Programação em Lógica: Escalonamento de Dispositivos Elétricos Optimização dos Custos e Recursos Energéticos

Grupo 5:
João Bordalo & João Soares

FEUP-PLOG,
Turma: 3MIEIC6, **Grupo:** Esc_Disp_Eletr_5 ,
Faculdade de Engenharia da Universidade do Porto,
Rua Roberto Frias, sn, 4200-465,Paranhos,Porto Portugal
{ei12040,ei12093}@fe.up.pt
http://www.fe.up.pt

Resumo Este trabalho pretende obter o escalonamento óptimo de dispositivos elétricos, ou seja, calendarizar o funcionamento de vários dispositivos, de acordo com o preço variante da energia ao longo do tempo, de modo a obter a solução com menor custo monetário ou com a menor potência energética contratada. Para isto dividimos as várias actividades de cada um dos dispositivos em diferentes tarefas com um consumo e duração associados e utilizamos a linguagem de programação em lógica Prolog, recorrendo também ao módulo de programação com restrições do SICStus Prolog, para nos auxiliar na tarefa do escalonamento sendo que, no final, recebemos a solução óptima do calendário de tarefas para as minimizações ambicionadas anteriormente. Sucintamente, a programação com restrições baseia-se principalmente em procurar um estado no qual uma grande quantidade de restrições sejam satisfeitas simultaneamente. Um problema define-se tipicamente como um estado da realidade no qual existe um número de variáveis com valor desconhecido. Um programa baseado em restrições procura valores possiveis para todas as variáveis, satisfazendo todas as restrições em simultâneo .Aplicando este tipo de programação foi possível obter combinações válidas, minimizando algumas variaveis pretendidas.

1 Introdução

Uma Smart grid (Rede Elétrica Inteligente), em termos gerais, é a aplicação de tecnologia da informação para um sistema elétrico de potência, integrada aos sistemas de comunicação e infra estrutura de rede automatizada. Estes sistemas analisam o funcionamento da rede e determinam o que deve ser feito para melhorar o desempenho desta.

O nosso projeto baseia-se neste conceito: melhorar o desempenho energético e poder ter várias aplicações no ramo da domótica, robótica e indústria em geral, sendo que poderia ser integrado numa smart grid para gerir e optimizar os gastos energéticos de um sistema.

Para que nos fosse possível resolver este problema, assumimos que a potência contratada pelo sistema é constante a qualquer hora do dia e esta nunca pode ser ultrapassada pelo consumo das diferentes tarefas, e ainda que o preço variante da energia ao longo do dia é conhecido previamente à execução.

Este trabalho apresenta a resposta desenvolvida em PLR ao problema de minimização do custo a pagar pela energia ao longo do dia e ao problema de minimizar a potência contratada, fazendo o escalonamento das tarefas a realizar numa janela temporal de 24h. Estas tarefas são classificadas em escalonáveis ou não-escalonáveis conforme a natureza da sua utilização e esta distinção tem de ser contemplada na resolução.

A nossa motivação para este projeto deriva do ganho que este tipo de redes possibilitam, e a sua utilidade efetiva no dia a dia. Neste artigo começamos por uma descrição do problema, sendo que os tópicos seguintes contemplam a sua resolução e abordagem em PROLOG. Finalmente apresentamos os resulatados e conclusões que retiramos deste trabalho, terminando com o código-fonte em anexo.

2 Descrição do Problema

O problema em questão consiste em minimizar o custo das tarefas de um sistema com dispositivos elétricos, sendo que esta otimização é feita de duas formas: minimizar o custo a pagar pela energia ao longo do dia e minimizar a potência contratada, obtendo um consumo energético mais uniforme.

Para ambos os casos as tarefas não-escalonáveis, devido à sua natureza, são alocadas primeiro. Após a alocação, os valores da potência para as outras tarefas são atualizados pois a potência deve ser partilhada naquela janela-temporal.

Para o primeiro caso, as tarefas são alocadas consoante a potência restante em cada instante e o seu custo, enquanto que no segundo caso, a alocação é feita com base a potência utilizada nos outros instantes de modo a tentar uniformizar o consumo em todas as horas do dia.

3 Abordagem

A abstração feita é que cada sistema de dispositivos é uma grande ≪máquina≫ com uma potência máxima permitida de consumir por todos os dispositivos inseridos nesse sistema por hora ao longo de um dia.

3.1 Variáveis de Decisão

As Variáveis de decisão são as variáveis sobre as quais fazemos Labeling e minimize:

- LimiteEnergetico: varia entre 0 (quando não há necessidade de Contratar Energia) e a Potência máxima contratada ou restante, no caso de se ter préviamente calculado o consumo dos equipamentos não escalonáveis. Este valor representa a potência disponível em cada janela-temporal e verifica se há potência disponível para alocar uma dada tarefa.
- CustoLimite: varia entre o Custo minimo, menor valor possivel tendo em conta a tarifa empregue, e o Custo Maximo, que tem em conta o consumo máximo necessário tendo em conta a tarifa.

