Instituto Superior de Engenharia do Porto

Mestrado em Engenharia Electrotécnica
Sistemas Eléctricos de Energia
Departamento de Engenharia Electrotécnica

Resolução do Problema do Escalonamento da Produção de Energia Eléctrica usando Programação Lógica por Restrições

Rui Pedro Teixeira Malheiro

Dissertação submetida para satisfação
dos requisitos do grau de mestre em
Engenharia Electrotécnica – Sistemas Eléctricos de Energia

Dissertação realizada sob a orientação do
Professor Doutor
Nuno Filipe da Fonseca Bastos Gomes
do Instituto Superior de Engenharia do Porto

Porto, Novembro de 2010

Resumo

A operação dos Mercados de Energia Eléctrica passa, actualmente, por uma profunda reestruturação, com o principal foco nas transacções do sistema de transmissão entre os diferentes agentes. Tendo isso em conta, o serviço de transmissão neste novo esquema de funcionamento do Mercado de Energia Eléctrica deve ser provido de máxima eficiência económica, atendendo sempre às restrições de segurança do sistema. Com esta reorganização do sector eléctrico da última década surgiu também a necessidade de rever os modelos tradicionais de optimização económica do Sistema Eléctrico de Energia, como por exemplo o despacho e pré-despacho (unit commitment).

A reestruturação e liberalização dos mercados de energia eléctrica trouxeram novas restrições a alguns dos problemas tradicionais associados aos Sistemas Eléctricos de Energia. Um desses problemas é o Escalonamento da Produção de Energia Eléctrica, que no contexto actual, implica quase sempre negociação entre os diferentes agentes do mercado e consequentemente reescalonamento. A maioria dos métodos usados para a resolução do problema não permitem reformular o pré-despacho, algo para que a Programação Lógica por Restrições é extremamente adequada.

O trabalho desenvolvido nesta dissertação visa criar uma aplicação computacional com base na Programação Lógica por Restrições, através da plataforma ECLiPSe, para resolver o problema do Escalonamento da Produção de Energia Eléctrica dos grupos térmicos, demonstrando assim a versatilidade e flexibilidade deste tipo de programação aplicada a problema combinatoriais deste género.

Abstract

The operation of the Electricity Markets is currently going through a major restructuration, with its primary focus over the transactions within the transmission system between the different agents. Taking this into account, the transmission service in this new scheme of operation of the Electricity Market should be provided with maximum economic efficiency, always considering the restrictions of system security. With this reorganization of the electricity sector over the last decade, it has also appeared a need to review the traditional models of the economic optimization of the Electrical Power Systems, especially within the dispatch and pre-dispatch (unit commitment).

The restructuration and the liberalization of electricity markets brought new restrictions to some of the traditional problems associated with the Electrical Power Systems. One of these problems is Scheduling the Production of Electricity, which within the present context, almost always involves negotiation between the different market agents and hence the rescheduling. Most of the methods used to solve the problem do not allow the reformulation of the pre-dispatch, which for the Constraint Logic Programming is adequated.

The work developed in this dissertation aims to create a computer application based on the Constraint Logic Programming, through the ECLiPSe platform, to solve the problem of the Scheduling of the Production of Electricity of the thermal groups, thus demonstrating the versatility and flexibility of this type of programming applied to combinational problems of this kind.

Agradecimentos

É com enorme prazer que manifesto publicamente a minha gratidão pela ajuda e apoio, de forma directa ou indirecta, que me foi dada e que foi primordial para a execução deste trabalho.

Gostaria de fazer um agradecimento especial ao Professor Doutor Nuno Filipe da Fonseca Bastos Gomes, responsável pela orientação desta dissertação, que me ajudou bastante e que se disponibilizou sempre a indicar-me o caminho certo.

Agradeço do fundo do meu coração à minha família por todo o apoio, suporte e por tudo o que me deram que me possibilitou concluir este trabalho. Para os meus pais, para a minha avó e para o meu irmão o meu mais sincero obrigado por me ajudarem de forma incondicional a trilhar este caminho. O vosso apoio teve um papel bastante importante na conclusão deste trabalho.

Á minha namorada e companheira de vida Ana quero dedicar esta dissertação com todo o amor e carinho que sinto por ela e que ela sabe que sinto. Sem a sua ajuda nunca teria conseguido chegar onde cheguei, ser o que sou e neste caso concluir esta dissertação. Pelo seu apoio incontestável, por tudo o que representa para mim e pelo que teve que aturar durante este tempo todo, o meu mais profundo e sincero obrigado.

A todos os meus verdadeiros amigos, que estiveram lá quando foi preciso, que sempre me motivaram e apoiaram, e que tiveram uma influencia bastante especial para a conclusão deste trabalho. O meu mais sincero obrigado e vocês sabem quem são.

Para a Mi

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

EPEE	Escalonamento da Produção de Energia Eléctrica
PLR	Programação Lógica por Restrições
MEE	Mercado de Energia Eléctrica
SEE	Sistema Eléctrico de Energia
SE	Sistema de Energia
FCT	Função Custo Total
PR	Programação com Restrições
IA	Inteligência Artificial
PSR	Problemas de Satisfação de Restrições
B&B	Branch & Bound

1

Introdução

1.1 Introdução

O Escalonamento da Produção de Energia Eléctrica (EPEE) é um dos mais importantes problemas combinatoriais s a ocorrer numa base diária a uma escala planetária. A procura por energia e a sua produção têm caminhado com ritmos semelhantes, o crescente número da população mundial obriga a uma constante evolução tanto na capacidade de produção energética, como na oferta dessa mesma energia. Este facto originou a reestruturação do Mercado de Energia Eléctrica (MEE) com vista à sua liberalização, passando assim a existir livre concorrência para o fornecimento energia eléctrica.

A profunda reestruturação do sector eléctrico a nível mundial começou a fazer-se notar a partir de 1990, tendo como ideia chave a liberalização dos segmentos potencialmente competitivos e a regulação dos segmentos considerados como monopólios naturais. Portugal começou por volta de 1995 a reestruturação do seu sector eléctrico, com a aplicação de legislação específica que pretendia separar o sistema regulado (SEP) do sistema liberalizado (SENV). Contudo só em 2007 teve lugar a aparição do MIBEL (Mercado Ibérico de Electricidade), integrando desta forma o mercado português e o mercado espanhol, que por seu lado já tinha sido liberalizado em 1998.

A reestruturação do MEE está a exigir às empresas que apresentem soluções dinâmicas e flexíveis, confrontando-as todos os dias com novos desafios. Com a intensificação da competição global, é obrigatório reduzir custos e aproveitar as oportunidades mal surjam. Com as constantes mudanças de legislações e de tecnologias inovadoras criam-se problemas que fazem com que um bom planeamento seja uma vantagem competitiva crucial. Não será obviamente com o tradicional planeamento manual que estes desafios serão ultrapassados, o mercado precisa de ferramentas flexíveis e readaptáveis, de entre as quais a Programação Lógica por Restrições (PLR) se destaca. Sendo um paradigma de programação que disponibiliza ferramentas que permitem desenvolver aplicações capazes de se adaptar rapidamente a qualquer tipo de problema, apenas com a redefinição das restrições a aplicar.

As empresas produtoras de electricidade e a(s) empresa(s) que gerem o Sistema Eléctrico de Energia (SEE) têm a responsabilidade de decidir qual a melhor forma de satisfazer as variações na procura de electricidade, que podem ser de ciclo diário ou ciclo semanal. O problema da optimização a curto prazo resume-se a definir um plano para produção de energia eléctrica minimizando os custos associados, como por exemplo o custo de combustível, satisfazendo sempre as inúmeras restrições do sistema.

Existem dois problemas de optimização de curto prazo, o pré-despacho (unit commitment) e o despacho económico. O pré-despacho é o processo de decisão de quais as unidades geradoras de energia eléctrica, que produzem, quando produzem, e quanto produzem. Ou seja, é quando se decide quais os grupos geradores que deverão arrancar, ou desligar, mediante todas as restrições técnicas que o sistema assim obriga. O despacho económico é o processo de decisão sobre a quantidade de energia a ser produzida por cada unidade em cada período de tempo, de forma a minimizar uma determinada função de custo.

O pré-despacho é um problema de optimização bastante complexo visto possuir um número astronómico de combinações possíveis entre todos os estados de unidades geradoras (ligadas/desligadas) durante todo o período do seu estudo. Como o objectivo da presente dissertação é a resolução do problema do EPEE, o desenvolvimento da aplicação informática, que serve de suporte a esta dissertação, versará simultaneamente sobre o pré-despacho e o despacho económico relativamente aos grupos térmicos.

1.2 Objectivos do Trabalho

O problema do Escalonamento da Produção da Energia Eléctrica (EPEE) apesar de ser bastante complexo obedece sempre a um número de restrições. A existência destas restrições torna a aplicação da Programação Lógica por Restrições (PLR) a este problema adequada.

Com o intuito de resolver o problema do EPEE nos grupos térmicos esta dissertação tem como principais objectivos:

- Criação de uma aplicação computacional em PLR na plataforma ECLiPSe que dê resposta a esse problema;
- Demonstração da grande flexibilidade e aplicabilidade da Programação Lógica por Restrições a este tipo de problemas;
- Comparação entre a solução encontrada e soluções já existentes para o mesmo tipo de problema e se possível melhoria a nível de resultados;
- Comprovação das melhores soluções aplicando a PLR a este tipo de problema, face a soluções já existentes.

1.3 Organização da Dissertação

A organização desta dissertação está dividida em seis capítulos. O presente capítulo consiste numa breve introdução ao tema da dissertação, no qual se apresenta o seu âmbito, enquadramento e uma descrição sucinta do problema sobre o qual esta incide. Refere-se ainda de modo genérico o trabalho realizado ao longo do desenvolvimento desta dissertação.

O segundo capítulo é dedicado à apresentação do Problema do Escalonamento da Produção da Energia Eléctrica. É efectuada uma abordagem à contextualização do problema e realçada a importância que este problema tem diariamente. É explicada toda a mecânica do problema abordando os seus temas mais importantes, como é o caso dos custos, das restrições e das variáveis associadas. Neste capítulo encontramos ainda uma explicação detalhada das restrições a serem aplicadas no problema, e no final apresenta-se a formalização do problema onde estão também todas as nomenclaturas utilizadas neste problema.

Durante o terceiro capítulo a PLR é tratada com algum detalhe. Neste capítulo pode-se encontrar uma breve introdução aos fundamentos da PLR e aos aspectos relacionados com o desenvolvimento de aplicações recorrendo à tecnologia das restrições com domínios finitos. Os conceitos gerais relacionados com esta tecnologia são aí introduzidos. Segue-se a caracterização dos problemas de satisfação de restrições, como ponto de partida para a descrição do paradigma da PLR com domínios finitos. Como suporte para os assuntos tratados nos capítulos seguintes, a descrição da PLR com domínios finitos inclui a especificação formal de uma linguagem genérica que permite especificar os problemas segundo o paradigma da PLR.

O quarto capítulo descreve-se a implementação do problema com recurso à plataforma ECLiPSe. É realizada uma introdução à plataforma onde se pode apreender as suas funcionalidades. De seguida é descrito todo o modelo de programação implementado para dar resposta ao problema do EPEE, onde se destacam o levantamento de variáveis e domínios, a implementação das restrições, a aplicação da função de optimização e a abordagem à pesquisa e optimização das soluções.

No quinto capítulo apresentam-se os diversos resultados da resolução dos casos práticos de 4 e 38 unidades de produção de energia eléctrica através da aplicação desenvolvida. Efectuam-se também análises aos valores encontrados e tecem-se considerações sobre as estratégias de optimização.

Por último, no sexto capítulo, resumem-se as principais conclusões da presente dissertação, salientando os tópicos mais importantes a reter. Termina-se com perspectivas de desenvolvimentos futuros na continuidade deste trabalho.

2

O Problema do Escalonamento da Produção da Energia Eléctrica

2.1 Abordagem Conceptual

O Problema do Escalonamento da Produção de Energia Eléctrica (EPEE) considerado nesta tese é um problema de optimização combinatorial. O seu principal objectivo é o de minimizar o custo total da produção de electricidade, satisfazendo sempre a procura de energia do sistema, através do escalonamento dos grupos geradores sujeitos a várias restrições. Estes grupos são escalonados dentro do Sistema de Energia Eléctrica ao longo de um determinado horizonte temporal , durante o qual estas unidades podem assumir o estado de ligadas (1) ou desligadas (0) em cada período unitário de tempo. Os métodos tradicionais de resolução do problema do EPEE são: Listas de Prioridades, Relaxamento de Lagrange, redes neuronais, programação dinâmica, algoritmos genéricos, ect. [26].

Neste trabalho o custo de produção podem dividir-se em duas categorias: custos de combustível e custos de transição. Os custos de combustível dependem não só da quantidade de energia a ser gerada pelas unidades mas também dos estados dessas mesmas unidades em determinada hora. Os custos de transição surgem quando os estados das unidades se alteram. Estes custos dependem do tempo que certa unidade esteve ligada ou desligada dividindo-se em custos de arranque ou de paragem [26].

De todas as restrições que estão implícitas no problema do EPEE, as restrições da procura são as mais comuns e as mais importantes. As restrições de procura (ou demanda) devem garantir que a energia total fornecida pelo sistema a qualquer hora terá que ser superior à demanda total das cargas. As restrições de produção de energia definem assim os limites superiores e inferiores de produção de energia de cada unidade. Estas restrições estão directamente relacionadas com a capacidade de produção de cada grupo gerador. Outras restrições comuns neste tipo de problema garantem o tempo mínimo de arranque e de paragem, a reserva girante, grupos obrigatórios, restrições de pessoal, ect. [26].

2.2 Características Gerais do Problema

O conjunto de decisões que leva à atribuição de uma certa carga a cada uma das unidades de produção do SEE começa pela definição das unidades que, num determinado período, estarão em funcionamento. Esta fase é necessária visto que cada grupo gerador, mesmo na potência mínima, fica com um custo associado que é constante e não desprezável, não sendo económico manter em funcionamento um número excessivo de unidades [29].

A conhecida variação diária do diagrama de cargas obriga à necessidade de ir ligando e desligando grupos ao longo do dia, o que envolve custos associados à ligação e ao corte, existindo também diversas restrições técnicas que limitam as opções de quem toma as decisões (tempos de arranque, tempos mínimos de paragem, ect.).

