OTIMIZAÇÃO EM PROBLEMAS DE LEIAUTE

João Carlos Furtado
LAC/INPE - Instituto Nacional de Pesquisas Espaciais
Av. dos astronautas 1758 - São José dos campos - SP
LACESM/CT - UFSM - Universidade Federal de Santa Maria
Campus Universitário - Santa Maria - RS

Luiz Antônio Nogueira Lorena
LAC/INPE- Instituto Nacional de Pesquisas Espaciais
Av. dos Astronautas 1758 - São José dos Campos - SP

1 - INTRODUÇÃO

O problema de leiaute é um problema antigo e após a sua apresentação por Armour e Buffa (1963) e o advento dos computadores, recebeu considerável atenção de muitos pesquisadores nas últimas quatro décadas. Ao longo deste tempo, por várias razões, diferentes formulações para o problema foram apresentadas na literatura. Dentre estas formulações, as de programação quadrática e teoria de grafos são as que mais se destacaram. Além disso, formulações mais recentes, usando como estrutura de dados uma árvore binária, têm demonstrado grande capacidade de envolver restrições mais abrangentes, bem como obter bons resultados.

O problema de leiaute foi modelado como sendo um problema de programação quadrática por Koopmans e Beckman (1957), ocorrendo refinamentos posteriores (Heragu e Kusiak - 1987). Muitos trabalhos seguiram esta modelagem (Bozer e Meller - 1993; Skorin-Kapov-1990) e até hoje ainda é uma das principais abordagens usadas.

Em 1976, Foulds e Robinson criaram uma nova formulação baseada na teoria de grafos. Nesta abordagem, um grafo é formado, onde cada vértice representa uma facilidade e cada aresta representa a possibilidade de atribuição de facilidades adjacentes.

Uma modelagem mais recente foi apresentada por Tam (1992). Nesta abordagem, uma árvore binária é usada como estrutura de dados e um algoritmo genético usado no processo de busca de soluções. Esta modelagem também foi usada por Furtado e Lorena (1997), mas com significativas alterações no processo de busca, usando-se o método heurístico de busca tabu (Glover, 1989a, 1989b, 1990).

Existem diferentes situações práticas que geram a necessidade de se construir leiautes eficientes. Por exemplo, o leiaute de componentes em placas de circuitos eletrônicos, de máquinas numa indústria, salas de um escritório, hospitais, instituições governamentais, aeroportos, etc. Assim, alguns termos podem assumir significados diferentes, como por exemplo, o termo facilidade (do inglês “facility”) que num leiaute de um escritório é atribuído a cada sala que irá compor este escritório. Já

num leiaute industrial, facilidade pode significar o prédio todo da indústria. Também os objetivos variam conforme a aplicação do problema. Desta forma, num leiaute de placa de circuito eletrônico geralmente deseja-se minimizar a área ocupada pelos componentes eletrônicos, o que não é preocupação em outras situações.

A importância do estudo do problema de leiaute apresenta dois aspectos: econômico e científico. Sob a ótica econômica, um leiaute eficiente numa indústria pode obter considerável redução nos custos de produção. A dimensão do investimento em novas facilidades nas indústrias e outras instituições, a cada ano, incentiva a busca de novas alternativas ao problema. Além disso, uma porcentagem significativa das facilidades construídas são modificadas anualmente e requerem um replanejamento. A reorganização do leiaute de facilidades precisa ser uma atividade constante em qualquer organização que pretenda ser competitiva e eficiente em sua área de atuação, devido principalmente a evolução tecnológica que produz novas máquinas e equipamentos, tornando modelos e métodos obsoletos. O problema também é muito estudado no leiaute de componentes eletrônicos, i.e., módulos retangulares de um circuito eletrônico que precisam ser distribuídos sobre uma superfície de uma placa e conectados entre si.

Sob a ótica científica, o problema de leiaute é interessante pois é um problema de otimização combinatória de difícil solução para problemas com grandes dimensões. O problema de programação quadrática, que é uma das principais abordagens, foi demonstrado ser NP-completo por Sahni e Gonzalez (1976). Vários métodos heurísticos tem sido propostos para obtenção de soluções satisfatórias.

Na tentativa de se obter soluções ótimas, foram propostos algoritmos "branch and bound" (Bazaraa e Elshafei-1979). Este tipo de algoritmos não consegue obter soluções ótimas para problemas de grandes dimensões, bem como o tempo e memória necessária para obtenção das soluções é muito grande. Assim, as pesquisas passaram a concentrar-se na busca de soluções sub-ótimas através de métodos heurísticos.

Hoje, temos diferentes tipos de algoritmos, que podem ser classificados em:

- algoritmos construtivos;
- algoritmos de melhoramento;
- algoritmos híbridos;
- algoritmos da teoria de grafos.

Apresentaremos na seção 2 conceitos e definições básicas sobre o problema de leiaute. Na seção 3 são descritas as diferentes abordagens de formulação do problema encontradas na literatura, e na seção 4, os métodos usados para obter as soluções. Já na seção 5, algumas considerações finais são apresentadas.

2 - DEFINIÇÕES E CONCEITOS BÁSICOS

Conforme comentado na seção 1, existem diferentes situações práticas que exigem a otimização de leiautes. No entanto, o problema é mais frequentemente estudado em leiautes de componentes numa placa de circuito eletrônico e em leiautes de facilidades, i.e., leiaute de salas num escritório, máquinas numa indústria, etc. Uma vez que este problema é bastante estudado nos ambientes industriais, é interessante mostrar

como ele se insere neste contexto. Desta forma, vamos inicialmente estudar o planejamento de facilidades na indústria, em seguida, veremos o leiaute de componentes num circuito eletrônico.

2.1 - O PLANEJAMENTO DE FACILIDADES

O problema de leiaute é parte importante de todo o planejamento de facilidades na indústria.

Precisamos esclarecer que o termo facilidade representa a organização industrial, a qual é dividida em departamentos. Entretanto, frequentemente o termo também pode significar cada departamento individualmente. Neste trabalho usaremos as duas formas, de acordo com a conveniência.