3.2 Restrições

- domain - impõe restriçoes sobre o dominio inicial das variáveis.
- minimize - faz optimizaçao da variavel.
- restricaoTempo - impõe restrições sobre variaveis que representam o tempo de inicio de uma tarefa e a sua baseline
- calculaCustoTotal - gera restrições referentes a custos.

3.3 Função de Avaliação.

A nossa função de avaliação, (ou função heuristica), consiste na função calculaCustoTotal que além de acumular restrições, também devolve o custo referente a um certo planeamento, podendo assim avaliar a "qualidade"da solução.

3.4 Estratégia de Pesquisa

A pesquisa pela solução ótima do problema é feita tendo em conta a Baseline e a Deadline de cada tarefa a escalonar individualmente, procurando uma solução que privilegia a minimização do valor da soma das potências a cada instante , ou uma solução que privilegia a minimização do valor do custo de todas as tarefas escalonadas. Estas duas pesquisas recorrem a estratégias semelhantes (ambas usam as funcionalidades do Prolog para obter os valores mínimos) sendo que diferem em objetivos. A primeira visa minimizar a potência contratada e a segunda o custo total das operações.

4 Visualização da Solução

Após o escalonamento a solução é mostrada em modo de texto na forma de uma lista com as tarefas a usar no escalonamento, uma lista com a hora de início e uma lista com a hora de término de cada uma e o resultado do cálculo do custo total ou da potência a contratar.

5 Resultados

Como era de esperar, o tempo de execução do programa é menor aquando a utilização de volumes de dados menores, aumentando conforme os dados aumentam, mas produzindo escalonamentos igualmente eficientes e respostas em tempo útil.

Os resultados são muito dependentes dos dados fornecidos, por exemplo, a potência ótima obtida numa situação em que o consumo de uma das tarefas seja vastamente maior que o consumo das outras muito provavelmente será igual ao consumo desta mesma tarefa, fazendo com que seja impossível contratar uma potência abaixo desse valor (se uma dada tarefa tiver um consumo 50 vezes maior que todas as outras, a potência a contratar terá de ser superior ao consumo desta, produzindo resultados possivelmente não ótimos para todas as outras).

6 Conclusões

Ao longo do desenvolvimento deste projeto, o nosso maior obstáculo foi, ao iniciar o trabalho, como iriamos traduzir os dados (tarefas e dispositivos) para Prolog. Após muita discussão e estudo da situação, encontramos uma solução adequada ao contexto do problema e iniciamos a lógica, tendo terminado com um programa que funciona plenamente nos casos contemplados no enunciado. Reconhecemos a dificuldade associada a estes problemas de otimização mas em suma achamos que conseguimos produzir resultados muito positivos para um problema de alta demanda atualmente e cujas únicas limitações são as tarefas a escalonar terem durações inteiras em horas e o seu consumo ser constante ao longo da execução.

