

UNIVERSIDADE FEDERAL DE SANTA CATARINA  
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

ESTUDO DA DISTRIBUIÇÃO AUTOMÁTICA DE GEOMETRIAS PLANAS APLICANDO  
CONCEITOS DE INTELIGÊNCIA ARTIFICIAL

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA  
PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA MECÂNICA

MARIA LETICIA ROSS

FLORIANÓPOLIS, DEZEMBRO DE 1989

ESTUDO DA DISTRIBUIÇÃO AUTOMÁTICA DE GEOMETRIAS PLANAS APLICANDO  
CONCEITOS DE INTELIGÊNCIA ARTIFICIAL

MARIA LETICIA ROSS

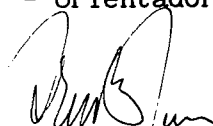
ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE

MESTRE EM ENGENHARIA

ESPECIALIDADE ENGENHARIA MECÂNICA, ÁREA DE CONCENTRAÇÃO FABRICA-  
ÇÃO, APROVADA EM SUA FORMA FINAL PELO CURSO DE PÓS-GRADUAÇÃO EM  
ENGENHARIA MECÂNICA.

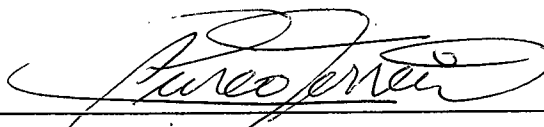


Prof. Áureo Campos Ferreira, Ph.D.  
- Orientador -

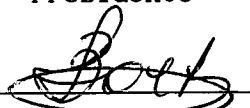


Prof. Arno Blass, Ph.D.  
- Coordenador do Curso -

BANCA EXAMINADORA:



Prof. Áureo Campos Ferreira, Ph.D.  
- Presidente -



Prof. Lourival Boehs, Dr. Eng. Mec.



Prof. Luiz Fernando Jacinto Maia, M.Sc.

Ofereço este trabalho ao meu tio  
Marcos João Reginato, pelo amor sempre dedicado e pelo exemplo de vida.

**HOMENAGENS ESPECIAIS**

**AOS MEUS PAIS**, José Carlos e Eduly, pela minha existência, pela educação, que me permitiu chegar até aqui, e pelo suporte financeiro durante a realização do curso de pós-graduação.

**À MINHA IRMÃ GÊMEA**, Maria Alice, pela torcida de 24 horas por dia e pelos momentos que eu gostaria de ter estado ao seu lado e não pude.

**AO MEU NAMORADO**, Édson, pelo carinho e companheirismo, cujo apoio, incentivo e cobranças (quando necessárias), foram decisivos na realização deste trabalho.



## AGRADECIMENTOS

À Universidade Federal de Santa Catarina, pela oportunidade de realizar o curso de pós-graduação.

Ao CNEN - Comissão Nacional de Energia Nuclear - pelo apoio financeiro.

Ao prof. Aureo, pela orientação deste trabalho e também pelo apoio, estímulo e amizade.

Aos amigos Luiz Márcio Spinosa e Ricardo José Rabelo, pelas valiosas opiniões na área de informática.

À estagiária Myriam de Fátima Borba de Oliveira, pela colaboração no desenvolvimento do sistema apresentado.

Ao prof. Stemmer, pela confiança depositada na minha pessoa.

Aos professores do curso de pós-graduação, Arno Blass, Berend Snoeijer, Ingeborg Kuhn Arroyo, Walter Lindolfo Weingaertner, Aureo Campos Ferreira e Abelardo Alves de Queiroz, pelas lições de vida, além dos ensinamentos técnicos.

Às secretárias Verinha, Soraya e Luciane, pelos serviços prestados e pela amizade.

Aos colegas do curso de pós-graduação e todos os amigos do GRUCON. A agradável convivência sempre amenizou as horas de cansaço.

À Dona Stella, pelo carinho e pelas orações.

Aos meus irmãos(ãs), cunhados(as) e sobrinhos(as).  
O amor de vocês sempre foi muito importante.

À Diná, pela ajuda espiritual.

À Sílvia Lopes Carneiro Leão, amiga muito querida.

À Rose, pelo apoio, sempre que necessário.

Ao colega Wiliam Alves Barbosa, pelas substituições de aulas, quando precisei.

A todos os meus amigos, que de uma maneira direta ou indireta deram sua ajuda.

Agradeço do fundo do coração a todos que acreditaram em mim, mas quero fazer um agradecimento especial àqueles que duvidaram. Provar que estavam errados era um obstáculo que eu ti-  
nha que superar.

Finalmente, agradeço a Deus, pelos bons momentos que tenho tido, e pela força, quando os momentos são mais difíceis.

## SUMÁRIO

1 - Introdução .....	01
1.1 - Definição do problema .....	01
1.1.1 - O processo de produção manual na indústria do vestuário .....	01
1.1.2 - O processo auxiliado por computador .....	09
1.2 - Objetivos do trabalho .....	14
2 - Revisão bibliográfica	
Comentário de artigos sobre o assunto. ....	15
3 - O uso da Inteligência Artificial na solução do problema. ....	28
4 - Sistema proposto .....	40
4.1 - Definição das heurísticas .....	40
4.2 - Implementação de um algoritmo proposto .....	42
4.3 - Resultados .....	69
4.4 - Limitações .....	69
5 - Conclusões e sugestões para trabalhos futuros. ....	71
Referências Bibliográficas .....	74
Apêndice A - Introdução aos termos técnicos de Pesquisa Operacio- nal .....	79
A.1 - Pesquisa Operacional .....	79
A.2 - Programação Linear .....	81
A.3 - Método Simplex ou Algoritmo Simplex de Dantzig .....	87

A.4 - Programação Inteira .....	103
A.5 - Algoritmo Branch-and-Bound .....	107
A.6 - Problema da Mochila .....	110
A.7 - Programação Dinâmica .....	111
Apêndice B - Inteligência Artificial .....	120
B.1 - Aspectos da inteligência .....	120
B.2 - Busca heurística .....	127
B.3 - Comparação entre programação convencional e programação em inteligência artificial .....	129
B.4 - Formas de representação do conhecimento .....	132
B.5 - Sistemas Especialistas .....	151
B.6 - Linguagens .....	157
B.6.1 - Lisp .....	160
B.6.2 - Prolog .....	163
B.6.3 - A linguagem C para Inteligência Artificial ....	165

## RESUMO

Quando se desejam obter peças oriundas de uma matéria prima plana, é importante aproveitar a maior quantidade possível dessa matéria prima. No caso da indústria do vestuário, onde os moldes são distribuídos sobre o tecido, primeiro é preciso reduzir o tamanho destes moldes, com o objetivo de possibilitar uma visualização geral do encaixe, que depois de aprovado é riscado no seu tamanho real. Cada molde utilizado nesses processos, devido a seu elevado custo, deve ser armazenado em local apropriado e arquivado de forma a facilitar sua recuperação, ocupando, com isso, um espaço físico considerável. Além disso, trata-se de um trabalho bastante moroso e cansativo, realizar o encaixe manualmente.

Os recursos computacionais gráfico-interativos, hoje disponíveis, possibilitam a solução de alguns desses problemas. No entanto, a realização do encaixe propriamente dito, não sendo totalmente automático, continua dependente de especialistas.

A formação técnica de um especialista requer tempo da ordem de meses, e a produção de uma indústria pode aumentar subitamente, exigindo de imediato um número maior de especialistas. A automatização da tarefa é interessante nesse caso, pois durante a noite os computadores realizariam os encaixes e, se necessário, durante o dia, os especialistas, além de realizarem encaixes normais, aperfeiçoariam os gerados pelo computador, com uma considerável redução no tempo exigido, possibilitando que a indústria responda de pronto ao aumento da demanda.

Nesse trabalho estudou-se uma maneira de abordar essa automatização.

Durante o levantamento das informações, foi confirmado o que outros trabalhos já haviam concluído, ou seja, que se trata de um problema extremamente complexo, que até o momento não apresenta uma solução satisfatória pelos meios convencionais, dirigindo os estudos do presente trabalho para as técnicas de Inteligência Artificial.

Baseando-se nessas técnicas, foi desenvolvido um sistema, em linguagem C, onde foram elaboradas heurísticas básicas.

Para concluir, são apresentados os resultados da aplicação dessas heurísticas no encaixe de peças do vestuário.

## ABSTRACT

When it is desired to obtain patterns from a rectangular piece of raw material, as it is in the garment industry, it is always necessary to make the most economical use possible of available material. In order to achieve this, the size of each pattern needs to be reduced, for a better visualization of the intended nesting. After this has been approved, it will then be re-drawn in full size. Because of the cost involved in this process, all the patterns need to be stored away, taking up considerable space. Besides, when this is done by hand it is a time-consuming and tiring task.

Some of these problems can be solved, due to the graph-interatives computing resources available today. However, if the nesting is not a fully automated operation itself, it still needs an expert.

The technical habilitation of an expert might take a few months, and when a sudden growth occurs in the industry production, it faces an urgency of more experts. The automation of the task is interesting in this case, because computers could do the nestings during nighttime, and, in the daytime, experts could improve those computer generated nestings, if needed. Experts may execute interactive nestings as well, and all this operation described reduces considerably the time needed, leading to a prompt answer from the industry to the increased demand.

The purpose of this dissertation is to suggest an approach to such automation.

Throughout all the informations survey, it was possible to assure what have other papers concluded. It is an extremely

complex problem with no satisfactory solution through conventional ways to the present date, orienting the present research towards Artificial Intelligence techniques.

A C Language system based on such techniques was developed where basic heuristics have been elaborated.

An application of these heuristics in garment nesting is shown as a conclusion.



## 1 INTRODUÇÃO

### 1.1 - DEFINIÇÃO DO PROBLEMA

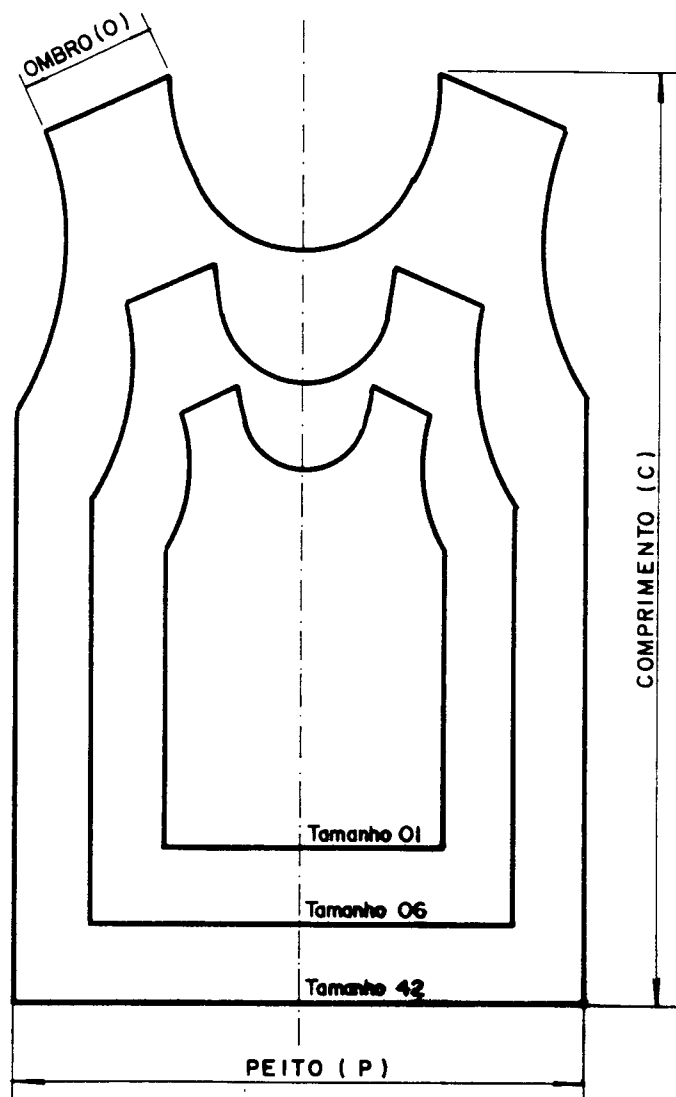
Para se poder abordar o assunto de encaixe automático é interessante definir como é feito o encaixe manual e localizá-lo no fluxo de produção. Para tal, tomou-se como exemplo indústrias do vestuário que foram visitadas com o intuito de obter uma série de informações referentes ao processo produtivo.

#### 1.1.1 - O PROCESSO DE PRODUÇÃO MANUAL NA INDÚSTRIA DO VESTUÁRIO

Levando em consideração tendências da moda, faixa etária, tecido, estação do ano, entre outros, os estilistas criam os modelos das roupas. A partir destes modelos são elaborados os moldes referentes a cada peça de roupa, e, baseados em tabelas padrão, é feito o escalonamento dos moldes. Entende-se por escalonamento como sendo o processo para obtenção dos diversos tamanhos de moldes conforme ilustrado na FIGURA 1.

Uma vez confeccionadas as roupas de acordo com os moldes definidos anteriormente, realizam-se as provas em manequins selecionados. Erros ou defeitos que eventualmente possam ocorrer são corrigidos até que as provas saiam todas satisfatórias.

Geram-se os moldes, agora aprovados, em papel prespam (quando existir uma ordem de produção) ou em cartolina (quando existir uma ordem de estudo). A estes moldes são agrupadas informações tais como: identificação e sentido do molde sobre o enfiesto. (Enfiesto são camadas de tecido sobrepostas. No caso de malharias, estas camadas são tubulares).



### TABELA DE PROPORÇÕES

	01	03	06	12	42	44	48	52	Tamanhos
<b>C</b>	36	42	51	60	70	73	79	85	Dimensões Em cm
<b>O</b>	06	08	10	12,5	15	16	17	18	
<b>P</b>	22	26	32	38	44	46	50	54	

FIGURA 1 - Escalonamento do molde da frente de uma camisa, com base em tabelas, de acordo com um certo bio-tipo [12].

Isto feito, os moldes estão prontos para a etapa do encaixe, no entanto, trabalhar com os moldes nos seus tamanhos reais significaria a necessidade de grande espaço físico, além da perda de

tempo da pessoa encarregada de realizar o encaixe, que teria que andar bastante em volta de uma grande mesa e, conforme sua posição, não teria uma boa visualização do encaixe geral.

Para resolver estes problemas, os moldes em papel prespam são reduzidos numa proporção adequada, facilitando sua manipulação bem como a visualização geral do encaixe. Esta redução é obtida através de um pantógrafo que na sua extremidade possui uma serra ou uma ponta aquecida, apoiada numa placa de poliestireno como mostrado na FIGURA 2.