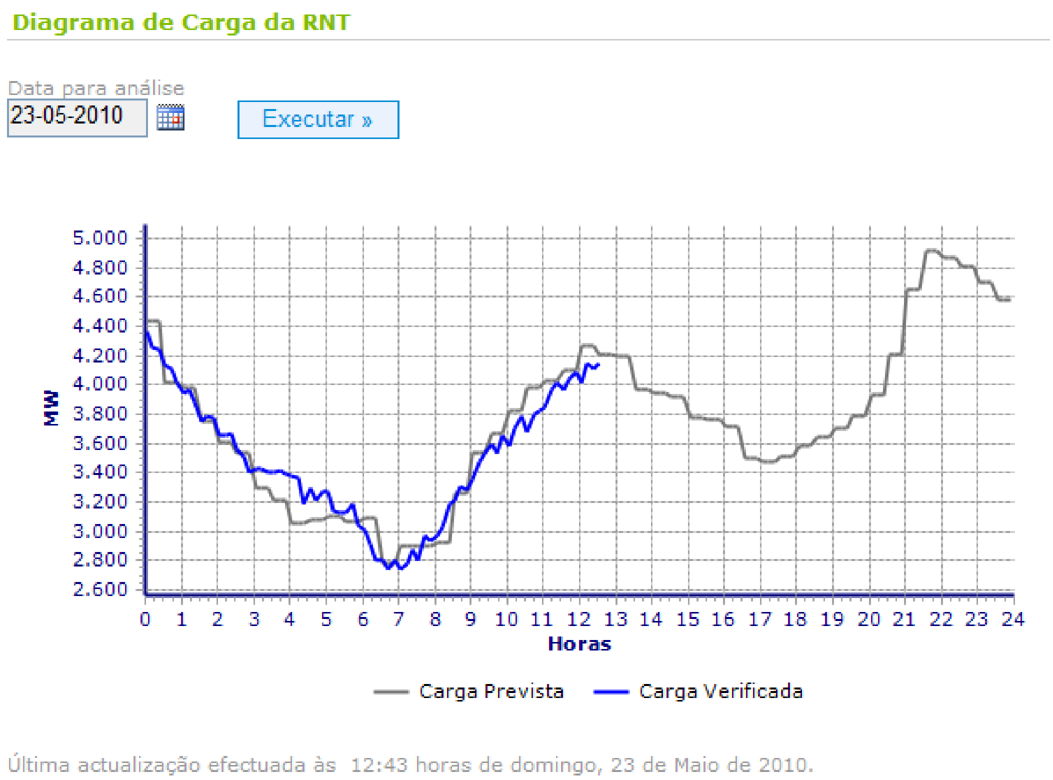

Figura 2.1 – Diagrama de cargas da REN [28]

Trata-se portanto de um problema impossível de ser resolvido no momento, sendo normalmente projectada a sua resolução para a semana seguinte, com intervalos de uma hora. É necessário também prever uma folga, designada por reserva girante, para ter em conta os aumentos inesperados de carga ou simplesmente para garantir o serviço em caso de avaria de alguma unidade.

Todos os aspectos acima referidos têm que ser tidos em conta no problema do escalonamento dos grupos (muitas vezes designado por *unit commitment*), onde se faz também o pré-despacho, a tal antecipação grosseira da carga a atribuir a cada grupo, tomando em conta os custos de funcionamento associados ao consumo de combustível.

É importante mencionar também a presença dos grupos hídricos no SEE nacional, uma vez que a abordagem que se realiza nesta dissertação corresponde unicamente aos grupos térmicos. A coordenação entre os grupos hídricos e térmicos normalmente funciona realizando um pré-despacho separado para o subsistema hídrico, onde são tidas em conta as previsões de afluências e as restrições próprias dos sistemas hidroeléctricos. São então fixados valores finais para os volumes armazenados nas albufeiras, decorrentes de estudos energéticos a longo prazo. A natureza dinâmica do problema geral de escalonamento, e as relações entre os subsistemas térmico e hídrico, particularmente no que respeita à bombagem, têm levado, no entanto, a uma tentativa de integração em modelos completos (coordenação hidro-térmica) [27].

2.2.1 Os Custos

Os custos associados aos grupos térmicos dependem, obviamente do tipo de máquina primária (turbina a vapor, turbina a gás, grupo diesel) e de outros aspectos, como o processo de geração do vapor (fuel-oil, carvão, nuclear), do reaproveitamento de outras formas de energia ou até mesmo da idade da máquina, o que implica uma análise caso a caso em problemas reais.

Como podemos observar na Figura 2, os grupos térmicos ainda são uma fonte de energia muito considerável no SEE Português.

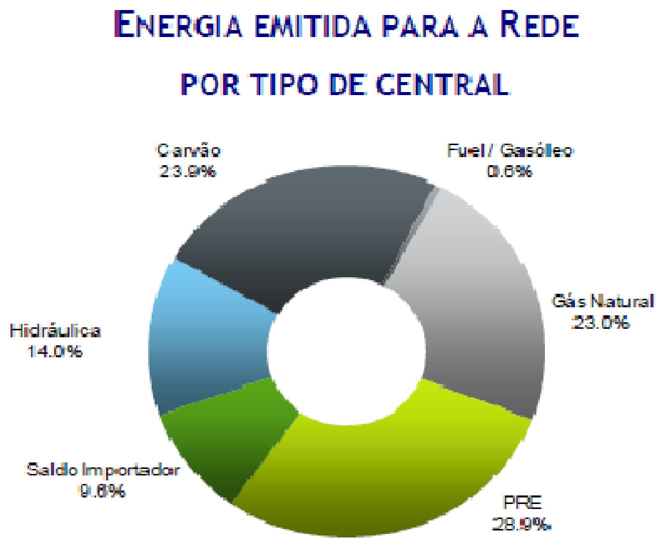

Figura 2.2 – Energia Emitida por central em 2009 [28]

Observemos então os diferentes custos associados aos grupos térmicos [27]:

- **Custo de Funcionamento**: é o custo associado ao consumo de combustível para a produção de energia, logicamente se a produção de energia aumentar numa unidade, este custo aumentará em igual proporcionalidade, como se demonstra na Figura 3. Pode-se reparar também que a função custo de funcionamento inclui o ponto (0,0), que corresponde ao estado de paragem da unidade, o que torna a função descontínua.

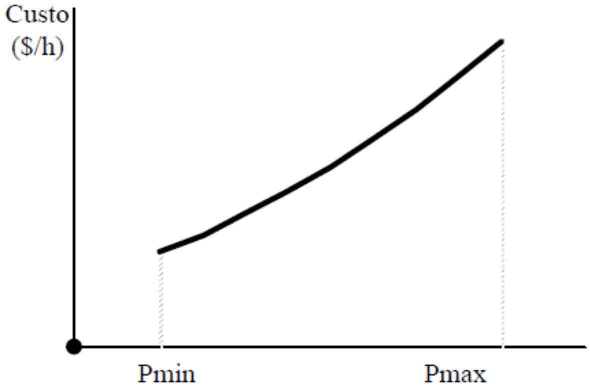

Figura 2.3 – Função custo de funcionamento

- **Custo de Arranque**: custo independente da potência que possa a vir ser produzida pelos grupos. Este custo está associado ao arranque que pode ser a frio (*cooling*) ou a quente (*banking*), e no caso das centrais com turbina a vapor depende do tempo de paragem anterior e do facto de se manterem ou não as caldeiras quentes durante o período de paragem. No caso dos grupos diesel, o arranque pode incluir patamares de aquecimento intermédio ou até mesmo mudança de combustível, daí não ser simples descrever o custo de arranque nestes casos.

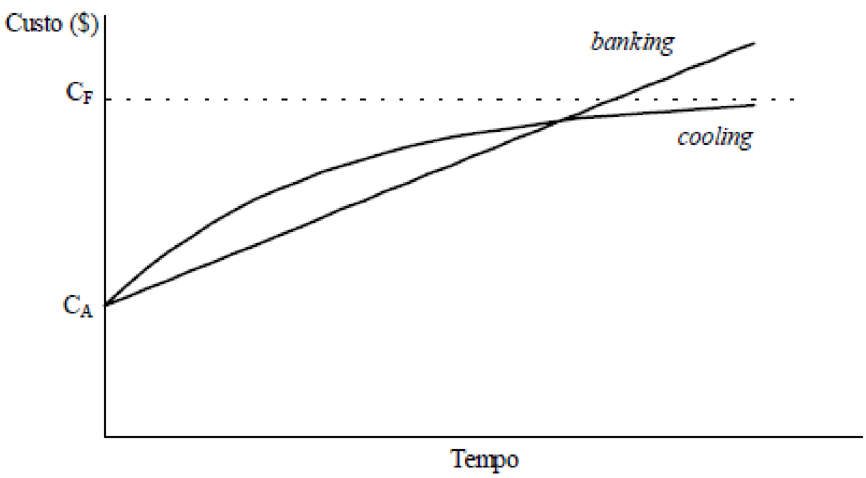

Figura 2.4 – Custos de arranque de centrais com turbina a vapor
(CF – Custo de arranque a frio CA – Custo de arranque a quente)

- **Custo de Paragem:** os custos de paragem estão associados às condições necessárias para um arranque a quente (*banquing*), estas condições reflectem-se em custos que se denominam de custos de paragem.

2.2.2 As Restrições

A restrição fundamental a considerar neste problema de EPEE é, como habitualmente nos SEE, a satisfação da carga, ou seja, a potência total disponível (soma das potências máximas de todos os grupos escalados) tem que ser superior à carga total prevista, em todos os intervalos de tempo [27]. Como se disse anteriormente, a diferença deverá corresponder à reserva girante definida para cada intervalo, de acordo com um dos princípios seguintes:

- Valor igual a uma percentagem da carga prevista para o intervalo;

- Valor igual à potência máxima da maior unidade em funcionamento;

- Reserva que garanta um risco de perda de carga inferior a um certo valor, tendo em conta as probabilidades de avaria dos grupos.

Os grupos térmicos, sobretudo aqueles em que a máquina primária é a turbina a vapor, não podem ser ligados de forma produzirem imediatamente a potência que se pretende, nem podem deslastrar imediatamente a carga que lhes está atribuída.

Há também motivos técnicos que excluem o funcionamento ou paragem durante períodos curtos. De forma abreviada, as restrições associadas são os que se apresentam a seguir:

- **Tempo de arranque**: para cada tipo de grupo, define-se um tempo mínimo de arranque que depende do tempo de paragem anterior e está relacionado com a necessidade de aquecer caldeiras, obter pressões de vapor e outros condicionalismos técnicos. Em consequência, a decisão de utilizar o grupo pode ter de ser tomada muito antes da hora a que a potência respectiva vai ser necessária.

- **Tempos mínimos de paragem e de funcionamento**: por razões fundamentalmente de ordem técnica, os períodos de paragem e funcionamento não devem ser demasiado 4 reduzidos. Valores mínimos típicos para grupos com turbinas a vapor são 2 a 12 horas para o tempo de paragem e 1 a 8 horas para o tempo de funcionamento. Os restantes tipos de máquinas apresentam tempos mínimos menores.

- **Limites de produção**: valor máximo e mínimo da potência produzida pelo grupo, fixados por razões técnicas e económicas. Por exemplo, nos grupos Diesel, a produção a potências baixas é economicamente inviável, embora fosse possível tecnicamente (usando óleo diesel em vez de fuel-oil). Os valores típicos da potência mínima para grupos com turbina a vapor são 40 a 70% da potência máxima. Estes limites também se utilizam no despacho.

- **Taxas máximas de tomada e deslastre de carga**: não sendo possíveis variações muito rápidas da potência produzida pelos grupos, definem-se taxas máximas de tomada e deslastre de carga (MW/h) que condicionam as alterações de produção em intervalos de tempo sucessivos. No pré-despacho, estes limites têm sobretudo influência nos períodos iniciais e finais de funcionamento.

Finalmente, uma referência a duas restrições um pouco diferentes, mas que também podem ter que ser consideradas:

- **Restrições sobre Recursos Humanos (materiais)**: numa central com diversos grupos produtores, podem surgir restrições ao arranque simultâneo de vários grupos por não existir recursos humanos suficientes para efectuar todas as operações necessárias. Esta restrição tem diminuído de importância devido à progressiva automatização do processo de arranque.

- **Grupos obrigatórios (must-run)**: por diversas razões, podem existir indicações de que certos grupos têm que estar obrigatoriamente ligados em certos períodos do horizonte de escalonamento. Este tipo de restrições pode aumentar o custo global, mas favorece a rapidez da resolução do problema, por diminuir o número de variáveis inteiras.

Os aspectos focados, nomeadamente a questão dos tempos de arranque, mostram que o pré-despacho tem uma escala temporal completamente diferente da do despacho, possuindo, além disso, muito maior incerteza na definição das cargas e maior complexidade na formulação matemática, dada a presença de variáveis inteiras (ter ou não ligado cada grupo em cada período) e de funções custo não convexas.

2.3 Formalização do Problema

Considerando as características apresentadas anteriormente podemos definir formalmente o Problema do EPEE. Esta formalização será feita numa linguagem próxima da linguagem matemática e servirá de base ao desenvolvimento do modelo baseado em restrições utilizado nos capítulos seguintes.

Nomenclaturas do Problema

C(U)	Função objectivo do custo total
$FC_i(P_{ij})$	Função do custo do combustível da unidade *i*
A_i, B_i, C_i	Coeficientes de custo de combustível constantes da unidade *i*
P_{ij}	Energia produzida pela unidade *i* no período *j*
SC_i	Custo de arranque da unidade *i*
SD_i	Custo de paragem da unidade *i*
U_{ij}	Estado ligado/desligado da unidade *i* à hora *j*, $Uij \in \{0,1\}$
U	Matriz de decisão dos elementos Uij
N	Número de unidades térmicas produtoras
T	Número de horas no período de estudo
D_j	Demanda do sistema de cargas à hora *j*
P_{imax}	Capacidade máxima de produção de energia da unidade *i*
P_{imin}	Capacidade mínima de produção de energia da unidade *i*

P_{srj}	Reserva do sistema, que consiste na maior capacidade de produção entre as unidades escaladas à hora j
MDT_i	Tempo mínimo de paragem da unidade *i*
MUT_i	Tempo mínimo de arranque da unidade *i*
j_{Si}	Hora a que cada unidade *i* arrancou
j_{Di}	Hora a que cada unidade *i* parou

A Função Custo Total (FCT), que será a função objectivo deste problema do EPEE, pode então ser formulada matematicamente através da seguinte expressão:

Função Objectivo

$$C(U) = \sum_{i=1}^{N} \sum_{j=1}^{T} \left(U_{ij}\, FC_i(P_{ij}) + SC_i\, U_{ij}\,(1 - U_{ij-1}) + SD_i\, U_{ij-1}\,(1 - U_{ij}) \right)$$

Esta função irá reflectir o custo total dos grupos térmico, durante o período de estudo. Efectuando o somatório dos custos de combustível, custos de arranque e de paragen, de todas as unidades e em todos os períodos de tempo, obtém-se o valor monetário pelo qual se irá avaliar a solução.

