5

OPTIMIZAÇÃO DE PROBLEMAS DE *LAYOUT* DE INSTALAÇÕES COM PROGRAMAÇÃO LÓGICA POR RESTRIÇÕES

Depois de ter sido apresentado o modelo proposto para a resolução de PPLI usando a tecnologia das restrições no capítulo anterior, nomeadamente com o recurso a meta-interpretadores de PLR(*DF*), é chegado o momento de tratar os aspectos práticos relacionados com a resolução destes problemas. Recorde-se que a resolução deste tipo de problemas consiste em encontrar a melhor solução e, como tal, este é essencialmente um problema de optimização. O estudo dos aspecto práticos relacionados com a resolução destes problemas resultou no desenvolvimento de um protótipo ao qual foi dada a designação de *LaRL0* (geração de ***La****yout* de instalações usando a tecnologia das **R**estrições e da **Ló**gica). Este protótipo permitiu, através da realização de algumas experiência, efectuar a avaliação

de alternativas para exploração do espaço de soluções relativamente à relação que existe entre a qualidade das soluções e o esforço computacional para as obter.

A primeira secção deste capítulo enumera os diferentes passos da *LaRLo*, desde a especificação do problema a solucionar até a obtenção da melhor solução. Na secção seguinte é descrito o formato usado para a formulação do problema que se pretende solucionar com o *LaRLo*. Por fim, a última secção tem por objectivo fundamental mostrar alternativas para a optimização destes problemas usando meta-interpretadores de PLR(*DF*).

5.1 Algumas Características do *LaRLo*

Tal como uma aplicação típica de optimização desenvolvida usando o paradigma da PLR(*DF*), o *LaRLo* pode ser dividido na seguinte sequência de etapas:

1. Enumeração das variáveis de decisão e seus domínios finitos. Estas variáveis de decisão estabelecem a posição e a forma das instalações a dispor na planta da unidade fabril;
2. Enumeração do conjunto de restrições do problema. De referir que, para os problemas de *layout* tratados, as restrições de 'não sobreposição' devem estar sempre presentes. As restrições específicas da instância do PPLI que se pretende solucionar são também enumeradas nesta etapa;
3. Definição da função de custo ou função objectivo. Esta função de custo traduz-se na prática num termo linear que contempla as variáveis de decisão do problema, quer directamente, quer através de terceiras variáveis que estão relacionadas com as variáveis de decisão por meio de restrições;
4. Optimização do problema propriamente dita. Esta etapa não envolve necessariamente a obtenção da solução óptima, podendo apenas circunscrever-se à pesquisa de uma solução de boa qualidade. A optimização é normalmente realizada com um algoritmo *Branch-and-Bound* (*B&B*) (Lawler e Wood,1966), cuja condição de paragem pode ser definida pelo utilizador, retornando a melhor solução encontrada até esse momento. Durante a optimização, a exploração do espaço de soluções é efectuado por um

processo de etiquetagem das variáveis de decisão. Em geral, uma boa escolha do processo de etiquetagem tem uma influencia dramática no desempenho global do processo de optimização.

As três primeiras etapas indicadas correspondem ao modelo para os PPLI descrito no capítulo anterior. Este identifica quais são as variáveis de decisão, os seus respectivos domínios e as relações que existem entre estas, ou seja, as restrições do problema. A última etapa é essencialmente o procedimento que soluciona o modelo para um dado problema em concreto.

Antes de se dar início a esta sequência de etapas é necessário concluir duas etapas preliminares. A primeira deve ser realizada pelo utilizador, aqui entendido como o projectista do *layout*, e consiste na recolha da informação necessária para solucionar o problema em questão e sua posterior descrição no formato aceite pelo *LaRLo*. O formato aceite pelo *LaRLo* será descrito com algum detalhe na secção seguinte e baseia-se nos requisitos de informação discutidos no capítulo anterior. A segunda etapa preliminar é realizada pelo *LaRLo* e consiste no cálculo do valor de fluxo entre os diferentes pares de instalações. Estes valores de fluxo são determinados em função do processo fabril, do volume de produtos a fabricar por unidade de tempo e da sequência de operações. A Figura 5-1 mostra a sequência de etapas seguida pelo *LaRLo* para solucionar o PPLI.

A forma como o cálculo do valor do fluxo entre os diferentes pares de instalações foi já descrito na secção 4.1. Estes valores de fluxo resultantes são, no entanto, valores pertencentes ao conjunto dos números reais ($\hat{A}$). Na PLR(*DF*) o tipo de valores considerados são apenas valores inteiros (Z). Como se refere na secção 4.1, o valor do fluxo entre os pares de instalações é um dos parâmetros que intervêm na avaliação da qualidade das soluções, e portanto, é necessário efectuar o arredondamento destes valores com consequente perda de precisão. Para minimizar esta perda de precisão o *LaRLo* usa um parâmetro denominado por *escala de valores de fluxo* (*EF*) para efectuar os arredondamentos através da expressão (5-1), onde *x* é um valor real e *v* é o inteiro resultante do seu arredondamento. Este parâmetro *EF* possui por defeito o valor 1, mas pode tomar um qualquer valor superior à unidade.

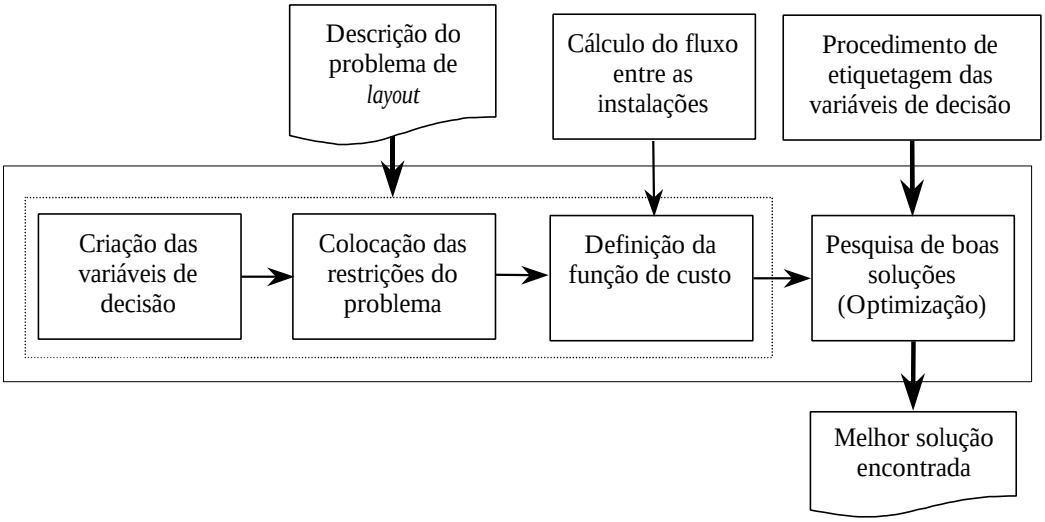

Figura 5-1: Sequência das etapas principais do *LaRLo*.

$$v = \lfloor x \cdot EF + 0.5 \rfloor \tag{5-1}$$

Estas questões de arredondamento também se colocam aos valores relacionados com as dimensões espaciais ou geométricas. Para estes valores é usado o parâmetro *escala espacial* (EE). A forma como o arredondamento é efectuado é semelhante, sendo agora efectuado através da expressão (5-2). O arredondamento dos valores relacionados com as dimensões espaciais é realizado essencialmente durante a etapa de criação das variáveis de decisão e durante a etapa de colocação das restrições do problema.

$$v = \lfloor x \cdot EE + 0.5 \rfloor \tag{5-2}$$

Finalmente, interessa referir qual a forma como o *LaRLo* organiza internamente as suas variáveis de decisão. A estrutura escolhida consiste numa lista com um tamanho igual ao número de instalações a dispor na planta. De referir que o número total de instalações é igual à soma do número de instalações de cada classe. Cada elemento da lista possui a estrutura seguinte:

$$\Phi\left(i(c,k), r\left(X_{ck}, Y_{ck}, C_{ck}, L_{ck}, F\right)\right)$$

onde

$i(c, k)$ é a instalação k da classe de instalações c ;

X é a variável de domínio que representa a coordenada *x* da posição da instalação *k* da classe *c* ;

Y_{ck} é a variável de domínio que representa a coordenada *y* da posição da instalação *k* da classe *c* ;

C_{ck} é a variável de domínio que representa a metade do valor do comprimento da instalação *k* da classe *c* ;

L_{ck} é a variável de domínio que representa metade do valor da largura da instalação *k* da classe *c* ;

F_{ck} é o valor da folga da instalação *k* da classe *c*.