Bibliografia

A Anexo A

Ficheiro interface.pl Neste ficheiro é sugerida uma interface com o utilizador proporcionando ao mesmo contacto com a aplicação de modo a obter os resultados procurados.

```
:-include('listagens.pl').
:-include('escalonamentos.pl').
:-use_module(library(clpfd)).
:-use_module(library(lists)).

:- dynamic escalonavel/3.
:- dynamic nescalonavel/2.
:- dynamic tarefa/6.
:- dynamic potenciacontratada/1.
:- dynamic subtask/5.
:- dynamic custo/1.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                          %%%
%%%        INTERFACE         %%%
%%%                          %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

intro:-

write('Escalonamento de Dispositivos Consumidores de Energia Elétrica').

start :-
write('CONFIGS INICIAIS'),
nl,
write('Introduza a potencia Contratada em W'),
nl,
read(Potenciacontratada),
assert(potenciaContratada(Potenciacontratada)),
startmenu.



startmenu :-
set_prolog_flag(fileerrors,off),
nl, intro, nl, nl,
write('1 - Mostrar todos os Dispositivos'), nl,
write('2 - Mostrar todas as Tarefas'), nl,
```

```
write('3 - Criar Dispositivo'), nl,
write('4 - Criar Tarefa'), nl,
write('5 - Importar Dispositivos do ficheiro'), nl,
write('6 - Importar Tarefas do ficheiro'), nl,
write('7 - Gerar Escalonamento'), nl,
write('8 - Limpar Base de Conhecimento'), nl,
write('9 - Configurar Preco de Energia'),nl,
write('10 - Importar config de preco de energia do ficheiro'),nl,
write('11 - Calcular Otimizacao do Custo'),nl,
write('0 - Sair'),nl,
repeat, read(Op), Op >= 0, Op =< 11,!,
menu(Op), repeat, skip_line, get_code(_), startmenu.



menu(0):-
abort.

menu(1):-
mostra_todos_dispositivos,
nl,
mostra_todas_tarefas.

menu(1):-
mostra_todos_dispositivos,
startmenu.

menu(2):-
mostra_todas_tarefas,
startmenu.


menu(3):-
write('1 - Dispositivo Escalonável'), nl,
write('2 - Dispositivo não escalonável'), nl,
write('0 - Voltar ao menu anterior'),nl,
repeat, read(Op), Op >= 0, Op =< 2,!,
menu_disp(Op), repeat, skip_line, get_code(_), menu(4).


menu(4):-
write('Adicionar Tarefa(minusculas e terminado com  \'.\' ):'),nl,
write('Nome da Tarefa:'),read(Nome),
get_todos_escalonaveis(Lista),
mostra_dispositivos(Lista),
```

```
write('id (numero inteiro) da Maquina a associar :'), read(Idmaq),
write('Baseline:'),read(Baseline),
write('Deadline:'),read(Deadline),
write('Duraçao:'), read(Duracao),
divide_tarefas(Duracao, _, Idmaq, _,SubtarefasFinal),
write(SubtarefasFinal),

assert(tarefa(Nome, Idmaq ,Baseline,Deadline,Duracao, SubtarefasFinal)).
%write('Tarefa '),write(Nome), write(' de Baseline '), write(Baseline),write(', com Dead
%write(Deadline), write(' que dura '), write(Duracao),write(' adicionado com sucesso à b

divide_tarefas(0, SubtarefasFinal, _, _,SubtarefasFinal).
divide_tarefas(Duracao, Subtarefas, Idmaq, S1,SubtarefasFinal):-
write('consumo na '), write(Duracao), write('º hora:'), read(Consumo), nl,
assert(subtask(S1, 1, E1, Consumo, Idmaq)),
append(Subtarefas,[Consumo],Subtarefas1),

Duracao1 is Duracao - 1,
divide_tarefas(Duracao1, Subtarefas1, Idmaq, E1,SubtarefasFinal).


menu(5):-
write('Adicionar dispositivos do Ficheiro (\'root\\\\path\\\\dispositivos.txt\'):'),nl,
repeat, read(Ficheiro),
adiciona_dispositivos_do_ficheiro(Ficheiro),
!.

menu(6):-
write('Adicionar tarefas do ficheiro (\'root\\\\path\\\\tarefa.txt\'):'),nl,
repeat,read(Ficheiro),
adiciona_tarefas_do_ficheiro(Ficheiro),
!.

menu(7):-
write('1 - Escalonamento Por Consumo Energetico'), nl,
write('2 - Escalonamento Pelo Custo Minimo'), nl,
write('3 - Escalonamento Por Consumo Energetico Com Potencia Contratada Restante Variave
write('4 - Escalonamento Pelo Custo Minimo Com Potencia Contratada Restante Variavel'),n
write('0 - Voltar ao menu anterior'),nl,
repeat, read(Op), Op >= 0, Op =< 4,!,
menu_escal(Op), repeat, skip_line, get_code(_), menu(7).

menu(8):-
retractall(nescalonavel(_,_)),
retractall(escalonavel(_,_,_)),
```

```
write('Todos os dispositivos instanciados foram eliminados com sucesso!'), nl,
retractall(tarefa(_,_,_,_,_,_)),
write('Todas as tarefas instanciadas foram eliminadas com sucesso!'), nl.


menu(9):-
write('Inserir preço hora a hora (inteiro terminado por ponto) 24 vezes'),nl,
cria_lista_custo(24, Lista),