No planejamento de facilidades, termos como: localização de facilidades, projeto de facilidades, leiaute de facilidades, não devem ser tratados como sinônimos. Para distinguir entre planejamento de facilidades e esses termos correlatos, é conveniente analisar uma facilidade em termos da sua localização e seus componentes.

A **localização** da facilidade refere-se ao local em relação aos clientes, fornecedores, e outras facilidades com a qual interage. Também, a localização inclui o local e orientação sobre o solo.

Os **componentes** da facilidade consistem da estrutura, do leiaute e o sistema de tráfego (manejo). A estrutura consiste da construção e serviços; o leiaute consiste da disposição de todos equipamentos, máquinas, e mobília dentro da estrutura; o sistema de tráfego consiste de mecanismos através dos quais todas interações exigidas pelo leiaute são satisfeitas.

A **estrutura** para uma facilidade de manufatura inclui a construção (prédio) e serviços como: água, gás, energia elétrica, ar e luz. O leiaute consiste do arranjo de áreas de produção, áreas de suporte, e áreas pessoais dentro da construção. O sistema de tráfego consiste de material, pessoas, informações, e sistema de manejo de equipamentos exigidos pelo leiaute.

Determinar como a **localização** de uma atividade (industrial) influi na obtenção dos objetivos dessa atividade é conhecido como **localização de facilidades.** A determinação de como os componentes de uma atividade influem nos objetivos dessa atividade é conhecido como **projeto de facilidades**. Portanto, planejamento de facilidades pode ser subdividido em: localização de facilidades e projeto de facilidades.

Planejamento de facilidades pode ainda ser subdividido através da separação de projeto de facilidades em três componentes da facilidade: a estrutura, leiaute e sistema de tráfego. A figura 2.1 apresenta a hierarquia de planejamento de facilidades.

Portanto, o problema/projeto de leiaute é parte importante do planejamento de facilidades. Assim, quando um leiaute é produzido numa organização industrial, deve-se considerar todo o contexto no qual esta inserido.

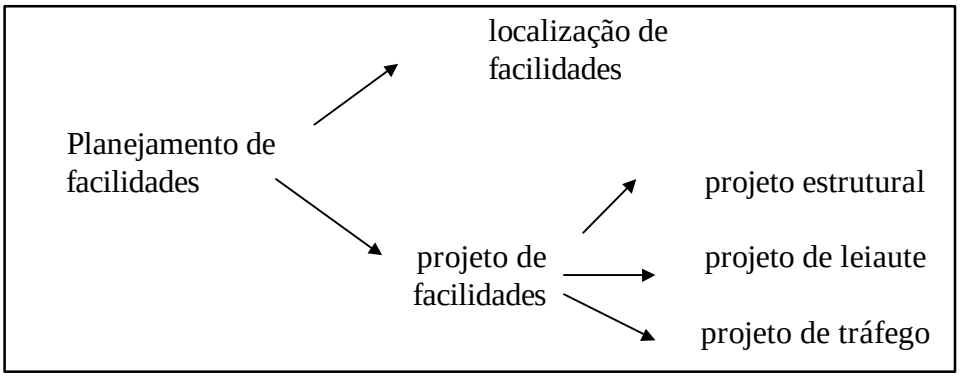

Figura 2.1: Esquema de como o planejamento de facilidades é constituído.

2.2 - O RELACIONAMENTO ENTRE FACILIDADES

Em leiaute de facilidades, teremos diferentes tipos de relacionamento entre departamentos, dependendo da natureza da atividade desenvolvida. Assim, num sistema de fluxo de material (tráfego) numa indústria podem ocorrer diferentes situações, de acordo com esta natureza. Temos quatro tipos básicos de disposição de departamentos numa indústria:

- departamentos de linha de produção;
- departamentos com localização fixa de material;
- departamentos com família de produção;
- departamentos de processos de produção.

Esta disposição irá influenciar diretamente o tráfego entre as facilidades. Para calcular leiautes alternativos, uma medida de tráfego precisa ser estabelecida. O tráfego pode ser especificado de uma forma quantitativa ou qualitativa. Medidas quantitativas podem incluir peças por hora, movimentos por dia, número de relatórios. Medidas qualitativas variam da absoluta necessidade que dois departamentos estejam próximos um ao outro à preferência que dois departamentos estejam afastados. Frequentemente as medidas quantitativas e qualitativas devem ser consideradas em conjunto.

Em leiaute de componentes eletrônicos, o relacionamento será determinado pela necessidade de conexões entre os componentes. Como o objetivo do leiaute é minimizar o comprimento total das conexões, componentes com grande fluxo de informações entre si, devem permanecer mais próximas no leiaute, enquanto componentes que não interagem podem permanecer fastados.

2.3 - DISTÂNCIA

Um dos elementos básicos para a formulação de problemas de leiaute é o conceito de distância. A distância retangular entre dois pontos $a_i = (a_{i1},a_{i2})$ e $a_j = (a_{j1},a_{j2})$ é dado por

$$d_1(a_i,a_j) = [\ |\ a_{i1} - a_{j1}|\ +\ |\ a_{i2} - a_{j2}|\ ] \qquad (2.1)$$

E a distância euclidiana é dada por

$$d_2(a_i,a_j) = [\ (a_{i1} - a_{j1})^2 + (a_{i2} - a_{j2})^2\ ]^{1/2} \qquad (2.2)$$

No caso de leiaute de departamentos, a distância centro-a-centro (euclidiana ou retangular) é usada para estimar a distância esperada entre dois departamentos, quando os pontos de entrada e saída (E/S) dos departamentos não são conhecidos. O número e a localização dos pontos de E/S, assim como o tráfego associado a cada ponto de E/S, são principalmente ditados pelos detalhes do leiaute, os quais são tipicamente desenvolvidos após o leiaute final ser obtido, através de métodos de otimização.

