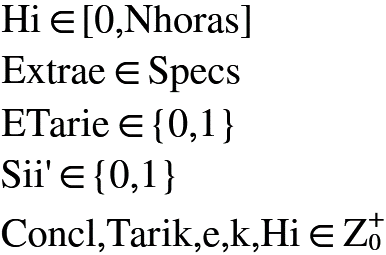
A tarefa i’ tem de começar pelo menos um número de horas depois da tarefa i acabar.

∀i,i' Di\*Nhoras+Hi+di+IMINii' ≤ Di'\*Nhoras+Hi'

A tarefa i’ tem de começar até um número máximo de horas depois da tarefa i acabar.

∀i,i' Di\*Nhoras+Hi+di+IMAXii' ≥ Di'\*Nhoras+Hi'

Domínio das variáveis.



## Resolução do Problema

Todas as variáveis de decisão foram criadas e os seus domínios restringidos.

Para cada uma das restrições foi criado um predicado que impõe as restrições nas variáveis. Cada “Para todo” implica percorrer uma lista.

Para facilitar a implementação para as datas e horas foram usados os índices das listas dos dias uteis e horas uteis, invés dos seus valores, e apenas passados para os valores quando comparando com o tempo máximo de conclusão.

Apos as restrições estarem feitas foi utilizado o predicado “minimize” para obter o valor mínimo de trabalhadores extra necessários, e fixando esse número foi usado um segundo “minimize” para obter o mínimo do tempo de conclusão do projeto.

Devido ao grande número de variáveis de decisão, quando comparado como o primeiro problema, invés de serem apresentados como argumentos os valores são escritos usando predicados de escrita.