write(Lista),nl,
assert(custo(Lista)),
write('Custo por hora confifurado com sucesso!!'),nl.




cria_lista_custo(0,Lista).
cria_lista_custo(Dias, Lista):-
write('preço:'),
read(X),nl,
append(Lista, [X], Lista1),
Dias1 is Dias - 1,
cria_lista_custo(Dias1, Lista1).


menu(10):-
write('Adicionar custo de energia do ficheiro (\'root\\\\path\\\\energia.txt\'):'),nl,
write('ex.: \'\\\\\\\\\samba.fe.up.pt\\\\ei07122\\\\plog\\\\energia.txt\''),nl,
repeat,read(Ficheiro),
adiciona_energia_do_ficheiro(Ficheiro),
!.


menu(11):-
gera_escalonamento_custo.

adiciona_energia_do_ficheiro(Ficheiro):-
file_to_list(Ficheiro, Custos),
write(Custos),nl,
assert(custo(Custos)),
write('custos adicionados com sucesso').
```

```
%%-------------------------------------------------------------------------------------------
%%Menu dos Dispositivos

menu_disp(0):-startmenu.

menu_disp(1):-
write('Adicionar Dispositivo Escalonável(minusculas e terminado com  \'.\' ):'),nl,
write('Nome do Dispositivo:'),read(Nome),
write('Id:'),read(Id),
write('Recursos Alocados:'), read(X),
assert(escalonavel(Nome,Id, X)).
%write('Dispositivo Escalonável '),write(Nome), write(' com id '), write(Id),write(' adi


menu_disp(2):-
write('Adicionar Dispositivo não Escalonável(minusculas e terminado com  \'.\' ):'),nl,
write('Nome do Dispositivo:'),read(Nome),
write('Consumo:'),read(Consumo),
assert(nescalonavel(Nome,Consumo)).
%write('Dispositivo não Escalonável '),write(Nome), write(' com consumo de '), write(Con

%%-------------------------------------------------------------------------------------------

%%Menu dos escalonamentos

menu_escal(0):- startmenu.


menu_escal(1):-

findall(Subtarefas,tarefa(_,_,_,_,_, Subtarefas),ListaConsumos),

( foreach(T,ListaConsumos),
foreach(A,AmountSubtasks),
count(I,1,N)
do
getLength(T,A),
true
),

findall(Baseline,tarefa(_,_,Baseline,_,_,_),ListaBaselines),
( foreach(T,ListaBaselines),
count(I,1,N)
do
true
```

```
),

findall(Deadline,tarefa(_,_,_,Deadline,_,_),ListaDeadlines),
( foreach(T,ListaDeadlines),
count(I,1,N)
do
true
),
geraLines(ListaBaselines,_,Baselines,ListaDeadlines,_,Deadlines,AmountSubtasks),
write(Baselines),write('\n'),
write(Deadlines),write('\n'),

findall(subtask(_,_,_,_,_),subtask(_,_,_,_,_),ListaSubT),%% numero de subtarefas existen
( foreach(_,ListaSubT),
count(I,1,NS)
do
true
),

length(ListaInicioTarefas,NS),
length(ListaFimTarefas,NS),

separaTasks(ListaConsumos,_,Tasks,_,ListaInicioTarefas,_,ListaFimTarefas),


cria_lista_n_escalonavel(LNE),

get_pot_max(PotMax),
calculaEmax(PotMax,PotenciaContratadaRestante,LNE),


escalonamentoPorConsumoEnergetico(ListaInicioTarefas,ListaFimTarefas,Tarefas,PotenciaCon

menu_escal(2):-

get_custos([Custos|_]),

findall(Subtarefas,tarefa(_,_,_,_,_, Subtarefas),ListaConsumos),
( foreach(T,ListaConsumos),
foreach(A,AmountSubtasks),
count(I,1,N)
do
getLength(T,A),
true
),
```

```
findall(Baseline,tarefa(_,_,Baseline,_,_,_),ListaBaselines),
( foreach(T,ListaBaselines),
count(I,1,N)
do
true
),

findall(Deadline,tarefa(_,_,_,Deadline,_,_),ListaDeadlines),
( foreach(T,ListaDeadlines),
count(I,1,N)
do
true
),
geraLines(ListaBaselines,_,Baselines,ListaDeadlines,_,Deadlines,AmountSubtasks),
write(Baselines),write('\n'),
write(Deadlines),write('\n'),

findall(subtask(_,_,_,_,_),subtask(_,_,_,_,_),ListaSubT),%% numero de subtarefas existen
( foreach(_,ListaSubT),
count(I,1,NS)
do
true
),

length(ListaInicioTarefas,NS),
length(ListaFimTarefas,NS),

separaTasks(ListaConsumos,_,Tasks,_,ListaInicioTarefas,_,ListaFimTarefas),

cria_lista_n_escalonavel(LNE),

get_pot_max(PotMax),
calculaEmax(PotMax,PotenciaContratadaRestante,LNE),


escalonamentoPeloCustoMinimo(ListaInicioTarefas,ListaFimTarefas,Tarefas,PotenciaContrata

menu_escal(3):-

findall(Subtarefas,tarefa(_,_,_,_,_, Subtarefas),ListaConsumos),

( foreach(T,ListaConsumos),
foreach(A,AmountSubtasks),
count(I,1,N)
```

```
do
getLength(T,A),
true
),

findall(Baseline,tarefa(_,_,Baseline,_,_,_),ListaBaselines),
( foreach(T,ListaBaselines),
count(I,1,N)
do
true
),

findall(Deadline,tarefa(_,_,_,Deadline,_,_),ListaDeadlines),
( foreach(T,ListaDeadlines),
count(I,1,N)
do
true
),
geraLines(ListaBaselines,_,Baselines,ListaDeadlines,_,Deadlines,AmountSubtasks),