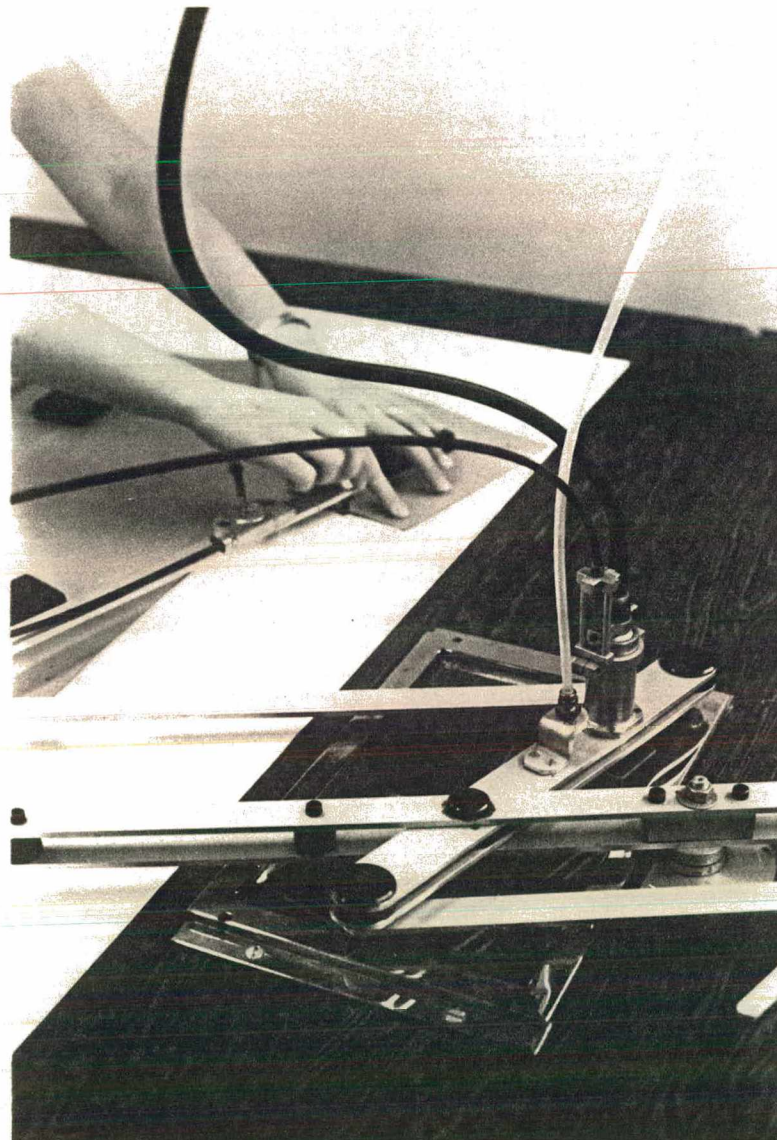


FIGURA 2 - Pantógrafo para redução de moldes em poliestireno.

Começa-se então o encaixe propriamente dito. As miniaturas dos moldes são encaixadas sobre uma área representativa do enfesto, observando-se o sentido do fio, o posicionamento do molde e sobretudo a economia do enfesto ilustrado na FIGURA 3.

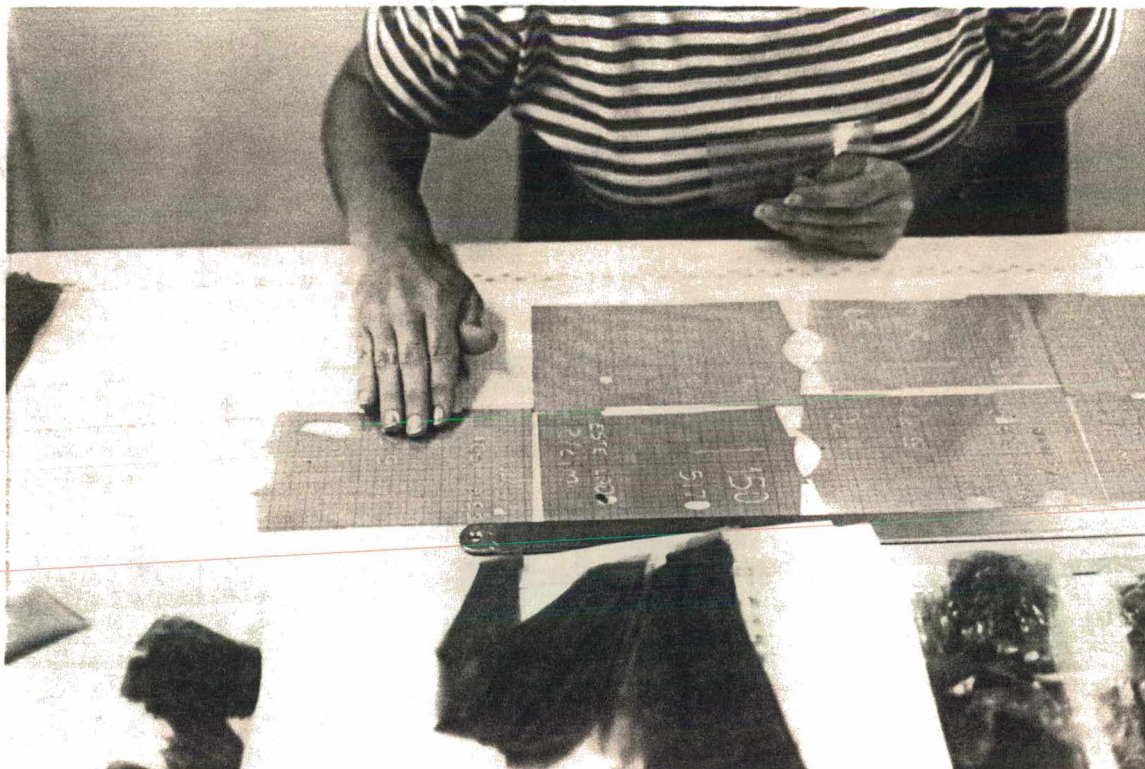


FIGURA 3 - Encaixe manual das miniaturas dos moldes.

Para melhor controle sobre o aproveitamento dos encaixes, bem como uma estimativa dos tempos de corte, é necessário o cálculo do perímetro e área de cada miniatura. Isto é obtido utilizando-se um pantógrafo que possui sensores para medir as variações angulares dos braços. Estes sensores fornecem os dados para um micro-computador que realiza os cálculos.

Se os encaixes não atingem os índices de aproveitamento definidos pela empresa, devem ser reestruturados.

Uma vez aprovado o encaixe, é feita uma cópia em papel,



que servirá de guia para as riscadeiras (FIGURA 4).

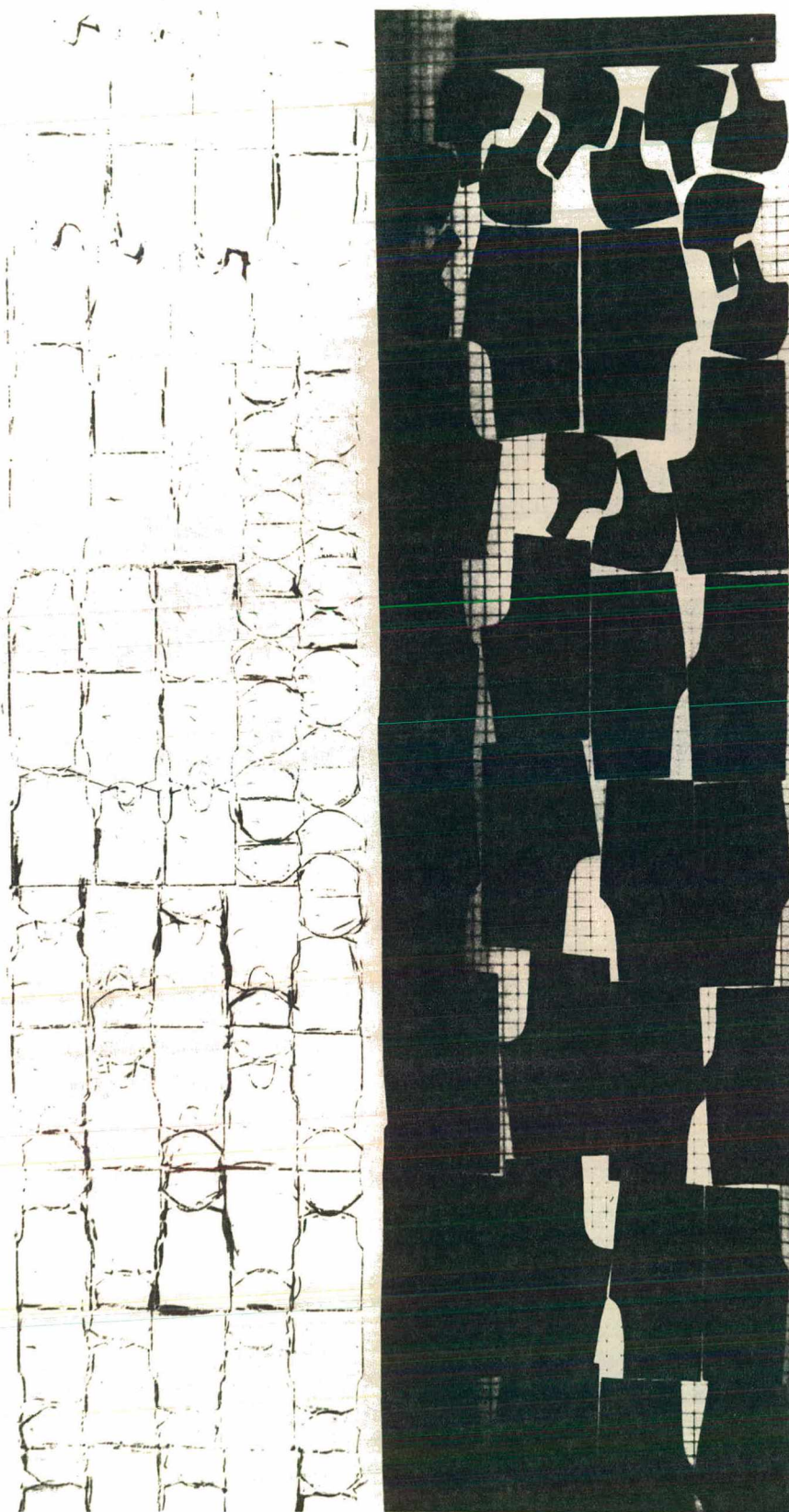


FIGURA 4 - Cópia do encaixe obtido a partir dos moldes miniaturizados.

A próxima etapa será, a partir do encaixe miniaturizado, dispor os moldes de papel prespam sobre um papel que represente o tamanho real do enfeite e riscá-lo, como mostrado na FIGURA 5.

Observe-se que há variação do tamanho de um enfeite para outro, sendo então este papel cortado de acordo com o tamanho do enfeite que está representando.

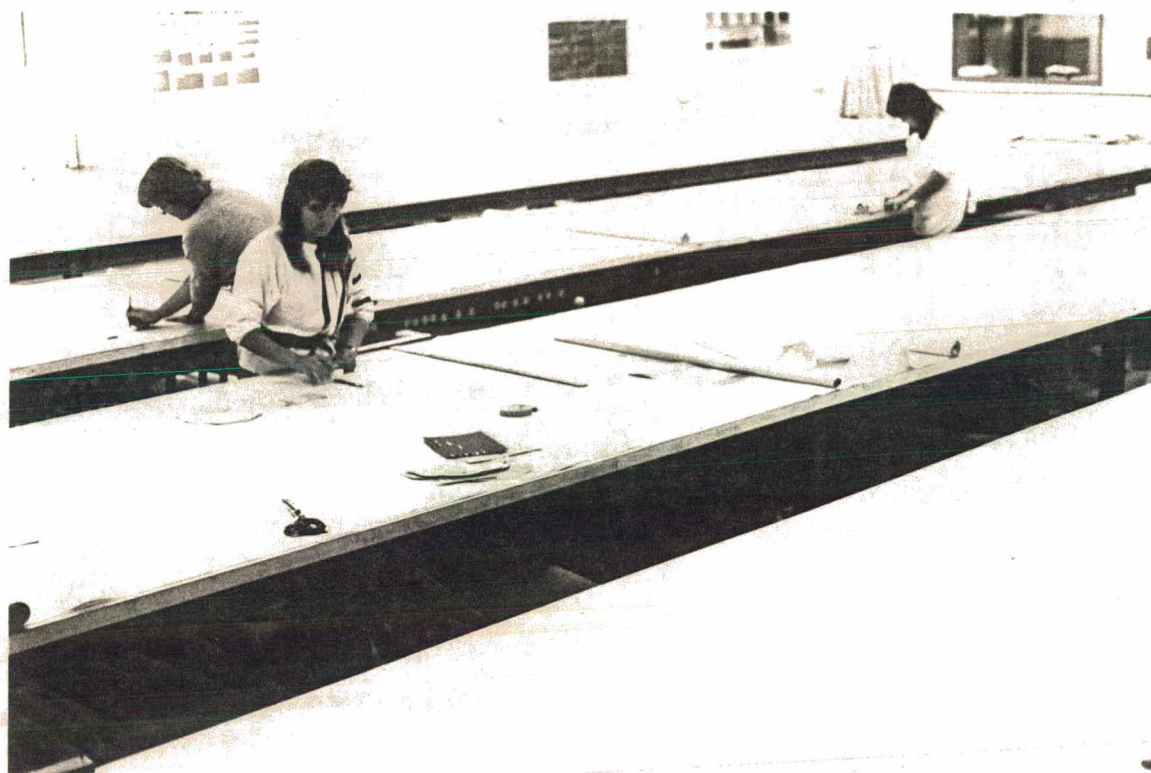


FIGURA 5 - Riscadeiras.

Obtém-se, então, um papel do tamanho do enfeite, riscado com os contornos dos moldes em tamanho real e que será colocado sobre este.

Para as ordens de estudo, é realizada uma estimativa dos custos referentes a uma produção efetiva. Se sua produção for economicamente adequada, uma ordem de produção é gerada. Caso contrá-



rio, será gerada uma nova ordem de estudo. Algumas vezes, também é necessário uma alteração na estética dos modelos criados.

Em resumo, partindo-se do princípio que o molde já foi aprovado, o processo produtivo passará pelas seguintes etapas:

- a) Escalonamento (FIGURA 1);
- b) execução dos moldes em papel prespam;
- c) miniaturização dos moldes com pantógrafo (FIGURA 2);
- d) encaixe dos moldes miniaturizados (FIGURA 3);
- e) cópia do encaixe em miniatura (FIGURA 4);
- f) seguindo-se o encaixe obtido no item "e", posicionam-se os moldes de papel prespam sobre um papel do mesmo tamanho que o enfeito, riscando-o (FIGURA 5);
- g) Coloca-se o papel riscado sobre o enfeito que é finalmente cortado.

Além de todas estas etapas serem feitas manualmente, existe ainda a necessidade de um depósito de moldes nas instalações da indústria, o que ocupa um espaço considerável (FIGURA 6).



FIGURA 6 - Depósito de moldes de papel prespam.