Função do custo do combustível

A função do custo do combustível é um polinómio de segunda ordem onde se expressam os coeficientes relativos a cada unidade.

$$FC_i(P_{ij}) = A_i + B_i\, P_{ij} + C_i\, P_{ij}^2 \text{ , para } i = 1, \ldots , N$$

Restrições

Atendendo a toda a envolvente do problema, as restrições que deverão ser tidas em conta para o Problema do Escalonamento da Produção da Energia Eléctrica nos grupos térmicos são as seguintes:

- **Equilíbrio de Cargas**

$$\sum_{j=1}^{T} (U_{ij} P_{ij}) = D_j$$

- **Limites de produção dos geradores**

$$P_{imin} \leq P_{ij} \leq P_{imax}$$

- **Reserva Girante**

$$D_j + P_{srj} \leq \sum_{j=1}^{T} (U_{ij} P_{imax})$$

$$\sum_{j=1}^{T} (U_{ij} P_{imin}) \leq D_j$$

- **Tempo Mínimo de Arranque**

$$U_{ij} = 1 \text{ para } \sum_{j=j_{si}}^{j-1} U_{ij} < MUT_i$$

- **Tempo Mínimo de Paragem**

$$U_{ij} = 0 \text{ para } \sum_{j=j_{Di}}^{j-1} (1 - U_{ij}) < MDT_i$$

Existem outras restrições que se utilizam diariamente como a restrição das unidades de produção contínua ou restrições de emissões poluentes, que não irão ser consideradas nesta dissertação. Estas restrições apresentadas é que realmente traduzem o Problema do EPEE do seu ponto de vista eléctrico.

3

Programação Lógica por Restrições

3.1 Introdução

A necessidade da implementação de novos métodos de resolução do problema do Escalonamento da Produção da Energia Eléctrica é real. Existe cada vez mais demanda por rapidez e optimização neste problema, que de dia para dia se torna cada vez mais complexo. A entrada nas redes eléctricas das energias renováveis; o constante aumento de grupos produtores de energia eléctrica; a procura cada vez maior de energia; a velocidade a que hoje em dia o mundo gira, sem se quer se ponderar como seria se existisse um *blackout* de alguns dias, são factores que fazem do EPEE um problema com características únicas e tão importante para cada um de nós. Através desta descrição podemos imaginar o impacto à escala mundial que este problema tem tido com as melhorias sucessivas na sua resolução, tanto a nível de tempos de optimização como a nível da própria solução final. Desta forma torna-se mandatório explorar todos os domínios programacionais, com todas as heurísticas possíveis de forma a se encontrar a melhor resolução possível. Com os constantes avanços a nível da tecnologia informática os métodos descobertos à 10 anos poderão hoje em dia ter futuro devido à elevada capacidade de processamento dos computadores de hoje em dia. Por esse motivo não podemos descurar os estudos realizados em torno de uma ferramenta tão poderosa para a resolução de problemas de escalonamento como é Programação Lógica por Restrições.

Ao longo das últimas duas décadas tem vindo a afirmar-se um novo paradigma da programação com imenso potencial, baseada na programação em lógica e na computação baseada em restrições. Este novo paradigma, conhecido por Programação Lógica por Restrições (PLR), tem vindo a conhecer uma aceitação crescente por parte das comunidades de investigadores e de utilizadores. Proporciona uma forma elegante, abstracta e declarativa para especificar problemas. Os sistemas baseados em restrições apresentam uma forte componente teórica, não deixando de ser apelativos aos sectores que requerem desenvolvimento tecnológico por esse facto [32].

A PLR corporiza um paradigma computacional que combina a natureza declarativa da programação em lógica com a eficiência dos métodos de resolução de problemas que recorrem a restrições. Este paradigma da programação tem-se mostrado eficaz na resolução de problemas que são intratáveis quando se recorrem a outras técnicas. Entre esses problemas encontram-se diversas classes de problemas de natureza combinatória, tais como os problemas de escalonamento, de planeamento e de atribuição de recursos. É de notar que uma das vantagens mais significativas deste modelo passa por oferecer um menor tempo de desenvolvimento de programas e uma eficiência comparável à das linguagens imperativas [32]. Em [2] afirma-se que a programação com restrições representa a abordagem mais aproximada, alguma vez efectuada, ao santo Graal da programação, em que um utilizador declara o problema e o computador soluciona-o.

Tendo em conta o Problema do EPEE e considerando a abordagem seguida para o solucionar, pretende-se neste capítulo fornecer uma visão global da PLR em termos de metodologia de resolução de problemas, a partir de restrições aplicadas a domínios finitos.

3.2 Conceitos Gerais sobre Restrições

A problemática dos sistemas que têm o seu comportamento condicionado pela existência de restrições surge naturalmente em muitas áreas da actividade humana. São meios naturais de expressão para formalizar as regularidades e dependências que estão na base dos mundos computacionais e físicos, bem como nas suas abstracções matemáticas. Na actividade humana as restrições constituem um ponto-chave do senso comum na condução do raciocínio. Frases como "eu posso estar lá amanhã das quatro às seis horas da tarde" englobam uma restrição típica usada para o planeamento de tempo. Dependendo do domínio do problema, poder-se-ão encontrar outros exemplos de restrições: "a soma dos ângulos de um triângulo é 180 graus"; "a soma das correntes que fluem num nó é igual a zero"; e "uma resistência num circuito eléctrico obedece à lei de Ohm" [32].

Intuitivamente, uma restrição pode ser considerada como uma condicionante num espaço de possibilidades [3]. As restrições matemáticas especificam de uma forma clara e precisa as relações entre diversos parâmetros desconhecidos (variáveis), cada um tomando um dado valor no domínio considerado. As restrições restringem os valores possíveis que as variáveis podem tomar e representam alguma informação (parcial) sobre as variáveis em jogo. Estas apresentam algumas propriedades deveras interessantes, e expressas no texto que se segue:

- Como se referiu, as restrições traduzem *informação parcial* acerca de uma dada questão ou problema, uma vez que de uma restrição, por si só, não é de esperar que se determine o valor das variáveis do problema;

- As restrições são *aditivas*. A uma restrição r_1 (e.g. $X+Y \geq Z$) pode ser adicionada uma outra restrição r_2 (e.g. $X+Y \leq Z$). A ordem segundo a qual as restrições são impostas é irrelevante, o que está em jogo é que no final à conjunção dos termos que denotam as restrições seja atribuído o valor verdadeiro;

- As restrições raramente são *independentes*; ou seja, geralmente partilham variáveis. A colocação das restrições r_1 e r_2 resulta na obtenção da restrição $X + Y = Z$;

- As restrições são ainda *não direccionais*: Considerando a restrição $X + Y = Z$, esta pode ser usada para determinar a sua forma equivalente em X ($X = Z - Y$) ou em Y ($Y = Z - X$); e

- Finalmente, as restrições são de natureza *declarativa* pelo facto de apenas denotarem as relações que devem ser asseguradas entre variáveis sem especificar um procedimento computacional para estabelecer esse relacionamento.

3.2.1 Programação com Restrições

A Programação com Restrições (PR) passa pelo estudo de sistemas computacionais baseados em restrições. O princípio base em que assenta a PR, e a sua utilização na resolução de problemas, passa pela indicação das restrições para o problema em causa e, consequentemente, por se encontrarem as soluções que satisfaçam essas condicionantes. Os primeiros desenvolvimentos que conduziram à PR podem ser encontrados no domínio da inteligência artificial (IA) e remontam aos anos sessenta e setenta [4], [5], [6], [7]. Estes avanços tecnológicos focalizavam-se na representação e manipulação explícita de restrições em sistemas computacionais. No entanto, apenas nas décadas de 80 e 90 se desenvolveu uma crescente consciencialização de que estas ideias podiam fornecer a base para uma abordagem poderosa à temática da programação, modelação e resolução de problemas [8]. Por outro lado foram desenvolvidos esforços para explorar estes pressupostos e unificá-los numa única estrutura conceptual [9].

Actualmente, nota-se o aparecimento de duas correntes, pese embora o facto de serem complementares, para o tratamento da PR: A satisfação de restrições; A resolução de restrições. Ambas partilham a mesma terminologia, mas possuem origens distintas e socorrem-se de tecnologias diferentes de resolução de problemas. A satisfação de restrições lida com os problemas definidos sobre domínios de valores discretos, do qual fazem parte os domínios finitos, sendo actualmente a mais usada na maioria das aplicações. A resolução de restrições possui todas as propriedades base da PR. No entanto, neste caso as restrições são definidas (na sua maior parte) sobre domínios infinitos, ou mais complexos [32].

3.2.2 Programação Lógica e Programação com Restrições

A Programação Lógica corporiza um paradigma de programação baseado num subconjunto da lógica de primeira ordem. Os programas em lógica são de natureza declarativa, dado que indicam os relacionamentos lógicos necessários para resolver um dado problema. O sistema subjacente é responsável por efectuar os cálculos lógicos que permitem que a computação ocorra. Um sistema baseado em lógica usa um conjunto de regras fornecidas pelo programador (programa) para responder às perguntas que lhe são endereçadas. Em geral, os programas em lógica podem ser usados para comprovar uma dada afirmação, ou para determinar qual o conjunto de instanciações das variáveis que faz com que a uma dada pergunta seja atribuído o valor de verdade verdadeiro. A propriedade declarativa das linguagens de programação em lógica não só é apelativa para os programadores, como facilita a existência de uma semântica clara e bem definida, com uma base teórica bem fundamentada.

As restrições surgem como a extensão natural à estrutura da programação em lógica, ao partilharem muitas das propriedades acima discutidas. Em [9] mostra-se que o *Prolog* pode ser visto como um exemplo de um esquema muito particular de PLR. De facto, a Programação Lógica é baseada num paradigma computacional de natureza declarativa em que um programa é uma teoria lógica e a cada passo computacional apresenta uma solução para um sistema de equações de termos lógicos por intermédio do algoritmo de unificação. A sua natureza declarativa faz com que a Programação Lógica esteja próxima do princípio base da programação com restrições, em que se declara o que tem de ser satisfeito mas não como. Por outro lado, o processo de pesquisa de soluções em profundidade com retrocesso é em tudo similar aos procedimentos de retrocesso padrão usados para solucionar problemas com restrições. De qualquer modo, interessa fundamentalmente observar que as equações de termos lógicos são restrições de um tipo específico e que o algoritmo de unificação corporiza um tipo especial de procedimentos para a resolução deste tipo de restrições [32].

Em PLR a lógica é usada para especificar um conjunto de possibilidades para resolução do problema que são exploradas por intermédio de um simples método de pesquisa, enquanto que as restrições são usadas para minimizar o espaço de soluções pela eliminação em avanço de vias alternativas para a resolução do problema que no futuro se irão mostrar inconsequentes. O programador de aplicações pode declarar os factores que têm de ser tidos em conta em qualquer solução (as restrições), declarar as possibilidades (o programa em lógica) e usar o sistema para combinar o raciocínio e a pesquisa. As restrições são usadas para restringir e guiar a pesquisa [32].

3.2.3 Algumas Definições

As definições aqui apresentadas não sendo necessariamente definições com carácter universal, servem, contudo, para estabelecer conceitos relacionados com a PR em geral.

- Uma *interpretação* consiste na atribuição a cada variável presente numa restrição de um valor do seu domínio

  (e.g. $\theta = \{X \rightarrow 3, Y \rightarrow 4, Z \rightarrow 2\} \Rightarrow \theta\,(X + 2Y) = (3+2\times4)=11$);

- Diz-se que uma interpretação *satisfaz* uma restrição se a essa restrição se puder associar o valor de verdade verdadeiro, ou seja, a restrição é verdadeira nessa interpretação. Uma restrição pode ser satisfeita se existir pelo menos uma interpretação que a satisfaça (e.g. $X \leq 3 \land Y = X+1$). Pelo contrário, uma restrição não pode ser satisfeita se não existir pelo menos uma interpretação que a satisfaça (e.g. $X \leq 3 \land Y = X+1 \land Y \geq 6$);

- Uma *solução* para um problema é uma interpretação que satisfaz todas as restrições do problema (e.g. θ ($X \geq 3 \land Y = X + 1$) = ($3 \geq 3 \land 4 = 3 + 1$) = *verdadeiro)*;

- Duas restrições dizem-se *equivalentes* se estas tiverem o mesmo conjunto de soluções, ou seja, duas restrições podem representar a mesma solução

$$X > 0 \Rightarrow 0 < X$$

$$X = 1 \land Y = 2 \Rightarrow Y = 2 \land X = 1$$

$$X = Y + 1 \land Y \geq 2 \Rightarrow X = Y + 1 \land X \geq 3$$

- Um conjunto de restrições σ, ou armazém de restrições, *implica* uma restrição *r* se para cada interpretação em que todas as restrições σ são verdadeiras, *r* é também verdadeira;

- Um conjunto de restrições σ é consistente com uma restrição r se existe pelo menos uma interpretação em que σ é verdadeiro e r também é verdadeir**a**;

- σ *contradiz* uma restrição *r* se não existe nenhuma interpretação em que σ é verdadeiro e *r* é também verdadeira.

3.2.4 Domínios Computacionais da Programação com Restrições

As linguagens de Programação Lógica tradicionais possuem o seu domínio assente em termos (estruturas) não interpretados baseados em constantes e símbolos funcionais. A linguagem de programação Prolog é um exemplo destas linguagens sendo, ao mesmo tempo, a mais representativa em termos de utilização. O Prolog trata a equação $X - 3 = Y + 5$ considerando que o termo X-3 não igual ao termo Y+5, sendo incapaz de interpretar cada um dos termos. A PLR amplia a Programação Lógica de modo a oferecer um ou mais domínios interpretados de restrições. A PLR é parametrizada por um ou mais domínios de discurso, sobre os quais assenta a resolução das restrições. Muitas das linguagens de PLR baseiam-se em diversos domínios, destacando-se o domínio dos booleanos, dos finitos, e dos reais. Outros exemplos incluem listas, conjuntos finitos e árvores [32].