Na ausência de um leiaute detalhado, a justificativa para se usar o centro da facilidade como um ponto de E/S é que a entrada (saída) de tráfego é igualmente possível em qualquer ponto no interior da facilidade. Como exemplo de um problema gerado ao se utilizar a medida centro-a-centro, considere o caso em que uma facilidade tenha o formato L, tal que seu centro de gravidade caia fora da mesma. Neste caso o tráfego não entrará, nem sairá da mesma.

2.4 - ÁREAS DAS FACILIDADES

Ao se projetar um leiaute, as facilidades podem apresentar diferentes exigências quanto a área, dependendo da aplicação e objetivo do leiaute proposto. Assim, no problema de leiaute de componentes eletrônicos numa placa (PCBs-printed-circuit boards e ICs), além de objetivar minimizar o comprimento total das conexões do circuito, geralmente busca-se minimizar a área total ocupada pelo leiaute (Quinn e Breuer,1979; Goto e Kuh, 1978; Yeap e Sarrafzadeh,1993).

No entanto, existem situações que o espaço destinado ao leiaute é previamente definido e as facilidades devem ser arranjadas no interior deste espaço.

Recentemente, foram apresentados (Tam-1992 e Furtado e Lorena-1997) estudos que também usam um espaço previamente determinado e destinado ao leiaute, mas que incluem regiões proibidas de serem utilizadas no interior deste espaço, conforme figura 2.2, que, em casos práticos representariam, pilares, escadas ou elevadores no interior de um prédio.

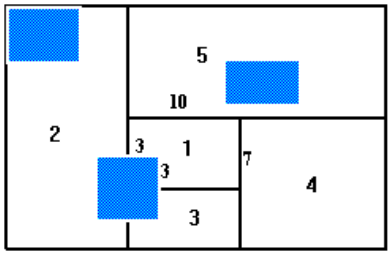

Figura 2.2: Leiaute com espaços ocupados.

2.5 - LEIAUTE DE COMPONENTES ELETRÔNICOS EM PCBs E ICs

Na sua forma mais simples, um problema de leiaute requer que objetos similares sejam atribuídos a posições fixas num plano, de tal forma que a medida da distância total das conexões do sistema inteiro seja minimizada. Isto pode ser facilmente visualizado como um número de componentes eletrônicos que ocupam posições particulares sobre uma placa e as conexões entre esses componentes. Numa abordagem mais geral, os componentes poderiam apresentar formas diferentes e não precisariam ser atribuídos a posições fixas, além de objetivos adicionais poderiam ser desejados, como por exemplo, minimizar a área total da superfície do circuito e a determinação de um leiaute que não apresente excessivo congestionamento de conexões bem como evitar que conexões se sobreponham.

O problema descrito ocorre em projetos de circuito, no entanto, o mesmo problema pode ser encontrado em uma grande diversidade de aplicações, onde se possa fazer analogia com componentes e conexões.

3 - MODELAGENS DO PROBLEMA DE LEIAUTE

O problema de leiaute de facilidades tem sido modelado de diferentes maneiras, entre as quais se destacam:

1. Programação quadrática;
2. Teoria de grafos;
3. Formulação usando estrutura de árvore binária.

Vamos estudar estas três modelagens, sem ter a pretensão de ser um estudo conclusivo, uma vez que existem outras possibilidades de modelagens.

3.1 - PROBLEMA DE LEIAUTE COMO SENDO UM PROBLEMA DE PROGRAMAÇÃO QUADRÁTICA

O problema de programação quadrática é um problema combinatorial que é de grande interesse e pode ser formalizado da seguinte forma:

$$Minimizar \sum_{i=1}^{m} \sum_{k=1}^{m} \sum_{j=1}^{m} \sum_{l=1}^{m} T_{ik} d_{jl} x_{ij} \; x_{kl} \qquad (3.1)$$

sujeito a

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \textit{para } j=1, \ldots, m$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \textit{para } I = 1, \ldots, m$$

x_{ij} *= 0 ou 1, para todos i, j.*

onde T_{ik} é o peso não-negativo que multiplica a distância entre as facilidades i e k; d_{jl} é a distância entre as localizações j e l; e

$x_{ij} =$ *1* se a facilidade *i* é atribuída a localização *j*,
$x_{ij} =$ *0* caso contrário.

O problema pode ser interpretado da seguinte maneira. Suponhamos que *m* facilidades ou objetos precisam ser atribuídos a *m* localizações. Cada localização tem uma certa quantidade de espaço que será ocupada pela facilidade atribuída a ela. Para cada par de facilidades um peso não-negativo é associado representado o nível de atividade entre estas facilidades (tráfego). O problema consiste em atribuir cada facilidade a uma localização diferente de todas outras facilidades de tal maneira a minimizar a soma dos pesos vezes a distância entre as facilidades.

Cada termo da soma quádrupla consiste da atribuição de um par de facilidades a um par de localizações. O produto do tráfego-x-distância $T_{ik}\ d_{jl}$ é o componente da função objetivo resultante do tráfego entre a facilidade *i* e a facilidade *k* e entre as localizações *j* e *l*. O primeiro conjunto de restrições exige que cada localização tenha uma única facilidade associada. O segundo conjunto de restrições assegura que cada facilidade é atribuída a uma única localização.

Podem ocorrer situações em que existam mais localizações do que facilidades ou cada facilidade poderia exigir diferentes quantidades de espaço. No primeiro caso, introduzem-se facilidades artificiais na formulação e no segundo caso, dividem-se as facilidades em sub-facilidades, cada uma, requerendo uma quantidade padrão de espaço e usando altos valores artificiais de tráfego entre estas sub-facilidades, de forma a permanecerem unidas.