A forma como o domínio de cada uma destas variáveis de decisão é determinado foi já descrita na secção 4.2. É necessário, no entanto, atender à forma como o arredondamento de valores é realizado, tal como foi referido nesta secção.

5.2 Descrição da Informação Necessária à Resolução de PPLI

A resolução de PPLI usando meta-interpretadores da PLR(*DF*) faz com que seja natural a escolha da sintaxe da linguagem *Prolog* para definir uma estrutura de dados que represente toda informação necessária à resolução de problemas de *layout*. A estrutura de dados que o *LaRLo* utiliza recorre, essencialmente, à extensão de quatro predicados: *planta*, *instalação*, *parte* e *restrições*. Quando a interacção entre as instalações depende também de outros factores não relacionados directamente com o processo produtivo recorre-se à extensão de um quinto predicado (*fluxos*) para especificar a interacção entre as diferentes instalações. Todos estes predicados possuem como argumento uma lista de propriedades que de alguma forma traduzem as propriedades que caracterizam as respectivas entidades. Estas propriedades podem tomar valores atómicos simples ou então são constituídas por sub-propriedades. A ordem em que as propriedades surgem na lista não é em princípio importante, existindo apenas uma excepção relacionada com a sequência de operações.

5.2.1 Planta

A descrição da planta de uma instalação fabril recorre à extensão do predicado planta e deve possuir as seguintes propriedades:

- **id** é usada para identificar a planta;
- **tamanhoX** corresponde ao comprimento do rectângulo que envolve a planta;
- **tamanhoY** corresponde à largura do rectângulo que envolve a planta;
- **interdições** é uma lista de regiões, possivelmente vazia, com uma forma rectangular e representam áreas onde não é possível a colocação nenhuma das instalações;

A extensão do predicado *planta* possui o seguinte formato:

```
planta([
    id( <ID> ),
    tamanhoX( <Valor> ), tamanhoY( <Valor> ),
    interdicoes([
        regiao( <Valor>, <Valor>, <Valor>, <Valor>),
        ...
    ])
]).
```

A Figura 5-2 mostra a especificação da planta representada na Figura 4-1. As propriedades relativas ao comprimento e à largura da planta fabril são usadas para construir o domínio das variáveis que definem a posição das instalações. Para cada uma das regiões interditas e para cada instalação é colocada uma restrição de posição absoluta que exclui a região dos locais possíveis para dispor a instalação.

```
planta([
        id( planta_exemplo ),
        tamanhoX( 500 ), tamanhoY( 400 ),
        interdicoes([
            regiao(   0,   0, 300,  40 ),
            regiao(   0,   0,  60,  50 ),
            regiao(   0, 360,  60,  50 ),
            regiao( 300, 380, 200,  20 )
        ])
]).
```

Figura 5-2: Especificação da planta de uma instalação fabril.

5.2.2 Instalações

A descrição das instalações passa pela utilização da extensão do predicado *instalação* e é feita em função das classes de instalações. Para a descrição de uma classe de instalações são necessárias apenas duas propriedades, a identificação da classe e a descrição das suas instâncias. A identificação da classe (*id*) é feita de forma idêntica à propriedade da planta com o mesmo nome. A segunda propriedade *instâncias* possui uma estrutura mais complexa e é constituída por uma lista de entidades (sub-propriedades) *instância* que descrevem as instalações da classe. Cada entidade *instância* possui por sua vez duas sub-propriedades obrigatórias e duas opcionais. As obrigatórias especificam a área mínima que a instalação ocupa e a lista de formas possíveis (*formatos*) que a instalação pode tomar. Em alternativa à área e lista de formas é possível indicar o valor do comprimento e da valor da largura da instalação quando a sua forma é fixa. Relativamente às opcionais, quando usadas, uma identifica a instalação (*id*) e a outra define a distância mínima que deve ser guardada em relação às instalações dispostas na vizinhança (*folga*).

Em geral a descrição de uma classe de instalações segue o seguinte formato:

```
instalacao([
    id( <ID> ),
    instancias([
        instancia([
            id( <ID> ),
            area( <Valor> ),
            formatos( <Lista de Valores ou Intervalos> ),
            folga( <Valor > )
        ]),
        instancia([
            id( <ID> ),
            comprimento( <Valor> ),
            largura( <Valor> ),
            folga( <Valor > )
        ]),
        ...
    ])
]).
```

A Figura 5-3 mostra a especificação da classe de instalações *C* indicada na Tabela 4-2. De notar que a propriedade *formatos* pode conter uma lista de valores ou um intervalo de valores para definir a forma das instalações.

```
instalacao([
    id( c ),
    instancias([
        instancia([ area( 3000 ), formatos([0.8 .. 1.5]), folga(2) ]),
        instancia([ area( 2500 ), formatos([0.8 .. 1.5]), folga(2) ])
    ])
]).
```

Figura 5-3: Especificação de duas instalações da mesma classe.

5.2.3 Produtos, Partes e Sequência de Operações

A descrição das partes processadas na planta requer seis propriedades: a identificação da parte (*id*); a capacidade da planta para produzir a quantidade que satisfaz a procura (*capacidade*); a composição da parte na suas subpartes (*composição*); o número de unidades da parte deslocadas pelo equipamento de transporte numa viagem (*lote*); a sequência de operações (*sequência*); e finalmente, o custo associado ao transporte por unidade de distância (*custo*). Esta última propriedade é opcional e quando o seu valor não é indicado considera-se que este possui um valor unitário.

A descrição de uma parte recorre à extensão do predicado *parte* que possui um argumento que contém uma lista de propriedades com o formato seguinte:

```
parte([
    id( <ID> ),
    capacidade( <Valor> ),
    composicao([
        subparte([ id( <ID> ),  quantidade( <Valor> ) ]),
        ...
    ]),
    lote( <Valor> ),
    sequencia([
        operacao([ id( <ID> ),  instalacao( <ID> ), taxas( <Lista de Valores> ) ]),
        ...
    ]),
    custo( <Valor> )
]).
```

As propriedades *capacidade*, *lote* e *custo* requerem apenas um valor numérico, enquanto que as propriedades *composição* e *sequência* possuem uma estrutura mais complexa. A propriedade *composição*, que é opcional, possui como argumento uma lista das subpartes que compõem a parte. Cada subparte na lista contém a identificação que refere a parte com o mesmo nome e também o número necessário destas subpartes para produzir a parte que se descreve. A ordem pela qual surgem as

sub-propriedades *operação* na propriedade *sequência*, ao contrário das restantes, é importante um vez que indica qual a ordem em que as operações serão realizadas. A Figura 5-4 mostra a descrição da parte *P*3 do problema exemplo do capítulo anterior.

```
parte([
    id( p3 ),
    capacidade( 400 ),
    composicao([
        subparte([ id( p1 ), quantidade( 2 ) ]),
        subparte([ id( p2 ), quantidade( 1 ) ])
    ]),
    lote( 10 ),
    sequencia([
        operacao([ id( p3_O1 ), instalacao( b ), taxas( [4,4,4] ) ]),
        operacao([ id( p3_O2 ), instalacao( g ), taxas( [4,4,3,3] ) ]),
        operacao([ id( p3_O3 ), instalacao( c ), taxas( [5,5] ) ]),
        operacao([ id( p3_O4 ), instalacao( f ), taxas( [4,4] ) ]),
        operacao([ id( p3_O5 ), instalacao( i ), taxas( [6,6] ) ]),
        operacao([ id( p3_O6 ), instalacao( j ), taxas( [4,6,6] ) ])
    ])
]).
```

Figura 5-4: Especificação das propriedades de uma parte.