- **As restrições booleanas** são tratadas por meta-interpretadores de restrições especializados, podendo, no entanto, ser tratadas como um caso particular das restrições associadas a domínios finitos para esse problema. Neste último caso as variáveis apenas podem tomar dois valores inteiros: 0 (falso) ou 1 (verdadeiro);
- **As restrições sobre domínios finitos** são utilizadas em muitas áreas do conhecimento. Para satisfação destas restrições usa-se uma combinação de técnicas para a preservação de consistência, propagação de valores e pesquisa com retrocesso. Cada variável possui associado um conjunto finito de valores inteiros, ao qual é dado o nome de domínio da variável. Os valores do domínio que levam a inconsistências são removidos do domínio das variáveis durante a fase de propagação, enquanto que pela pesquisa se tenta instanciar cada variável do problema;
- **As restrições sobre intervalos reais** são o equivalente das consideradas para os domínios finitos, só que aqui trabalha-se com valores reais em vez de valores inteiros. As técnicas de remoção de inconsistências são similares às técnicas usadas para os domínios finitos, ou então são baseadas em técnicas matemáticas de diferenciação automática ou em séries de Taylor;
- **As restrições lineares** denotam restrições construídas a partir de variáveis cujo domínio é dado pelo conjunto dos números reais. Para este tipo de restrições têm sido implementados meta-interpretadores de restrições bastante eficientes que utilizam o algoritmo *Simplex* como ponto de partida.

3.2.5 Meta-Interpretadores de Restrições

Um dos aspectos que permite distinguir os diferentes meta-interpretadores de restrições relaciona-se com os domínios do conhecimento em que se desenvolvem os processos computacionais e que definem os objectos usados para expressar as restrições. Outro aspecto deste problema relaciona-se com o volume de restrições a propagar e no esforço computacional colocado na propagação. Este último aspecto permite classificar os meta-interpretadores de restrições em completos e incompletos, cujas características são expressas no texto que se segue.

Os meta-interpretadores de restrições, do tipo completo, podem, a cada passo computacional, esperar pela satisfação de um qualquer conjunto de restrições, o que é uma propriedade muito desejável. Infelizmente, a satisfação de um conjunto de restrições para um dado domínio pode ter um custo excessivo ou pode mesmo não se verificar. Esta característica faz com que os meta-interpretadores do tipo completo sejam quase só usados em domínios bem específicos, sendo as restrições dadas por expressões lineares estruturadas ou em meta-interpretadores específicos. Por outro lado, os meta-interpretadores de restrições do tipo incompleto podem ser definidos para os mais variados domínios do conhecimento, podendo, no entanto, diferir na quantidade de restrições e no esforço computacional dispensado à sua propagação [32].

3.3 Problemas de Satisfação de Restrições

A resolução de problemas em termos do paradigma da PLR com domínios finitos faz uso de uma combinação de técnicas de verificação de consistência e propagação de restrições, ao que se associa um algoritmo de pesquisa com retrocesso. Estas técnicas de consistência e propagação foram inicialmente desenvolvidas para a resolução dos designados Problemas de Satisfação de Restrições (PSR). Um PSR caracteriza-se por possuir um conjunto $X = \{x_1, x_2, \ldots, x_n\}$ de variáveis, cada uma com o seu domínio $D = \{d_1, d_2, \ldots, d_n\}$ de valores possíveis e um conjunto $R = \{r_1, r_2, \ldots, r_n\}$ de restrições que condicionam os valores que as variáveis podem tomar em simultâneo. Estas variáveis são normalmente designadas por variáveis de domínio e são representadas pelo par $<x_i, d_i>$, onde x_i é o símbolo que identifica a variável i e d_i é um conjunto finito designado por domínio de i. Por seu lado, cada restrição é dada pela extensão de um predicado $r(x_1, x_2, \ldots, x_k)$ em que os k argumentos denotam as variáveis de domínio envolvidas na restrição [32].

É uma prática comum nos PSR restringir a discussão a restrições binárias com domínios finitos. Pode ser demonstrado que qualquer restrição envolvendo n variáveis,

$n > 2$, pode sempre ser representada por um conjunto de restrições binárias [10]. Os domínios das variáveis são usualmente dados por restrições unárias. Um PSR com restrições binárias pode ser representado por um grafo em que cada nó é uma variável e cada restrição é um ramo que liga as variáveis envolvidas [6].

3.3.1 Técnicas de Consistência e Propagação de Restrições

A resolução de PSR pode ser efectuada de forma simples por via de uma exploração exaustiva do espaço de soluções. Este método corporiza a essência do algoritmo gerar e testar e passa, basicamente, por instanciar as variáveis do problema com valores do seu domínio, seguindo-se então o teste de verificação da consistência das restrições. Sendo um procedimento pouco "inteligente" e ineficiente, não é usado na prática. O algoritmo que implementa uma pesquisa com retrocesso é uma outra forma de efectuar uma busca sistemática [11]. Este opera de uma forma incremental, ao estender, passo a passo, uma solução parcial para uma global, até se obter a solução desejada, (i.e., uma solução parcial é obtida pela atribuição de um valor do domínio a uma variável ainda não instanciada, valor este consistente com os valores já atribuídos a outras variáveis da solução parcial corrente). Embora este método apresente melhor desempenho que o anterior, a complexidade do algoritmo que o corporiza aconselha a que os problemas a que se aplique não sejam de grandes dimensões.

Uma outra abordagem para atacar os PSR passa pela remoção dos valores dos domínios das variáveis de decisão que levam a inconsistências, até se obter uma solução. Estes métodos, conhecidos por técnicas de consistência, são determinísticos, ao contrário dos algoritmos que efectuam busca sistemática. As técnicas de verificação de consistência vão desde a simples consistência de nó, passando pela muito popular consistência de arco, até à mais complexa e computacionalmente pesada consistência de caminho [10], [12]. Estas técnicas são derivadas das teorias sobre grafos, pelo que é normal que um PSR seja muitas vezes transformado num PSR binário.

A técnica de verificação de consistência mais simples é conhecida como *consistência de nó*. O nó que representa a variável x no grafo de restrições é consistente se para todos os valores do domínio de x, todas as restrições unárias em x são satisfeitas. Se o domínio D da variável x contém um valor a que não satisfaz a restrição unária em x então a instanciação de x com a não ocorrerá. Consequentemente, as inconsistências de nó são eliminadas simplesmente pela remoção dos valores do domínio D de cada variável x que não satisfaçam as restrições unárias em x. Se o grafo de restrições é consistente de nó, então as restrições unárias podem ser eliminadas, dado que estão à partida satisfeitas.

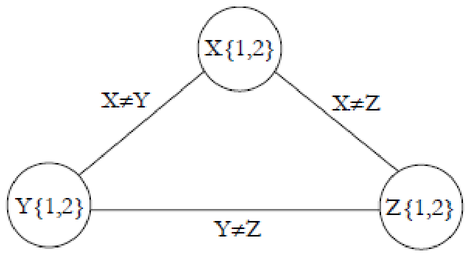

Figura 3.1: O grafo possui todos os arcos consistentes, mas não existe nenhuma solução que satisfaça todas as restrições

A *consistência de arco* denota a técnica de consistência de maior utilização. Um arco (x_i, x_j) é consistente se para todos os valores a do domínio de x_i existem valores b no domínio de x_j de forma a que $x_i=a$ e $x_j=b$ sejam permitidos pela restrição binária entre x_i e x_j. A consistência de arco é direccional, ou seja, o facto de o arco (x_i, x_j) ser consistente não significa que o arco (x_j, x_i) seja também consistente. Existem diversos algoritmos para o tratamento da consistência de arco. Os mais frequentemente utilizados são conhecidos por AC3 e AC4 [10]. Embora tanto as técnicas de consistência como os algoritmos de pesquisa sistemática possam ser usados separadamente para solucionar de forma completa um PSR, em situações práticas utiliza-se uma combinação das duas abordagens. Com a integração de algoritmos de pesquisa sistemática com técnicas de consistência, é possível obter algoritmos de satisfação de restrições mais eficientes. Deste modo o algoritmo de pesquisa com retrocesso é modificado pela introdução de técnicas de verificação de consistência baseadas na consistência de arco conhecidas por verificação para diante, ver à frente e ver atrás [1].

[1] Os termos usados em inglês são respectivamente *forward checking*, *look-ahead* e *look-back*.

A Figura 3-2 mostra quais as restrições que são verificadas quando estas técnicas de propagação são aplicadas.

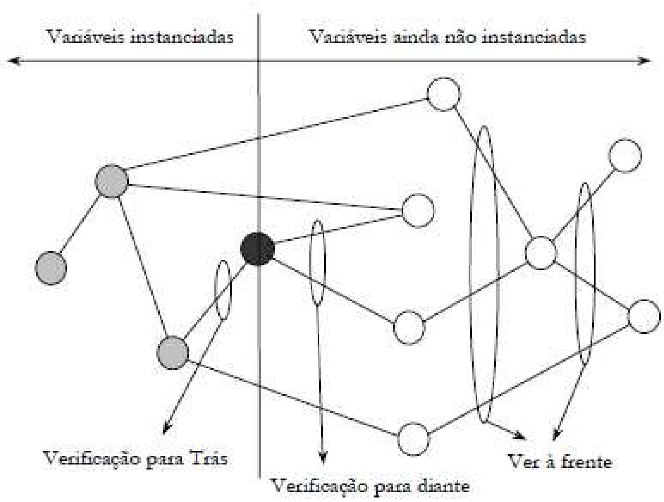

Figura 3.2: Comparação das técnicas de propagação

Melhorar a qualidade de propagação de restrições em cada nó resulta numa árvore de pesquisa que contém menos nós, embora à custa dum maior esforço computacional. No extremo, um elevado grau de verificação de consistência de arco e de caminho para um dado problema pode eliminar completamente a necessidade de pesquisa, mas costuma normalmente ser uma técnica muito mais pesada que o retrocesso simples. Deste modo, em situações reais, é frequente usar apenas a técnica de verificação para diante com o retrocesso simples [32].

3.3.2 Ordenação de Valores e Variáveis

A ordem como as variáveis e os valores são considerados para instanciação tem um impacto considerável no desempenho dos algoritmos de pesquisa na PR. A ordenação de variáveis e valores é determinada usualmente por métodos que seguem diversas heurísticas. Estes diferentes métodos de ordenação são agrupados em dois grandes grupos: *ordenação estática*, em que a ordem da variáveis (ou valores) é especificada antes da pesquisa começar, e não se altera durante o processo de pesquisa;

e *ordenação dinâmica*, em que a escolha da próxima variável (ou valor) depende em qualquer momento do estado corrente da pesquisa.

A ordenação dinâmica não pode ser usada por todos os algoritmos de pesquisa. O algoritmo de retrocesso simples constitui um exemplo em que esta situação se verifica, e tal deve-se ao facto de não existir informação extra durante a pesquisa que possa ser usada para efectuar uma escolha diferente face à ordem inicial. No entanto, com a verificação para diante, cada estado possui o domínio das variáveis tal como foram truncados pelo conjunto de instanciações nesse estado, e assim é possível basear a escolha da próxima variável nessa informação.

Ordenação de Variáveis

Diversas heurísticas têm sido desenvolvidas para a ordem de selecção de variáveis. A mais comum é baseada no princípio "falhar-primeiro"[2]. Este princípio pode ser melhor entendido considerando a seguinte expressão "para ter sucesso, tentar primeiro onde é mais provável falhar". Com isto pretende-se seleccionar em primeiro lugar a variável que possui o menor número de alternativas possíveis. Assim, a ordem de instanciação de variáveis é, em geral, diferente em diferentes ramificações da árvore de pesquisa, sendo determinada dinamicamente. Dado que não se pretende falhar na pesquisa de soluções, o princípio "falhar-primeiro" pode inicialmente parecer paradoxal. No entanto, o que se pretende é que, num dado momento, se a solução parcial não originar uma solução completa, então é melhor obter essa conclusão o mais cedo possível.

A Figura 7 mostra como a ordem de selecção de variáveis modifica a forma duma árvore de pesquisa. Se forem escolhidos em primeiro lugar valores para x_1 e só depois valores para x_2, então a árvore de pesquisa tem uma dada forma particular. Se pelo contrário forem escolhidos em primeiro lugar valores para x_2 e só depois para x_1, então obtém-se uma árvore de pesquisa com uma forma diferente. É possível, ainda, verificar se os domínios das variáveis possuem tamanhos diferentes, nesse caso a ordem de selecção de variáveis pode mudar o número de nós internos da árvore,

[2] Em inglês este principio é conhecido por *first-fail*.

embora não o número das folhas. Para minimizar o número de nós, as variáveis que possuem os menores domínios devem ser tentadas em primeiro lugar.

Por vezes, quando as variáveis possuem o mesmo número de valores no seu domínio, é usada uma outra heurística para desempatar. Esta baseia-se também no princípio de lidar em primeiro lugar com os casos mais difíceis ao tratar em primeiro lugar a variável que participa no maior número de restrições [32].

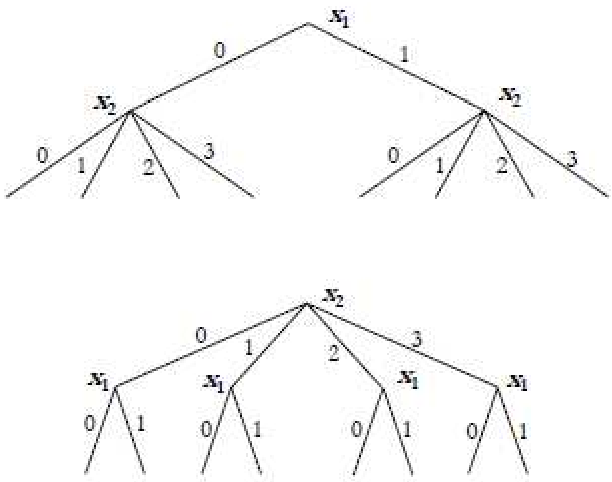

Figura 3.3: Efeito da ordem de selecção das variáveis

Ordenação de Valores

Após a selecção da variável segue-se a escolha de um valor do seu domínio actual para a instanciar. Também a ordem por que é feita esta escolha de valores pode ter um impacto substancial no tempo despendido para encontrar uma solução. Note-se, no entanto, que se todo o espaço de soluções tem de ser explorado então a ordenação de valores é indiferente.

O uso de uma diferente ordenação de valores provoca um rearranjo das ramificações que emanam a partir de cada nó da árvore de pesquisa. Este rearranjo constitui uma vantagem se for possível assegurar que uma dada ramificação permite encontrar uma solução mais rapidamente do que outra. No melhor caso, se o problema possui pelo menos uma solução, e se o valor correcto para cada variável é seleccionado, então a solução é encontrada sem necessidade de retrocesso.

A Figura 3.4 mostra como a ordenação de valores do domínio pode mudar a ordem de visita às folhas relativamente à Figura 3.3.