write(Baselines),write('\n'),
write(Deadlines),write('\n'),

findall(subtask(_,_,_,_,_),subtask(_,_,_,_,_),ListaSubT),%% numero de subtarefas existen
( foreach(_,ListaSubT),
count(I,1,NS)
do
true
),

length(ListaInicioTarefas,NS),
length(ListaFimTarefas,NS),

separaTasks(ListaConsumos,_,Tasks,_,ListaInicioTarefas,_,ListaFimTarefas),


cria_lista_n_escalonavel(LNE),

get_pot_max(PotMax),
calculaListaEmax(PotMax,PotenciaContratadaRestante,LNE),

escalonamentoPorConsumoEnergeticoComPotenciaContratadaRestanteVariavel(ListaInicioTarefa

menu_escal(4):-

get_custos([Custos|_]),
```

```
findall(Subtarefas,tarefa(_,_,_,_,_, Subtarefas),ListaConsumos),
( foreach(T,ListaConsumos),
foreach(A,AmountSubtasks),
count(I,1,N)
do
getLength(T,A),
true
),

findall(Baseline,tarefa(_,_,Baseline,_,_,_),ListaBaselines),
( foreach(T,ListaBaselines),
count(I,1,N)
do
true
),

findall(Deadline,tarefa(_,_,_,Deadline,_,_),ListaDeadlines),
( foreach(T,ListaDeadlines),
count(I,1,N)
do
true
),
geraLines(ListaBaselines,_,Baselines,ListaDeadlines,_,Deadlines,AmountSubtasks),
write(Baselines),write('\n'),
write(Deadlines),write('\n'),

findall(subtask(_,_,_,_,_),subtask(_,_,_,_,_),ListaSubT),%% numero de subtarefas existen
( foreach(_,ListaSubT),
count(I,1,NS)
do
true
),

length(ListaInicioTarefas,NS),
length(ListaFimTarefas,NS),

separaTasks(ListaConsumos,_,Tasks,_,ListaInicioTarefas,_,ListaFimTarefas),

cria_lista_n_escalonavel(LNE),

get_pot_max(PotMax),
calculaListaEmax(PotMax,PotenciaContratadaRestante,LNE),
```

```
escalonamentoPeloCustoMinimo(ListaInicioTarefas,ListaFimTarefas,Tarefas,PotenciaContrata


escalonamentoPeloCustoMinimoComPotenciaContratadaRestanteVariavel(ListaInicioTarefas,Lis




calculaListaEmax(PotMax,PotenciaContratadaRestante,[]).

calculaListaEmax(PotMax,PotenciaContratadaRestante,[H|T]):-
PotMaxNext is PotMax - H,
append(PotenciaContratadaRestante,[PotMaxNext],EMax),
calculaListaEmax(PotMax,EMax,T).



mostra_todos_dispositivos:-
get_todos_dispositivos(Dispositivos),
nl,
mostra_dispositivos(Dispositivos).

mostra_dispositivos([]):- write('Fim dos Dispositivos'), nl.
mostra_dispositivos([Este_dispositivo|Outros_dispositivos]):-
write(Este_dispositivo), nl,
mostra_dispositivos(Outros_dispositivos).


mostra_todas_tarefas:-
get_todas_tarefas(Tarefas),
nl,
mostra_tarefas(Tarefas).

mostra_tarefas([]):- write('Fim das Tarefas'), nl.
mostra_tarefas([Esta_tarefa|Outras_tarefas]):-
write(Esta_tarefa),nl,
mostra_tarefas(Outras_tarefas).

%%-------------------------------------------------------------------------------------------

%-----------------------------------------------
%% SINTAXE DO FICHEIRO DE DISPOSITIVOS
%% 1 Disp por linha
%% 'Nome1. Consumo1.'
%% 'Nome2. Consumo2.'
```

```
%%
%-----------------------------------------------

adiciona_dispositivos_do_ficheiro(Ficheiro):-
file_to_list(Ficheiro, Dispositivos),
write(Dispositivos),nl,
importar_dispositivos_para_kb(Dispositivos),
write('Dispositivos importados!!').

importar_dispositivos_para_kb([]).
importar_dispositivos_para_kb([Nome,Consumo|Proximo]):-
assert(dispositivo(Nome, Consumo)),
write('fez assert'), nl,
importar_dispositivos_para_kb(Proximo).


%-----------------------------------------------
%% SINTAXE DO FICHEIRO DE TAREFAS
%% 1 Disp por linha
%% 'Nome1. Idmaq1, Baseline1. Deadline1. Duracao1. [consumos].'
%%
%%
%-----------------------------------------------

adiciona_tarefas_do_ficheiro(Ficheiro):-
file_to_list(Ficheiro, Tarefas),
write(Tarefas),nl,
importar_tarefas_para_kb(Tarefas),
write('Tarefas importadas!!').

brinca_consumos([],_).
brinca_consumos([Consumo|Nconsumo], Idmaq):-
assert(subtask(S1, 1, E1, Consumo, Idmaq)),
brinca_consumos(Nconsumo, Idmaq).


importar_tarefas_para_kb([]).
importar_tarefas_para_kb([Nome,Idmaq, Baseline, Deadline, Duracao, Consumos|Proximo]):-
write('unificou'),nl,
assert(tarefa(Nome, Idmaq ,Baseline,Deadline,Duracao, Consumos)),
brinca_consumos(Consumos, Idmaq),
write('fez assert das tarefas'),nl,
importar_tarefas_para_kb(Proximo).
```

```
file_to_list(FILE,LIST) :-
   see(FILE),
   inquire([],R),
   reverse(R,LIST),
   seen.

inquire(IN,OUT):-
   read(Data),
   (Data == end_of_file ->
      OUT = IN
        ;    % more
      inquire([Data|IN],OUT) ) .
```