5.2.4 Fluxo entre Instalações

O fluxo de material entre as diferentes instalações é calculado a partir da descrição de duas de entidades: as instalações e as partes. É, no entanto, possível existirem situações em o valor do fluxo entre instalações é já conhecido ou então é calculado de uma forma diferente daquela que usa a informação relacionada com a procura de produtos e com a sequência de operações. Nesta situação, e em alternativa à descrição das partes processadas na planta, é possível indicar directamente os valores de fluxo entre as instalações. A forma para indicar estes valores de fluxo recorre à extensão do predicado *fluxos* que possui o formato seguinte:

```
fluxos([
    fluxo( <ID>, <ID>, <Valor> ),
    ...
]).
```

A identificação das instalações deve ser feita de acordo com o nome da sua classe e um valor que especifica o índice da instalação na classe. Se disponível, este

índice pode ser substituído pela seu nome identificador.

5.2.5 Restrições do Problema

As restrições aqui referidas são apenas as usadas para traduzir as especificidades de cada problema em concreto e, como tal, devem ser indicadas pelo utilizador, aqui entendido como o projectista do *layout*. São apenas três as situações em que as restrições são colocadas automaticamente pelo *LaRLo*, e portanto, comuns a todos os PPLI. Estas situações relacionam-se com as restrições de 'não sobreposição' das instalações, com as áreas que não pertencem à planta real quando a forma desta não é rectangular e com a construção do termo linear que representa a função de custo, termo este que envolve variáveis presentes em restrições de distância.

As restrições específicas de cada problema concreto são indicadas com a utilização da extensão do predicado *restrições* e possui o formato seguinte:

```
restricoes([
    <Restricao>,
    ...
]).
```

Como se pode observar a extensão do predicado *restrições* possui como argumento uma lista de restrições. A Tabela 5-1 mostra alguns exemplos para cada tipo de restrição que podem surgir bem como a sua tradução para as restrições internas correspondentes.

Também se pode observar que as instalações são identificadas pela nome da sua classe e pelo número da instalação na classe. Dada a organização interna das variáveis de decisão do *LaRLo* é simples obter as variáveis de decisão da instalação correspondente.

Resta ainda referir que a restrição distância, nesta situação, refere-se à medida de distância relativamente à periferia. Além disso, o código de cada uma destas restrições, na sua forma interna, corresponde à colocação do respectivo conjunto de restrições básicas descrito na secção 4.2.

Tabela 5-1: Exemplos para cada tipo de restrições e a sua tradução interna.

Restrição	**Exemplos**	**Tradução Interna**
Vizinhança	vizinha(i(e,3), i(f,1))	*viz*(R_{e3}, R_{f1}).
Adjacência	adjacente(i(b,1), i(c,1))	*adj*(R_{b1}, R_{c1}).
Distância	distancia(entre(i(a,3), i(b,1)), >=, 100) distancia(entre(i(c,1), ponto(30,50)),=<,100)	*dl*(R_{a3}, R_{b1}, D), D >= 100. *dl*(R_{c1}, P, D), D =< 100.
Posição absoluta	dentro_de(i(g,2), regiao (20, 50, 40, 40)) fora_de(i(e,5), regiao(40, 10, 30, 70))	*dd*(R_{c1}, *Reg*). *fd*(R_{e5}, *Reg*).
Posição relativa	esquerda_de(i(g,1), i(a,2)) frente_de(i(g,1), i(a,2)) direita_de(i(e,3), i(d,4)) atras_de(i(e,3), i(d,4))	*esq*(R_{g1}, R_{a2}). *fre*(R_{g1}, R_{a2}). *dir*(R_{e3}, R_{d4}). *atr*(R_{e3}, R_{d4}).
Orientação absoluta	orientacao(i(a,1), x) orientacao(i(b,1), y)	*ori_x*(R_{a1}, 1). *ori_y*(R_{b1}, 1).
Orientação relativa	orientacao(i(d,4)) = orientacao(i(h,2)) orientacao(i(a,2)) \= orientacao(i(d,5))	*ori_igual*(R_{a1}, R_{h2}). *ori_dif*(R_{a2}, R_{d5}).

5.3 Pesquisa e Optimização

Muitos problemas práticos são de facto problemas de optimização, isto é, não interessa apenas uma solução ou todas as soluções, mas a melhor solução. Este é o caso dos PPLI tratados neste trabalho. 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 *B&B* e, de uma forma simplificada, funciona da seguinte forma com os meta-interpretadores de PLR(*DF*):

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.

Nas próximas secções, este algoritmo *B&B* será usado como algoritmo de optimização para solucionar os PPLI. 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(*DF*) relacionam-se com a:

- Estrutura do problema e modelo adoptado;
- Qualidade da propagação das restrições;
- Escolha de um bom procedimento de etiquetagem das variáveis de decisão.

Relativamente aos dois primeiros pontos, de certa forma, estes foram já discutidos no capítulo 4. Este capítulo trata apenas os aspectos relacionados com os métodos de etiquetagem das variáveis de decisão.

5.3.1 Colocação das Restrições e a Avaliação das Soluções

Após a etapa de criação das variáveis de decisão segue-se a colocação das restrições do problema. Como já se referiu, as restrições que são sempre colocadas, independentemente das diferentes instâncias dos PPLI, são as restrições de 'não sobreposição'. Esta restrição é colocada para cada um dos diferentes pares possíveis de instalações a dispor na planta e, deste modo, o número de restrições (*NR*) colocadas deste tipo é dado pela expressão (5-3), sendo *n* o número de instalações.

$$NR = \binom{n}{2} = \frac{n \times (n-1)}{2} \tag{5-3}$$

Outras restrições colocadas automaticamente são restrições de posição absoluta que excluem determinadas áreas da planta da unidade fabril onde não é possível dispor as instalações. Isto ocorre quando a descrição da planta inclui uma ou mais

regiões interditas. Recorde-se que estas regiões interditas surgem dado que a planta real da unidade fabril é modelada por um rectângulo que envolve a sua forma real, sendo que, estas regiões indicam áreas que não pertencem à planta real. Atendendo a que são *n* as instalações a dispor na planta e que são declarada *r* regiões interditas então o número de restrições deste tipo que se colocam é $n \times r$.

A quantidade de restrições de outro tipo é à partida desconhecida. Esta depende de cada problema concreto a solucionar e da descrição que o utilizador efectuar desse problema. De referir que a colocação destas restrições, em particular as relacionadas com a distância e as de posição absoluta, considera também as questões de arredondamento já discutidas na secção 5.1. Há ainda a apontar as restrições que se relacionam com as variáveis de decisão relativas a cada uma das instalações. O número deste tipo de restrições é também proporcional ao número de instalações.

Foi já referido que, durante o processo de optimização de problemas usando meta-interpretadores de PLR(*DF*), são colocadas restrições que limitam o custo das soluções a procurar, impedindo que se explorem soluções com custo superior ao da melhor solução encontrada até ao momento. Estas restrições adicionais, para o caso dos PPLI, são baseadas na função de custo dada pela expressão (4-11) do capítulo 4. Para que estas restrições sejam colocadas é necessário definir esta função de custo na forma de um termo linear (t_{custo}) (ver secção 3.3.2). Na construção deste termo, cada uma das variáveis de t_{custo} corresponde à distância entre um dado par de instalações. O coeficiente associado a cada uma das variáveis é o correspondente valor de fluxo. Para o cálculo do custo das soluções considera-se que a distância entre duas instalações é medida relativamente ao seu ponto central. Assumindo que a variável D_{iujv} representa a distância entre a instalação *u* da classe *i* e a instalação *v* da classe *j*, o seu valor é dado pela relação estabelecida ao colocar a restrição (5-4), sendo que R_{iu} e R_{jv} são as estruturas que contêm as variáveis de decisão das respectivas instalações.

$$dc\left(R_{iu}, R_{jv}, D_{iujv}\right) \tag{5-4}$$

De referir que só entram na construção do termo linear, que representa a função de custo, os pares de instalações que possuem um valor de fluxo não nulo. Portanto, o número destas restrições de distância que necessitam de ser colocadas depende do número de pares de instalações que possuem um valor de fluxo não nulo. No

entanto, o número máximo de restrições a colocar é igual ao número de restrições de "não sobreposição" que, como se viu, é dado por (5-3).

Considerando todos os pares de instalações que possuem um valor de fluxo não nulo entre si (*NPI* pares), o valor de fluxo para cada par de instalações *k* é definido pelo predicado φ_k. Considere-se, ainda, o predicado Φ que associa uma dada instalação com as respectivas variáveis de decisão. O termo linear que representa a função de custo é, deste modo, definido pela expressão (5-5)

$$t_{Custo} = \sum_{k=1}^{NPI} t_k \tag{5-5}$$

sendo t_k determinado pelo procedimento indicado na Figura 5-5.