3.2 - PROBLEMA DE LEIAUTE COMO UM PROBLEMA DA TEORIA DE GRAFOS

Na formulação em termos da teoria de grafos é assumido que a localização desejável de cada par de facilidades adjacentes seja conhecida “a priori”. Sejam:
$G = (V, E)$ - grafo com conjunto V não vazio de vertices (facilidades), e E conjunto de arestas;
T_{ij} - razão de aproximação, indicando a medida desejável de localizar a facilidade *i* adjacente à facilidade *j* (tráfego)*;*
V - conjunto de facilidades;
N - conjunto de pares de facilidades que necessitam estar adjacentes em qualquer possível solução;
F - conjunto de pares de facilidades que não necessitam estar adjacentes em qualquer possível solução; e

$E' = \{ \{ i, j\} : x_{ij} = 1, \{i, j\}\ \hat{I}\ E \},$
$x_{ij} =$ *1* se a facilidade *i* é adjacente à facilidade *j*,
$x_{ij} =$ 0 caso contrário.
Então a formulação na teoria de grafos é:

$$Maximizar \sum_{i \in E} \sum_{j \in E} T_{ij} x_{ij}, \qquad (3.2)$$

sujeito a $x_{ij} = 1, \{i, j\} \hat{I} N$,

$x_{ij} = 0, \{i, j\} \hat{I} F$,

$(V, E' \cup N)$ é um grafo planar.

Um grafo planar é aquele que pode ser mapeado no plano, sem que quaisquer duas arestas se intercessionem.

3.3 - PROBLEMA COM ESTRUTURA DE ÁRVORE BINÁRIA

Tam (1992) e Furtado e Lorena (1997) modelaram o problema de leiaute de forma a permitir ampliar as restrições e aproximar o problema da realidade.

Nesta formulação, inicialmente é produzida uma árvore binária. A árvore binária é produzida baseada no tráfego entre as facilidades, de forma que facilidades com alto tráfego entre si, fiquem próximas na árvore. A partir da árvore binária é possível construir um leiaute, conforme explicação a seguir.

Dispondo da árvore binária e das dimensões da região onde as facilidades serão introduzidas, inicia-se o processo de particionamento desta região. Para isso, percorre-se recursivamente a árvore binária a partir da raiz e verificam-se as áreas correspondentes na região de particionamento. Por exemplo, na figura 3.1, inicia-se na raiz (nó -5) e calcula-se o somatório das áreas das facilidades que estão no ramo esquerdo (as áreas dos nós 1, 2 e 3). O valor obtido é usado na região de particionamento, onde varre-se a região até obter a área correspondente, figura 3.2. Neste momento um particionamento é realizado. Em seguida, continua-se a percorrer a árvore binária (seguindo para o nó -4) e novamente a área do ramo esquerdo é calculada (área do nó 1) e traduzida em particionamento na região de particionamento (é colocada a identificação da região particionada, ou seja, facilidade número 1). O processo continua até toda árvore binária ser percorrida, o que corresponde, no final do processo, a um leiaute na região de particionamento. A cada particionamento é necessário decidir se o particionamento será horizontal ou vertical. O critério é o seguinte: verificamos a distância entre os limites inferior e superior da região de particionamento para os cortes vertical e horizontal no local considerado naquele instante a ser cortado. Se a distância vertical é menor do que a horizontal realizamos um corte vertical, caso contrário, o corte será horizontal. Usamos este critério por acreditar que desta maneira produziremos leiautes que violem menos as limitações da razão de aspecto máxima e mínima.

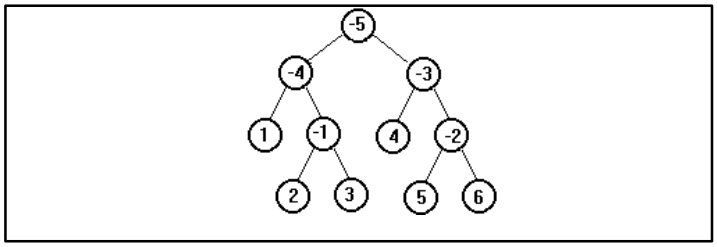

Figura 3.1:Árvore binária

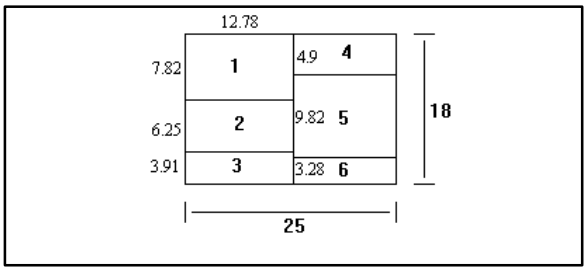

Figura 3.2: Leiaute produzido pela processo de particionamento

A forma da facilidade é descrita por dois parâmetros: razão de aspecto e razão de "área morta". A razão de aspecto a_i da facilidade é definida como:

$$a_i = \frac{\text{Altura da partição alocada à facilidade } i}{\text{Largura da partição alocada à facilidade } i} \tag{3.3}$$

Podemos impor restrições na forma da facilidade, restringindo sua razão de aspecto a $[a_{i(min)}, a_{i(max)}]$, onde $a_{i(min)}$ e $a_{i(max)}$ são os limites inferior e superior da razão de aspecto a_i. Nós podemos também impor restrições na orientação da facilidade, classificando facilidades em duas categorias: facilidades de orientação livre e facilidades com orientação fixa. A facilidade com orientação livre permite orientação vertical ou horizontal. Para estas facilidades, sua razão de aspecto pode ser a_i ou $1/a_i$. Então, o possível intervalo da razão de aspecto torna-se:

$$\text{intervalo da razão de aspecto do facilidade } i = \left[min\left\{ a_{i(min)}, 1/a_{i(max)} \right\}, max\left\{ a_{i(max)}, 1/a_{i(min)} \right\} \right] \tag{3.4}$$

A facilidade com orientação fixa, permite somente orientação vertical (ou horizontal), e seu intervalo permitido para razão de aspecto é simplificado:

$$\text{intervalo da razão de aspecto do facilidade } i = \left[a_{i(min)}, a_{i(max)} \right] \tag{3.5}$$

Os intervalos das razões de aspecto dependem das restrições impostas pelo tipo de trabalho a ser realizado por cada facilidade. Por exemplo, num ambiente industrial, cada facilidade poderia representar uma sala de produção em que sua forma ainda esteja sendo definida pela disposição das máquinas que nela devem operar.