### 1.1.2 - O PROCESSO AUXILIADO POR COMPUTADOR

Neste trabalho, tomou-se como exemplo um sistema cuja configuração básica consta de (FIGURA 7):

- A) monitor gráfico
- B) mesa digitalizadora
- C) cursor de digitalização
- D) teclado alfa-numérico
- E) caneta magnética + "tablet"
- F) Unidade Central de Processamento

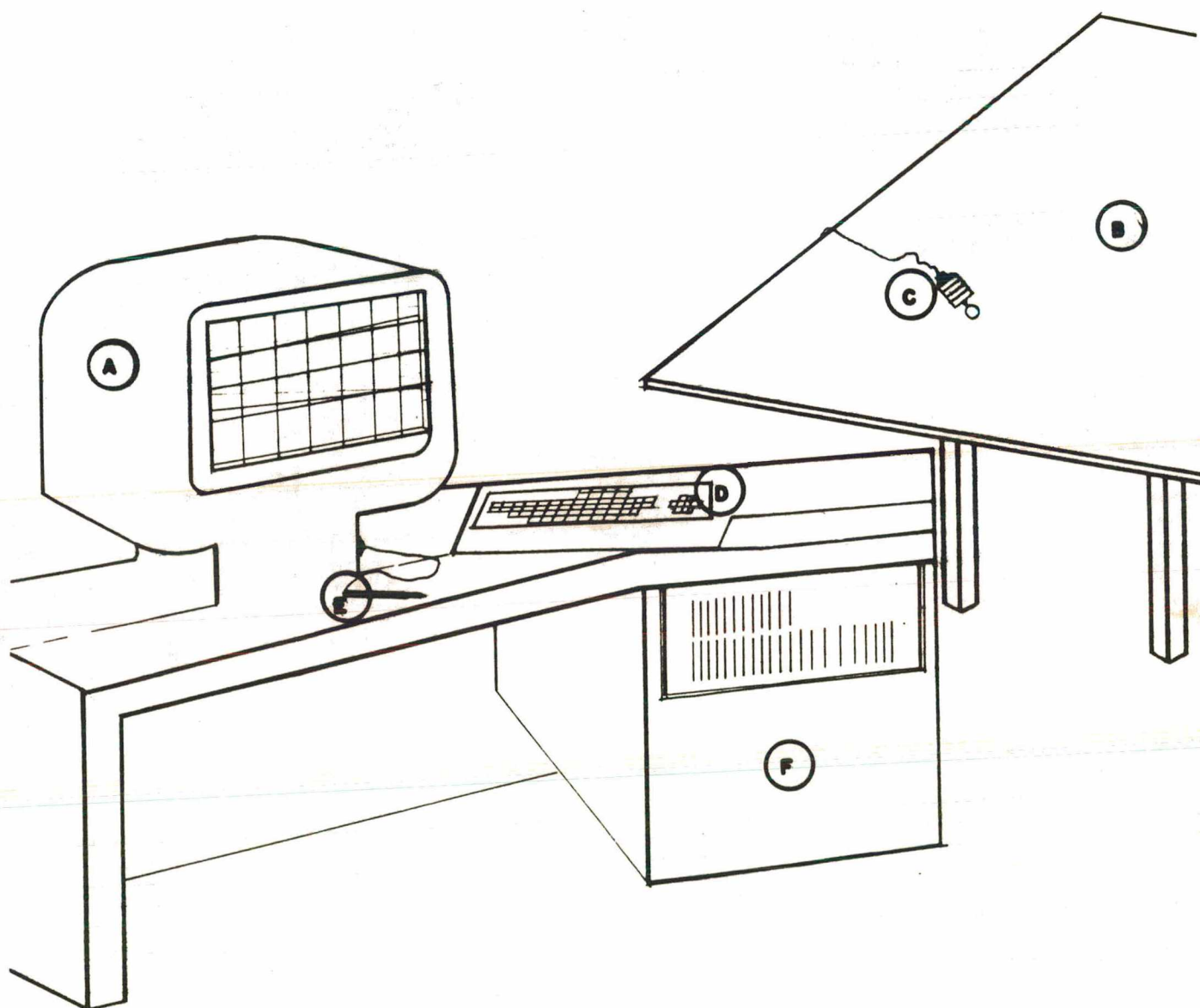


FIGURA 7 - Configuração básica do sistema.

A etapa inicial do sistema é a digitalização. Através da mesa digitalizadora e do cursor de digitalização, informa-se ao sistema a geometria e a identificação do molde criado. Este processo é bem simples: com o cursor contorna-se o desenho do molde, identificando os pontos de referência. Os pontos de referência são a base para o processo de escalonamento. Na FIGURA 8, os pontos 1, 2, 3, 4, 5, 6, 7 e 8 são pontos de referência.

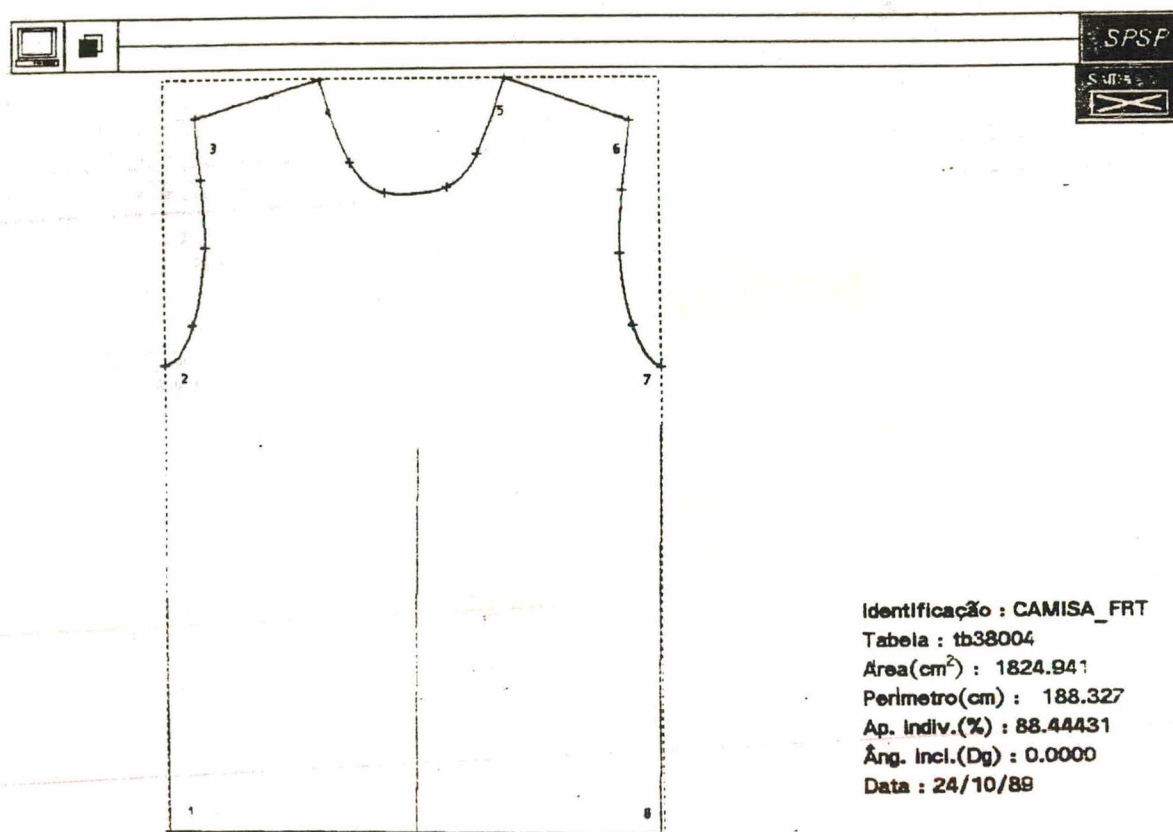


FIGURA 8 - Exemplo de um molde.

Através do escalonamento obtém-se toda a numeração desejada de um determinado molde. A geração dos números maiores ou menores baseia-se principalmente nos pontos de referência. Uma vez identificado o ponto de referência, define-se uma variação em X e Y para os demais números.

Algumas vezes a tabela de graduação não corresponde ao desejado estilisticamente. Existe então a possibilidade de se alterar o posicionamento dos pontos de referência (FIGURA 9).

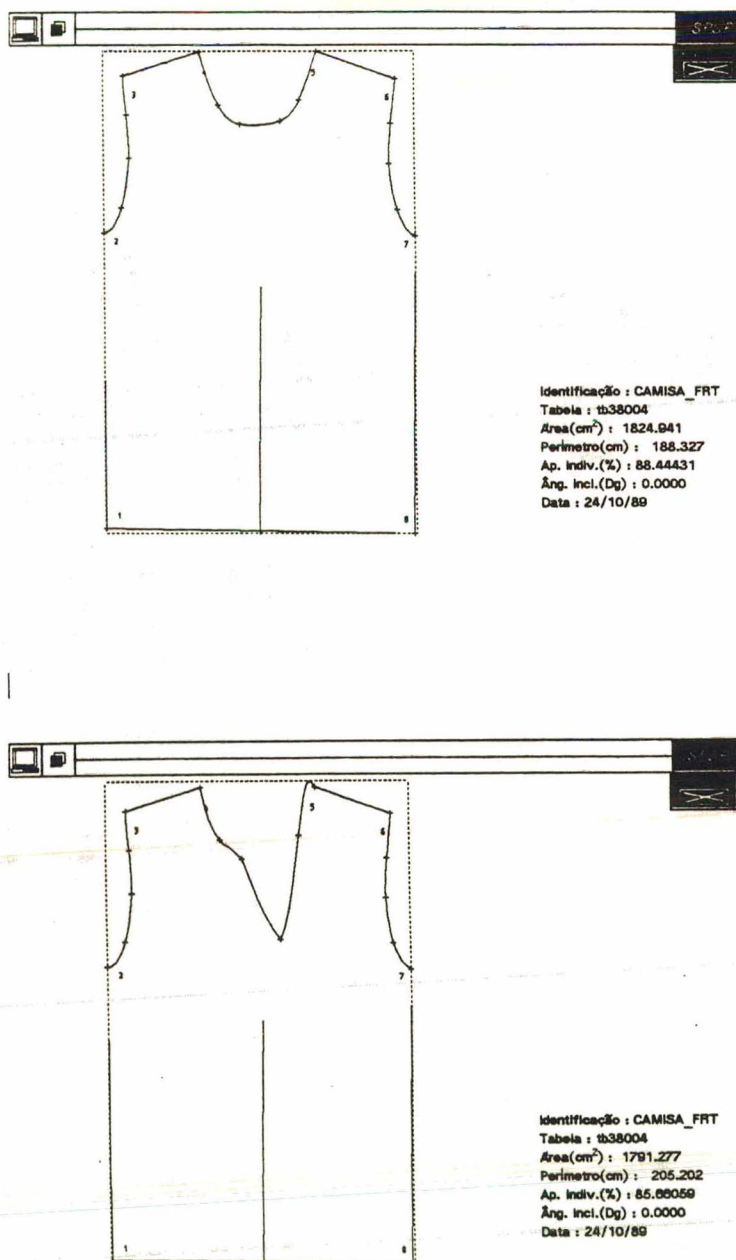


FIGURA 9 - Alteração no molde através da mudança nos pontos de referência

Uma vez que se saibam principalmente:

- a) a quantidade e os tipos de moldes a serem encaixados;
- b) o tipo de enfesto, e
- c) o aproveitamento desejado, são definidos os lotes de moldes a se encaixar.

Esta etapa do encaixe evidencia uma das maiores vantagens do sistema auxiliado por computador. O usuário pode definir o posicionamento de cada molde de um lote sobre o enfesto, através de uma caneta magnética e guiando-se pelas imagens do monitor.

O processo todo consiste basicamente em tentativas de movimentação dos moldes até que se obtenha um aproveitamento adequado do enfesto. Estes movimentos são feitos com bastante facilidade através do sistema gráfico interativo.

- se o usuário deseja IDENTIFICAR um molde para movimentação, basta deslocar a caneta sobre o "tablet" (um cursor no monitor espelhará estes movimentos), e indicar com uma leve pressão da caneta.

- se o usuário desejar TRANSLADAR o molde, primeiro ele terá que ser identificado, depois com movimentos rápidos da caneta pressionada sobre o "tablet", a peça se deslocará até uma distância mínima de outra peça ou os limites do enfesto, no sentido do risco feito pela caneta.

- se o usuário desejar ROTACIONAR o molde, primeiro ele terá que ser identificado, depois, com movimentos prolongados da caneta pressionada sobre o "tablet", a peça rotacionará segundo seu grau de liberdade (por exemplo,  $90^\circ \pm 2^\circ$ ) no sentido do risco feito pela caneta.

Finalizando o processo vem a geração de riscos, que é feita pelos traçadores gráficos ("plotters"), e tem como resultado



o encaixe no seu tamanho real. Estes riscos servirão de guia para o setor de talharia.

Comparando-se este processo com o processo manual tem-se o seguinte:

a) O escalonamento agora é feito automaticamente pelo computador e não mais manualmente;

b) não há necessidade de se confeccionar moldes em papel prespam;

c) não há necessidade de se reduzir o tamanho dos moldes com o pantógrafo;

d) o encaixe é realizado semi-automaticamente, usando-se recursos gráfico-interativos;

e) não há necessidade de uma cópia do encaixe miniaturizado;

f) o risco do papel que servirá de guia para a talharia é gerado automaticamente e no tamanho natural (escala 1:1) segundo os diversos tamanhos especificados na ordem de produção.

g) não há necessidade de um depósito de moldes na indústria.

O presente trabalho visa especificamente o item "d" onde mesmo com auxílio do computador, o encaixe ainda não está totalmente automatizado.

A necessidade da automação total na área do vestuário se faz sentir principalmente nos países mais industrializados. Segundo Ralph King Jr. [24], a V.F.Corp., uma grande fábrica de jeans norte-americana, está investindo 4% do valor de suas vendas para automatizar sua produção, desde o projeto de novos estilos, passando pelo encaixe automático, até a costura automática supervi-

sionada por câmeras, onde o objetivo é triplicar a velocidade de produção.

## 1.2 - OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho será indicar em que direção devem seguir os estudos no sentido de se obter um sistema computadorizado que faça encaixes automaticamente, com um enfoque especial na indústria do vestuário.

A indicação desta direção será importante porque no momento não há como prever o rumo a ser tomado:

- a) se é possível desenvolver um algoritmo usando apenas as técnicas da pesquisa operacional;
- b) se serão utilizados métodos matemáticos;
- c) se serão necessárias outras técnicas.

Para tal, será feita uma busca exaustiva nos artigos sobre o assunto, analisando-se as técnicas utilizadas com os resultados obtidos. Isto deverá direcionar os estudos para um dos itens mencionados, que deverá ser o próximo passo deste trabalho.

Uma vez realizado este estudo, pretende-se dar início ao desenvolvimento do sistema.

Embora este não seja um problema só da área da engenharia mecânica, é um problema da área de CAD/CAM, onde o objetivo é a automação integrada desde o projeto até a fabricação. Além do que, grande parte da indústria mecânica, e não apenas a indústria do vestuário, têm inúmeras aplicações para este tipo de sistema toda vez que se trata de planejamento da fabricação de peças planas.