$t_k \leftarrow$
$\varphi_k(I_{iu} , I_{jv} , F_{iuju})$,
$\Phi(I_{iu} , R_{iu})$,
$\Phi(I_{jv} , R_{jv})$,
$dc(R_{iu} , R_{jv} , D_{iujv})$,
$t_k = F_{iuju} \times D_{iujv}$

Figura 5-5: Determinação da contribuição do par de instalações *k* para a função custo.

Durante o processo de optimização sempre que uma nova solução é encontrada possuindo um custo *C* é colocada a restrição (5-6). Após a colocação desta restrição a pesquisa continua, mas agora as novas soluções que serão eventualmente encontradas possuem todas um custo inferior a *C*. De notar que a colocação desta restrição nestas condições quebra, no entanto, a semântica lógica subjacente à programação em lógica.

$$t_{Custo} < C \tag{5-6}$$

Em geral, os meta-interpretadores de PLR(*DF*) já possuem embebidos primitivas que permitem realizar estas tarefas de optimização que seguem o processo descrito. Estas primitivas requerem apenas como argumentos um procedimento de etiquetagem das variáveis de decisão e um termo linear que represente a função de custo.

5.3.2 Dimensão do Espaço de Soluções

Em geral, os modelos para PPLI são problemas que possuem um espaço de soluções bastante vasto, sendo a sua resolução computacionalmente difícil. Os algoritmos usados na resolução destes problemas são (NP)-completos. Considerando o QAP, por exemplo, que é talvez o modelo mais simples usado para solucionar os problemas de *layout*, o tamanho do seu espaço de soluções é $n!$ (n é o número de instalações), o que significa que os algoritmos usados na sua resolução, que exploram todas as soluções, possuem uma ordem de complexidade temporal $O(n!)$.

Relativamente ao modelo proposto e descrito no capítulo 4, verifica-se que na melhor das situações este apresenta a mesma complexidade do QAP. E para que a melhor situação se verifique, o problema a solucionar deve apresentar algumas características particulares, sendo estas as seguintes:

- As instalações possuem uma área igual;
- A sua forma é fixa e a orientação é sempre a mesma. Portanto, o comprimento e largura das instalações são constantes;
- O produto do tamanho do domínio da variável *x* com o tamanho do domínio da variável *y* de cada instalação é igual ao número de instalações;
- Não se consideram restrições específicas de cada problema em concreto.

De referir que os problemas com as características indicadas também podem ser solucionados através do QAP.

No caso geral, estima-se que o tamanho do espaço de soluções, para o formulação do problema efectuada no capítulo 4, é sempre igual ou inferior ao valor *T*, sendo este calculado através da expressão (5-7).

$$T = \prod_{i=1}^{m} \prod_{u=1}^{NI_i} \left(\Delta(X_{iu}) \times \Delta(Y_{iu}) \times \Delta(C_{iu}) \right) \tag{5-7}$$

em que

m é o número de classes de instalações;

NI_i é o número de instalações da classe i ;

X_{iu} e Y_{iu} são variáveis de domínio relativas às coordenadas da instalação *u* da classe *i* ;

C_{iu} e L_{iu} são variáveis de domínio relativas a metade do comprimento e metade da largura da instalação *u* da classe *i* ;

$\Delta(X)$ é a função que fornece o tamanho do domínio de uma dada variável.

Observando em detalhe a expressão (5-7) verifica-se que o produto $\Delta(X_{iu})\times\Delta(Y_{iu})$ está relacionado com as dimensões da planta e com a escala adoptada para os valores relacionados com os parâmetros espaciais. Para as mesmas dimensões da planta, quando maior for o valor deste produto maior será a precisão das soluções e, naturalmente, maior o espaço de soluções. Para que todas as instalações de todas as classes possam ser dispostas o valor de $\Delta(X_{iu})\times\Delta(Y_{iu})$ deve ser sempre superior a *n*, sendo *n* dado por (5-8).

$$n = \sum_{i=1}^{m} NI_i \tag{5-8}$$

Portanto, considera-se que $\Delta(X_{iu})\times\Delta(Y_{iu})$ é proporcional a *n* dada uma constante de proporcionalidade *a*. É claro que à medida que as variáveis de decisão forem sendo instanciadas o valor deste produto irá diminuir, no entanto, em média, o seu valor será sempre proporcional a *n*. Por outro lado, o valor de $\Delta(C_{iu})$ corresponde ao número de formas possíveis para a instalação *i* da classe *u*. De notar que, de acordo com o modelo, o valor de $\Delta(C_{iu})$ é igual a $\Delta(L_{iu})$. Considerando-se também que o número médio de formas possíveis que as instalações podem ter é igual a uma constante *b* obtém-se então a expressão (5-9) para cálculo do valor de T .

$$T = \prod_{i=1}^{n} a \times n \times b \tag{5-9}$$

Manipulando a expressão (5-9) chega-se à expressão (5-10) que define o tamanho máximo para o espaço de soluções.

$$T = a^n \times b^n \times n^n \tag{5-10}$$

Em termos práticos, o tamanho do espaço de soluções praticáveis depende também das restrições específicas de cada problema. Estas restrições podem,

potencialmente, diminuir o espaço de soluções que é necessário explorar.

Em conclusão, o modelo proposto para a resolução de PPLI usando meta-interpretadores PLR(*DF*) dá origem a um espaço de soluções bastante vasto. A exploração de todo o espaço de soluções usando um algoritmo *B&B* é demasiado pesado computacionalmente, sendo até mesmo impraticável para a maior parte dos problemas reais. Tendo em conta o valor estimado para o tamanho mínimo e máximo do espaço de soluções a ordem de complexidade temporal de um algoritmo que explore todas as soluções é sempre igual ou superior a O($n!$) e inferior ou igual a O(n^n). Considerando todo este panorama, o uso de um algoritmo *B&B* tem de recorrer à colaboração de algumas técnicas heurísticas que permitam guiar a pesquisa para boas soluções sem ser necessário efectuar a exploração completa do espaço de soluções.

5.3.3 Exploração do Espaço de Soluções

Tal como se mostrou no capítulo 3, os meta-interpretadores de PLR(*DF*) são incompletos, ou seja, para um dado problema, não é suficiente o conhecimento dos domínios para as suas variáveis de decisão nem o conhecimento das suas restrições para que este seja solucionado e, portanto, é necessário recorrer a um procedimento de etiquetagem para o solucionar. No entanto, a maioria dos problemas práticos possuem mais do que uma solução, pretendendo-se normalmente a melhor solução de acordo com um dado critério. Neste caso, o procedimento de etiquetagem destina-se não só a solucionar o problema, ao procurar uma solução, como também deve ser capaz de explorar todas as soluções que satisfazem as restrições.

Em geral, o procedimento de etiquetagem deve atender a duas questões importantes: a ordem pela qual as variáveis são seleccionadas para instanciação e a ordem com que os valores do domínio de cada variável são seleccionados para as instanciar. Relativamente à primeira questão, e como já foi referido no capítulo 3, o número de nós internos da árvore de pesquisa pode variar, dependendo da ordem com que as variáveis são seleccionadas para instanciação. Esta variação do número de nós internos da árvore de pesquisa relaciona-se com a quantidade de valores eliminados do domínio das variáveis por instanciar, por intermédio do mecanismo de

propagação, em função dos valores atribuídos às variáveis já instanciadas. Deste modo é importante que o procedimento de selecção das variáveis considere uma ordem de instanciação das variáveis que minimize o número de nós internos da árvore de pesquisa. De facto, quanto menor for o número de nós da árvore de pesquisa que for necessário visitar melhor é o desempenho conseguido.

Ao contrário do que acontece com a ordem de selecção das variáveis, a ordem de selecção de valores não altera o número de nós internos da árvore de pesquisa. A ordem de selecção de valores apenas influencia a forma da árvore de pesquisa, ou seja, a ordem pela qual os nós são visitados. Quando se pretende encontrar apenas uma solução, a escolha de uma boa heurística para a ordem de selecção de valores pode permitir que a solução seja encontrada mais rapidamente. Quando se pretende encontrar todas as soluções, a ordem de selecção de valores é indiferente. No entanto, para problemas de optimização, que usam determinados algoritmos de optimização como, por exemplo, o algoritmo *B*&*B*, a ordem de selecção de valores já é importante. Neste caso a ordem de visita aos nós permite que melhores soluções possam ser encontradas mais cedo e, desta forma, maximizar a quantidade de ramificações que não necessitam de ser visitadas por garantidamente não conduzirem a soluções melhores do que as que foram já encontradas.