Freqüentemente, existem regiões no leiaute, que não podem ser usadas para alocar integralmente as facilidades, porque já estão parcialmente ocupadas, com construções, tais como elevadores, escadas, pilares, *etc.* Estas áreas ocupadas precisam ser subtraídas da área a ser alocada à facilidade e distorcerão a forma retangular da área usável. Portanto, é necessário providenciar uma medida que reflita o grau de distorção da forma. A razão de "área morta" é introduzida com este propósito.

A razão de "área morta" da facilidade i denotada por s_i é definida como

$$s_i = \frac{\text{área total do espaço ocupado na partição alocada a facilidade } i}{\text{área da partição alocada a facilidade } i} \qquad (3.6)$$

Se o leiaute estiver livre de áreas ocupadas, a razão de "área morta" de todas facilidades será zero. Para cada facilidade *i*, podemos restringir o valor de s_i a cair dentro de $[0, s_{i(max)}]$, onde , $s_{i(max)}$ é o limite superior de s_i..

O objetivo é alocar espaço para as facilidades de tal maneira que facilidades com grande volume de tráfego estejam próximas entre si, ao mesmo tempo em que satisfazem individualmente às restrições de área e forma. Dada uma árvore de particionamento s ∈ S, o problema de alocação de espaço pode ser formulado como:

$$min\ F = \sum_{i=1}^{n}\sum_{j=1}^{n} T_{ij}\, d_{ij}\ , \qquad s\ \hat{I}\ S \qquad (3.7)$$

sujeito a: $a_{i(min)} \pounds a_i \pounds a_{i(max)}$

$0 \pounds s_i \pounds s_{i(max)}$

$i, j = 1, 2, \ldots , n.$

onde

T_{ij} = Volume de tráfego entre as facilidades *i* e *j*.

d_{ij} = Distância retangular entre o centro da partição alocada às facilidades *i* e *j*.

a_i = Razão de aspecto da partição alocada a facilidade *i*.

$a_{i(min)}$ = Limite inferior de a_i.

$a_{i(max)}$ = Limite superior de a_i.

$s_{i(max)}$ = Limite superior de s_i.

Na formulação, as restrições são convertidas em funções de penalidades. Desta forma, temos:

$$min\ F = \sum_{i=1}^{n}\sum_{j=1}^{n} T_{ij}\, d_{ij} + \sum_{k=1}^{n} (w^a{}_k\, a_k + w^b{}_k\, b_k)\ , \quad s\ \hat{I}\ S \qquad (3.8)$$

onde:

$a_k = max\ \{ 0, max\{(a_k - max\{a_{k(max)}, 1/a_{k(min)}\}), (min\{a_{k(min)}, 1/a_{k(max)}\} - a_k)\}\},$

$b_k = max\ \{ 0,\ s_k - s_{k(max)}\},$

$w^a{}_k, w^b{}_k\ ³\ 0,$

$i, j, k = 1, 2, \ldots., n.$

O primeiro termo de *F* mede o volume de tráfego. É definido como o produto da distância retangular d_{ij} entre duas facilidades e o volume de tráfego T_{ij}. O segundo termo representa uma função de penalidades para restrições geométricas. O primeiro termo da função de penalidades mede quanto a razão de aspecto é violada. Uma vez que cada facilidade tem um único nível de tolerância para a violação destas restrições, a_k é multiplicada por um fator $w_k{}^a$.

Igualmente, o segundo termo b_k penaliza a existência de espaços ocupados dentro da facilidade. Pesos individuais $w_k^b \geq 0$ podem ser atribuídos às facilidades para refletir seus níveis de tolerância de irregularidade de forma.

4 - ALGORITMOS PARA O PROBLEMA DE LEIAUTE

Neste capítulo vamos apresentar os principais algoritmos desenvolvidos na tentativa de obter soluções para o problema de leiaute.

Nas primeiras tentativas para se obter a solução do problema, buscou-se obter soluções ótimas, principalmente através de algoritmos "branch-and-bound". Este tipo de algoritmo não consegue obter soluções ótimas para problemas de grandes dimensões, bem como o tempo e memória necessária para obtenção das soluções é muito grande. Assim, as pesquisas passaram a concentrar-se na busca de soluções sub-ótimas através de métodos heurísticos.

Estes algoritmos heurísticos podem ser divididos em:

- algoritmos construtivos;
- algoritmos de melhoramento;
- algoritmos híbridos;
- algoritmos baseados na teoria de grafos.

Vamos então, estudar inicialmente os algoritmos para obter soluções ótimas e, em seguida, os que buscam soluções sub-ótimas.

4.1- ALGORITMOS PARA OBTENÇÃO DE SOLUÇÃO ÓTIMA

Como vimos no capítulo 3, a maioria das modelagens do problema de leiaute são formulações de programação quadrática. Bazaraa e Elshafei (1979) procuraram resolver o problema de programação quadrática usando o algoritmo "branch-and-bound". O algoritmo "branch-and-bound" para o problema de leiaute de facilidades apresenta um leiaute parcial *P* que está disponível a cada estágio. Um limite inferior *LB* do custo de todos possíveis complementos do leiaute parcial *P* é determinado. Se *LB* é menor que o custo do melhor leiaute disponível C^o, o algoritmo prossegue com a atribuição de uma nova facilidade e portanto incrementa o tamanho /*P*/ do leiaute parcial. Caso contrário, a busca nesta direção é terminada e a última atribuição é proibida e uma nova atribuição é procurada. A busca continua usando o procedimento acima até que um leiaute completo seja obtido.

4.2 - ALGORITMOS PARA OBTENÇÃO DE SOLUÇÕES SUB-ÓTIMAS

Os algoritmos para obtenção de soluções ótimas apresentam as seguintes desvantagens:

- necessidade de grande memória e tempo computacional para obtenção da solução;
- o maior problema resolvido otimamente apresenta 19 facilidades (Heragu, 1992).

Isto tem levado as pesquisas a concentrarem-se em heurísticas para resolver o problema de leiaute de facilidades.

4.2.1 - ALGORITMOS CONSTRUTIVOS

Nos algoritmos construtivos as facilidades são atribuídas a uma localização, uma de cada vez, até que um leiaute completo seja obtido.

Vamos estudar os principais algoritmos construtivos.