## 2 - REVISÃO BIBLIOGRÁFICA COMENTÁRIO DE ARTIGOS SOBRE O ASSUNTO

Para um leigo, a solução do problema do encaixe automático feito pelo computador pode parecer bem simples. Basta examinar todos os arranjos possíveis, compará-los entre si e escolher o que melhor atende aos objetivos que serão:

- minimizar as sobras inúteis, ou,
- minimizar a quantidade de matéria prima necessária.

Quando a demanda é conhecida com exatidão, ambos os objetivos são satisfeitos, uma vez que uma solução que implique na menor perda implicará simultaneamente na menor quantidade de matéria prima necessária e vice-versa [06].

A dificuldade está no grande número de variáveis envolvidas (ou, em termos de *programação linear*<sup>1</sup>, o problema do grande número de colunas que podem ser geradas), tornando o problema intratável. Por exemplo, para um rolo de papel padrão com 5,0 metros e uma demanda de 40 comprimentos diferentes, variando de 0,50 a 2,00 metros, o número de colunas pode exceder a 100 milhões, elevando demasiadamente o tempo de processamento [14].

Antes da metade da década de 50 os problemas de encaixe automático não foram abordados pela literatura científica anglo-americana e alemã [06].

Um modelo de solução clássica foi desenvolvido a partir de 1961 por Gilmore e Gomory [13][14][15][16]. Este modelo fundamentava-se numa formulação baseada na programação linear para a so-

1 No Apêndice A encontram-se explicações básicas dos termos técnicos de Pesquisa Operacional.



lução do problema de encaixe automático, por eles denominado "*the cutting-stock problem*", envolvendo geração de colunas, cujo subproblema não precisava ser necessariamente de programação linear, observando-se que no caso do encaixe automático, o subproblema recaía no chamado "*problema da mochila*"<sup>1</sup> ("*knapsack problem*" ou "*knapsack functions*").

A grande contribuição de Gilmore e Gomory foi então encontrar uma solução eficiente para o problema da mochila, o que resolve o problema do grande número de colunas geradas ao se resolver o problema pelo *método simplex*<sup>1</sup>. Para se ter uma idéia da evolução do trabalho, entre um artigo e outro, eles desenvolveram um algoritmo da mochila cinco vezes mais rápido que o modelo de *programação dinâmica*<sup>1</sup> aplicado no artigo anterior [17].

Minimizar a perda é o principal objetivo desta técnica por eles desenvolvida. Este caminho, no entanto, dificilmente leva em conta que a maior parte dos problemas implicam em condições adicionais em relação à produção, ou existe a necessidade de se levar em consideração outras áreas de planejamento durante sua solução. Como exemplos destas "condições adicionais" tem-se que parte do estoque pode apresentar defeitos ou ainda, o fato de alguns materiais serem isotrópicos enquanto outros não o são. Para materiais isotrópicos, a orientação de um retângulo a ser cortado em relação ao comprimento e a largura do retângulo da matéria prima é irrelevante, enquanto que para materiais anisotrópicos a orientação pode ser muito importante. No caso da indústria do vestuário, por exemplo, a orientação adequada do fio do tecido (ou dos desenhos do tecido) é muito importante.

Estas deficiências dirigiram os esforços no sentido de buscar outras soluções, principalmente baseadas em técnicas de *heurísti-*



cas<sup>2</sup>, que como regra, estão inteiramente direcionadas para situações específicas de planejamento. Isto está refletido numa variedade de contribuições que têm sido e continuam sendo publicadas, sendo inclusive, em certos casos, analisadas e condensadas por outros autores [06], [17], [21] e [34]. A respeito delas, existe uma conclusão tecida por Hinxman [21] sobre os problemas de encaixe automático, por ele denominado "*the trim loss problem*":

"No que diz respeito ao estado da arte, se alguém tiver que resolver um problema de encaixe automático, terá que desenvolver seu próprio método, a não ser que a técnica de Gilmore e Gomory seja aplicável a seu caso. Este método a ser desenvolvido, quase sempre terá que ser heurístico. O estudo de artigos, se existir algum, em problemas similares pode se tornar útil. Entretanto, métodos heurísticos tendem a ser fortemente baseados nos problemas particulares a que são aplicados e problemas aparentemente similares podem precisar de métodos heurísticos radicalmente diferentes."

Em 1968, R.W.Haessler escreveu "*An Application of Heuristic Programming to a Nonlinear Cutting-Stock Problem Occuring in the Paper Industry*" como tese de doutorado pela Universidade de Michigan, Ann Arbor e em 1969 publicou um artigo [18] em que desenvolvia heurísticas integradas com as técnicas de Gilmore e Gomory. Este estudo era direcionado à indústria de papel e devido ao seu baixo preço por unidade de peso, o objetivo não era apenas minimizar as perdas nos cortes do papel, mas minimizar todos os custos controláveis do processo, onde se incluíam os custos decorren-

2 As explicações da técnica de busca heurística podem ser encontradas no capítulo 3 e no Apêndice B.

tes das operações de acabamento, necessárias para satisfazer o pedido do cliente.

Estes custos de acabamento são controláveis à medida que se pode escolher o tipo de corte e conseqüentemente a quantidade de tempo requerida para completar cada operação. A inclusão de custos de acabamento no problema de encaixe introduz não linearidades, o que torna muito mais difícil de se achar a solução ótima.

Neste estudo, testou-se um procedimento heurístico computado-rizado para uma situação específica, envolvendo pedidos que seriam processados através de uma operação de acabamento bem definida, considerando custos não lineares.

Ele concluiu que os modelos de programação linear com o objetivo de otimizar encaixes automáticos na indústria de papel, que haviam sido desenvolvidos até a época, tinham sido inadequados porque haviam ignorado a economia do processo de acabamento. O objetivo foi então mostrar que havia a possibilidade de se desenvolver modelos que, na média, poderiam proporcionar melhores soluções, se comparados com procedimentos manuais comuns em problemas de encaixe que envolvessem não-linearidades significantes.

Com este objetivo, entretanto, era necessário renunciar à elegância de uma solução matemática fechada e dirigir a atenção no sentido de desenvolver um procedimento heurístico para a geração de soluções praticáveis com características satisfatórias. Desta forma foi possível considerar e resolver as realidades do problema. Com o procedimento heurístico houve um aumento da qualidade da solução em 16% se comparado com o método manual.

Em 1969 Charles M. Eastman [07] iniciou estudos sobre linguagens computacionais para problemas que envolviam o arranjo de ob-

jetos em espaços bi ou tri-dimensionais a que chamou de "problemas de planejamento do espaço" ("*space planning problems*").

Partindo do princípio de que a forma como os humanos representam o espaço físico é um desenho, foi feita uma revisão de como os projetistas resolviam os problemas de planejamento do espaço no desenho.

Qualquer alternativa de planejamento do espaço poderia ser gerada e avaliada se as seguintes condições estivessem disponíveis e facilitadas:

- a) Representação de domínios de qualquer forma;
- b) Determinação de qualquer domínio ou atribuição de um domínio representado;
- c) Identificação de qualquer grupo de domínios adjacentes que se quiser;
- d) Determinação de qualquer dimensão ou atributo de um grupo de domínios adjacentes.

Numa análise das linguagens gráficas para computação, as CGLs (Computer Graphic Languages), disponíveis na época, concluiu-se que eram boas para a análise de vários tipos de sistemas claramente estruturados como processos hidráulicos ou circuitos elétricos. Também eram apropriadas para análises do fluxo de diversos grupos de elementos através de arranjos topológicos diferentes, mas não eram adequadas para o planejamento do espaço pois as análises e operações necessárias não eram facilitadas por tais estruturas de dados, devido às seguintes limitações:

- a) Apenas os elementos locados no espaço eram representados na estrutura de dados. Os vazios entre os elementos eram ignorados.



- b) Segmentos de linhas especificavam um elemento. Domínios tinham que ser obrigatoriamente derivados destas linhas.
- c) A especificação de cada elemento era ordenada na estrutura de dados de acordo com uma organização topológica definida pelo programador e não espacialmente.

Como não existiam especificações do tamanho e formato dos espaços vazios, não era possível testar diretamente se um espaço vazio poderia abrigar um sólido particular.

Uma variedade de outras estruturas de dados que respondiam mais diretamente às necessidades do planejamento do espaço haviam sido desenvolvidas. Estas estruturas representavam tanto os espaços preenchidos como os vazios, especificavam as adjacências dos domínios espaciais e também proporcionavam facilidades para verificar a sobreposição de domínios.

Foram apresentados então 4 possíveis representações para o problema:

- a) arranjo simples;
- b) arranjo hierárquico;
- c) representação em string;
- d) representação com domínios de tamanho variável.

Eastman explorou o uso de strings para representar problemas de locação no espaço, usando como linguagem básica o SNOBOL4.

Todas as representações apresentadas serviam para poliedros retangulares convexos como domínio. Compreendem domínios cujos lados são perpendiculares às coordenadas de maneira que cálculos computacionais muito simples podem somar distâncias e áreas de domínios adjacentes. Esta regra falha ao tratar com elementos não

paralelos às coordenadas. Também não serve para lidar precisamente com formas curvilíneas. Para isto deveriam ser utilizados domínios não retangulares e teriam que ser desenvolvidas estratégias para somar. Outro aprimoramento seria usar segmentos de linhas, como os usados nas CGLs, mas sub-dividindo todos os domínios em formas convexas através de uma regra de decomposição apropriada.

Em 1971, apresentou o trabalho "Algoritmos heurísticos para o planejamento do espaço automatizado" [08], na Segunda Conferência Internacional em Inteligência Artificial, realizada em Londres.

Na continuação de seus estudos, em 1971 Eastman [09] apresentou uma organização e princípios de operação de uma linguagem computacional e um programa que podia determinar a posição de equipamentos (divisões, móveis ou espaço necessário para o trabalho) numa sala. Além disto, incorporou algoritmos que poderiam resolver estes problemas. Chamou o sistema de "General Space Planner" (GSP), que foi implementado primeiramente em Algol e depois em FORTRAN IV, num computador IBM 360 modelo 67, na Universidade de Carnegie-Mellon.

Ao observar a resolução de uma variedade de problemas de planejamento do espaço, constatou que, às vezes, algumas variáveis eram desconhecidas em certos níveis, e em outros, eram definidas. Supondo então que uma variável poderia mudar durante um período de tempo, usou o termo "unidade de projeto" (DU de "*Design Unit*") para se referir a qualquer elemento com características temporariamente fixadas e "espaço" ao elemento cujas características estavam sendo manipuladas e definidas.

Em 1973 publicou outro artigo [10], sempre com resultados melhorados, mas ainda somente para problemas de "*lay-out*", que embora apresentem características diferentes dos problemas de encaixe

que ocorrem na indústria do vestuário, têm uma mesma lógica que é a de posicionar elementos no espaço levando-se em consideração tanto os espaços com elementos já posicionados como os espaços ainda não utilizados.

Em 1975, Adamowicz e Albano [02] desenvolveram um algoritmo, baseado nas técnicas de programação dinâmica, que não se preocupava apenas com a minimização do desperdício, mas também com o tempo de computação, evitando dessa forma procedimentos de busca exaustivos.

Como ponto de partida, observaram pessoas com experiência em encaixar manualmente formatos retangulares em grandes chapas, também retangulares, e notaram que geralmente elas tentavam arrumar os retângulos, subdividindo-os em grupos. Para simplificar o problema e agilizar a solução, muitas vezes elas tentavam agrupar séries de retângulos que tivessem pelo menos uma dimensão em comum, a fim de formar grandes tiras retangulares ou blocos. Além disto, às vezes elas voltavam atrás, retirando alguma parte do arranjo já feito e reposicionando os retângulos.

Ainda que tal caminho raramente proporcionasse uma solução ótima, as soluções geradas geralmente eram muito boas.

Com o objetivo de reduzir o tempo de computação e capacidade de armazenamento, ao desenvolver o algoritmo, Adamowicz e Albano adotaram um caminho similar.

A solução apresentada pelo algoritmo não representa o melhor encaixe **POSSÍVEL**, entretanto, em problemas com um grande número de retângulos numa grande variedade de tipos, as heurísticas, usadas para decidir a subdivisão, produziram uma solução próxima da solução ótima e extremamente eficiente em termos de tempo de computação.



ção.

Em 1983, Sanders e Bronsoiler [35] publicaram o desenvolvimento de um algoritmo para o encaixe de peças irregulares. O algoritmo desenvolvido aceita uma variedade de peças irregulares e quando é adicionada uma nova peça ao encaixe já existente, o algoritmo tenta as várias orientações possíveis, rotacionando-a, nos vários lugares possíveis ao longo do encaixe já existente. O preço pago pela flexibilidade é a necessidade de um tempo de computação considerável. Para cinco variedades de peças, o tempo requerido para cada chapa de encaixe, gira em torno de 3 minutos. Para situações onde o número de peças e a variedade é grande, como é o caso da indústria do vestuário, é preciso um método heurístico mais rápido.

Em 1984, Dov Dori e Moshe Ben-Bassat [05] publicaram um artigo em que apresentavam um algoritmo que fazia um encaixe eficiente de figuras convexas. O algoritmo baseia-se em dois passos principais. Primeiro, a figura convexa dada é circunscrita por um polígono convexo  $P_n$  com um número de lados suficientemente grande para que as perdas sejam negligenciáveis. Segundo, o polígono convexo  $P_n$  é circunscrito por um polígono paralelo, hexagonal ou pentagonal  $P^*$ , que é então duplicado e encaixado como num pavimento.

As limitações do algoritmo são, em primeiro lugar, o fato de que ele só é eficiente para polígonos hexagonais ou pentagonais paralelos. Poderiam ser obtidos melhores resultados se a figura original pudesse ser circunscrita por todos os tipos de polígonos paralelos existentes. Uma segunda limitação do algoritmo vem da suposição de que o plano a ser trabalhado é infinito, enquanto na

prática, a mesa onde as figuras serão cortadas, é de dimensões finitas; conseqüentemente, as perdas na margem também devem ser consideradas. Se a área da figura for pequena em relação à mesa, então esta perda advinda da margem pode ser negligenciável.

Este algoritmo é aplicável somente para o encaixe de figuras convexas. Esta desvantagem pode ser superada parcialmente com um prévio encaixe manual das poucas figuras diferentes, algumas delas não-convexas e/ou com buracos, de forma que o contorno externo da área resultante deste encaixe manual seja convexo.

A eficiência do algoritmo cai com o aumento no número de peças, de 93% para  $n=15$  e 87,2% para  $n=50$ . Esta eficiência considera o plano de trabalho infinito e, no caso da indústria do vestuário, os moldes não são tão pequenos em relação à mesa de corte, o que torna necessária a consideração das perdas nas margens.

Em 1985, H. Tokuyama e N. Ueno [41] desenvolveram um algoritmo heurístico que dividia o problema de encaixe em duas fases. Na primeira fase, são analisados os grupos de ordens de serviço e os grupos de material em estoque, encontrando-se os melhores pares "ordem de serviço/material em estoque". A seguir, encontra-se a seqüência ótima para as ordens de serviço. Na segunda fase, realizam-se os encaixes, na ordem definida pela fase I.