Tendo em conta modelo proposto para o PPLI, foram experimentadas inicialmente duas heurísticas para a ordem de selecção das variáveis. A primeira considera uma ordenação estática em que a ordem de selecção das variáveis é conhecida á partida. A segunda é dinâmica e baseia-se no princípio do "falhar primeiro". Recorde-se que, em termos práticos, este princípio consiste em seleccionar em primeiro lugar a variável que possua o domínio com o menor tamanho. Estas heurísticas experimentadas inicialmente agrupam as variáveis sem terem em conta quais as instalações a que estão associadas. A selecção das variáveis é feita em função do conjunto de todas as variáveis relativas à coordenada *x*, do conjunto de todas as variáveis relativas à coordenada *y*, do conjunto de todas as variáveis relativas ao comprimento e do conjunto de todas as variáveis relativas à largura. No entanto, verificou-se que os procedimentos de etiquetagem com estas características mostraram possuir um desempenho de muito fraca qualidade e em que na maior parte das situações não foram capazes de encontrar qualquer solução ao fim

de uma hora de processamento.

Como tentativa para melhorar o desempenho foram desenvolvidos procedimentos de etiquetagem das variáveis, que ao contrário de considerarem como unidade atómica de selecção as variáveis de decisão, consideram desta vez as instalações como unidades atómicas de selecção. Neste caso é seleccionada uma instalação de cada vez e são instanciadas imediatamente todas as variáveis de decisão que lhe estão associadas. Com estes procedimentos foi verificada uma melhoria substancial do desempenho, tendo-se encontrado pelo menos uma solução em escassos minutos de processamento em quase todas as situações. Alguns destes procedimentos de etiquetagem serão descritos com algum detalhe numa das secções seguintes. Com os procedimentos de etiquetagem desenvolvidos foram usados 4 (quatro) heurísticas para a ordem de selecção de valores. Estas heurísticas são de aplicação geral dado que não dependem do problema em questão e consistem em:

- Seleccionar primeiro o menor valor;
- Seleccionar primeiro o maior valor;
- Seleccionar primeiro o valor médio e depois alternar em torno do valor médio até aos valores situados nos dois extremos do domínio, o inferior e o superior;
- Seleccionar primeiro o valor médio e partir o domínio em dois sub-domínios, um com os valores inferiores ao valor médio e o outro com os valores superiores. O processo prossegue recursivamente, em primeiro lugar com o sub-domínio que contém os valores inferiores e depois contém o sub-domínio com os valores superiores.

A Tabela 5-2 mostra um exemplo de um domínio com dez valores e a ordem resultante para cada um dos quatro critérios de ordenação de valores referidos.

Tabela 5-2: A ordem de selecção dos valores para quatro heurísticas diferentes.

Domínio	**1 .. 10**
Menor	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Médio	{6, 5, 7, 4, 8, 3, 9, 2, 10, 1}
Maior	{10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
Partição	{6, 3, 2, 1, 5, 4, 9, 8, 7, 10}

As experiências realizadas com os diferentes procedimentos de etiquetagem desenvolvidos foram realizadas considerando 5 (cinco) PPLI, utilizados como casos de teste. Estes possuem diferentes graus de complexidade e encontram-se resumidos na Tabela 5-1. A sua especificação completa encontra-se representada no anexo B e no formato descrito na secção 5.2.

Tabela 5-3: Resumo dos diferentes casos de teste.

Problema	**Número de Instalações**	**Restrições específicas**	**Observações**
ppli8	8	Nenhuma.	Instalações de comprimento e largura fixa.
ppli10	10	Planta com um forma diferente de um rectângulo.	Instalações com várias formas possíveis e com folga entre elas.
ppli10c	10	• Planta com uma forma diferente de um rectângulo; • Uma restrição de distância entre duas instalações; • Uma restrição de posição absoluta.	É igual a 'ppli10' mas com restrições adicionais.
ppli15	15	Nenhuma	Instalações com várias formas. O valor de fluxo entre as diferentes instalações é fornecido directamente.
ppli24	24	Planta com uma forma diferente de um rectângulo.	Instalações com várias formas. Algumas requerem uma folga em relação às restantes.

5.3.4 Procedimentos de Etiquetagem na Resolução dos Casos de Teste

Nesta secção são descritos os procedimentos de etiquetagem desenvolvidos que consideram como unidade atómica de selecção de variáveis as instalações. Isto significa que as instalações são seleccionadas uma a uma para colocação na planta. A colocação de uma instalação na planta corresponde, em termos práticos, a instanciar as variáveis de decisão que lhe estão associadas, tanto as variáveis que definem a sua forma como as variáveis que definem a sua posição na planta.

Cinco procedimentos de etiquetagem serão tratados nesta secção, embora, obviamente, poderão ser considerados muitos mais. De igual modo não se considera que estes sejam os melhores procedimentos de etiquetagem, sendo antes de mais aqueles que, de todos os que foram experimentados, mostraram dar soluções relativamente boas em tempo útil. Foi já mostrado que o espaço de soluções do modelo proposto é muito vasto e, como tal, a preocupação principal residiu inicialmente em encontrar um ou mais procedimentos de etiquetagem que proporcionem boas soluções em tempo útil. Estes procedimentos de etiquetagem são usados em colaboração com um algoritmo *B*&*B* que pode ser interrompido ao fim de um período de tempo pré-determinado.

Estes procedimentos de etiquetagem foram postos à prova na resolução dos cinco casos de teste apresentados na secção anterior. O custo das soluções obtidas bem como o tempo de processamento (em segundos) gasto para as obter são apresentados nas subsecções seguintes. Para cada um dos procedimentos de etiquetagem foram realizadas vinte experiências que correspondem às quatro heurísticas da ordem de selecção dos valores indicadas na Tabela 5-2 e aos cinco casos de teste referidos. Para cada experiência foi imposto um tempo máximo de processamento igual a uma hora.

Relativamente ao ambiente em que ocorreram as experiências interessa referir que o *LaRLo* foi escrito para o meta-interpretador de PLR(*DF*) *ECLiPSe* 4.2 (Schimpf *et al*, 1999) e a máquina utilizada foi uma SUNSparc Interprise com o Solaris 7. Como curiosidade, verificou-se que o desempenho do *ECLiPSe* numa

SUNSparc é da mesma ordem de grandeza ao de uma máquina com um Intel Pentium III com o MSWindows NT4.0.

Procedimento de Etiquetagem nº 1

Este primeiro procedimento de etiquetagem (*PE*1) das variáveis de decisão considera uma ordem de selecção das instalações estática e pseudo-aleatória (depende da ordem como internamente a informação é armazenada). Em termos funcionais o *PE*1 começa por escolher a forma para todas as instalações e só depois é que escolhe a posição de cada uma das instalações. O código para este *PE*1 encontra-se representado na Figura 5-6. O primeiro argumento de *PE*1 (implementado pela extensão do predicado *etiq*/2) é a lista das instalações a dispor na planta considerando a estrutura que as associam com as respectivas variáveis de decisão e o segundo argumento indica qual a heurística a usar para a ordem de selecção de valores.

```
etiq( {Φ}, λ ) ←
    etiq_cl( {Φ}, λ ),
    etiq _xy( {Φ}, λ ).

etiq_cl( [ ], _ ).
etiq_cl( [ ( _, r( _, _, C, L, _ ) ) | T ], λ ) ←
    nodominio( C, λ ),    nodominio ( L, λ ),
    etiq_cl( T, λ ).

etiq_xy( [ ], _ ).
etiq_xy( [ ( _, r( X, Y, _, _, _ ) ) | T ], λ ) ←
    nodominio( X, λ ),    nodominio( Y, λ ),
    etiq_xy( T, λ  ).
```

Figura 5-6: Código para o *PE*1.