CORELAP

CORELAP - "COmputerized RElationship LAyout Planning" (Lee e Moore-1967) constrói um leiaute calculando a média total de proximidade (MTP) de cada facilidade, onde MTP é a soma dos valores numéricos atribuídos aos relacionamentos (A=6, E=5, I=4, O=3, U=2, X=1) entre a facilidade e todas outras facilidades. A facilidade tendo a maior MTP é então atribuído ao centro da região de leiaute. Se existir um empate na verificação da maior MTP a facilidade com maior área é usada. Em seguida a tabela de relacionamento é varrida e se uma facilidade é encontrada tendo relacionamento ***A*** com a facilidade selecionada, esta é levada ao leiaute. Se não existe, a tabela de relacionamento é varrida procurando um relacionamento ***E***, em seguida ***I***, e assim por diante. Se duas ou mais facilidades são encontradas tendo o mesmo relacionamento com a facilidade selecionada, então a facilidade tendo a maior MTP é selecionada. A terceira facilidade a compor o leiaute é determinada varrendo a tabela de relacionamento para ver se existe uma facilidade ainda não atribuída que possui relacionamento ***A*** com a primeira facilidade selecionada. Se existir, esta facilidade é levada ao leiaute. Se existir empate, o MTP é usado, conforme anteriormente. Se não existe nenhuma facilidade com relacionamento ***A*** com a segunda facilidade, o procedimento é repetido considerando relacionamento ***E***, então ***I***, e assim por diante. O mesmo procedimento é repetido para a quarta facilidade, exceto pelo fato de que as três primeiras facilidades não mais entram na busca. Assim, o procedimento continua até que todas facilidades sejam selecionadas a compor o leiaute.

ALDEP

ALDEP - "Automated Layout DEsign Program" foi apresentado por Seehof e Evans (1967) e possui os mesmos dados básicos de entrada e objetivos como CORELAP. A diferença básica entre CORELAP e ALDEP é que CORELAP seleciona a primeira facilidade a entrar no leiaute e resolve empates usando a medida MTP, enquanto ALDEP seleciona a primeira facilidade e resolve empates aleatoriamente. A diferença filosófica básica entre CORELAP e ALDEP é que CORELAP procura produzir um único melhor leiaute, enquanto ALDEP produz muitos leiautes e deixa ao projetista escolher o melhor.

Como dito anteriormente, a primeira facilidade selecionada por ALDEP para compor o leiaute é selecionada aleatoriamente. A tabela de relacionamento é então varrida para determinar se existe uma facilidade com relacionamento ***A*** com a facilidade selecionada aleatoriamente. Se existe uma facilidade, ela é selecionada a compor o leiaute. Se existem mais de uma facilidade, uma delas é selecionada aleatoriamente. Se não existe nenhuma facilidade com um grau mínimo de relacionamento especificado pelo usuário, a segunda facilidade a compor o leiaute será escolhida aleatoriamente. Uma vez

que a segunda facilidade seja selecionada, o procedimento de seleção continuará até todas as facilidades terem sido selecionadas para compor o leiaute.

ALDEP inicia alocando a primeira facilidade no lado esquerdo da região onde o leiaute será construído (figura 4.1). Cada facilidade adicional será acrescentada à direita da última facilidade atribuída à região do leiaute, conforme esquema da figura 4.1.

Quando todas as facilidades forem selecionadas e atribuídas à região do leiaute, ALDEP avalia a qualidade do leiaute atribuindo valores aos relacionamentos entre facilidades adjacentes. Se uma facilidade é adjacente à outra facilidade com relacionamento ***A***, um valor 64 é atribuído à medida da qualidade. Um relacionamento ***E*** adiciona 16, ***I*** adiciona 4, e um relacionamento ***O*** adiciona 1 à qualidade do leiaute. Um relacionamento ***U*** não adiciona nada e se duas facilidades estão adjacentes e possuem um relacionamento ***X,*** o valor 1024 é subtraído da qualidade do leiaute.

Toda vez que ALPED é executado pode gerar até 20 diferentes leiautes, que são apresentados mostrando a qualidade do leiaute, conforme avaliação comentada anteriormente.

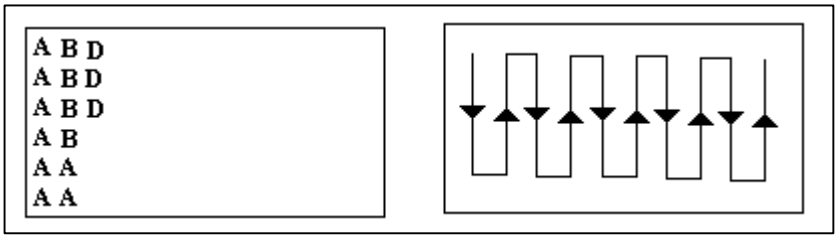

Figura 4.1: Procedimento de alocação.

4.2.2 - ALGORITMOS DE MELHORAMENTO

Os algoritmos de melhoramento iniciam o procedimento a partir de uma solução inicial, onde através de permutações de facilidades, busca-se melhorar esta solução inicial.

CRAFT

CRAFT - "Computerized Relative Allocation of Facilities Technique" (Armour e Buffa-1963) foi a primeira técnica computacional a procurar resolver o problema de leiaute. A técnica procura desenvolver um leiaute que minimiza o custo de transporte, onde custo de transporte é o produto dos tráfegos, das distâncias e custos unitários das distâncias.

CRAFT inicia determinando os centróides dos departamentos no leiaute inicial. Então é calculada a distância retangular entre os centróides dos departamentos e estas distâncias são armazenadas numa matriz. CRAFT em seguida avalia permutações de departamentos com áreas iguais ou que tenham uma borda comum. Os seguintes tipos de permutações podem ser consideradas:

- permutação de pares de departamentos;
- permutação de três departamentos;
- permutação de um par, seguida por permutação de três departamentos;
- permutação de três departamentos, seguida da permutação de um par;

- o melhor entre a permutação de um par ou três departamentos.