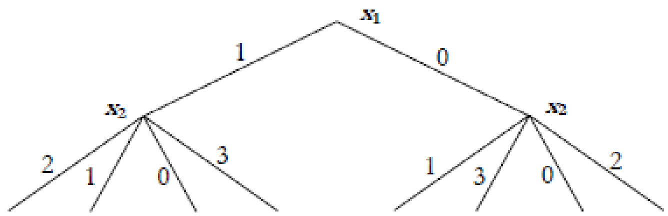

Figura 3.4: Efeito da ordem de selecção de valores

3.4 Programação Lógica por Restrições com Domínios Finitos

Os meta-interpretadores de PLR com domínios finitos pertencem à categoria dos meta-interpretadores de restrições incompletos e, deste modo, a enumeração das restrições não é normalmente suficiente para solucionar um problema. Um meta-interpretador de restrições deste tipo necessita de ser combinado com procedimentos que atribuam às variáveis de decisão valores admissíveis. A escolha de um bom procedimento deste tipo possui um enorme impacto no desempenho de um dado programa.

Nesta secção e nas seguintes usar-se-á a abreviatura PLR(*DF*) para referir genericamente a PLR com domínios finitos. Para além de se descrever as características e as facilidades que se podem encontrar nos meta-interpretadores de PLR(*DF*), esta secção também define parcialmente uma linguagem formal para a PLR(*DF*) [32].

3.4.1 Domínio Finitos

Tal como para as técnicas de verificação de consistência, um dos elementos básicos dos meta-interpretadores de PLR(*DF*) são as variáveis de domínio. Estas têm associado um conjunto finito de valores, normalmente numéricos, designado por domínio finito. Este conjunto finito é definido como um subconjunto dos números inteiros. A definição de uma variável de domínio recorre a uma restrição particular usualmente designada por restrição de domínio. Esta consiste numa expressão na forma $x \in D_x$ em que x é a variável e D_x é o seu domínio. A Tabela 9 mostra três formas possíveis para a especificação de domínios finitos.

Intervalo	$a .. b$
Lista de intervalos	$\{a .. b, c .. d, ...\}$
Lista de valores e intervalos	$\{a, b, c .. d, e, ...\}$

Tabela 3.1: Diferentes formas de representação de domínios finitos.

3.4.2 Restrições

Para além da restrição de domínio, os meta-interpretadores PLR(*DF*) fornecem, em geral, diversos tipos de restrições. Estas podem agrupar-se em duas classes: as aritméticas e as simbólicas. Relativamente à classe das restrições simbólicas, encontram-se ainda as restrições baseadas em métodos sintácticos e as restrições baseadas em métodos semânticos. As primeiras são geralmente independentes do domínio enquanto que as outras não.

Restrições Aritméticas

As restrições aritméticas usam normalmente, e dependendo da implementação, métodos de propagação semelhantes ao *ver à frente* parcial. Em geral, estas restrições tratam termos lineares sob domínios finitos. Estes termos lineares possuem normalmente a forma $t = a_0 + a_1 x_1 + a_2 x_2 + ... + a_n x_n$.

Restrições Aritméticas Básicas

As restrições aritméticas básicas surgem sob forma de equações ($t_1 = t_2$), de inequações ($t_1 < t_2$, $t_1 > t_2$, $t_1 \leq t_2$ ou $t_1 \geq t_2$) ou sob a forma de desigualdades ($t_1 \neq t_2$). Os termos t_1 e t_2 são termos lineares de variáveis de domínio finito. Certos meta-interpretadores de PLR(*DF*) permitem que as restrições possuam termos não lineares de um tipo particular. Estes termos não lineares apenas consideram produtos entre duas variáveis e têm a forma $t = a_0 + a_1\, x_1\, y_1 + a_2\, x_2\, x_2 + \ldots + a_n\, x_n\, y_n$. No entanto, o produto entre três ou mais variáveis pode sempre ser convertido em produtos entre duas variáveis recorrendo a variáveis auxiliares. Por exemplo, o produto $t = xyz$ pode ser substituído por $xa = t \land a = yz$.

Conectivas Lógicas

Num linguagem de PLR(*DF*) típica, é usual encontrar um conjunto de conectivas lógicas que permitem combinar restrições aritméticas básicas de modo a especificar relações mais complexas. A Tabela 3-2 mostra um conjunto de cinco conectivas lógicas. De referir que r, r_1 e r_2 são normalmente restrições aritméticas básicas.

Conjunção	$r_1 \land r_2$
Disjunção	$r_1 \lor r_2$
Implicação	$r_1 \Rightarrow r_2$
Equivalência	$r_1 \Leftrightarrow r_2$
Negação	$\neg r$

Tabela 3.2: As diferentes conectivas lógicas que permitem formar restrições compostas.

Cardinalidade

O raciocínio do operador de cardinalidade tem mostrado ser capaz de substituir as conectivas lógicas, constituindo uma primitiva bastante poderosa [13]. A forma básica deste operador é $\#(l, L, u)$, onde $L=[r_1, \ldots, r_k]$ é uma lista de restrições, l é limite inferior do número de restrições de L que têm de ser satisfeitas, e u é o limite superior do número de restrições de L que têm de ser satisfeitas.

Quando o operador não está disponível, é possível defini-lo considerando a conjunção de restrições (3-1) onde $\land$ representa conjunção de todos *i*.

I

$$l \leq \sum_i b_i \leq u \wedge \bigwedge_i (r_i \Leftrightarrow b_i = 1) \wedge \bigwedge_i (\neg r_i \Leftrightarrow b_i = 0) \qquad (3\text{-}1)$$

Por outro lado, se as conectivas lógicas de conjunção não estão disponíveis, o operador de cardinalidade pode ser usado para as substituir como se mostra na Tabela 3-3.

$r_1 \wedge \ldots \wedge r_n$	$\#(n, [r_1, \ldots, r_n], n)$
$r_1 \vee \ldots \vee r_n$	$\#(1, [r_1, \ldots, r_n], n)$
$\neg r$	$\#(0, r, 0)$

Tabela 3.3: Substituição das conectivas lógicas com o operador de cardinalidade.

Restrições Simbólicas

No que refere às restrições simbólicas, estas são úteis para expressar condições não aritméticas entre conjuntos de variáveis de domínio. São aqui descritas algumas das restrições simbólicas baseadas em métodos sintácticos e semânticos que registam uma utilização mais ampla no desenvolvimento de aplicações usando a PLR(*DF*) [32].

Uma das restrições simbólicas baseadas em métodos sintácticos mais usada é a restrição *element*/3. Esta permite expressar uma dependência funcional entre duas variáveis. A definição desta restrição [14] especifica que o predicado *element*(i, L, v), em que L=[x_1, …, x_k], é verdadeiro se (3-2) se verifica, ou seja, se x_i é igual a v. Considera-se que todos os x_j ($1 \leq j \leq k$) representam os valores inteiros e que o raciocínio desta restrição é bidireccional.

$$\bigvee_{j=1}^{k} (i = j \wedge x_j = v) \qquad (3\text{-}2)$$

A outra restrição muito frequente é a *all_different*/1. Esta especifica que todos os valores das variáveis de domínio têm de ser diferentes. Uma definição para esta restrição especifica que *all_different*([x_1, …, x_k]) é verdadeiro se 3-3) se verifica, ou seja, se todas as variáveis de domínio x_i possuem valores diferentes.

$$\bigwedge_{i=1}^{k-1} \bigwedge_{j=i+1}^{k} x_i \neq x_j \qquad 3\text{-}3)$$

Esta definição considera a colocação de restrições de desigualdade para todos os pares possíveis de variáveis de domínio. No entanto, é possível inferir mais informação de consistência, para além daquela que pode ser inferida por cada restrição de desigualdade. Por exemplo, é possível deduzir que a restrição não se verifica se o tamanho do domínio de todas as variáveis não instanciadas é inferior ao número de variáveis não instanciadas. Ao efectuar esta verificação, é possível detectar inconsistências que as restrições de desigualdade por si só não detectariam [32].

Outras restrições baseadas em métodos sintácticos podem ser encontradas nos meta-interpretadores de PLR(*DF*), como por exemplo, *outof*, *atmost*, *atleast*, *relation*. Este tipo de restrições proporcionam uma propagação de restrições de menor qualidade quando comparadas com as restrições baseadas em métodos semânticos. No entanto, são normalmente independentes do domínio do problema, o que as torna de uso geral.

As restrições baseadas em métodos semânticos usam o conhecimento específico do domínio do problema para obter melhores resultados de propagação. Este tipo de restrições são conhecidas por restrições globais [15], [16] e combinam algumas propriedades importantes:

- Modelam condições complexas baseadas em conjuntos de variáveis;
- As condições podem ser usadas em diferentes contextos;
- O raciocínio sobre as restrições detecta inconsistências num maior número de situações e reduz significativamente o espaço pesquisa;
- Podem ser aplicadas a diversos problemas de grande dimensão.

Podendo ser encontrada em muitos meta-interpretadores de PLR(*DF*), a restrição *comulative*/4 é, porventura, a mais conhecida desta classe de restrições e foi desenvolvida especialmente para solucionar problemas de escalonamento [15]. Esta incorpora um conjunto eficiente de algoritmos desenvolvidos para solucionar problemas de escalonamento. Em geral, nos problemas de escalonamento pretende-se escalonar a partir do instante inicial (s_i) um conjunto de operações de diferentes durações (d_i) que consomem uma dada quantidade de recursos (r_i). Em cada instante o total de recursos consumidos não deve exceder o limite disponível (l). Formalmente, esta restrição representada pelo predicado *cumulative*($[s_1, \dots, s_k], [d_1, \dots, d_k], [r_1, \dots, r_k]$ l), é verdadeira quando o consumo acumulado em cada instante de tempo i não excede o limite total de recursos, ou seja, quando as condições (3-4) e (3-5) se verificam.

$$\sum_{j=1}^{n} b_j r_j \le l \qquad (3\text{-}4)$$

$$(b_j = 1 \Leftrightarrow s_j \le i \le s_j + d_j\text{-}1) \wedge (b_i = 0 \Leftrightarrow s_j > i \vee i > s_j + d_j\text{-}1), \quad \forall I \qquad (3\text{-}5)$$

Sistemas do Tipo Caixa Preta e do Tipo Caixa Transparente

Um meta-interpretador de PLR(*DF*) que permite o desenvolvimento de aplicações usando restrições como as descritas na secção anterior é classificado como sistema do tipo caixa preta.

Com estes sistemas o programador tem pouco, ou mesmo nenhum, controlo na forma como a propagação do conjunto de restrições do problema é feita. Ele deve apenas especificar as relações do problema por intermédio das restrições primitivas embebida no meta-interpretador. Embora sistemas deste tipo sejam eficientes, uma vez que o meta-interpretador pode ser escrito a um nível baixo, esta abordagem apresenta algumas desvantagens, como por exemplo, o facto de ser difícil de modificar ou construir um novo meta-interpretador para um novo domínio, tornando a depuração mais complexa.

Em oposição aos sistemas do tipo caixa preta, surgiram os sistemas do tipo caixa transparente. Estes sistemas seguem uma abordagem que permite um controlo da propagação de restrições a um nível mais fino. Os meta-interpretadores que seguem esta abordagem possuem mecanismos que permitem ao programador de aplicações implementar restrições especiais mais adaptadas à estrutura do problema e ao mesmo tempo melhorar a qualidade da propagação de restrições. Estas restrições são usualmente designadas por restrições definidas pelo utilizador [17].

Diversas propostas têm sido feitas para proporcionar mecanismos que permitem oferecer uma maior flexibilidade e adaptabilidade dos meta-interpretadores de restrições.

Estas propostas não se limitam aos domínios finitos, e em alguns casos permitem a modificação completa dos meta-interpretadores de restrições pelos programadores de aplicações. Algumas destas propostas são as seguintes:

- Serviços activados por eventos[3] que permitem definir a propagação de restrições de uma forma limitada, por exemplo, o CHIP [13], [14];
- Combinação ou agrupamento de restrições, cc(*FD*) [13], que permite a construção de restrições mais complexas a partir de restrições mais simples;
- Restrições ligadas a variáveis booleanas que permitem expressar qualquer fórmula lógica com restrições primitivas, como, por exemplo, o BNR-Prolog [18], ou as "restrições encadeadas" [19];
- Indexantes que permitem a implementação de restrições de domínios finitos num nível de abstracção médio, sendo o caso do clp(*FD*) [20];
- Meta variáveis e variáveis com atributos que permitem ligar restrições a variáveis [21];
- Regras sentinela[4] que definem condições lógicas de compromisso e permitem que as restrições sejam substituídas por restrições mais simples até o problema ser solucionado [22]. Esta técnica é seguida pelo sistema CHR [23].

---

[3] O termo original para estes serviços é *"demon constructs"*.

[4] Originalmente estas regras são designadas por *Guarded Rules*.

3.4.3 Pesquisa e Optimização

Considerando que os meta-interpretadores de PLR(*DF*) são incompletos na resolução da generalidade dos problemas, o mecanismo de propagação por si só não é capaz de solucionar a grande maioria dos problemas. Para solucionar completamente um dado problema é necessário recorrer a uma estratégia de pesquisa que permita obter uma ou mais soluções para o problema. Geralmente, o programador deve implementar a sua própria estratégia de pesquisa de modo a explorar o espaço de soluções, embora os meta-interpretadores de PLR(*DF*) já possuam algumas facilidades para a pesquisa, que em termos gerais permitem bons resultados. Estas facilidades consideram essencialmente as estratégias de ordenação de variáveis e de valores já discutidas na secção 3.3.2. Estas estratégias de ordenação de variáveis e valores condicionam a forma da árvore de pesquisa. Em geral, os nós da árvore de pesquisa representam escolhas, sendo estas mutuamente exclusivas, originando a divisão do espaço de soluções em dois ou mais sub-espaços disjuntos [32].

Basicamente, para os problemas formulados usando variáveis de decisão do tipo dos domínios finitos, a exploração da árvore de pesquisa é feita pela escolha, em cada nó, de uma variável e de um valor do seu domínio para a instanciar. Esta técnica é conhecida por instanciação de variáveis e valores[5]. A quantidade de escolhas possíveis de valores é igual ao tamanho do domínio da variável, significando que, sendo *n* o tamanho do domínio da variável escolhida, então do nó correspondente partem no máximo *n* ramos. No entanto, a escolha de valores não envolvem necessariamente a escolha de um valor concreto para a variável. É possível efectuar escolhas disjuntas pela partição do domínio em dois ao mais subdomínios disjuntos, ou então pela escolha de um valor num ramo e pela sua exclusão no outro [32].

[5] Esta designação surge do termo original em língua inglesa *labelling*.