De referir ainda que o código para o *PE*1 usa a extensão do predicado *nodomínio*/2 que tem a função de atribuir à variável presente no seu primeiro argumento um valor do seu domínio de acordo com a heurística da ordem de selecção de valores λ indicada no segundo argumento. Antes de falhar, este predicado tenta instanciar, por retrocesso, a variável presente no primeiro argumento com todos os valores do seu domínio.

Os resultados da resolução dos diferentes casos de teste usando este *PE*1 encontram-se representados na Tabela 5-4. Para cada um dos diferentes casos de

teste e para cada heurística da ordem de selecção de valores é apresentado o custo da primeira solução encontrada e o custo da melhor solução encontrada dentro do período de processamento máximo considerado. É também indicado o tempo de processamento que foi gasto para encontrar a primeira solução e a melhor solução.

Tabela 5-4: Custo das soluções pelo uso do *PE*1.

Problema		**ppli8**	**ppli10**	**ppli10c**	**ppli15**	**ppli24**
Menor	Primeira	42712	40405	33964	30519	113239
	Tempo	1,07	2,12	2,03	10,68	15,55
	Melhor	40032	28635	**32085**	**30096**	111950
	Tempo	37,52	214,41	**1795,73**	**269,88**	3361,15
Médio	Primeira	41768	28013	34644	34159	112276
	Tempo	1,54	5,08	4,37	28,15	35,71
	Melhor	**33402**	**27791**	34332	33925	**109372**
	Tempo	**3547,14**	**2832.88**	2972,92	2482,58	**2727,31**
Maior	Primeira	58483	35571	38462	33951	115241
	Tempo	0,97	2,18	2,09	8,98	21,37
	Melhor	40963	34390	32687	33331	114474
	Tempo	723,37	1800.00	2664,59	354,49	3221,06
Partição	Primeira	47563	54222	45945	43310	137520
	Tempo	95,17	1,76	1,26	2,79	4,61
	Melhor	44928	49937	41233	40537	131525
	Tempo	3367,09	3116,24	2902,87	914,25	3294,79

Procedimento de Etiquetagem nº 2

Este procedimento de etiquetagem (*PE*2) apresenta algumas características comuns com o *PE*1. A diferença fundamental está em que, em vez de escolher em primeiro lugar a forma para todas as instalações e só depois a posição destas, a posição de uma instalações é escolhida imediatamente após a escolha da sua forma. Isto significa que as instalações que ainda não foram colocadas na planta não têm ainda a sua forma definida. O código para *PE*2 encontra-se representado na Figura 5-7 e os resultados com ele obtidos na resolução dos casos de teste estão representados na Tabela 5-5. Esta é semelhante à Tabela 5-4.

```
etiq( [ ], _ ).
etiq( [ ( _, r( X, Y, C, L, _ ) ) | T ], λ ) ←
   nodominio( C, λ ),   nodominio( L, λ ),
   nodominio( X, λ ),   nodominio( Y, λ ),
   etiq( T, λ ).
```

Figura 5-7: Código para o *PE*2.

Tabela 5-5: Custo das soluções pelo uso do *PE*2.

Problema		**ppli8**	**ppli10**	**ppli10c**	**ppli15**	**ppli24**
Menor	Primeira	42712	40405	33964	30519	113239
	Tempo	1,09	2,86	2,37	9,93	47,27
	Melhor	40032	28670	**32801**	**30070**	112211
	Tempo	107,42	3530,54	**3,85**	**269,81**	3557,62
Médio	Primeira	41768	28013	34644	34159	112276
	Tempo	1,38	5,31	4,68	19,99	63,86
	Melhor	**33358**	**27711**	34386	33973	**109493**
	Tempo	**3458,57**	**655,19**	456,59	3309,94	**3560,86**
Maior	Primeira	58483	35571	38461	33951	115241
	Tempo	1,48	7,14	6,42	25,29	66,15
	Melhor	40963	34896	33911	33307	114646
	Tempo	1741,85	1934,71	1106,97	1366,75	3600,00
Partição	Primeira	440623	37746	45945	43310	137520
	Tempo	0,92	1,44	1,26	2,60	5,06
	Melhor	42513	35844	40678	40531	133223
	Tempo	3587,29	3543,03	2667,09	3231,65	3493,75

Procedimento de Etiquetagem nº 3

Este procedimento de etiquetagem *PE*3 (ver Figura 5-8) considera uma ordem dinâmica de selecção das instalações. Esta ordem dinâmica depende do tamanho do domínio das variáveis relacionadas com as coordenadas das instalações. De resto *PE*3 é em tudo semelhante ao *PE*1 quando escolhe em primeiro lugar a forma para todas as instalações e só depois as suas posições. A escolha da próxima instalação para colocar na planta é realizada pela extensão do predicado *remove_pf*/3. Este escolhe a instalação ainda não colocada na planta que possui o menor valor resultante do produto entre tamanho dos domínio das variáveis relativas à coordenada *x* e

coordenada *y* respectivamente. Os resultados da utilização do *PE*3 para solucionar os casos de teste encontram-se representados na Tabela 5-6. Note-se que a ausência de resultados significa que não foi encontrada qualquer solução dentro do limite de tempo considerado.

```
etiq( {Φ}, λ ) ←
   etiq_cl( {Φ}, λ ),    etiq _xy( {Φ}, λ ).

etiq_cl( [ ], _ ).
etiq_cl( [ ( _, r( _, _, C, L, _ ) ) | T ], λ ) ←
   nodominio( C, λ ),    nodominio ( L, λ ),
   etiq_cl( T, λ ).

etiq_xy( [ ], _ ).
etiq_xy( {Φ}, λ ) ←
   remove_pf( ( _, r( X, Y, _, _, _ ) ), {Φ}, T ),
   nodominio( X, λ ),    nodominio( Y, λ ),
   etiq_xy( T, λ ).
```

Figura 5-8: Código para o *PE*3.

Tabela 5-6: Custo das soluções pelo uso do *PE*3.

Problema		**ppli8**	**ppli10**	**ppli10c**	**ppli15**	**ppli24**
Menor	Primeira	41039	30549	34567	36332	115229
	Tempo	1,34	2,73	2,26	16,36	15,82
	Melhor	**35359**	**29589**	33585	36160	**114787**
	Tempo	**2192,34**	**347,71**	3590,81	2880,34	**3509,41**
Médio	Primeira	56019	32926	-	35142	124034
	Tempo	1,6	4,47	-	34,04	71,98
	Melhor	43954	31933	-	34983	122265
	Tempo	3226,41	3562,03	-	235,81	3596,75
Maior	Primeira	35999	36580	32525	34183	123405
	Tempo	1,13	3,39	4,27	14,06	59,54
	Melhor	35824	33760	**31723**	**34073**	122352
	Tempo	3366,97	382,52	**1857,54**	**27,55**	184,69
Partição	Primeira	55254	40548	37211	42564	148654
	Tempo	0,93	1296,87	1,25	2,62	4,91
	Melhor	47349	40097	33777	39247	144526
	Tempo	3495,44	2621,43	393,98	2726,26	1510,71

Procedimento de Etiquetagem nº 4

O desenvolvimento deste procedimento de etiquetagem *PE*4 surgiu da constatação de que as melhores soluções obtidas com os procedimentos de etiquetagem anteriores tendem a aproximar instalações que possuem fluxo entre elas, especialmente para o caso dos pares com maiores valores de fluxo. Para solucionar os problemas, o *PE*4 começa por coleccionar todos os pares de instalações com um valor de fluxo diferente de zero. De seguida ordena os pares de instalações por ordem decrescente do seu valor de fluxo. Finalmente, seguindo a sequência ordenada e começando pelo par com maior valor de fluxo, coloca na planta as instalações ainda não colocadas de cada par. O tamanho do domínio das variáveis relativas às coordenadas serve para desempate quando as duas instalações de um par ainda não estão colocadas na planta. O código de *PE*4 encontra-se representado na Figura 5-9.

```
etiq( {Φ}, λ ) ←
   colecciona_todos_pares( Pares ),
   ordenar_pares( Pares, Ord_Pares ),
   etiq( {Φ}, Ord_Pares, λ ).

etiq( _, [ ], _ ).
etiq( {Φ}, [ φ( Ii , Ij , _) | Pares ],λ ) ←
   Φ( Ii , r( Xi, Yi, Ci, Li, _ ) ),
   Φ( Ij , r( Xj, Yj, Cj, Lj, _ ) ),
   tamanho_dominio( Xi, TXi ),   tamanho_dominio( Yi, TYi ),
   tamanho_dominio( Xj, TXj ),   tamanho_dominio( Yj, TYj ),
   (
      TXi × TYi ≤ TXj × TYj ,
      !,
      etiq( Ci, Li, Xi, Yi, λ ),   etiq( Cj, Lj, Xj, Yj, λ ),
   ;
      etiq( Ci, Li, Xi, Yi, λ ),   etiq( Cj, Lj, Xj, Yj, λ ),
   ),
   etiq( T, Pares, λ ).

etiq( C, L, X, Y, λ ) ←
      nodominio( C, λ ),   nodominio( L, λ ),
      nodominio( X, λ ),   nodominio( Y, λ ).

colecciona_todos_pares( Pares ) ←
   findall( φ( Ii , Ij , Fij), φ( Ii , Ij , Fij), Pares ).
```

Figura 5-9: Código para o *PE*4.