A permutação de departamentos que oferece a maior redução do custo de transporte é realizada. CRAFT continua considerando a permutação de departamentos para o novo leiaute. O procedimento continua até que nenhuma permutação encontrada reduza o custo de transporte. A busca então termina.

Vemos que o resultado final depende grandemente da solução inicial, assim, é necessário executar o procedimento diversas vezes para obter melhores resultados.

Além de CRAFT, existem outros algoritmos de melhoramento (Heragu e Kusiak, 1987).

4.2.3 - ALGORITMOS HÍBRIDOS

Scriabin e Vergin (1985) propuseram um algoritmo que consiste de três estágios. No primeiro estágio, as facilidades são alocadas de tal forma que as distâncias entre si, sejam inversamente proporcionais aos tráfegos. No segundo estágio, as facilidades são alocadas usando o princípio do estágio 1, mas restrições de área são consideradas. O terceiro estágio consiste de ajuste fino usando um algoritmo de permutação.

Elshafei (1977) propôs um algoritmo que é a combinação de um algoritmo construtivo e outro de melhoramento. O algoritmo construtivo emprega duas estratégias. Na primeira estratégia, localizações são ordenadas em ordem ascendente de R_j , onde R_j é a soma das distâncias da localização j para todas outras localizações. Facilidades também são ordenadas em ordem ascendente de L_i , onde L_i é baseada no número de facilidades tendo tráfego com facilidade i e a soma destes tráfegos. Usando a primeira estratégia, a qualquer estágio das atribuições, a facilidade ainda não atribuída com o maior L_i é alocada para a localização ainda não usada com mínimo R_j. Na segunda estratégia, a qualquer estágio k, a facilidade não alocada e que tenha o maior tráfego com a facilidade alocada no estágio k - 1, é atribuída a uma localização não usada que cause o mínimo incremento no custo total. Usando as duas estratégias, um leiaute completo é obtido e melhorado (se possível) através de permutações.

4.2.4 - ALGORITMOS NA TEORIA DE GRAFOS

Foulds e Robinson (1976) propuseram um algoritmo que inicialmente determina um tetraedro, isto é, um tipo particular de grafo onde cada um de seus quatro vértices são conectados aos outros três vértices. Note (figura 4.2) que o tetraedro possui quatro faces incluindo a face externa: f_1 , f_2 , f_3 , f_4 .

Os vértices restantes são inseridos um de cada vez, em uma das faces do grafo. A qualquer estágio do algoritmo, um lista de vértices *V*, arestas *E* e faces *F* são mantidas. Por exemplo, se o vértice *a* é inserido na face f_1 que consiste das arestas *pq*, *pr* e *qr*, então as arestas correspondentes *ap*, *aq* e *ar* são também adicionadas ao grafo. A lista *V*, *E* e *F* são atualizadas da seguinte forma:

- *V* consiste dos vértices *a, p, q, r, s;*
- *E* consiste das arestas *ap, aq, ar, pq, pr, qr, ps, rs, qs;*
- *F* consiste das faces f_1 , f_2, f_3, f_4, f_5 , f_6 .

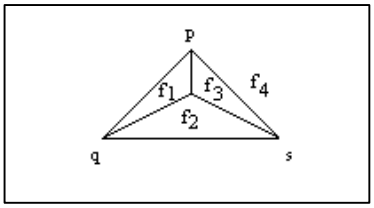

Figura 4.2: Tetraedro

4.2.5 - BUSCA TABU E ALGORITMO GENÉTICO

A busca tabu é um método heurístico aplicado, com muito sucesso, a um grande número de problemas de otimização combinatória. A forma básica moderna é de Glover (Glover, 1989a e 1989b), bem como seus melhoramentos (Glover, 1990). A idéia básica é evitar mínimos locais na busca, usando algumas estruturas para soluções (ou movimentos) proibidas (tabu).

O algoritmo foi usado por Furtado e Lorena (1997) para resolver o problema de leiaute, usando a formulação descrita na seção 3. Segue abaixo, a descrição do algoritmo.

Inicialização gera a árvore binária (usando um algoritmo de aglomeração ou produzida aleatoriamente)
solução inicial para o leiaute $s \in S$, e *nbmax*;
calcule f(s) (função objetivo)*;*
Pegue uma lista tabu *T*; *nbiter* := 0 (*nbiter* = número de iterações);
while *nbiter* < *nbmax* ***do***
Gere repetidas soluções $s_t \in N(s)$ com $[s \rightarrow s_t] \notin T$ ou $f(s_t) \leq A(f(s))$;
Verifique o melhor s_t gerado;
atualize a list tabu *T*;
atualize *A*(*f(s)*); {remova o último movimento tabu e introduza o movimento $s' \rightarrow s$}
s := *s'*;
nbiter := *nbiter* + 1;
endwhile

A árvore binária inicial é produzida de duas formas: através do processo de aglomeração ou aleatoriamente. A solução inicial $s \in S$ é um vetor de dimensão *N*. Cada componente é uma facilidade diferente e corresponde às folhas na árvore binária. *S* é o espaço de todas soluções possíveis, um espaço de dimensão *N!*.

Caracterizamos como sendo um *movimento* duas situações: a) o *movimento* corresponde à permutação de folhas da árvore binária; b) o *movimento* correspondente à permutação de nós internos da árvore binária.

N(s) é a vizinhança de *s*, isto é, o conjunto das soluções que podem ser obtidas através dos movimentos a partir de *s*. *N(s)* é dada através de todas as possíveis permutações (movimento) de duas diferentes facilidades ou dois diferentes nós internos. O parâmetro *nbmax* fornece o número de iterações do algoritmo e *A* representa o

critério de aspiração, ou seja, quando um movimento classificado como tabu pode ser ignorado.

Tam (1992) usando a formulação descrita na seção 3.3, usou o algoritmo genético para resolver o problema de leiaute. Neste trabalho, a árvore binária apresenta uma representação para se adaptar ao algoritmo genético. Cada nó interno da árvore binária representa a forma de corte da partição retangular. O tipo de corte é denotado por uma letra atribuída a cada nó interno. Partições reservadas às facilidades residem nas folhas da árvore. A cada folha é atribuído um único número inteiro correspondendo à identificação da facilidade. A estrutura de árvore é representada numa forma pós-fixada. Por exemplo, a figura 4.3 apresenta a seguinte representação: 65*c*4*e*32*c*1*bd*, onde: *d:* direita, *e:* esquerda, *c:* em cima, *b:* em baixo.