Este algoritmo gera soluções próximas da ótima, em tempo real, no entanto o encaixe é uni-dimensional e aplica-se apenas para peças de grandes seções, típicas da indústria do aço.

Em 1986, Weishuang Qu e Jerry L. Sanders [30] desenvolveram um algoritmo heurístico que fazia o encaixe automático de peças irregulares, gastando no máximo 25 segundos por chapa de encaixe.



Não foi usada a representação real da figura para que não fosse necessário armazenar no computador a geometria completa da peça irregular, diminuindo assim o tempo de computação e a memória necessária. Sempre que possível, a forma da peça foi reduzida a um retângulo.

A eficiência do encaixe gira em torno de 87,5% para peças retangulares e 80% para peças irregulares. O resultado é bastante promissor, principalmente se comparado com os algoritmos conhecidos na época, e a sugestão de não se armazenar todos os pontos da figura serão aproveitadas no desenvolvimento deste trabalho.

Em 1988, Robert W. Haessler [19], publicou um trabalho sobre como selecionar e projetar procedimentos heurísticos para a solução de problemas de encaixe, aplicáveis para a indústria do papel, onde escreveu uma lista de diretrizes gerais:

a) Nenhum procedimento heurístico simples é capaz de tratar todos os problemas de uma forma geral. A heurística tem que ser desenvolvida para uma aplicação específica;

b) O problema deve ser completamente definido. Todas as restrições têm que ser identificadas e todos os fatores que serão avaliados para gerar a solução têm que ser especificados. Devem ser entendidos alguns tratamentos econômicos de forma que possam ser avaliadas e classificadas mais de uma solução alternativa do mesmo problema;

c) Deve ser feito um julgamento, avaliando-se a importância e a dificuldade de se conseguir mais de uma solução com características diferentes;

d) Só se deve começar se o retorno financeiro parecer promissor. Na realidade, isto envolve duas questões. A mais óbvia

é a necessidade de se determinar se as soluções podem ser melhoradas o suficiente para justificar os gastos envolvidos no desenvolvimento e na implementação de um procedimento heurístico. A mais sutil é a disposição de se comprometer a desenvolver um procedimento numa forma de código de computador, ainda que todos os detalhes com respeito a como a heurística trabalhará não estejam disponíveis. Geralmente, não é possível saber exatamente o que se tem que fazer antes de se dispor de alguns resultados preliminares.

e) Deve-se coletar um conjunto representativo de problemas testados onde tenham sido obtidas as melhores soluções para o problema. É importante que o grupo de problemas testados seja extensivo o suficiente para cobrir as várias situações que o procedimento terá que ser capaz de manusear. Ter a melhor solução conhecida é importante pois impõe uma disciplina no processo de desenvolvimento, determinando uma qualidade específica para a solução que se procura.

f) Os resultados preliminares do procedimento heurístico inicial, quando disponíveis, devem ser usados para ajudar a definir e incrementar os requisitos para o procedimento heurístico final a ser desenvolvido. Este procedimento heurístico final será o resultado de um processo criativo que é melhorado continuamente durante o seu desenvolvimento.

Nos artigos encontrados, não fica evidente um aumento da complexidade dos problemas resolvidos, nem um progresso significativo nos resultados. Francis J. Vasko [42] publicou um artigo em 1989, onde trata do problema de encaixe para corte bi-dimensional realizável com guilhotina. Problemas desta complexidade já eram aborda-

dos em 1974 por Nicos Christofides e Charles Whitlock [04] com bons resultados para problemas de tamanho médio.

Nota-se a tendência do aparecimento de correntes de pensamento diversas, onde o segredo parece ser a alma do negócio.

Roberts [33] comenta que "é sabido que foram desenvolvidos alguns algoritmos para encaixe de peças irregulares, mas poucos foram publicados devido aos interesses comerciais. Os que são encontrados na literatura, ou são muito limitados e não podem ser usados para aplicações mais gerais, ou as características de seu desempenho não são investigadas completamente."

Como Eastman abordou técnicas heurísticas para a solução do problema e esta tendência foi seguida pelos outros pesquisadores, direcionou-se este trabalho neste mesmo sentido, partindo-se para um estudo da inteligência artificial.

Neste aspecto, a contribuição dos trabalhos de Eastman são de considerável valor.



### 3 O USO DA INTELIGÊNCIA ARTIFICIAL NA SOLUÇÃO DO PROBLEMA

A história da Inteligência Artificial<sup>3</sup> (IA), também conhecida como AI (do inglês "*Artificial Intelligence*"), confunde-se com a história da computação de um modo geral, tanto que os primeiros computadores receberam o apelido de "cérebro eletrônico". No entanto, o termo "inteligência artificial" só apareceu na década de 60 com o surgimento de programas que procuravam resolver problemas tais como jogos e prova de teoremas, que até então não eram exclusividade apenas do "homem", mas de homens com "inteligência privilegiada" [29].

O termo "Inteligência Artificial" é creditado a Marvin Minsky, do M.I.T., que em 1961 escreveu um artigo cujo título era "*Steps towards artificial intelligence*", publicado pelo "*Institute of Radio Engineers Proceedings 49*" de janeiro de 1961 [36].

Essas primeiras tentativas para se escrever programas buscando-se uma máquina mais "inteligente" foram feitas de maneira isolada, apenas por alguns pesquisadores, mais como uma curiosidade [36].

Esses estudos foram evoluindo, sempre ao nível de laboratório, sem maior expressão e sem aplicabilidade no mundo comercial, sendo colocados em segundo plano por administradores e financiadores de projetos, julgando-se um "castelo de cientista maluco"[23].

Tal postura foi vencida e a aplicabilidade da IA tem se tornado cada vez mais óbvia nos dias de hoje [31].

Até pouco tempo atrás, a maior parte da pesquisa que estava

3 - O Apêndice B contém maiores detalhes da Inteligência Artificial.

sendo realizada em IA era difícil de ser compreendida. Isto não se deve a uma dificuldade intrínseca da IA mas porque apresentava-se como o resultado de muitos anos de pesquisa de acadêmicos e engenheiros, cujo interesse maior era a comunicação com seus pares, que já conheciam muito sobre o assunto, ficando de fora o principiante interessado [23].

Felizmente esta situação mudou e hoje em dia as técnicas de inteligência artificial ocupam lugar de destaque sempre que se fala em desenvolvimento de tecnologia de ponta.

Mas o que vem a ser exatamente inteligência artificial? Na falta de uma definição universalmente aceita, serão apresentadas algumas delas, o que poderá auxiliar no entendimento do que vem a ser o conceito de IA.

- Segundo Marvin Minsky [27], "a IA é a ciência que estuda um modo para que as máquinas façam coisas que iriam requerer inteligência, se feitas pelo homem."

- Elaine Rich [32] inicialmente a define como o "estudo de como fazer os computadores realizarem tarefas em que, no momento, as pessoas são melhores", e depois como "o estudo de técnicas destinadas a resolver problemas exponencialmente difíceis, utilizando conhecimentos a respeito do domínio do problema".

- Robert I. Levine [25] a define como sendo "simplesmente uma maneira de fazer o computador pensar inteligentemente." Isto é conseguido estudando a maneira como as pessoas pensam quando estão tentando tomar decisões ou resolver problemas; uma vez identificado o processo de raciocínio, ele é dividido em etapas básicas, fazendo-se a seguir um programa de computador que utilize estas mesmas etapas ao tentar resolver um problema. Em resumo, a inteligência artificial fornece um método simples e estruturado de se pro-



jetar programas complexos de tomada de decisão.

- Herbert Schildt [36] define um programa inteligente como "aquele que exhibe um comportamento similar ao comportamento humano quando confrontado com o mesmo problema, não sendo necessário que o programa realmente resolva, ou tente resolver, o problema da mesma maneira que um humano o faria".

- Heinz Schawrtzel [37] comenta que "a IA proporciona um apoio valioso onde o cérebro humano encontra dificuldades e o computador se sobressai, particularmente devido a alta velocidade de processamento de grande quantidade de dados.

Enfim, o objetivo central da IA é entender os princípios da inteligência e tornar os computadores mais úteis [31].

Com o seu desenvolvimento, surge a necessidade de um novo profissional, que seria alguém capaz de entender os diversos métodos de armazenar o conhecimento e recuperá-lo de acordo com suas relações e propriedades; este profissional recebe o nome de "engenheiro do conhecimento".

A base fundamental da ciência dos computadores e da programação tem sido formada pelos *algoritmos*.

Um algoritmo pode ser definido como uma organização lógica de todos os passos necessários para a solução de um problema, assim como de todos os seus aspectos [25] o que vem a ser uma lista de instruções para dar uma solução segura para um problema, ou indicar que não existe solução.

Quando um humano vai resolver um problema mais complexo, ele não utiliza um procedimento similar ao de um algoritmo. Na realidade aplicam-se várias regras que se sabe, pela experiência, que funcionam. Estas regras fazem com que se chegue mais rápido à solução e seu uso diferencia uma pessoa com prática de um iniciante.

Da mesma forma, ao se fazer um programa de computador que resolva problemas complexos com eficiência, muitas vezes é necessário desistir da estrutura rígida dos algoritmos e construir, sobre eles, uma estrutura de controle sob a forma de regras, que não tenha mais a garantia de encontrar a melhor resposta, mas que quase sempre encontre uma resposta muito boa.

Estas regras que criam a possibilidade de se chegar mais perto de uma solução são conhecidas como heurísticas.

A palavra heurística vem da palavra grega "*heuriskein*", que significa "descobrir" [32].

Uma maneira muito usada para a implementação de estratégias de busca é a travessia de uma "árvore" [32]. Esta estrutura de árvore é constituída por nós e cada nó pode gerar um conjunto de nós sucessores, que por sua vez, podem ser expandidos, continuando até que seja encontrado um nó que represente a solução.

Numa busca dita "exaustiva", todos os nós são pesquisados. O problema é que a cada novo nó adicionado à árvore, aumenta o número de possíveis soluções. E como este aumento não é linear, num certo ponto, existirão muitas possibilidades a serem pesquisadas e cai-se no problema de "explosão combinatorial".

No caso do problema do encaixe automático, onde são analisadas as possíveis maneiras de se arranjar os moldes, tem-se que, para  $N$  moldes serão possíveis  $N!$  combinações. Então, para 4 moldes seriam possíveis 24 tipos de encaixes, e para 10 moldes haveria a possibilidade de 3.628.800 encaixes!

Sem a heurística a maioria dos programas atuais cairia numa explosão combinatorial. Isto por si só poderia ser um argumento suficiente a favor de sua utilização. Mas há ainda outro argumento:



- Raramente se precisa da solução ótima; basta que ela satisfaça e servirá muito bem. Na verdade, há alguma evidência de que as pessoas, quando resolvem problemas, não exigem obrigatoriamente uma solução ótima, mas aceitam soluções que *satisfizem* de uma maneira razoável. Em outras palavras, elas buscam qualquer solução que satisfaça um conjunto de exigências, e tão logo encontrem uma, a busca é interrompida. Exemplificando, ao realizar o encaixe manual, o profissional humano observa o aproveitamento do encaixe obtido. Se for igual ou maior que o segundo, ele poderá aceitar este encaixe, ainda que um melhor aproveitamento pudesse ser obtido, porque esta melhoria no aproveitamento pode não compensar o tempo gasto ao se fazer um novo encaixe.

Existem certas áreas de problemas para os quais, pelo menos atualmente, os algoritmos são incapazes de fornecer quaisquer soluções, só sendo possível uma abordagem destes problemas através de heurísticas. Tais áreas incluem toda a gama de problemas tais como fixação de horários, planejamento de rotas, encaixe de peças, etc...

Heurísticas não são capazes de garantir uma solução para um problema e podem não indicar quando a solução não existe, mas podem ser usadas para uma grande variedade de situações. Além disto, quando elas conduzirem a uma solução, isto se dará em muito menos tempo do que um algoritmo gastaria para o mesmo problema.

Mesmo que seja difícil aplicar o método heurístico em alguns casos, pelo menos ele pode dar uma alternativa para quando o método algorítmico falha.

A heurística é uma ferramenta imprescindível na solução dos problemas humanos e, se os computadores forem realmente se transformar em máquinas pensantes, deverão, inevitavelmente, empregar

os métodos de busca heurística, lado a lado com os algoritmos exatos [23].

Segundo Elaine Rich [32], a resolução de problemas de inteligência artificial pode ser descrita mais precisamente como um processo de busca heurística.

Num programa de encaixe automático é muito importante que haja flexibilidade, isto é, deve ser possível a adição de novas regras, ou a eliminação de algumas, de acordo com o tipo da indústria que está sendo a usuária do sistema.

Um programa comum de computador só pode fornecer respostas aos problemas para os quais está especificamente programado. Se um programa comum precisar ser modificado para acomodar novas informações, todo o programa terá que ser analisado visando a introdução destas possíveis modificações. Isto não toma apenas tempo como também pode afetar outras partes do programa, de maneira desfavorável, podendo resultar em erros.

E a mente humana pode incorporar novos conhecimentos sem alterar seu funcionamento e sem interferir em todos os outros fatos que já estão armazenados no cérebro.

Um programa de IA funciona de forma similar. Mudanças feitas em programas de inteligência artificial são muito mais simples de implementar do que aquelas feitas em programas convencionais [25].

Comparando-se esses dois tipos de programação, tem-se [31]:

a) programação convencional

- primariamente, processamento numérico;
- soluções algorítmicas;
- estruturas de controle e de conhecimento integradas;
- difícil modificação e atualização;



- só a resposta correta interessa;
- só a melhor solução satisfaz.

b) programação em Inteligência Artificial

- primariamente tem processamento simbólico;
- soluções heurísticas;
- estrutura de controle separada do domínio de conhecimento;
- fácil de se modificar e de se atualizar;
- algumas respostas erradas são toleradas;
- respostas satisfatórias usualmente são aceitas.

Dentro da programação em inteligência artificial, uma área onde têm sido obtidos excelentes resultados, principalmente do ponto de vista comercial, são os sistemas especialistas.

Sistemas especialistas (S.E.) são sistemas de armazenamento e acesso de conhecimento especializado (cuja forma de raciocínio simula a humana), empregados na resolução de problemas complexos e com muitos dados, que requereriam um especialista humano para resolvê-lo.

Sistema especialista é a tradução mais consagrada para "*expert system*", embora haja quem use "sistema perito". O perito é, mais apropriadamente, quem faz perícias, isto é, um perito é normalmente um especialista, mas nem todos os especialistas fazem perícias [45].

Basicamente os S.E. são constituídos por três componentes [40]:

- a) Base de conhecimento;
- b) Máquina de inferência e

c) Interface com o usuário.

A base de conhecimento é o elemento onde estão codificadas todas as informações relativas ao universo do conhecimento para o qual o sistema foi desenvolvido. Podem ser expressas por fatos ou procedimentos padrão, ou pelas regras utilizadas por especialistas humanos na resolução dos problemas.