Ficheiro Escalonamentos.pl É neste ficheiro onde é feita toda a lógica de restrições com o objetivo de optimizar o a solução do problema imposto.

```
:-use_module(library(lists)).
:-use_module(library(clpfd)).
:-include('listagens.pl').

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                       %%%
%%%    ESCALONAMENTOS     %%%
%%%                       %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


calculaEmax(EMax,EMax,[]).
calculaEmax(PotMax,EMax,[H|T]) :-
PotMaxNext is PotMax - H,
calculaEmax(PotMaxNext,EMax,T).


separaTasks([],Tasks,Tasks,LS,LS,LE,LE).
separaTasks([SubTask|Next],SubTasks,Tasks,ListaStarts,LS,ListaEnds,LE) :-
criaSubTask(SubTask,SubTasks,SubTasks1,ListaStarts,ListaStarts1,ListaEnds,ListaEnds1,_),
separaTasks(Next,SubTasks1,Tasks,ListaStarts1,LS,ListaEnds1,LE).

criaSubTask([],Tasks,Tasks,LS,LS,LE,LE,_).
criaSubTask([Consumo|Next],ListaSubTasks,Tasks,ListaStarts,LS,ListaEnds,LE,S1) :-
append(ListaSubTasks,[task(S1,1,E1,Consumo,0)],ListaSubTasks1),
append(ListaStarts,[S1],ListaStarts1),
append(ListaEnds,[E1],ListaEnds1),
```

```
criaSubTask(Next,ListaSubTasks1,Tasks,ListaStarts1,LS,ListaEnds1,LE,E1).

getLength([ ], 0).
getLength([_|T],N) :- getLength(T,M), N  is  M+1.

doWhile(BaselinesFinal,BaselinesFinal,_,DeadlinesFinal,DeadlinesFinal,_,Amount1) :- Amou
doWhile(Baselines,BaselinesFinal,BaseLinesActuais,Deadlines,DeadlinesFinal,DeadLinesActu
append(Baselines,[BaseLinesActuais],Baselines1),
append(Deadlines,[DeadLinesActuais],Deadlines1),
Amount1 is Amount - 1,
doWhile(Baselines1,BaselinesFinal,BaseLinesActuais,Deadlines1,DeadlinesFinal,DeadLinesAc

geraLines(_,ListaBaseLinesFinal,ListaBaseLinesFinal,_,ListaDeadlinesFinal,ListaDeadlines
geraLines([BaseLinesActuais|NextBaseline],BaseLines,ListaBaseLinesFinal,[DeadLinesActuai
doWhile(BaseLines,BaseLines1,BaseLinesActuais,Deadlines,Deadlines1,DeadLinesActuais,Amou
geraLines(NextBaseline,BaseLines1,ListaBaseLinesFinal,NextDeadline,Deadlines1,ListaDeadl



restricaoTempo([],[],_).

%% Para o Baseline
restricaoTempo([Baseline|NextBase],[LS|NextS],0) :-
LS #>=Baseline,
restricaoTempo(NextBase,NextS,0).

%% Para o Endline
restricaoTempo([Baseline|NextBase],[LS|NextS],1) :-
LS #=<Baseline,
restricaoTempo(NextBase,NextS,1).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ListaInicioTarefas= lista dos tempos iniciais das tarefas em que o primeiro elemento é
%ListaFimTarefas=lista dos tempos finais das tarefas em que o primeiro elemento é a tare
%Tarefas= lista de task(...)
%PotenciaContratada


% Sem considerar os não escalonaveis caso o valor passado em PotenciaContratadaRestante
% PotenciaContratadaRestante= PotenciaContratada-Potencia consumida pelas tarefas nao es
% Assume-se que o pior caso é as tarefas nao escalonaveis terem uma duracao de 24h, assi

escalonamentoPorConsumoEnergetico(ListaInicioTarefas,ListaFimTarefas,Tarefas,PotenciaCon
```

```
domain(ListaInicioTarefas,0,23),
domain(ListaFimTarefas,1,24),

%%fazer as restricoes do tempo
restricaoTempo(ListaInicioTarefas,ListaInicioTarefasRestricoes,0),
restricaoTempo(ListaFimTarefas,ListaFimTarefasRestricoes,1),

LimiteEnergetico is 0..PotenciaContratadaRestante,

cumulative(Tarefas,limit(LimiteEnergetico)),

append(ListaInicioTarefasRestricoes,ListaFimTarefasRestricoes, Vars1),
append(Vars1,[LimiteEnergetico], Vars),

labeling([minimize(LimiteEnergetico)],Vars),


% Mostrar Resultados
write('Lista de Principios:'),write(ListaInicioTarefasRestricoes),write('\n'),
write('Lista de Terminos:'),write(ListaFimTarefasRestricoes),write('\n'),
write('Potencia Maxima Recomendada:'),write(LimiteEnergetico),write('\n'),
write('Lista de Tarefas:'),write(Tarefas),write('\n').



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/*
calculaCustoTotal(_,[],CustoTotal,CustoTotal).

calculaCustoTotal(LCustos,[task(S,_,_,C,_)|NextTask],CustoActual,CustoTotal) :-
element(S,LCustos,Custo),
Mult #= Custo*C,
CustoActual1 #= CustoActual + Custo*C,
calculaCustoTotal(LCustos,NextTask,CustoActual1,CustoTotal).*/


calculaSomaCustos([],CustoMaximo,CustoMaximo).

calculaSomaCustos([C|NextC],CustoActual,CustoMaximo) :-
CustoActual1 is CustoActual + C,
calculaSomaCustos(NextC,CustoActual1,CustoMaximo).