Resta referir que a extensão do predicado *colecciona_todos_pares*/1 usado por *PE*4 colecciona todos os pares de instalações que possuem um valor de fluxo entre si não nulo. Todos estes pares são ordenados em função do valor de fluxo por intermédio da extensão do predicado *ordenar_pares*/2. Finalmente, a extensão do predicado *tamanho_dominio*/2 tem como finalidade a determinação do número de elementos presentes no domínio da variável de domínio do seu primeiro argumento.

Os resultados da utilização do *PE*4 para solucionar os casos de teste encontram-se representados na Tabela 5-7.

Tabela 5-7: Custo das soluções pelo uso do *PE*4.

Problema		**ppli8**	**ppli10**	**ppli10c**	**ppli15**	**ppli24**
Menor	Primeira	39519	29208	-	32996	124969
	Tempo	1,17	2,68	-	8,97	68,93
	Melhor	39443	28852	-	32237	124557
	Tempo	209,78	3582,56	-	1028,21	3472,36
Médio	Primeira	39263	28644	27604	35602	116091
	Tempo	1,67	5,39	3507,83	18,86	85,51
	Melhor	36035	**28564**	**27604**	35140	**115443**
	Tempo	3377,43	**3547,51**	**3507,83**	3286,49	**3552,51**
Maior	Primeira	35786	29439	-	31345	117301
	Tempo	1,12	4,25	-	16,66	72,23
	Melhor	**33942**	28592	-	**31345**	117187
	Tempo	**19,17**	2380,62	-	**16,66**	3587,51
Partição	Primeira	42858	31118	-	40288	128811
	Tempo	0,94	1,29	-	2,76	5,17
	Melhor	38211	29668	-	38873	128290
	Tempo	1599,33	2528,92	-	3355,73	1500,43

Procedimento de Etiquetagem nº 5

Com os procedimentos de etiquetagem anteriores, as melhorias conseguidas entre duas soluções consecutivas devem-se muitas vezes a uma ligeira deslocação de uma ou mais instalações na planta, o que até é natural, considerando a ordem com que as variáveis de decisão são instanciadas. Nestas circunstâncias uma ligeira

deslocação de uma instalação acaba por não introduzir uma melhoria significativa no custo da nova solução encontrada relativamente à precedente. Por outro lado, os procedimentos de etiquetagem acabam por perder bastante tempo na exploração de soluções vizinhas (pesquisa local) que apresentam na maior parte das situações um custo superior à última solução encontrada.

De modo a tentar evitar estas situações foi desenvolvido o procedimento de etiquetagem *PE*5 que apresenta algumas semelhanças com o *PE*2. O seu código encontra-se representado na Figura 5-10.

```
etiq( [ ], _ ).
etiq( {Φ}, λ ) ←
    remove( (_, r( X, Y, C, L, _ ) ), {Φ}, T ),
    etiq( C, L, X, Y, λ ),
    etiq( T, λ ).

etiq( C, L, X, Y, λ ) ←
    nodomínio( C, λ ),
    nodomínio( L, λ ),
    nodomínio( X, λ ),
    nodomínio( Y, λ ),    !.
```

Figura 5-10: Código para o *PE*5.

Tal como com o *PE*2, as instalações são colocadas na planta uma a uma. A colocação de uma instalação consiste em atribuir-lhe uma forma e uma posição na planta. A principal diferença relativamente a *PE*2 reside na forma como o retrocesso é efectuado. Em vez de serem esgotadas todas as posições e formas possíveis para uma dada instalação antes de se tentar uma colocação alternativa para a instalação precedente na ordem de selecção, o procedimento de etiquetagem *PE*5 tenta encontrar apenas uma posição e uma forma para a instalação. O retrocesso, neste caso, envolve a escolha de uma instalação alternativa na ordem de selecção de instalações.

O procedimento de etiquetagem *PE*5 não é, no entanto, um procedimento de etiquetagem completo, dado que o espaço de pesquisa não é explorado exaustivamente. Apesar de tudo, e pelos resultados obtidos na resolução dos diferentes casos de teste indicados na Tabela 5-8, acaba por fornecer soluções de melhor qualidade na maior parte das situações, quando comparados com os

procedimentos de etiquetagem anteriores. Recorde-se que o algoritmo de optimização é interrompido ao fim de um determinado período de tempo.

Tabela 5-8: Custo das soluções pelo uso do *PE*5.

Problema		**ppli8**	**ppli10**	**ppli10c**	**ppli15**	**ppli24**
Menor	Primeira	42712	40405	33964	30519	113239
	Tempo	1,08	2,64	2,21	8,9	48,1
	Melhor	**31377**	27571	**25926**	**29286**	112211
	Tempo	**3454,32**	2046,47	**2126,37**	**3426,63**	3598,52
Médio	Primeira	41768	28013	34644	34159	112276
	Tempo	1,39	4,81	4,21	18,66	64,11
	Melhor	34839	**25836**	33091	33510	**109486**
	Tempo	3570,0	**3444,11**	3295,38	3414,35	**3600,95**
Maior	Primeira	58483	35571	38462	33510	115241
	Tempo	1,39	6,51	5,96	3414,35	65,07
	Melhor	34814	33278	32223	32821	114638
	Tempo	3555,64	3299,14	3129,47	3560,93	3560,39
Partição	Primeira	44063	37746	45945	43310	137520
	Tempo	0,91	1,31	1,24	2,41	5,0
	Melhor	37504	33236	31251	39483	132157
	Tempo	3374,9	3526,27	3592,65	2133,01	417,57

Comparação dos Resultados

Pela análise da Tabela 5-9, que contém o custo da melhores soluções encontradas na resolução dos diferentes casos de teste, verifica-se que o procedimento de etiquetagem que forneceu melhores resultados foi o *PE*5. A excepção foi a resolução do caso de teste "ppli24". Relativamente à heurística para a ordenação de valores, não houve nenhuma que mostrasse ser melhor que as restantes. Verifica-se apenas que a heurística "partição" apresentou os piores resultados para todos os procedimentos de etiquetagem.

De acordo com a análise efectuada na secção 5.3.2, para a dimensão do espaço de soluções em função do número de instalações, a percentagem do espaço de soluções explorada é inferior para os casos de teste com mais instalações se

atendermos ao facto de que foi considerado o mesmo tempo de processamento. No entanto, o objectivo destas experiências foi, fundamentalmente, encontrar um procedimento de etiquetagem que para a maior parte dos casos forneça boas soluções o mais cedo possível sem ser necessário explorar todo o espaço de soluções, considerando heurísticas para definir a ordem de selecção das variáveis e de valores. Recorda-se que para os problemas reais não é praticável, em termos temporais e computacionais, a exploração de todo o espaço de soluções.

Tabela 5-9: Os melhores resultados das experiências realizadas.

Problema	**ppli8**	**ppli10**	**ppli10c**	**ppli15**	**ppli24**
Melhor Solução	31377	25836	25926	29286	109372
λ	Menor	Médio	Menor	Menor	Médio
PE	*PE*5	*PE*5	*PE*5	*PE*5	*PE*1

5.3.5 Resolução de Problemas com Restrições de Adjacência Adicionais

Pela análise das melhores soluções obtidas com as experiências descritas na secção anterior constatou-se que os pares de instalações que possuem entre si o maior valor de fluxo têm a tendência para serem colocadas na planta próximas entre si. Isto não é de estranhar dado que o custo das soluções depende não só do fluxo entre as instalações como da distância a que são colocadas. Este conhecimento foi já usado no procedimento *PE*4. No entanto, este conhecimento pode ser também usado para ajudar na resolução de problemas através da colocação de restrições de adjacência adicionais para alguns pares de instalações, especialmente aqueles com maior valor de fluxo.