A máquina de inferência é um mecanismo que acessa as estruturas de conhecimento, buscando gerar soluções para um problema em particular.

Inferir significa deduzir pelo raciocínio. Quando se alcança um objetivo, não se está apenas resolvendo um problema imediato; também se está adquirindo novos conhecimentos.

A tarefa da máquina de inferência é selecionar e então aplicar a regra mais apropriada em cada passo da execução do S.E..

Há sistemas que operam a inferência usando técnicas "forward", enquanto outros utilizam-se de técnicas "backward". Existe ainda a possibilidade de serem utilizadas ambas as técnicas [03].

Na técnica "backward", o sistema parte do objetivo (como, por exemplo, a identificação de uma espécie de planta) e pesquisa a cadeia de premissas de todos os fatos envolvidos (como, por exemplo, características botânicas), que permite chegar a uma conclusão (por exemplo, o nome da espécie).

Na técnica "forward", o sistema trabalha com peças elementares de informação (como o lote de moldes a serem encaixados) e tenta seguir para o objetivo, combinando estes elementos (de modo que os espaços entre eles sejam os menores possíveis).

Muitos sistemas que empregam a técnica "backward" são usados



para propósitos de diagnóstico; eles iniciam com algum objetivo (uma lista de sintomas, por exemplo) e tentam descobrir quais premissas podem ocasionar tais sintomas. Muitos sistemas que utilizam técnica "forward" são usados para o propósito de construção: eles partem de uma série de premissas, combinando-as para produzir algum resultado final.

Além das técnicas citadas, existem diversos processos para se fazer a inferência, pois ela está diretamente ligada à *representação do conhecimento*<sup>4</sup>. Contudo, o método mais usado é o da avaliação de regras, onde o mecanismo de inferência busca as regras na base de conhecimento e as avalia. Esta busca depende dos fatos e das hipóteses que existem e que se quer determinar a cada momento [31].

Em muitos casos a escolha de qual técnica o sistema deverá executar é feita reproduzindo-se a maneira utilizada por uma pessoa ao resolver o problema [03].

Para que o S.E. seja uma ferramenta eficaz, as pessoas deverão ser capazes de interagir com ele facilmente. Em muitos dos domínios onde os S.E. operam, as pessoas não aceitarão os resultados a menos que estejam convencidas da precisão do processo de raciocínio que produziu estes resultados. Assim, é importante que o processo de raciocínio, utilizado nesses programas, possua passos compreensíveis e que se tenha disponível o conhecimento a respeito do processo de raciocínio, para que possam ser geradas as explicações dos passos [32].

O quadro negro, também denominado rascunho, é uma área da memória onde o sistema vai gravando e apagando os dados que vai

4 - O Apêndice B trata de algumas formas de representação do conhecimento.

usando no processo de inferência, até chegar a uma solução.

Para se chegar a uma solução, é necessário que se recuperem certas regras da base de conhecimento numa área de trabalho da memória, onde são ordenadas em uma nova ordem para que sejam avaliadas. As conclusões dessas regras irão gerar novos fatos e novas hipóteses que serão armazenadas no quadro-negro [31].

Na FIGURA 10 tem-se o esquema básico da arquitetura de um sistema especialista.

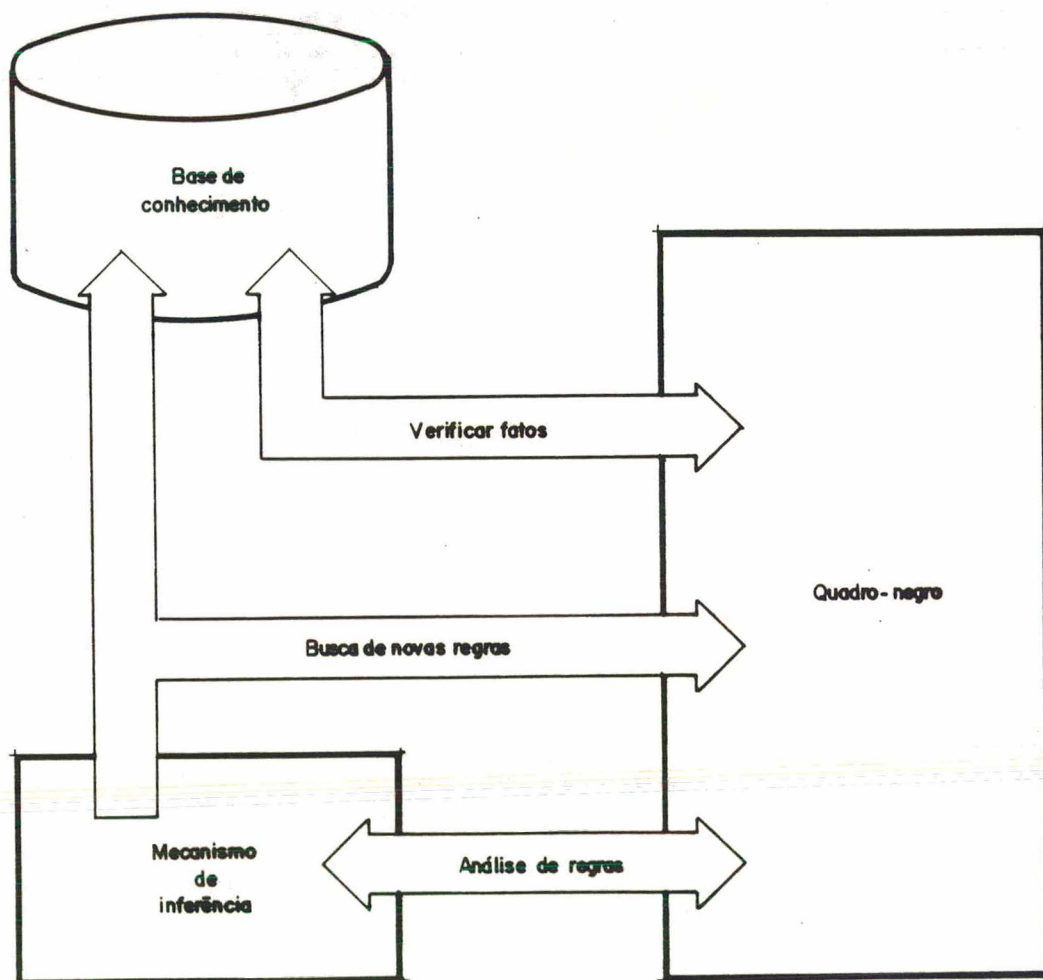


FIGURA 10 - Elementos básicos de um sistema especialista.



Como os sistemas especialistas derivam seu poder da riqueza das bases de conhecimento que exploram, é extremamente importante que essas bases sejam tão completas e confiáveis quanto possível. Muitas vezes, entretanto, não existe nenhuma codificação padrão deste conhecimento, ou seja, só o especialista humano o conhece. Assim, a única maneira de levar este conhecimento para o programa será pela interação com o especialista humano.

O engenheiro do conhecimento deve trabalhar com o especialista da área em que se deseja desenvolver o sistema, devendo perguntar diretamente qual regra ou método foi utilizado para a solução de um problema particular do domínio.

Os especialistas normalmente têm grande dificuldade em exprimir tais regras.

Um aspecto interessante que não pode ser esquecido, chamado de *paradoxo da perícia*, é o seguinte [03]:

**"O perito mais competente será o menos capaz de descrever o conhecimento que usa para resolver problemas."**

Por isto, as informações fornecidas pelos especialistas deverão ser testadas, para que se verifique sua praticidade.

Se um problema for tão difícil que nenhum humano saiba como iniciar, nenhum sistema de computador poderá igualmente resolvê-lo. Se o problema requerer julgamentos intuitivos sobre situações novas - como calcular o potencial de mercado para uma nova invenção ou um roteiro de novela - então a intuição não poderá ser for-

malizada num sistema especialista.

As aplicações ótimas para tais sistemas são aquelas que requerem grande volume de conhecimento bem definido e formalizável.

A quantidade de conhecimento exigida depende da tarefa. Pode oscilar de dezenas de regras a milhares delas. A TABELA 1 reflete uma apreciação de Hayes-Roth [20]:

OBJETIVO	NÚMERO DE REGRAS
Demonstração de uma tecnologia	50
Demonstração convincente de um sistema	250
Sistema comercial prático	menos que 50
Desempenho equivalente a um especialista humano numa área limitada	de 500 a 1000
Conhecimento a nível profissional	10.000
Limite do conhecimento humano especializado	100.000

TABELA 1 - Quantidade de regras necessárias.

Um sistema especialista não pode ser construído rapidamente. Com a tecnologia disponível atualmente, são necessários recursos da ordem de 5 a 10 homens-ano para resolver um problema de complexidade moderada [03].

A meta final da IA é criar inteligência do tipo humano em uma máquina. Se isto pode ser atingido num período razoável de tempo, ou se será eventualmente conseguido, não é uma questão que altere a importância prática de se produzir programas que levem, cada vez mais adiante, no caminho desta meta. A nova geração de programas, que a IA está produzindo, deve constituir o início de uma modificação gradual da maneira atual das máquinas procederem para uma maneira mais flexível e inteligente [23].

## 4 SISTEMA PROPOSTO

O desenvolvimento do sistema consistiu na implementação de heurísticas consideradas básicas.

Constatou-se na prática o fato levantado por alguns autores, de que é difícil para um especialista detalhar as regras que ele criou e/ou adotou para desempenhar suas tarefas.

Os profissionais que realizam o encaixe manual consultados, não souberam definir uma única regra que ajudasse a definir as heurísticas. Portanto, para sua execução, utilizou-se a criatividade e o bom-senso porque não havia "conhecimento especializado" em forma de regras à disposição.

A linguagem utilizada<sup>5</sup> foi C, para que o sistema possa ser incorporado ao projeto SAPRO - Sistema de Apoio à Produção - em desenvolvimento no GRUCON, e que trata, entre outras coisas, de toda a parte gráfica interativa do encaixe.

### 4.1 - DEFINIÇÃO DAS HEURÍSTICAS

Foram definidas quatro heurísticas básicas, a saber:

#### Heurística 1:

A ordem em que os moldes são colocados no encaixe tem uma influência direta na eficiência deste. Logo, é necessário que seja dada uma prioridade aos moldes mais "fáceis" de serem encaixados. Para tal, procede-se da seguinte maneira:

Circunscreve-se o molde com um retângulo, obtendo-se um "en-

5 - O uso da linguagem C para a IA é tratado no Apêndice B.

velope" e calcula-se o aproveitamento deste pela fórmula:

$$\eta_{env.} = \frac{\text{Área do molde}}{\text{Área do retângulo}} \quad (01)$$

Desta forma, os moldes de maior  $\eta_{env.}$  serão os primeiros a serem encaixados.

Durante o encaixe, o algoritmo trabalha com o molde e não com o seu envelope. Uma vez ordenados os moldes, não existe mais o envelope.

#### Heurística 2:

Esta heurística faz com que o encaixe seja realizado na direção vertical, pois a largura do enfiesto é constante, enquanto o comprimento é variável. Quanto menor o comprimento do encaixe obtido, melhor o aproveitamento. Se o encaixe fosse realizado primeiro na direção horizontal, não haveria garantia do menor comprimento possível.

#### Heurística 3:

Após a alocação de cada molde, calcula-se o aproveitamento em relação ao espaço vertical ainda disponível ao redor do molde.

Se não existir nenhum espaço vago, é sinal que o molde está encaixado com um aproveitamento ótimo e parte-se para a alocação de um novo molde.

Se existir um espaço onde caiba o próximo molde da fila a ser alocado, o encaixe o posiciona e repete esta heurística.

Se o espaço existente for menor que o próximo molde da fila,



procura-se outro molde menor que caiba neste espaço. Caso este molde não seja encontrado, retira-se o molde já alocado e procura-se um grupo de moldes que aproveite melhor este espaço.

#### Heurística 4:

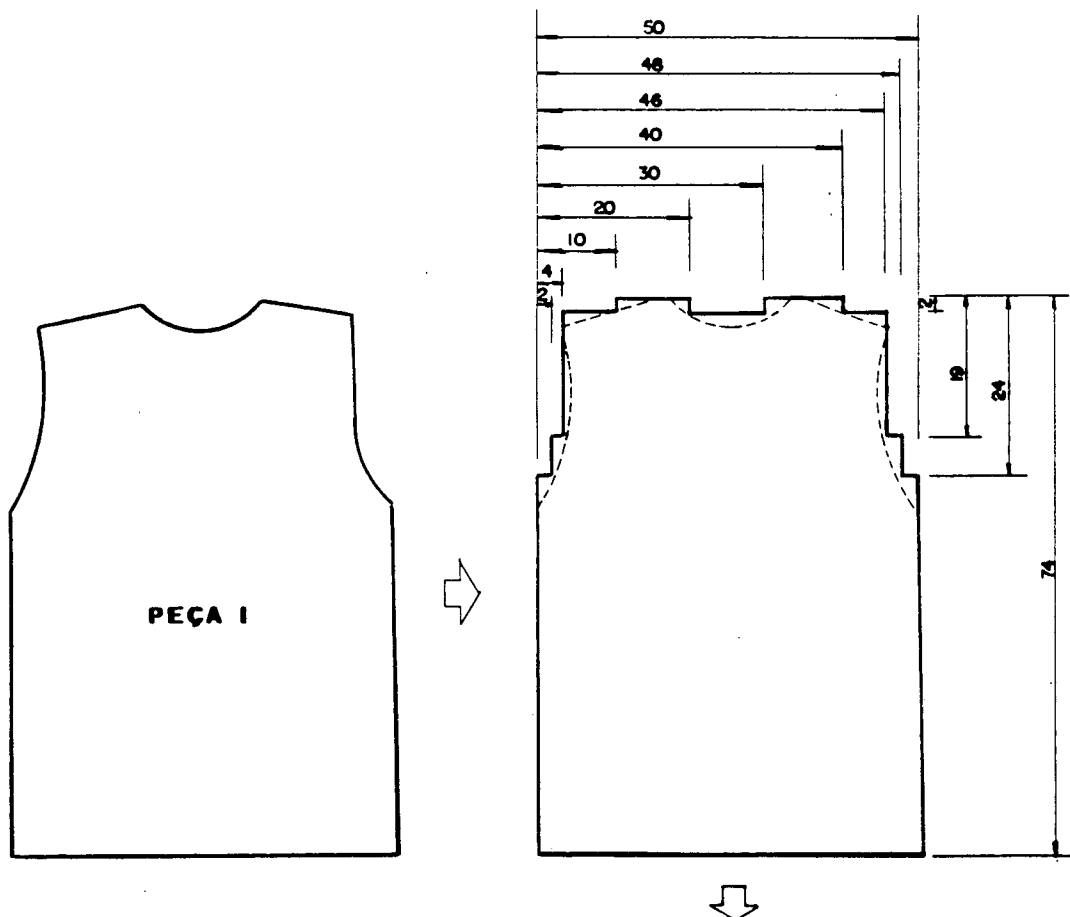
Para cada nova alocação de um molde, o algoritmo deve percorrer todo o encaixe, procurando um lugar. Acontece que onde o encaixe já está repleto de moldes, é perda de tempo procurar um espaço. Para evitar este desperdício de trabalho, estipula-se um comprimento padrão, 1,5 m por exemplo, e a cada intervalo deste comprimento o algoritmo faz o cálculo do aproveitamento que já tem. Se for melhor ou igual ao aproveitamento estipulado, ele assume que não vai mais procurar espaços lá. Mas para se assegurar que a eficiência do encaixe será a melhor possível, ainda tenta alocar os menores moldes disponíveis neste espaço aparentemente já lotado. Feito isto, esta parte do encaixe não será mais pesquisada.