calculaConsumoTotal([],ConsumoTotal,ConsumoTotal).
```

```
calculaConsumoTotal([task(_,_,_,C,_)|NextTask],ConsumoActual,ConsumoTotal) :-
ConsumoActual1 is ConsumoActual + C,
calculaConsumoTotal(NextTask,ConsumoActual1,ConsumoTotal).

minList([H|T], Min) :-
minList(T, H, Min).

minList([], Min, Min).

minList([H|T], Min0, Min) :-
    Min1 is min(H, Min0),
    minList(T, Min1, Min).

% Para um Custo constante ao longo do tempo
% Sem considerar os não escalonaveis caso o valor passado em PotenciaContratadaRestante
% PotenciaContratadaRestante= PotenciaContratada-Potencia consumida pelas tarefas nao es
% Assume-se que o pior caso é as tarefas nao escalonaveis terem uma duracao de 24h, assi

escalonamentoPeloCustoMinimo(ListaInicioTarefas,ListaFimTarefas,Tarefas,PotenciaContrata

domain(ListaInicioTarefas,0,23),
domain(ListaFimTarefas,1,24),

%%fazer as restricoes do tempo
restricaoTempo(ListaInicioTarefas,ListaInicioTarefasRestricoes,0),
restricaoTempo(ListaFimTarefas,ListaFimTarefasRestricoes,1),

calculaSomaCustos(Custo,0,CustoMaximo),
calculaConsumoTotal(Tarefas,0,ConsumoTotal),
CustoTotal is CustoMaximo * ConsumoTotal,
minList(Custo,CustoMinimo),


%calculaCustoTotal(LCustos,Tasks,0,ConsumoCusto),
CustoTotal #= ConsumoCusto,


LimiteEnergetico is 0..PotenciaContratadaRestante,

cumulative(Tarefas,limit(LimiteEnergetico)),

CustoLimite in CustoMinimo..CustoTotal,


append(ListaInicioTarefasRestricoes,ListaFimTarefasRestricoes, Vars1),
```

```
append(Vars1,[CustoLimite], Vars),

labeling([minimize(CustoLimite)],Vars),

% Mostrar Resultados
write('Lista de Principios:'),write(ListaInicioTarefasRestricoes),write('\n'),
write('Lista de Terminos:'),write(ListaFimTarefasRestricoes),write('\n'),
write('Custo Otimo:'),write(CustoLimite),write('\n'),
write('Lista de Tarefas:'),write(Tarefas),write('\n').


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


escalonamentoPorConsumoEnergeticoComPotenciaContratadaRestanteVariavel(ListaInicioTarefa

domain(ListaInicioTarefas,0,23),
domain(ListaFimTarefas,1,24),

%%fazer as restricoes do tempo
restricaoTempo(ListaInicioTarefas,ListaInicioTarefasRestricoes,0),
restricaoTempo(ListaFimTarefas,ListaFimTarefasRestricoes,1),


%assumir que o pior caso é o conjunto de tarefas que num determinado periodo de 1h consu
minList(ListaPotenciaContratadaRestante,PotenciaContratadaRestante),

LimiteEnergetico is 0..PotenciaContratadaRestante,

cumulative(Tarefas,limit(LimiteEnergetico)),

append(ListaInicioTarefasRestricoes,ListaFimTarefasRestricoes, Vars1),
append(Vars1,[LimiteEnergetico], Vars),

labeling([minimize(LimiteEnergetico)],Vars),

% Mostrar Resultados
write('Lista de Principios:'),write(ListaInicioTarefasRestricoes),write('\n'),
write('Lista de Terminos:'),write(ListaFimTarefasRestricoes),write('\n'),
write('Potencia Maxima Recomendada:'),write(LimiteEnergetico),write('\n'),
write('Lista de Tarefas:'),write(Tarefas),write('\n').

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
escalonamentoPeloCustoMinimoComPotenciaContratadaRestanteVariavel(ListaInicioTarefas,Lis

domain(ListaInicioTarefas,0,23),
domain(ListaFimTarefas,1,24),

%%fazer as restricoes do tempo
restricaoTempo(ListaInicioTarefas,ListaInicioTarefasRestricoes,0),
restricaoTempo(ListaFimTarefas,ListaFimTarefasRestricoes,1),

calculaSomaCustos(Custo,0,CustoMaximo),
calculaConsumoTotal(Tarefas,0,ConsumoTotal),
CustoTotal is CustoMaximo * ConsumoTotal,
minList(ListaCustos,CustoMinimo),


%calculaCustoTotal(LCustos,Tasks,0,ConsumoCusto),
CustoTotal #= ConsumoCusto,

minList(ListaPotenciaContratadaRestante,PotenciaContratadaRestante),

LimiteEnergetico is 0..PotenciaContratadaRestante,

cumulative(Tarefas,limit(LimiteEnergetico)),

CustoLimite in CustoMinimo..CustoTotal,


append(ListaInicioTarefasRestricoes,ListaFimTarefasRestricoes, Vars1),
append(Vars1,[CustoLimite], Vars),

labeling([minimize(CustoLimite)],Vars),

% Mostrar Resultados
write('Lista de Principios:'),write(ListaInicioTarefasRestricoes),write('\n'),
write('Lista de Terminos:'),write(ListaFimTarefasRestricoes),write('\n'),
write('Custo Otimo:'),write(CustoLimite),write('\n'),
write('Lista de Tarefas:'),write(Tarefas),write('\n').
```