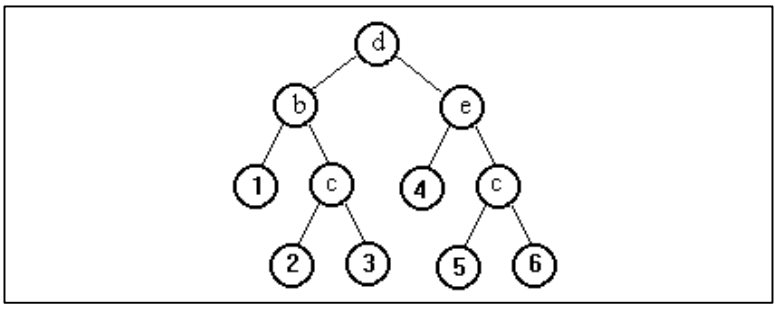

Figura 4.3 - Árvore binária usada com algoritmo genético.

A representação pós-fixada constitui uma seqüência de operadores, que por sua vez, constitui uma estrutura para o algoritmo genético. Se mantivermos as folhas fixas, isto é, se os números permanecerem fixos na estrutura, podemos ocultá-los. Por exemplo, a estrutura 65*c*4*e*32*c*1*bd* pode ser representada como *cecbd.*

Usando a terminologia do algoritmo genético, uma estrutura é o mesmo que uma seqüência de operadores. Todas as estruturas são geradas do alfabeto $\{e,c,b,d\}^{n-1}$, onde n é o número de facilidades.

Uma vez tendo uma estrutura e usando os operadores genéticos mutação e recombinação, soluções são geradas e avaliadas.

5 - CONSIDERAÇÕES FINAIS

Neste trabalho, vemos que o problema de leiaute de facilidades é um problema antigo e muitos esforços foram dispendidos para tentar resolvê-lo.

Verificamos que o problema surge em muitas situações práticas e definições podem sofrer alterações em função deste problema estudado.

As primeiras tentativas para solucionar o problema foram através de métodos exatos, como o procedimento “branch and bound”. Considerando que ele apresentava desvantagens, como a necessidade de grande memória e tempo computacional, além de não conseguir resolver problemas com grandes dimensões, os métodos para obtenção de soluções sub-ótimas ganharam impulso.

Devido à importância do problema, tanto econômica quanto científica e devido ao surgimento de novos métodos heurísticos, um renovado interesse tem surgido por novas tentativas de solucionar o problema de forma mais eficiente.

6 - REFERÊNCIAS BIBLIOGRÁFICAS

Armour, G.C.; Buffa, E. S.; A heuristic algorithm and simulation approach to the relative location of facilities. *Management Science,* 9, 295-309, 1963.

Bazaraa, M.S.; Elshafei, A.N.; An exact branch and bound procedure for quadratic assignment problems. *Naval Research Logistics Quarterly* 26, 109-121, 1979.

Bozer, Y. A.; Meller, R. D.; A reexamination of the general facility layout problem. *Working Paper, Department of Industrial Engineering, Auburn University, Auburn,* 1993.

Elshafei, A. N.; Hospital layout as a quadratic assignment problem. *Operations Research Quarterly,* 28(1), 167-179, 1977.

Foulds, L.R.; Robinson, D.F.; A strategy for solving the plant layout problem. *Operations Research Quarterly* 27(4), 845-855, 1976.

Furtado, J.C.; Lorena, L.A.N; Otimização de leiaute usando busca tabu. *Gestão & Produção,* vol. 4, no. 1, 88-107, 1997.

Glover, F.; Tabu Search - Part I. *ORSA Journal on Computing,* vol. 1, no. 3, 190-206, 1989a.

Glover, F.; Tabu Search - Part II, *ORSA Journal on Computing,* vol. 2, no. 1, 4-32, 1989b.

Glover, F.; Tabu Search - A tutorial. *Interfaces,* vol. 20, no. 4, 74-94, 1990.

Goto, S.; Kuh, E.; An approach to the two-dimensional placement problem in circuit layout. *IEEE Transactions Circuits Systems,* CAS-25, 208-214, 1978.

Heragu, S.S. e Kusiak, A.; The facility layout problem. *European Journal of Operational Research* 29, 229-251, 1987.

Heragu, S.S.; Recent models and techniques for solving the layout problem *European Journal of Operational Research* 57, 136-144, 1992.

Lee, R.C.; Moore, J.M.; CORELAP - Computerized Relationship Layout Planning, *Industrial Engineering,* 18, 195-200, 1967.

Quinn Jr., N.R.; Breuer, M.A.; A force directed component placement procedure for printed circuit boards. *IEEE Transactions Circuits Systems*, CAS-26, 377-388, 1979.

Sahni, S.; Gonzalez, T.; P-complete approximation problem", *Journal of Associated Computing Machinery*, vol. 23, no. 3, 555-565, 1976.

Scriabin, M.; Vergin, R.C.; A cluster-analytic approach to facility layout. *Management Science*, 31(1), 33-49, 1985.

Seehof, J.M.; Evans, W.O.; Automated layout design program, *Industrial Engineering*, 18, 690-695, 1967.

Skorin-Kapov, J.; Tabu search applied to the quadratic assignment problem. *ORSA Journal on computing*, vol. 2, no. 1, 33-45, 1990.

Tam, K.Y.; Genetic algorithms, function optimization, and facility layout design. *European Journal of Operational Reseach* 63, 322-346, 1992.

Tompkins, J.A.; White, J.A.; *Facilities planning*, John Wiley & Sons, 1984.

Yeap, G. K.; Sarrafzadeh, M.; A unified approach to floorplan sizing and enumeration. *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 12, no. 12,1858-1867, 1993.