Muitos problemas reais são caracterizados por possuírem mais do que uma solução, sendo até frequente haver muitas soluções. Nestas situações, pretende-se fundamentalmente efectuar a escolha da melhor solução. Esta melhor solução é aquela que satisfaz todas as restrições e que minimiza uma dada função de custo. Na PLR(*DF*) o algoritmo de optimização mais amplamente usado é o *B&B*[6] [24]. Este algoritmo encaixa bem na resolução de restrições devido à sua abordagem incremental e a maior parte dos meta-interpretadores de PLR(*DF*) possuem já uma implementação deste algoritmo. Estas implementações, nas suas formas mais simples, trabalham em conjunto com a estratégia de pesquisa, colocando sempre uma restrição adicional que impõe a pesquisa de uma nova solução com um custo melhor do que o custo da melhor solução até então encontrada [32].

O algoritmo *B&B* é completo no sentido em que o espaço de soluções é explorado de forma a que garantidamente se possa encontrar a melhor solução. No entanto, para a maior parte dos problemas reais é computacionalmente impraticável explorar o espaço de soluções completo e, como tal, não se consegue provar que a melhor solução encontrada até um dado momento é a óptima. Felizmente, nem sempre é necessário obter a solução óptima, bastando apenas uma boa solução. Neste caso a exploração incompleta do espaço de soluções pode ser suficiente. Alguns métodos para efectuar a exploração incompleta podem ser os seguintes:

- Pesquisa com um período de tempo limitado em que o algoritmo de optimização termina ao fim de um dado intervalo temporal e retorna a melhor solução encontrada até então;
- A pesquisa com retrocesso limitado é outro método que faz terminar o algoritmo de optimização quando se atinge um dado volume de retrocessos;

---

[6] Algoritmo *Branch and Bound.*

- A pesquisa com crédito é um método de pesquisa que limita o número de escolhas não determinísticas. A pesquisa inicia com um valor de crédito na raiz da árvore de pesquisa. O crédito atribuído a cada nó é repartido por cada nó filho. Apenas são explorados os trajectos das sub-árvores que obtêm pelo menos um crédito. Considera-se que uma unidade de crédito é indivisível;

- Pesquisa limitada por discrepância [25] é um método que assume a existência de uma boa heurística para guiar a pesquisa. A "discrepância" é uma medida do grau em que o método falha em seguir a heurística. O método começa com um valor de discrepância 0, e sempre que este falha na busca de uma solução dentro do dado valor de discrepância, este valor é aumentado e a pesquisa recomeça.

4

Resolução do Problema do EPEE utilizando a PLR através do ECLiPSe

4.1 Introdução

Após o estudo do problema do EPEE e de toda a envolvente que o caracteriza, apresenta-se neste capítulo a implementação do modelo matemático descrito no Capítulo 2, através da Programação Lógica por Restrições (PLR), utilizando o software ECLiPSe. Para chegar a este ponto foi necessário estudar e compreender todos os conceitos da linguagem da Programação Lógica por Restrições, assim como dominar todas as ferramentas que o ECLiPSe [30] disponibiliza para filtrar e optimizar soluções em problemas de escalonamento como o tratado neste trabalho.

O ECLiPSe é um sistema *freeware* de programação lógica por restrições sucessor do programa CHIP. É um sistema baseado em Prolog cujo propósito é servir de plataforma para integrar várias extensões de Programação Lógica, em particular a PLR. A maior parte das extensões do ECLiPSe baseiam-se numa estrutura de dados especial denominada por *Metaterm*. São várias as bibliotecas disponibilizadas por este sistema, nomeadamente a de Intervalos de Restrições (utilizada neste trabalho), Propagação Generalizada, Eplex, ect. A escolha deste sistema ficou a dever-se, por um lado, à sua arquitectura aberta que o torna adequado para o desenvolvimento de aplicações, e por outro, a restrições financeiras. Este sistema dispõe também de um interface para o *tcl/tk*, o que permite o desenvolvimento das interfaces gráficas [31].

A modelização do problema utilizada foi a já mencionada na **Secção 2.3**, traduzida obviamente para linguagem lógica por restrições. Para isso foi necessária uma compreensão do conceito matemático para que a tradução em linguagem lógica fosse totalmente fidedigna à modelização dos conceitos do problema, e das restrições que definem as relações entre as variáveis. Finalmente, foram desenvolvidos procedimentos de selecção de variáveis e enumeração que, conjuntamente com os mecanismos de pesquisa pré-definidos, constituem o métodos de resolução do problema.

4.2 O Modelo de Programação Implementado

O modelo de programação implementado nesta dissertação segundo a PLR aplicada ao ECLiPSe, passa por 5 etapas:

- Levantamento de variáveis e respectivos domínios
- Implementação das Restrições definidas
- Aplicação da função de optimização
- Elaboração do método de pesquisa e de optimização

De seguida é apresentado e explicado cada uma das 4 etapas do programa de resolução do problema de EPEE.

4.2.1 Levantamento de variáveis e domínios

A primeira tarefa a realizar em qualquer programa informático é o levantamento dos *inputs* do problema a resolver.

Neste paradigma os dados do problema são representados como variáveis de domínio. Uma variável de domínio possui um domínio que se pode apresentar como a correspondência ao conjunto finito de todos os valores com que a variável pode ser instanciada. Os valores são os elementos do domínio. Se X for uma variável, D_x representa o seu domínio. O tamanho do domínio D_x corresponde ao número de valores que o constitui e é representado por T_d.

Podemos considerar 3 tipos de variáveis de domínio, conforme o tipo de elemento do seu domínio. Se o domínio é constituído por números, então a variável diz-se *Variável Numérica*. Estes números podem ainda ser restringidos ao conjunto dos inteiros, naturais ou não, ao conjunto dos racionais, ou dos reais. Se o domínio é constituído por elementos booleanos, então a variável diz-se *Variável Booleana*. Uma variável booleana pode ser vista como um caso particular da variável numérica em que o domínio é o conjunto {0,1}. Nesta dissertação o estado de cada gerador de energia será uma variável booleana, em que se irá representar o estado de Desligado/{0} ou de Ligado/{1}. Finalmente se o domínio é constituído por objectos não numéricos, a variável diz-se *Variável Simbólica*. Em PLR sobre domínios finitos os elementos podem ser representados por números naturais.

A existência de uma variável pressupõe que, em alguma fase da execução do programa esta tomará um valor do seu domínio. Este processo é definido como instanciação de variável e será apresentado posteriormente.

Devido à necessidade de trabalhar com todas as variáveis do problema quase sempre ao mesmo tempo, e como forma de organizar os vários tipos de variáveis diferentes, optou-se por criar listas de listas (matrizes) tanto para as variáveis dos Estados, como para as variáveis das Potências dos geradores. Estas listas de listas são necessárias, devido aos vários grupos geradores existentes que funcionarão nos 24 períodos correspondentes às horas de um dia. Desta forma, o número de índices de cada lista irá corresponder ao número de geradores que irão constituir o nosso sistema, assim como o número de listas será definido pela quantidade de períodos de tempo que o utilizador estabeleça.

4.2.2 Implementação das Restrições

Informalmente o Problema de Satisfação de Restrições consiste em determinar para cada uma das variáveis do problema, um valor do seu domínio, que satisfaça o conjunto finito das restrições do problema. Basicamente, uma restrição define os valores permitidos para um determinado subconjunto das variáveis. Uma restrição pode ser dada explicitamente, por enumeração dos tuplos válidos, ou implicitamente, por exemplo, por uma expressão algébrica.

A implementação das restrições vistas no Capítulo 2 terá que ser transversal a todos os elementos variáveis do problema para que se possa iniciar o processo de Satisfação das Restrições. Para que essa transversalidade ocorra foi criado um ciclo que percorre todos os elementos, tanto da lista de listas dos Estados, como do das Potências dos geradores em todos os períodos. Neste ciclo, para cada um dos períodos são impostas as restrições técnicas de forma a garantir a sua correcta propagação. Ainda neste capítulo da implementação de restrições, especial destaque para as restrições do tempo mínimo de arranque e de paragem, que devido ao seu carácter intransigente são suspensas e "acordam" assim que qualquer elemento da matriz dos

Estados dos geradores é restringido. Desta forma é possível garantir a consistência das restrições e e consequentemente garantir que todas as soluções apresentadas são soluções válidas.

De seguida explica-se o significado de cada restrição implementada no programa, significado esse que ajudou a compreender o problema do EPEE e facilitou a sua integração na PLR.

4.2.2.1 O significado das Restrições

- **Equilíbrio de Cargas**

$$\sum_{j=1}^{T} (U_{ij} P_{ij}) = D_j$$

Esta restrição obriga a que os grupos geradores durante todos os períodos de funcionamento satisfaçam sempre a procura existente de energia. Esta restrição salvaguarda o constante fornecimento de energia eléctrica aos consumidores.

- **Limites de produção dos geradores**

$$P_{imin} \leq P_{ij} \leq P_{imax}$$

A restrição dos limites de produção dos geradores, é uma restrição técnica que impõe os limites de bom funcionamento das máquinas. Cada gerador tem uma potência mínima e máxima e, em caso de estar a funcionar, poderá fornecer à rede energia unicamente dentro desses parâmetros.

- **Reserva Girante**

$$D_j + P_{srj} \leq \sum_{j=1}^{N} (U_{ij} P_{imax})$$

$$\sum_{i=1}^{N} (U_{ij} P_{imin}) \leq D_j$$

Para além de existir a restrição do Equilíbrio de Cargas no sentido de garantir a continuidade de serviço, a restrição da Reserva Girante garante que mesmo em caso de falha de algum grupo gerador por alguma anomalia a procura de energia seja sempre satisfeita.

- **Tempo Mínimo de Arranque**

$$U_{ij} = 1 \text{ para } \sum_{j=j_{si}}^{j-1} U_{ij} < MUT_i$$

Esta restrição garante o tempo mínimo de funcionamento do gerador desde o seu arranque, ou seja, desde o momento em que o seu estado seja de ligado/{1} deverá ficar a funcionar até ao MUT(tempo mínimo de funcionamento).

- **Tempo Mínimo de Paragem**

$$U_{ij} = 0 \text{ para } \sum_{j=j_{Di}}^{j-1} (1 - U_{ij}) < MDT_i$$

A restrição do Tempo Mínimo de Paragem é exactamente inversa à do Tempo Mínimo de Arranque, quer dizer que quando o gerador passar para o seu estado de desligado/{0} deverá permanecer parado durante o MDT (tempo mínimo de paragem).

4.2.3 Aplicação da Função de Optimização

O objectivo principal do problema do EPEE é encontrar um plano de escalonamento dos grupos geradores que satisfaça a procura de energia, sem violar as restrições. A medida principal de avaliação de desempenho do problema é o custo de produção da energia que obviamente depende, no caso dos grupos térmicos, da quantidade de combustível consumido. Isto quer dizer que quanto melhor a optimização realizada na função objectivo, menor custo teremos na produção da mesma quantidade de energia e, à partida, menos combustível será utilizado.

Tal como visto no ponto 2.3 desta dissertação a Função Objectivo é a seguinte:

$$C(U) = \sum_{i=1}^{N} \sum_{j=1}^{T} \left(U_{ij}\, FC_i(P_{ij}) + SC_i\, U_{ij}\,(1 - U_{ij-1}) + SD_i\, U_{ij-1}\,(1 - U_{ij}) \right)$$

De forma a utilizarmos esta função como função de optimização da nossa aplicação é necessário atribuir todos os elementos variáveis e constantes, conforme está demonstrado na equação, a uma variável *C (U)*. Esta variável de optimização irá corresponder ao custo total de funcionamento dos *N* geradores e dos *T* períodos, estando estes períodos definidos como sendo as 24 horas que cada dia contém.

Esta função divide-se em 3 somas que em ECLiSPe também terão que ser tratadas separadamente. Na realidade a Função de Optimização *C(U)* irá corresponder ao somatório destas 3 somas afectas a cada um dos geradores, em todos os períodos de tempo.

A primeira parte destas 3 somas corresponde à função do custo do combustível, apresentada também no Capítulo 2:

$$FC_i(P_{ij}) = A_i + B_i\, P_{ij} + C_i\, P_{ij}^2 \text{ , para } i = 1, \ldots , N$$

Os elementos Ai , Bi , Ci desta função irão ser constantes para cada gerador ao longo dos períodos, sendo que irão variar de gerador para gerador de acordo com o consumo de cada um. A função $FC_i\ (P_{ij})$ irá então variar consoante o gerador *j* em questão e irá variar exponencialmente conforme a Potência P_{ij}, que no período *i* estiver a fornecer à rede.

Com a multiplicação da função do custo do combustível pelo estado do gerador U_{ij}, garantimos que só serão contabilizados os geradores que estiverem a produzir energia, uma vez que os estados variam entre o valor {0}, quando não está em funcionamento, e {1} quando está em funcionamento.

A segunda parte da Função de Custo Final está relacionada com o custo do arranque dos geradores. Em cada gerador, ao longo de todos os períodos, existe a possibilidade de arranque e início de funcionamento, a menos que já esteja a funcionar. Para isso temos na segunda soma da Função de Optimização a expressão $(1 - U_{ij-1})$ que faz com que, caso no período anterior (U_{ij-1}) o gerador já esteja em funcionamento (estado {1}), o resultado desta parcela seja zero visto não ter nenhum *Start Up Cost* (custo de arranque) associado.

A multiplicação pelo estado do gerador mantém-se como forma de contabilizar apenas os custos dos geradores em funcionamento.

A última parcela da Função de Optimização está associada ao custo de paragem dos geradores em funcionamento e funciona exactamente da forma inversa à segunda parte da função. Ao invés de termos um *Start Up Cost*, iremos ter um *Shut Down Cost* (custo de paragem), que apenas irá ser contabilizado se o gerador tiver estado em funcionamento no período anterior e no período actual parar. Esta premissa é controlada pela expressão U_{ij-1} (1 - U_{ij}).

4.2.4 Pesquisa e Optimização

Em geral um problema de restrições consiste num conjunto de variáveis e restrições. As constantes são predicados que expressam propriedades de variáveis, tal como relações entre variáveis. Uma solução de um problema de restrições é uma atribuição de valores às variáveis de maneira que todas as restrições sejam satisfeitas.

No fim dos anos 70 as restrições começaram por serem integradas em ferramentas e linguagens de programação. Ao mesmo tempo, estavam a ser feitos esforços no sentido de fazer a programação lógica (*logic programming*, *LP*), mais declarativa (com uma estratégia de selecção flexível), mais rápida (com pesquisa melhoradas) e mais geral (com uma igualdade mais expansiva).