#### **4.2 - IMPLEMENTAÇÃO DO ALGORITMO PROPOSTO**

As representações dos moldes e dos encaixes serão feitas por matrizes, pois se os moldes fossem armazenados ponto por ponto, haveria necessidade de muita memória e o tempo de execução seria muito elevado.

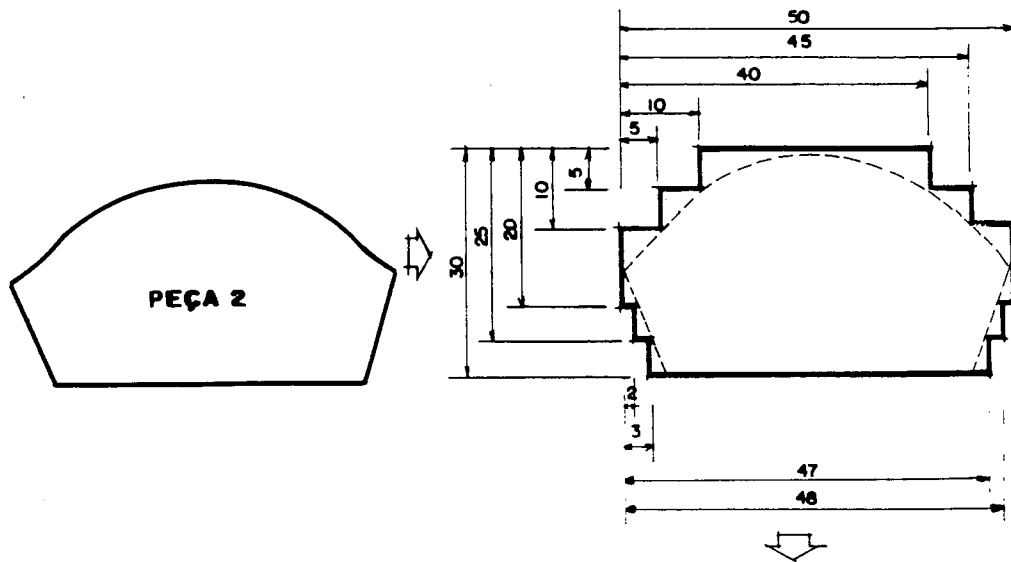
Uma vez que as peças não serão rotacionadas, pois a possível melhoria da eficiência do encaixe acarretaria num aumento de tempo de resposta indesejado, a restrição da direção do fio do tecido fica amarrada na matriz de entrada do molde.

O uso destas matrizes foi feito baseado nos artigos de Eastman [07], [08], [09] e [10] e pode ser melhor compreendido pelas FIGURAS 11 e 12.



	0	1	2	3	4	5	6	7	8	9	10
0		2,0	4,0	10,0	20,0	30,0	40,0	46,0	48,0	50,0	0
1	2,0	1	1	1	901	1	901	1	1	1	999
2	19,0	1	1	901	901	901	901	901	1	1	999
3	24,0	1	901	901	901	901	901	901	901	1	999
4	74,0	901	901	901	901	901	901	901	901	901	999
5	0	999	999	999	999	999	999	999	999	999	999

FIGURA 11 - Representação do molde "peça 1" e sua respectiva matriz.



	0	1	2	3	4	5	6	7	8	9	10
0		2	3	5	10	40	45	47	48	50	0
1	5					902					999
2	10				902	902	902				999
3	20	902	902	902	902	902	902	902	902	902	999
4	25		902	902	902	902	902	902	902		999
5	30			902	902	902	902	902			999
6	0	999	999	999	999	999	999	999	999	999	999

FIGURA 12 - Representação do molde "peça 2" e sua respectiva matriz.

Este sistema é um módulo de um projeto maior, o SAPRO, já citado anteriormente, sendo que o núcleo do objetivo deste trabalho está na obtenção do encaixe automático das peças propostas. As interfaces de entrada e saída dos dados foram criadas apenas para se testar os encaixes gerados pelo sistema. Sabe-se que esta forma de entrada e saída dos dados será uma limitação, mas não se trata aqui de resolver este problema.

Uma vez definidas as matrizes dos respectivos moldes, os dados são fornecidos ao sistema via teclado.

O aproveitamento mínimo desejado também é introduzido via teclado e seu valor é estipulado pelo usuário, levando-se em consideração o tipo de roupa que está sendo feita. Durante as visitas feitas para o levantamento de dados, verificou-se que este aproveitamento pode variar muito, por exemplo, na indústria de calças compridas, é razoável pedir um aproveitamento mínimo de 95%, enquanto no caso da indústria de calcinhas e "soutiens", o aproveitamento mínimo gira em torno de 60%.

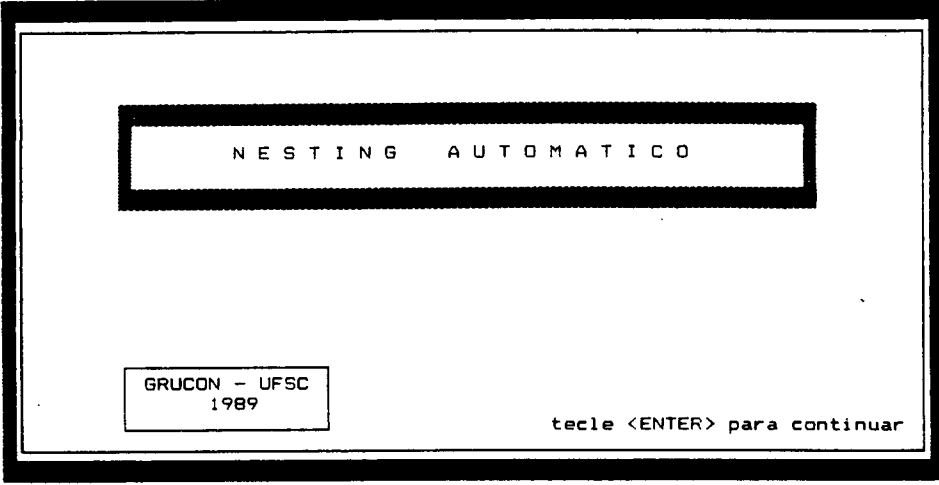
Com as interfaces criadas para os testes, o sistema apresenta telas para a entrada de dados de cada molde e do aproveitamento mínimo, e à medida que o encaixe vai evoluindo, as linhas e as colunas da matriz gerada são apresentadas seqüencialmente, sem que se tenha uma visualização do encaixe.

Para o encaixe de 10 moldes "peça 1", representado na FIGURA 11, e 10 moldes "peça 2", representado na FIGURA 12, o sistema mostraria as telas apresentadas nas FIGURAS 13, 14 e 15.

A FIGURA 13 apresenta três telas. A primeira simplesmente identifica o sistema. Através da segunda tela são introduzidos dados como comprimento e largura do enfiesto, aproveitamento mínimo estipulado e número de peças. Uma vez fornecidos estes dados, o



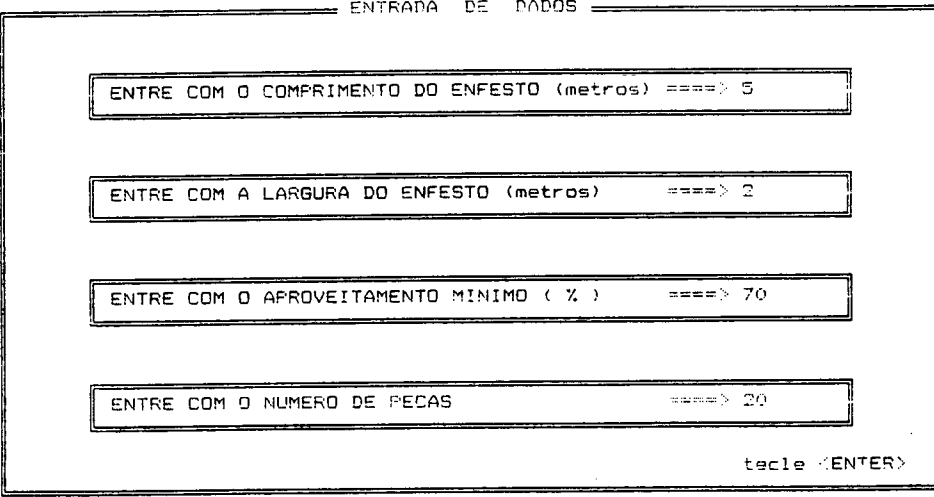
sistema pergunta o número de linhas e de colunas da matriz representativa de cada peça.



NESTING AUTOMATICO

GRUCON - UFSC  
1989

tecle <ENTER> para continuar



ENTRADA DE DADOS

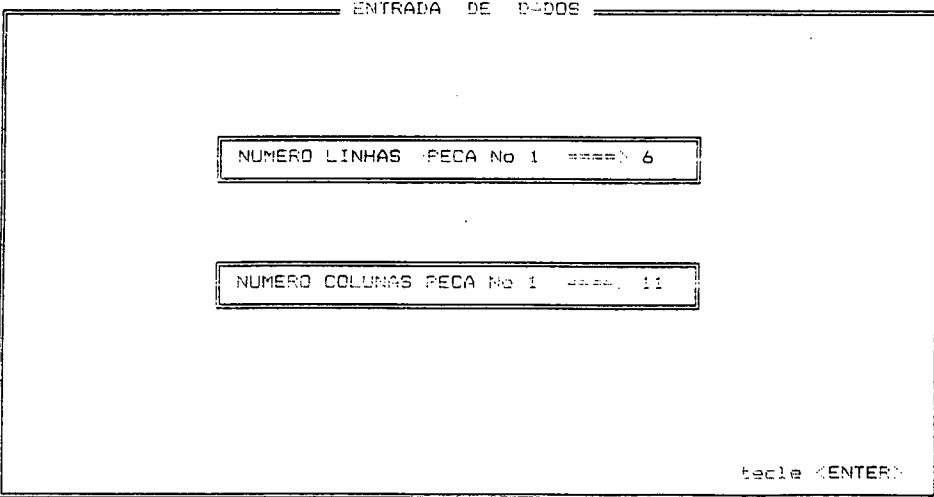
ENTRE COM O COMPRIMENTO DO ENFESTO (metros) ====> 5

ENTRE COM A LARGURA DO ENFESTO (metros) ====> 2

ENTRE COM O APROVEITAMENTO MINIMO ( % ) ====> 70

ENTRE COM O NUMERO DE PECAS ====> 20

tecle <ENTER>



ENTRADA DE DADOS

NUMERO LINHAS PECA No 1 ====> 6

NUMERO COLUNAS PECA No 1 ====> 11

tecle <ENTER>

FIGURA 13 - Tela inicial do sistema e telas de entrada de dados.

A FIGURA 14 mostra a entrada do número de linhas e colunas das matrizes das peças n.º2, n.º3 e n.º4, todas iguais (serão encaixados 10 moldes "peça 1"), e por isto, as telas com a entrada destes dados para as peças n.º5 a n.º10 não precisam ser apresentadas.

A FIGURA 15 representa as telas de entrada do número de linhas e colunas para as peças n.º11, n.º12 e n.º13, que são do tipo "peça 2". Pelo mesmo motivo apresentado anteriormente, não há necessidade de se mostrar as telas de entrada de dados das peças n.º14 a n.º20.

Nas FIGURAS 16 e 17, tem-se a entrada da matriz da peça n.º1. As matrizes das peças n.º2 a n.º10 são idênticas à matriz da peça n.º1, não havendo então, necessidade de se repetir as telas do sistema.

Note-se que nas FIGURAS 11 e 12, os moldes estão com suas dimensões em centímetros, que é a dimensão utilizada na indústria têxtil. E da FIGURA 16 em diante, o sistema está trabalhando em metros. Isto não chega a ser uma limitação, visto que a mudança se dá apenas na localização do ponto decimal.

As FIGURAS 18, 19 e 20 representam as telas de entrada da matriz da peça n.º11, que é a mesma para as peças de n.º12 a n.º20.

As FIGURAS 21 a 26 mostram as telas necessárias para representar a matriz do encaixe que o sistema gerou, depois de encaixadas as três primeiras peças.

Neste teste o sistema encaixou todas as vinte peças propostas. Para facilitar a leitura deste trabalho, é apresentada apenas a matriz do encaixe das três primeiras peças. Estas 18 telas podem ser visualizadas na FIGURA 27, onde se nota que foram encaixadas duas peças do tipo "peça 1" e uma peça do tipo "peça 2", que era o que se esperava que o sistema executasse.

ENTRADA DE DADOS

NUMERO LINHAS PEÇA No 2 =====> 6

NUMERO COLUNAS PEÇA No 2 =====> 11

tecle <ENTER>

ENTRADA DE DADOS

NUMERO LINHAS PEÇA No 3 =====> 6

NUMERO COLUNAS PEÇA No 3 =====> 11

tecle <ENTER>

ENTRADA DE DADOS

NUMERO LINHAS PEÇA No 4 =====> 6

NUMERO COLUNAS PEÇA No 4 =====> 11

tecle <ENTER>

FIGURA 14 - Entrada do número de linhas e colunas das peças n.º2,  
n.º3 e n.º4.

ENTRADA DE DADOS

NUMERO LINHAS PECA No 11 <====>7

NUMERO COLUNAS PECA No 11 <====>11

tecle <ENTER>

ENTRADA DE DADOS

NUMERO LINHAS PECA No 12 <====>7

NUMERO COLUNAS PECA No 12 <====>11

tecle <ENTER>

ENTRADA DE DADOS

NUMERO LINHAS PECA No 13 <====>7

NUMERO COLUNAS PECA No 13 <====>11

tecle <ENTER>

FIGURA 15 - Entrada do número de linhas e colunas das peças n.º11,  
n.º12 e n.º13.