Foram realizadas algumas experiências que mostraram ser possível obter melhores soluções em certas situações. Verificou-se que para encontrar boas soluções é necessário escolher os pares de instalações correctos, sob pena de se transformar o problema original num problema de difícil resolução ou mesmo sem soluções. Surge então a questão de como determinar quais são os melhores pares de instalações

adjacentes. Uma solução para esta questão passa pela resolução de um problema da teoria dos grafos que consiste na determinação da *Correspondência de Peso Máximo*[1] (CPM). A determinação da CPM envolve a construção de um grafo em que as instalações são representadas por nós. Dois nós são ligados por um arco se o valor de fluxo entre as respectivas instalações é diferente de zero. O peso associado com esse arco é o respectivo valor de fluxo. A determinação da CPM de um grafo consiste em encontrar o conjunto de pares de nós, ou seja, o conjunto de arcos que satisfaça as seguintes condições:

1. Um nó não pode estar ligado a mais do que um ramo;
2. O número de ramos deve ser o maior possível. Portanto, se for adicionado um outro ramo qualquer a condição anterior é quebrada;
3. O valor que resulta da soma dos pesos da combinação de ramos escolhidos deve ser máximo, ou seja, nenhuma outra combinação de ramos deve originar uma soma de pesos superior;
4. Tendo em conta o tipo problemas que se pretende solucionar, acrescenta-se ainda a condição de que nenhum ramo escolhido crie inconsistências com as restrições do problema a solucionar (como, por exemplo, as restrições de distância).

A Figura 5-11 mostra um grafo construído a partir do problema "ppli10" em que os arcos com uma linha mais grossa pertencem ao grafo resultante da determinação da CPM (*m*6-*m*8, *m*5-*m*10, *m*1-*m*7, *m*4-*m*9). Refira-se que a complexidade temporal de um algoritmo para determinar a CPM de um grafo com *n* nós é de $O(n^3)$ (Lengauer, 1990).

Para explorar o espaço de soluções foi usado um procedimento de etiquetagem (*PE*6) semelhante ao *PE*4 que para colocar as instalações na planta considera uma ordenação decrescente dos valores de fluxo entre pares de instalações. Neste caso, a ordenação dos pares de instalações considera em primeiro lugar todos os pares que estão envolvidos numa restrição de adjacência e só depois os restantes. Os resultados

[1] *Maximum Weight Matching*.

obtidos na resolução dos casos de teste com o *PE*6 são apresentados na Tabela 5-10.

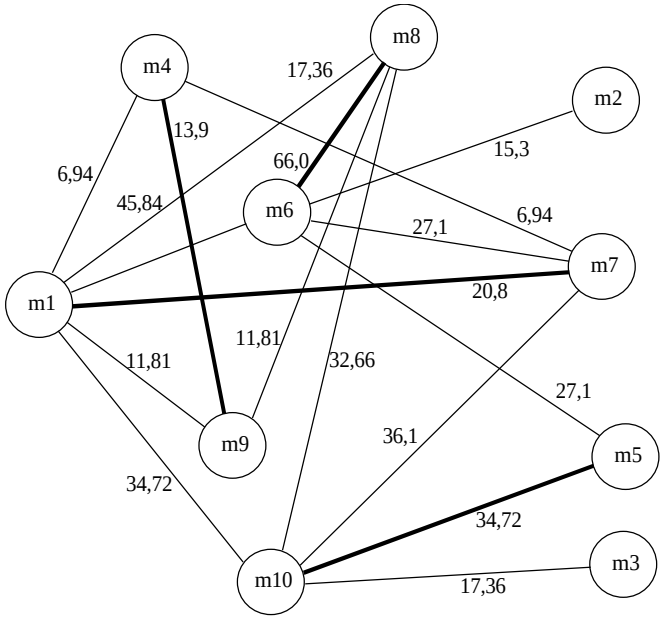

Figura 5-11: A correspondência de peso máximo de um grafo.

Tabela 5-10: Custo das soluções pelo uso do *PE*6.

Problema		**ppli8**	**ppli10**	**ppli10c**	**ppli15**	**ppli24**
Menor	Primeira	27679	24402	23965	31557	114601
	Tempo	0,83	2,62	1,22	10,94	88,24
	Melhor	24079	24177	**22985**	31442	**114401**
	Tempo	542,53	877,02	**1056,25**	856,49	**1688,49**
Médio	Primeira	34239	24563	-	29565	-
	Tempo	1,17	9,74	-	32,56	-
	Melhor	25239	23766	-	**29361**	-
	Tempo	246,23	867,79	-	**498,86**	-
Maior	Primeira	34479	31090	25744	31416	115194
	Tempo	1,26	3,46	8,31	33,09	91,09
	Melhor	31599	29583	25727	31065	115085
	Tempo	2,09	223,11	160,50	926,95	2267,78
Partição	Primeira	30134	25533	28720	29556	119227
	Tempo	0,98	2,06	136,26	2,34	5,92
	Melhor	**22784**	**23422**	28719	29394	116635
	Tempo	**824,74**	**905,27**	734,93	724,32	3367,25

Comparando os resultados obtidos usando o *PE*6 com os restantes procedimentos de etiquetagem verifica-se que existe um incremento na qualidade das soluções maior para os casos de teste de menores dimensões e resultados da mesma ordem de grandeza para os restantes casos de teste. Uma possível explicação para este facto pode ser o argumento já apresentado anteriormente e que consiste no facto de se dedicar um tempo de processamento idêntico para a resolução de problemas de diferente dimensão. Embora este argumento possa explicar em parte os resultados obtidos, existe um outro factor que potencialmente contribui para o sucedido e que se relaciona com a razão entre o valor fluxo total das instalações envolvidas em restrições de adjacência e o valor de fluxo total de todos os pares de instalações. Quanto maior for esta razão maior é a possibilidade de se encontrarem melhores soluções com um melhor desempenho. Na Tabela 5-11 observa-se que para os casos de teste em que esta razão é superior a qualidade das soluções foi também melhor.

Tabela 5-11: Relação entre o valor de fluxo total para as instalações envolvidas em restrições de adjacência e o valor de fluxo total de todos os pares possíveis de instalações para os diferentes casos de teste.

Problema		**ppli8**	**ppli10**	**ppli10c**	**ppli15**	**ppli24**
Nº Pares	Adjacentes	4	4		7	12
	Total	11	17		75	129
% Pares Adjacentes		36,36 %	23,53 %		9,33 %	9,30 %
% Fluxo	Adjacentes	41,01 %	31,75 %		20,20 %	11,45 %

Como conclusão final, e atendendo à dimensão destes problemas, refira-se que não é praticável para os casos reais a exploração completa do espaço de soluções. É necessário recorrer a heurísticas que permitam conduzir os algoritmos de optimização o mais depressa possível para boas soluções e ao mesmo tempo minimizar a quantidade das soluções exploradas. O *PE*5 e o *PE*6 são procedimentos de etiquetagem que permitem uma exploração incompleta do espaço de soluções e que em geral podem dar soluções melhores.

Deve-se notar que a velocidade com que estes procedimentos de etiquetagem estudados encontram boas soluções, relativamente à forma como a ordem de selecção das variáveis e valores é efectuada, depende também, muitas vezes, do problema que está a ser solucionado. Um exemplo desta situação relaciona-se com os

resultados obtidos com o *PE*6 considerando as diferentes heurísticas de ordem de selecção de valores. A informação representada na Tabela 5-9 é outro exemplo que reforça esta situação de dependência.

O que realmente se pode reter destas experiências é que a optimização destes problemas requer um bom algoritmo capaz de realizar uma amostragem de boas soluções em todo o espaço de soluções ou então um algoritmo que seja capaz de encontrar boas soluções sem que seja necessário explorar todo o espaço de soluções. Este assunto será tratado nos capítulos seguintes com descrição de uma abordagem para a utilização de algoritmos genéticos em tarefas de optimização.