A estratégia de selecção flexível significa que o manuseamento de certos elementos pode ser atrasado, e consequentemente afastada a ordem fixa da esquerda para a direita. A pesquisa melhorada significa a detecção de falhas de derivações mais cedo. Igualdades mais expansivas significa considerar símbolos de funções interpretados. Por exemplo, interpretando o símbolo + como uma adição, o termo $2+1$ será equivalente a 3. Consequentemente esta sintaxe seria generalizada numa equação que seria solucionada.

Estas equações poderão ser interpretadas como restrições. Estas restrições são manuseadas por um algoritmo especial num *solver* de restrições. Elas podem ser atrasadas até haver informação suficiente na forma de outras restrições de maneira a poder serem solucionadas. Se as restrições se tornam inconsistentes, então a derivação actual falha. Por exemplo, o problema de restrições $X - Y = 3 \wedge X + Y = 7$ irá levar-nos à solução $X = 5 \wedge Y = 2$. A restrição objectivo $X < Y \wedge Y < X$ falha logo antes de se atribuir valores às variáveis.

Obviamente que a maneira mais fácil de resolver os problemas seria gerar uma possível solução, ou seja tentar todos os valores possíveis para todas as variáveis e então testar se as restrições são satisfeitas. Este é o método que é usado em *LP* (*generate-and-test*) e que infelizmente é impraticável na maioria dos casos por este método só usar as restrições para testar os valores já aplicados às variáveis em vez de as usar para impor valores nas variáveis de modo a conduzir a uma solução. No chamado método *constraint-and-generate*, primeiro as restrições são aplicadas para reduzir o espaço de pesquisa (o número possível de soluções) e então, se necessário, uma solução é gerada. Por exemplo, a solução do seguinte problema $X \in \{1,2\} \wedge Y \in \{1,2\} \wedge Z \in \{1,2\} \wedge X = Y \wedge X \neq Z \wedge Y < Z$ é encontrada depois de sete escolhas de soluções possíveis usando o método *generate-and-test* ao passo que com o método *constraint-and-generate* chegamos a uma solução sem ter que atribuir um único valor às variáveis. Basta pegar na restrição $Y < Z$ que determina logo que $Y = 2$ e $Z = 1$. A restrição $X = Y$ propaga a informação de que $X = 2$ e a restrição $X \neq Y$ continua satisfeita.

Desta forma chegamos ao ponto mais crítico do problema do EPEE que é de facto o problema de optimização, isto é, não interessa apenas uma solução ou todas as soluções, mas a melhor solução. Felizmente, existe um método geral para encontrar a solução óptima baseada na possibilidade de explorar todas as soluções.

No entanto, a dimensão do problema pode tornar a resolução deste impraticável em tempo aceitável. Este método geral é o algoritmo *Branch* & *Bound* e, de uma forma simplificada, funciona da seguinte forma com os meta-interpretadores de PLR:

1. Procurar a primeira solução e determinar o seu custo;

2. Adicionar uma restrição adicional que impõe que a próxima solução a procurar possua um custo inferior ao custo da melhor solução que foi encontrada até ao momento;

3. Procurar uma solução que satisfaça também esta nova restrição. Se esta existe, então foi encontrada uma nova solução que é melhor que a anterior. Volta-se então ao passo 2 até não ser possível encontrar uma nova solução;

4. Não sendo possível encontrar uma nova solução prova-se então que a última solução encontrada é a óptima.

Na próxima secção, este algoritmo *B&B* será usado como algoritmo de optimização para solucionar os exemplos práticos. É de referir ainda que, em termos de optimização, alguns dos principais factores que têm uma grande influência no desempenho de uma aplicação baseada nos meta-interpretadores de PLR relacionam-se com a:

- Estrutura do problema e modelo adoptado;
- Qualidade da propagação das restrições;
- Escolha de um bom procedimento de instanciação de variáveis e valores (labeling).

Em relação ao método de labeling utilizado nesta dissertação optou-se por instanciar primeiro as variáveis dos Estados dos geradores, uma vez que a primeira definição é saber se o gerador irá estar a funcionar ou não. Após esta definição instanciamos as variáveis das Potências dos geradores, que de acordo com as soluções encontradas na instanciação dos Estados dos geradores irão ser optimizadas ou não, caso o seu estado seja de funcionamento ou de paragem.

O critério escolhido para a instanciação das variáveis foi o de seguir a ordem de introdução dos dados do utilizador, percorrendo de seguida todos os elementos variáveis da matriz dos Estados e das Potências da esquerda para a direita e de cima para baixo. O algoritmo do B&B foi inserido englobando todo o método de instanciação de variáveis e valores, tendo como função objectivo de optimização a função Custo Total da Produção. Desta forma iremos obter para todos os geradores e em todos os períodos de tempo a melhor solução económica, sem comprometer as questões técnicas, já aplicadas quando efectuamos a propagação das restrições.

5

Resolução de Problemas Práticos

5.1 Introdução

Para o propósito de avaliação dos algoritmos e do sistema desenvolvido no âmbito deste trabalho serão usados dois casos práticos baseados em [1]. O primeiro caso corresponde a um problema de EPEE com 4 unidades produtoras e 8 horas de período de estudo.

O segundo caso é a simulação do funcionamento de 38 unidades térmicas da Taipower (Empresa de Electricidade de Taiwan) durante um período de 24 horas, onde as unidades que têm funcionamento obrigatório (*must-run*) não são consideradas (unidades nucleares).

Os exemplos práticos foram implementados e processados num computador pessoal com um processador Intel Pentium 5 de 2.66GHz e 2.49GB de memória RAM.

5.2 Métodos de Pesquisa

Os métodos de pesquisa presentes neste trabalho foram aplicados, de igual modo, nos dois exemplos práticos apresentados em cima.

A estratégia aplicada na selecção de variáveis do método de pesquisa teve por base a escolha do Menor Valor (MinVar). Esta estratégia selecciona a variável com menor valor no domínio. Este tipo de estratégia é bastante utilizada em problemas de escalonamento visto que permite selecionar a variável que terá menor custo. Desta forma já se reduz o espaço de procura uma vez que neste tipo de problema o objectivo é a optimização do custo.

Do mesmo modo que se optou por recorrer à escolha do menor valor de domínio na selecção de variáveis também na selecção de valores se recorreu à mesma estratégia. A aplicação da estratégia de Menor Valor (MinVal) na selecção de valores surge normalmente associada à estratégia MinVar para a selecção de variáveis, com o intuito de se seleccionar a variável com menor custo e instanciar essa variável com esse mesmo custo.

O objectivo do EPEE consiste em determinar quais os grupos geradores que entrarão em funcionamento e quanta energia irão produzir, desta forma realizamos testes considerando a construção da árvore de pesquisa sobre um tipo de variáveis que representam o custo total de produção de energia C(U).

A ideia da aplicação do método MinVar+MinVal+C(U) consiste em implementar uma heurítica que tenta escalonar os grupos geradores nos períodos com menor custo. Se todas as unidades pudessem ser escalonadas para os períodos aos quais correspondem o menor custo a primeira solução obtida seria óptima. Para implementar esta heurística é selecionada a variável C(U), que possui o menor valor no seu doínio, com o qual é instanciada.

A nível de estratégia de optimização serão aplicadas duas estratégias de *B&B* diferentes nos dois problemas práticos, o primeiro será o *B&B* normal tanto para o valor mais económico da solução como para o pior valor que se poderia encontrar. A segunda estratégia a ser testada é a estratégia *dichotomic*. Esta estratégia contém uma particularidade bastante interessante que é a seguinte, depois de ser encontrada uma solução para o problema, o intervalo de custo remanescente é dividido a meio e uma nova procura é iniciada no sub-intervalo inferior. Caso essa procura não tenha sucesso, é assumido o sub-intervalo superior como o limite de procura remanescente e é novamente dividido a meio, voltando o processo a repetir-se.

5.3 Exemplo Prático de EPEE com 4 unidades

Neste primeiro caso prático de 4 unidades geradoras que têm de ser escalonadas durante um período de 8 horas a restrição da reserva girante foi desactivada como em [1].

Na Tabela A.1 é indicada a procura de energia ao longo das 8 horas equivalente a um diagrama de cargas. Na Tabela A.2 são apresentadas as características das 4 unidades geradoras.

As Tabelas A.1 e A.2 servem de dados de entrada para as simulações que se apresentam em seguida.

5.3.1 Resultados aplicando *Branch & Bound* normal

Esta simulação foi executada com recurso à optimização *Branch* & *Bound* e à estratégia de continuação que é aplicada por defeito. Esta estratégia de *Branch* & *Bound* está descrita no 3º Capítulo desta dissertação. Apresentam-se de seguida os resultados obtidos:

- Tempo de processamento até se encontrar a 1ª solução: 0.02s
- Tempo de processamento até se encontrar a solução óptima: 0.02s
- Resultado final de custo: 72.280$
- Resultado máximo admissível: 112.200€

5.3.2 Resultados aplicando Branch & Bound dichotomic

Recorreu-se à estratégia *dichotomic* do *Branch* & *Bound* com o intuito de verificar se as soluções ou os tempos de processamento conseguiam ser melhorados. Foi apresentado em **5.2** o princípio de funcionamento desta estratégia. Apresentam-se de seguida os resultados obtidos:

- Tempo de processamento até se encontrar a 1ª solução: 0.02s
- Tempo de processamento até se encontrar a solução óptima: 0.02s
- Resultado final de custo: 72.280$
- Resultado máximo admissível: 112.200€

Os resultados destes dois exemplos de 4 unidades, tanto a nível de estados finais dos geradores, como a nível de potências, foram idênticos e podem ser consultados em A.3 e em A.4.

5.4 Exemplo Prático de EPEE de 38 unidades

Na Tabela A.4 apresentam-se as características de geração de energia das 38 unidades onde se encontram os limites das capacidades de produção, os coeficientes de custo do combustível, os custos de arranque e os tempos mínimos de arranque e de paragem. Para uma maior celeridade de processamento dos algoritmos, os custos de arranque foram assumidos como sendo constantes e foram omissos os custos de paragem como em [1].

Os valores de custos apresentados em [1] estão em Novos Dólares de Taiwan (TWN ou mais comum NT$)[8], neste trabalho entendeu-se fazer a sua conversão para Euros aproximando o problema mais à nossa realidade. Todavia esta conversão é demasiado desfavorável e visto que os valores em NT$ já são relativamente baixos, optou-se por considerar os coeficientes de custo de combustível B e C como nulos, uma vez que são tão baixos que se podem considerar desprezáveis.

Na Figura 5.1 podemos agora observar os valores que foram utilizados a partir do diagrama de cargas neste exemplo prático de 38 unidades geradoras de energia.

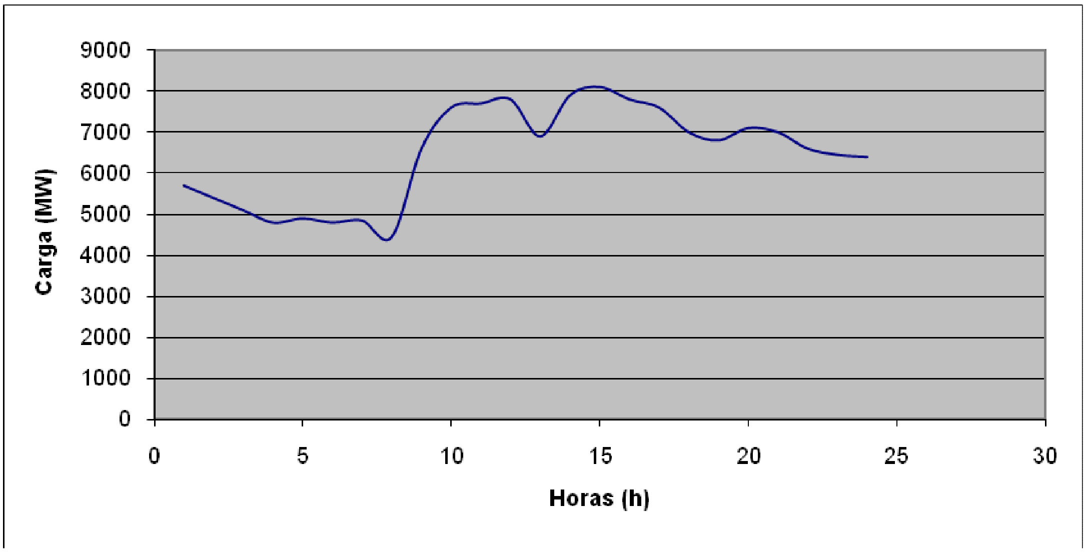

Figura 5.1: Diagrama de cargas para o sistema de 38 unidades

[8] 1 NT$ = 0.0244 €

5.4.1 Resultados aplicando *Branch & Bound* normal

Esta simulação foi executada com recurso ao *Branch* & *Bound* com a mesma estratégia de optimização utilizada em **5.3.1**. Apresentam-se de seguida os resultados obtidos:

- Tempo de processamento até se encontrar a 1ª solução: 16.77s
- Tempo de processamento até se encontrar a solução óptima: após 8 horas de processamento não se conseguiu encontrar outra solução para o problema.
- Resultado de custo da 1ª solução: 23.190.000€
- Resultado máximo admissível: 110.740.000€

5.4.2 Resultados aplicando *Branch & Bound dichotomic*

Esta simulação foi executada com recurso ao *Branch* & *Bound* com a mesma estratégia de optimização utilizada em **5.3.2**. Apresentam-se de seguida os resultados obtidos:

- Tempo de processamento até se encontrar a 1ª solução: 16.72s
- Tempo de processamento até se encontrar a solução óptima: após 8 horas de processamento não se conseguiu encontrar a solução óptima para o problema.
- Melhor resultado de custo encontrado ao fim de 8 horas: 22.770.000€
- Resultado máximo admissível: 110.740.000€

Na Figura 5.2 apresenta-se a evolução das soluções encontradas durante um período de 8 horas com a estratégia de optimização *Branch* & *Bound Dichotomic* .