Ficheiro listagens.pl Encontra-se neste ficheiro os predicados de auxilio para gerar as listas pretendidas para serem utilizadas nos predicados em escalonamentos

```
get_todos_dispositivos(Lista):-
write('Dispositivos na Base de Conhecimento: '),
```

```
findall(Nome-Id-X, escalonavel(Nome,Id, X), Lista1),
findall(Nome-X , nescalonavel(Nome,X), Lista2),
append(Lista1, Lista2, Lista).
get_todos_dispositivos(_):- write('Nao existem dispositivos').

get_todos_escalonaveis(Lista):-
write('Dispositivos na Base de Conhecimento: '),
findall(Nome-Id-Consumo, escalonavel(Nome, Id, Consumo), Lista).
get_todos_escalonaveis(_):- write('Nao existem nescalonaveis').


get_todos_nomes_dispositivos(Lista):-
findall(Nome, dispositivos(Nome, _), Lista).


get_todas_tarefas(Lista):-
write('Tarefas na Base de Conhecimento: '), nl,
findall(Nome-Idmaq-Baseline-Deadline-Duracao-Subtarefas, tarefa(Nome, Idmaq ,Baseline,De
get_todas_tarefas(_):- write('Nao existem tarefas').


%% LEE %Lista de consumos de dispositivos esc.
%% LED %Lista de energias disponíveis no decorrer de 24 horas
%% PotMax %Potencia contratada por defeito
%% EMax %Potência contratada com n esc.
%% EMaxCalc %Potência contratada optimizada
%% LS %Lista de starting times
%% LD %Lista de durações de dispositivos esc.
%% LE %Lista de ending times
%% Machines %Lista das máquinas
%% Tasks %Lista das Tarefas

cria_lista_disponivel(LED):-
findall(Potenciacontratada, potenciacontratada(Potenciacontratada), [Pot|_]),
(for(X,1,24), foreach(I,LED) do I is Pot).

cria_lista_consumo(LEE):-
findall(Consumo, subtask(_,_,_,Consumo,_), LEE).

cria_lista_starting(LS):-
findall(St,subtask(St,_,_,_,_), LS).

cria_lista_duracao(LD):-
findall(Duracao, subtask(_,Duracao,_,_,_), LD).
```

```
cria_lista_ending(LE):-
findall(Et,subtask(_,_,Et,_,_), LE).

cria_lista_machines(Machines,EMax):-
findall(machine(Id, EMax), escalonavel(_,Id, _), Machines).

cria_lista_tasks(Tasks):-
findall(task(Start,Dur,End,Consumo,Idmaq), subtask(Start, Dur, End, Consumo, Idmaq), Tas

cria_lista_n_escalonavel(LNE):-
findall(Consumo, nescalonavel( _, Consumo), LNE).

get_pot_max(PotMax):-
findall(Potenciacontratada, potenciacontratada(Potenciacontratada), [Pot|_]),
PotMax is Pot.

get_custos(LCustos) :-
findall(Custo,custo(Custo),LCustos).
```