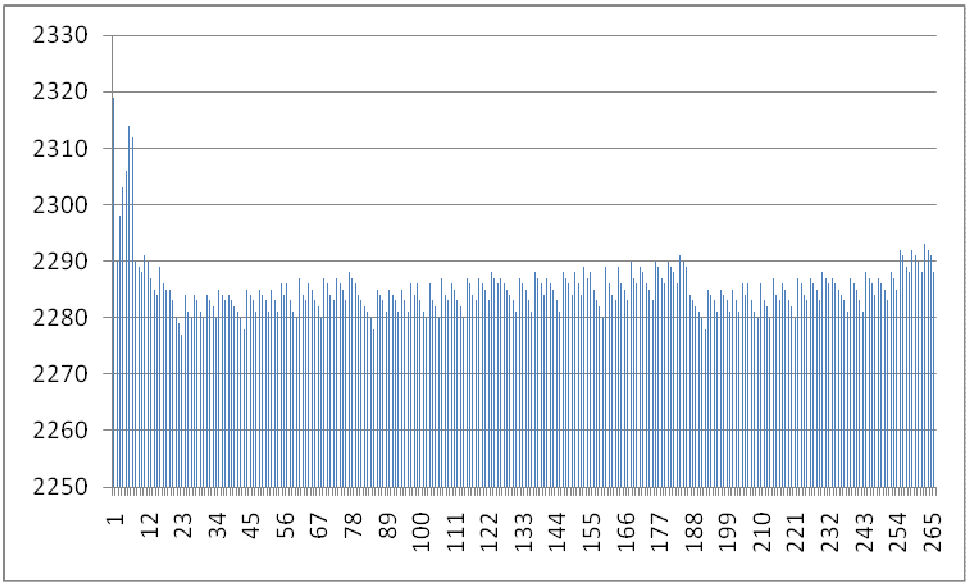

Figura 5.2: Soluções encontradas através do *B&B Dichotomic*

Visto a dimensão deste caso prático ser extensíssima dado que temos 38 unidades produtoras que terão esse mesmo número de estados e de potências durante cada uma das 24 horas, o que daria 38(unidades) x 2(estados e potências) x 24(horas) = 1824 variáveis para serem instanciadas, optou-se por não publicar os resultados finais dos estados e das potências dos geradores para os testes deste caso prático.

Apresenta-se a seguir na Tabela 5.1 a tabela resumo com os resultados dos problemas práticos descritos neste capítulo.

Exemplo Prático	**Estratégia de Optimização**	**Tempo de Processam.**	**Melhor Resultado de Custo Encontrado**
4 Unidades	***B&B Normal***	0.02s	72.280$
4 Unidades	***B&B* Dichotomic**	0.02s	72.280$
38 Unidades	***B&B Normal***	16.77s	23.190.000€
38 Unidades	***B&B* Dichotomic**	1h 41m	22.770.000€

Tabela 5.1: Resumo com os resultados dos problemas práticos

6

Conclusão

6.1 Conclusão

O objectivo principal desta dissertação consistiu no desenvolvimento duma ferramenta computacional para a Resolução do Problema de Escalonamento da Produção de Energia Eléctrica baseada em Programação Lógica por Restrições. Este objectivo era extremamente ambicioso pois a PLR é uma linguagem muito complexa, baseada na Programação Lógica tradicional, e ainda pouco divulgada. O desenvolvimento deste sistema pressupôs um estudo intenso tanto sobre a Programação Lógica em primeiro plano, como em seguida num segundo plano, sobre a Programação Lógica por Restrições. Após bastante estudo e bastantes tentativas o programa em PLR que dá resposta ao Problema de Escalonamento da Produção de Energia Eléctrica foi desenvolvido.

Com esta dissertação conclui-se que a aplicação da PLR nos problemas de escalonamento da produção de energia eléctrica é uma solução extremamente viável, como se pôde comprovar nas simulações efectuadas no exemplo das 38 unidades e nos resultados alcançados, é também uma solução altamente flexível visto que se pode adaptar a qualquer tipo de restrições e é uma solução que permite melhorar os planos parcialmente executados. A sua aplicabilidade neste momento é justificável devido ao paradigma da liberalização do mercado eléctrico, que hoje é real. Esta liberalização exige quase sempre negociação entre os diferentes agentes do mercado e com isso um consequentemente reescalonamento que tem de ser executado o mais rapidamente possível. Com o constante avanço das capacidades informáticas este tipo de solução é extremamente eficaz nesse campo. Este é um ponto deveras importante para o sucesso deste tipo de resolução de problemas de escalonamento, uma vez que a execução destes algoritmos exige imenso da capacidade de processamento das máquinas. Se na década de 90 se ponderava a utilização deste tipo de resolução mas os meios informáticos tornavam-na pouco atractiva, nos dias de hoje correm e com o avanço tecnológico a fazer-se sentir de dia para dia, julgo ser bastante importante a consideração da eventual aplicação da PLR na Resolução do Problema de Escalonamento da Produção de Energia Eléctrica.

Em suma julgo que esta dissertação levanta uma questão pertinente quanto à utilização da PLR na resolução dos problemas de EPEE, não só pelos resultados alcançados a nível de valores de custo com recurso ao melhoramento das heurísticas, mas também pelos resultados obtidos a nível de tempos de processamento até se encontrar a solução mais viável. A PLR é claramente uma solução quando existem problemas com várias restrições pois pode-se facilmente expressá-las, modifica-las e trata-las.

6.2 Trabalho Futuro

O tema abordado nesta dissertação tem ainda muita investigação a ser realizada, não só a nível académico como a nível profissional. É sempre necessário optimizar todos os recursos disponíveis e quando estamos a falar de focos tão poluidores como os grupos térmicos de produção de energia, deveremos encarar o problema com bastante seriedade.

Com o constante aumento, nos sistemas eléctricos, de energia oriunda de fontes renováveis, torna-se primordial conseguir conciliar o escalonamento da produção de energia eléctrica entre todas as fontes. Esse trabalho não é fácil devido às variâncias de produção das fontes renováveis, mas é um trabalho que tem que ser desenvolvido e a Programação Lógica por Restrições poderá ajudar nesse sentido.

A constante procura por melhores heurísticas que tragam novos resultados é decisiva para a implementação deste tipo de solução nos problemas de escalonamento. Com o constante avanço tecnológico a nível informático, julgo que se poderá utilizar este método na vida real sem comprometer o sistema. É necessário também testar a sua aplicação com os sistemas eléctricos de energia existentes pois poderão existir lacunas que comprometam este tipo de solução, como é o caso do desconhecimento dos limites de produção dos grupos geradores.

Referências

[1] Kun-Yuan Huang, Hong-Tzer Yang and Ching-Lien Huang. "A New Thermal Unit Commitment Approach Using Constrait Logic", in IEEE Trans. on Power Systems, Vol.13, No. 3, pp. 936-945, 1998.

[2] E. C. Freuder. "In Pursuit of the Holy Grail" in An International Journal, Kluwer Academic Publishers, 2, pp. 57-61, 1997.

[3] P. van Hentenryck and V. Sarawat. "Constraint Programming: Strategic Directions, Constraints" in An International Journal, Kluwer Academic Publishers, 2, pp. 7-33, 1997.

[4] I. Sutherland. "Sketchpad: a man-machine graphical communication system.", in *Proceedings of the IFIP Spring Joint Computer Conference*, 1997.

[5] R. Fikes. "*A heuristic program for solving problems stated as non-deterministic procedures*" in PhD thesis, Carnegie Mellon University,1968.

[6] U. Montanari. "Network of constraints: Fundamentals properties and applications to picture processing" in *Information Science*, 7, pp. 95-132, 1974.

[7] D. L. Waltz. "Understanding line drawings of scenes with shadows" in P. Winston, editor, *The Psychology of Computer Vision*, McGraw-Hill, 1975.

[8] G. L. Steele. "The definition and implementation of a computer programming language based on constraints" in PhD thesis, MIT, 1980.

[9] J. Jaffar and J.L. Lassez. "Constraint logic programming" in POPL'87: Proceedings 14th ACM Symposium on Principles of Programming Languages, pp. 111-119, 1987.

[10] V. Kumar. "Network Algorithms for constraint satisfaction problems: A survey" in *AI Magazine*, 13(1), pp. 32-44, 1992.

[11] N. J. Nilsson. "*Principles of Artificial Intelligence*" in Tioga, Palo Alto, 1980.

[12] A. K. Mackworth. "Consistency in networks of relations" in Artificial Intelligence, 8(1), pp. 99-118., 1977.

[13] P. van Hentenryck and Y. Deville. "The cardinality operator: A new logical connective for constraint logic programming" in *Proceedings of the 8th Int. Conf. on Logic Programming*, MIT Press, pp. 745-759., 1991.

[14] M. Dincbas, P. van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier. "*The Constraint Logic Programming Language CHIP*" in Fifth Generation Computer Systems, Tokyo, Japan, 1988.

[15] A. Aggoun and N. Beldiceanu. "Extending CHIP in order to solve complex scheduling problems" in *Journal of Mathematical and Computer Modelling*, 17(7), pp. 57-73, 1993.

[16] N. Beldiceanu and E. Contejean. "Introducing global constraints in CHIP" in *Journal of Mathematical and Computer Modelling*, 20(12), pp. 97-123, 1994.

[17] T. Frühwirth, A. Herold, V. Küchenhoff, T. Le Provost, P. Lim, E. Monfroy and M. Wallace. "*Constraint logic programming: An informal introduction*" in Technical Report ECRC-93-5, ECRC, European Computer-Industry Research Centre, 1993.

[18] F. Benhamou and W. J. Older. "Applying interval arithmetic to Integer and Boolean constraints" in Bell Northern Research, Technical Report, 1992.

[19] G. A. Sidebottom. "*A Language for Optimizing Constraint Propagation*" in Thesis, Simon Fraser University, Canada, 1993.

[20] P. Codognet and D. Diaz. "Compiling constraints in clp(fd)" in *Journal of Logic Programming*, 27(3), 1996.

[21] C. Holzbaur. "Meta-structures vs. Attributed Variables in the Context of Extensible Unification" in *International Symposium on Programming Language Implementation and Logic Programming (PLILP'92)*, pp 260-268, Springer LNCS 631, 1992.

[22] M. J. Maher. "Logic semantics for a class of committed-choice programs" in *Proceedings of 4th International Conference on Logic Programming*, pp. 858-876, 1987.

[23] T. Frühwirth. "Theory and Practice of Constraint Handling Rules" in *Journal of Logic Programming, Special Issue on Constraint Logic Programming,* 37(1-3), pp. 95-138, 1998.

[24] E. L. Lawler and D. E. Wood. "Branch-and-bound methods: a survey" in *Operations Research*, 14(4), pp 699-719, 1966.

[25] Harvey and Ginsberg. "Limited Discrepancy Search" in *Proceedings of International Joint Conferences of Artificial Intelligence*, pp. 607-613, 1995.

[26] Xiao-Qiang Cai and Kam-Ming Lo. "Unit Commitment By a Genetic Algotithm" in *Nonlinear Analysis, Theory, Mrthods & Applications*, Vol.30, No7, pp 4289-4299, 1997.

[27] Manuel António Matos. "Introdução ao Problema de Escalonamento e Pré-Despacho" in *Apontamentos para a Disciplina de DOSE*, 2000.

[28] http://www.ren.pt/home.asp.

[29] B. Venkatesh, T Jamtsho and H. B. Gooi "Unit Commitment – a fuzzy mixed integer Linear Programming solution" in *IET Gener. Transm. Distrib., 1(5)*, pp 836-846, 2007.

[30] J. Schimpf, P. Brisset, H. Sakkout, T. Frühwirth, C. Gervet, M. Meier, S. Novello, T. Provost, K. Shen, and M. Wallace. "ECLiPSe 4.2 User Manual. International Computers Limited and Imperial College", 1999.

[31] Nuno Filipe da Fonseca Bastos Gomes "Escalonamento Inteligente de Tarefas de Manutenção nos Sistemas Eléctricos de Energia" in Tese de Doutoramento, 2004.

[32] José António dos Reis Tavares "Geração de Configurações de Sistemas Industriais com o Recurso à Tecnologia das Restrições e Computação Evolucionária" in Tese de Doutoramento, 2000.

Anexo A

A.1 Dados e Resultados Relativos ao Capítulo 5

Hora	1	2	3	4	5	6	7	8
Carga	450	530	600	540	400	280	290	500

Tabela A.1: Procura de energia das cargas ao longo das 8 horas

Unidade	**Tempos Mínimos (h)**		**Custo médio a plena carga ($/MWh)**	**Max. (MW)**	**Min. (MW)**
	Subida	**Descida**			
U_a	5	4	20	300	75
U_b	5	3	20	250	60
U_c	4	2	24	80	25
U_d	1	1	28	60	20

Tabela A.2: Características de geração de energia das 4 unidades

	Hora							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Unidade								
U$_a$	1	1	1	1	1	1	1	1
U$_b$	1	1	1	1	1	1	1	1
U$_c$	0	0	0	0	0	0	0	0
U$_d$	0	0	1	0	0	0	0	0

Tabela A.3: Resultados do exemplo de 4 unidades, estados finais dos geradores

	Hora							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Unidade								
U$_a$	200	280	290	290	150	75	75	250
U$_b$	250	250	250	250	250	205	215	250
U$_c$	25	25	25	25	25	25	25	25
U$_d$	20	20	60	20	20	20	20	20

Tabela A.4: Resultados do exemplo de 4 unidades, potências finais dos geradores

Unidade	P_{max} (MW)	P_{min} (MW)	A (k€)	B (k€/MW)	C (k€/MW^2)	SC (k€)	MUT (h)	MDT (h)
1	272	120	1	0	0	14	9	8
2	272	120	1	0	0	14	9	8
3	260	110	1	0	0	11	11	8
4	500	200	2	0	0	20	18	8
5	500	200	2	0	0	20	18	8
6	500	200	2	0	0	20	18	8
7	500	200	2	0	0	20	1	8
8	500	200	2	0	0	20	1	8
9	500	200	2	0	0	20	13	8
10	550	220	2	0	0	20	13	8
11	550	220	2	0	0	20	13	8
12	125	60	1	0	0	3	2	8
13	190	80	1	0	0	2	11	7
14	110	55	1	0	0	7	3	7
15	75	35	1	0	0	6	2	7
16	38	20	1	0	0	2	11	8
17	38	20	1	0	0	2	3	8
18	500	114	4	0	0	10	2	7
19	500	114	4	0	0	10	11	7
20	500	114	4	0	0	10	7	7
21	500	114	4	0	0	10	7	7
22	500	110	2	0	0	14	7	8
23	365	90	2	0	0	14	7	8
24	365	82	2	0	0	14	9	8
25	325	120	2	0	0	14	12	8
26	315	65	5	0	0	1	12	1
27	315	65	7	0	0	1	10	1
28	315	65	7	0	0	1	1	1
29	60	20	3	0	0	1	1	1
30	60	25	2	0	0	1	1	1
31	70	20	2	0	0	1	1	1
32	70	20	3	0	0	1	1	1
33	70	20	3	0	0	1	1	1
34	60	18	3	0	0	1	1	1
35	60	8	2	0	0	1	1	1
36	70	20	3	0	0	1	1	1
37	60	25	3	0	0	1	1	1
38	150	10	3	0	0	1	1	1

Tabela A.5: Características de geração de energia das 38 unidades