ENTRADA DE DADOS

PEÇA No 1

```

posicao[0][0] ====> 0
posicao[0][1] ====> .02
posicao[0][2] ====> .04
posicao[0][3] ====> .10
posicao[0][4] ====> .20
posicao[0][5] ====> .30
posicao[0][6] ====> .40
posicao[0][7] ====> .46
posicao[0][8] ====> .48
posicao[0][9] ====> .50
posicao[0][10] ====> 0

```

tecle <ENTER>

ENTRADA DE DADOS

PEÇA No 1

```

posicao[1][0] ====> .02
posicao[1][1] ====> 1
posicao[1][2] ====> 1
posicao[1][3] ====> 1
posicao[1][4] ====> 901
posicao[1][5] ====> 1
posicao[1][6] ====> 901
posicao[1][7] ====> 1
posicao[1][8] ====> 1
posicao[1][9] ====> 1
posicao[1][10] ====> 999

```

tecle <ENTER>

ENTRADA DE DADOS

PEÇA No 1

```

posicao[2][0] ====> .19
posicao[2][1] ====> 1
posicao[2][2] ====> 1
posicao[2][3] ====> 901
posicao[2][4] ====> 901
posicao[2][5] ====> 901
posicao[2][6] ====> 901
posicao[2][7] ====> 901
posicao[2][8] ====> 1
posicao[2][9] ====> 1
posicao[2][10] ====> 999

```

tecle <ENTER>

FIGURA 16 - Entrada de dados da matriz da peça n.º1.

ENTRADA DE DADOS

PEÇA No 1

```

posicao[3][0]  ====> .24
posicao[3][1]  ====> 1
posicao[3][2]  ====> 901
posicao[3][3]  ====> 901
posicao[3][4]  ====> 901
posicao[3][5]  ====> 901
posicao[3][6]  ====> 901
posicao[3][7]  ====> 901
posicao[3][8]  ====> 901
posicao[3][9]  ====> 1
posicao[3][10] ====> 999

```

tecle <ENTER>

ENTRADA DE DADOS

PEÇA No 1

```

posicao[4][0]  ====> .74
posicao[4][1]  ====> 901
posicao[4][2]  ====> 901
posicao[4][3]  ====> 901
posicao[4][4]  ====> 901
posicao[4][5]  ====> 901
posicao[4][6]  ====> 901
posicao[4][7]  ====> 901
posicao[4][8]  ====> 901
posicao[4][9]  ====> 901
posicao[4][10] ====> 999

```

tecle <ENTER>

ENTRADA DE DADOS

PEÇA No 1

```

posicao[5][0]  ====> 0
posicao[5][1]  ====> 999
posicao[5][2]  ====> 999
posicao[5][3]  ====> 999
posicao[5][4]  ====> 999
posicao[5][5]  ====> 999
posicao[5][6]  ====> 999
posicao[5][7]  ====> 999
posicao[5][8]  ====> 999
posicao[5][9]  ====> 999
posicao[5][10] ====> 999

```

tecle <ENTER>

FIGURA 17 - Continuação da entrada de dados da matriz da peça nº1.

ENTRADA DE DADOS

PEÇA No 11

```

posicao[01][01]  ==>  0
posicao[01][11]  ==> .02
posicao[01][21]  ==> .03
posicao[01][31]  ==> .05
posicao[01][41]  ==> .10
posicao[01][51]  ==> .40
posicao[01][61]  ==> .45
posicao[01][71]  ==> .47
posicao[01][81]  ==> .48
posicao[01][91]  ==> .50
posicao[01][101] ==>  0

```

tecle <ENTER>

ENTRADA DE DADOS

PEÇA No 11

```

posicao[11][01]  ==> .05
posicao[11][11]  ==>  1
posicao[11][21]  ==>  1
posicao[11][31]  ==>  1
posicao[11][41]  ==>  1
posicao[11][51]  ==> 911
posicao[11][61]  ==>  1
posicao[11][71]  ==>  1
posicao[11][81]  ==>  1
posicao[11][91]  ==>  1
posicao[11][101] ==> 999

```

tecle <ENTER>

ENTRADA DE DADOS

PEÇA No 11

```

posicao[21][01]  ==> .10
posicao[21][11]  ==>  1
posicao[21][21]  ==>  1
posicao[21][31]  ==>  1
posicao[21][41]  ==> 911
posicao[21][51]  ==> 911
posicao[21][61]  ==> 911
posicao[21][71]  ==>  1
posicao[21][81]  ==>  1
posicao[21][91]  ==>  1
posicao[21][101] ==> 999

```

tecle <ENTER>

FIGURA 18 - Entrada de dados da matriz da peça n.º11.

ENTRADA DE DADOS

PEÇA No 11

```

posicao[3][0]  ==> .20
posicao[3][1]  ==> 911
posicao[3][2]  ==> 911
posicao[3][3]  ==> 911
posicao[3][4]  ==> 911
posicao[3][5]  ==> 911
posicao[3][6]  ==> 911
posicao[3][7]  ==> 911
posicao[3][8]  ==> 911
posicao[3][9]  ==> 911
posicao[3][10] ==> 999

```

tecle <ENTER>

ENTRADA DE DADOS

PEÇA No 11

```

posicao[4][0]  ==> .25
posicao[4][1]  ==> 1
posicao[4][2]  ==> 911
posicao[4][3]  ==> 911
posicao[4][4]  ==> 911
posicao[4][5]  ==> 911
posicao[4][6]  ==> 911
posicao[4][7]  ==> 911
posicao[4][8]  ==> 911
posicao[4][9]  ==> 1
posicao[4][10] ==> 999

```

tecle <ENTER>

ENTRADA DE DADOS

PEÇA No 11

```

posicao[5][0]  ==> .20
posicao[5][1]  ==> 1
posicao[5][2]  ==> 1
posicao[5][3]  ==> 911
posicao[5][4]  ==> 911
posicao[5][5]  ==> 911
posicao[5][6]  ==> 911
posicao[5][7]  ==> 911
posicao[5][8]  ==> 1
posicao[5][9]  ==> 1
posicao[5][10] ==> 999

```

tecle <ENTER>

FIGURA 19 - Continuação da entrada de dados da matriz da peça



ENTRADA DE DADOS

PEÇA No 11

posicao[6][0]	====>	0
posicao[6][1]	====>	999
posicao[6][2]	====>	999
posicao[6][3]	====>	999
posicao[6][4]	====>	999
posicao[6][5]	====>	999
posicao[6][6]	====>	999
posicao[6][7]	====>	999
posicao[6][8]	====>	999
posicao[6][9]	====>	999
posicao[6][10]	====>	999

tecle <ENTER>

FIGURA 20 - Continuação da entrada de dados da matriz da peça  
nº11.

```

MATRIZ ENFESTO

posicao[0][0] = 0.00
posicao[0][1] = 0.03
posicao[0][2] = 0.04
posicao[0][3] = 0.05
posicao[0][4] = 0.06
posicao[0][5] = 0.11
posicao[0][6] = 0.21
posicao[0][7] = 0.31
posicao[0][8] = 0.41
posicao[0][9] = 0.46
posicao[0][10] = 0.47
posicao[0][11] = 0.48
posicao[0][12] = 0.49
posicao[0][13] = 0.51
posicao[0][14] = 5.00

tecle <ENTER>

```

```

MATRIZ ENFESTO

posicao[1][15] = 0.00
posicao[1][0] = 0.03
posicao[1][1] = 1
posicao[1][2] = 1
posicao[1][3] = 1
posicao[1][4] = 1
posicao[1][5] = 1
posicao[1][6] = 901
posicao[1][7] = 1
posicao[1][8] = 901
posicao[1][9] = 1
posicao[1][10] = 1
posicao[1][11] = 1
posicao[1][12] = 1
posicao[1][13] = 1

tecle <ENTER>

```

```

MATRIZ ENFESTO

posicao[1][14] = 1
posicao[1][15] = 999
posicao[2][0] = 0.20
posicao[2][1] = 1
posicao[2][2] = 1
posicao[2][3] = 1
posicao[2][4] = 901
posicao[2][5] = 901
posicao[2][6] = 901
posicao[2][7] = 901
posicao[2][8] = 901
posicao[2][9] = 901
posicao[2][10] = 901
posicao[2][11] = 1
posicao[2][12] = 1

tecle <ENTER>

```

FIGURA 21 - Matriz do encaixe das três primeiras peças.

```

MATRIZ ENFESTO

posicao[2][13] = 1
posicao[2][14] = 1
posicao[2][15] = 999
posicao[3][0] = 0.25
posicao[3][1] = 1
posicao[3][2] = 901
posicao[3][3] = 901
posicao[3][4] = 901
posicao[3][5] = 901
posicao[3][6] = 901
posicao[3][7] = 901
posicao[3][8] = 901
posicao[3][9] = 901
posicao[3][10] = 901
posicao[3][11] = 901

tecle <ENTER>

```

```

MATRIZ ENFESTO

posicao[3][12] = 901
posicao[3][13] = 1
posicao[3][14] = 1
posicao[3][15] = 999
posicao[4][0] = 0.75
posicao[4][1] = 901
posicao[4][2] = 901
posicao[4][3] = 901
posicao[4][4] = 901
posicao[4][5] = 901
posicao[4][6] = 901
posicao[4][7] = 901
posicao[4][8] = 901
posicao[4][9] = 901
posicao[4][10] = 901

tecle <ENTER>

```

```

MATRIZ ENFESTO

posicao[4][11] = 901
posicao[4][12] = 901
posicao[4][13] = 901
posicao[4][14] = 1
posicao[4][15] = 999
posicao[5][0] = 0.78
posicao[5][1] = 1
posicao[5][2] = 1
posicao[5][3] = 1
posicao[5][4] = 1
posicao[5][5] = 1
posicao[5][6] = 902
posicao[5][7] = 1
posicao[5][8] = 902
posicao[5][9] = 1

tecle <ENTER>

```

FIGURA 22 - Continuação da matriz do encaixe das três primeiras peças.

```

MATRIZ ENFESTO

posicao[5][10] = 1
posicao[5][11] = 1
posicao[5][12] = 1
posicao[5][13] = 1
posicao[5][14] = 1
posicao[5][15] = 999
posicao[6][0] = 0.95
posicao[6][1] = 1
posicao[6][2] = 1
posicao[6][3] = 1
posicao[6][4] = 902
posicao[6][5] = 902
posicao[6][6] = 902
posicao[6][7] = 902
posicao[6][8] = 902

tecle <ENTER>

```

```

MATRIZ ENFESTO

posicao[6][9] = 902
posicao[6][10] = 902
posicao[6][11] = 1
posicao[6][12] = 1
posicao[6][13] = 1
posicao[6][14] = 1
posicao[6][15] = 999
posicao[7][0] = 1.00
posicao[7][1] = 1
posicao[7][2] = 902
posicao[7][3] = 902
posicao[7][4] = 902
posicao[7][5] = 902
posicao[7][6] = 902
posicao[7][7] = 902

tecle <ENTER>

```

```

MATRIZ ENFESTO

posicao[7][8] = 902
posicao[7][9] = 902
posicao[7][10] = 902
posicao[7][11] = 902
posicao[7][12] = 902
posicao[7][13] = 1
posicao[7][14] = 1
posicao[7][15] = 999
posicao[8][0] = 1.50
posicao[8][1] = 902
posicao[8][2] = 902
posicao[8][3] = 902
posicao[8][4] = 902
posicao[8][5] = 902
posicao[8][6] = 902

tecle <ENTER>

```

FIGURA 23 - Continuação da matriz do encaixe das três primeiras peças.



```

MATRIZ ENFESTO

posicao[8][7] = 902
posicao[8][8] = 902
posicao[8][9] = 902
posicao[8][10] = 902
posicao[8][11] = 902
posicao[8][12] = 902
posicao[8][13] = 902
posicao[8][14] = 1
posicao[8][15] = 999
posicao[9][0] = 1.56
posicao[9][1] = 1
posicao[9][2] = 1
posicao[9][3] = 1
posicao[9][4] = 1
posicao[9][5] = 1

tecle <ENTER>

```

```

MATRIZ ENFESTO

posicao[9][6] = 911
posicao[9][7] = 911
posicao[9][8] = 911
posicao[9][9] = 1
posicao[9][10] = 1
posicao[9][11] = 1
posicao[9][12] = 1
posicao[9][13] = 1
posicao[9][14] = 1
posicao[9][15] = 999
posicao[10][0] = 1.61
posicao[10][1] = 1
posicao[10][2] = 1
posicao[10][3] = 1
posicao[10][4] = 1

tecle <ENTER>

```

```

MATRIZ ENFESTO

posicao[10][5] = 911
posicao[10][6] = 911
posicao[10][7] = 911
posicao[10][8] = 911
posicao[10][9] = 911
posicao[10][10] = 1
posicao[10][11] = 1
posicao[10][12] = 1
posicao[10][13] = 1
posicao[10][14] = 1
posicao[10][15] = 999
posicao[11][0] = 1.71
posicao[11][1] = 911
posicao[11][2] = 911
posicao[11][3] = 911

tecle <ENTER>

```

FIGURA 24 - Continuação da matriz do encaixe das três primeiras peças.

```

MATRIZ ENFESTO

posicao[11][4] = 911
posicao[11][5] = 911
posicao[11][6] = 911
posicao[11][7] = 911
posicao[11][8] = 911
posicao[11][9] = 911
posicao[11][10] = 911
posicao[11][11] = 911
posicao[11][12] = 911
posicao[11][13] = 911
posicao[11][14] = 1
posicao[11][15] = 999
posicao[12][0] = 1.76
posicao[12][1] = 1
posicao[12][2] = 911
tecle <ENTER>

```

```

MATRIZ ENFESTO

posicao[12][3] = 911
posicao[12][4] = 911
posicao[12][5] = 911
posicao[12][6] = 911
posicao[12][7] = 911
posicao[12][8] = 911
posicao[12][9] = 911
posicao[12][10] = 911
posicao[12][11] = 911
posicao[12][12] = 911
posicao[12][13] = 1
posicao[12][14] = 1
posicao[12][15] = 999
posicao[13][0] = 1.81
posicao[13][1] = 1
tecle <ENTER>

```

```

MATRIZ ENFESTO

posicao[13][2] = 1
posicao[13][3] = 911
posicao[13][4] = 911
posicao[13][5] = 911
posicao[13][6] = 911
posicao[13][7] = 911
posicao[13][8] = 911
posicao[13][9] = 911
posicao[13][10] = 911
posicao[13][11] = 911
posicao[13][12] = 1
posicao[13][13] = 1
posicao[13][14] = 1
posicao[13][15] = 999
posicao[14][0] = 2.00
tecle <ENTER>

```

FIGURA 25 - Continuação da matriz do encaixe das três primeiras peças.

```
MATRIZ ENFESTO

posicao[14][1] = 1
posicao[14][2] = 1
posicao[14][3] = 1
posicao[14][4] = 1
posicao[14][5] = 1
posicao[14][6] = 1
posicao[14][7] = 1
posicao[14][8] = 1
posicao[14][9] = 1
posicao[14][10] = 1
posicao[14][11] = 1
posicao[14][12] = 1
posicao[14][13] = 1
posicao[14][14] = 1
posicao[14][15] = 999

tecle <ENTER>
```

```
MATRIZ ENFESTO

posicao[15][0] = 0.00
posicao[15][1] = 999
posicao[15][2] = 999
posicao[15][3] = 999
posicao[15][4] = 999
posicao[15][5] = 999
posicao[15][6] = 999
posicao[15][7] = 999
posicao[15][8] = 999
posicao[15][9] = 999
posicao[15][10] = 999
posicao[15][11] = 999
posicao[15][12] = 999
posicao[15][13] = 999
posicao[15][14] = 999

tecle <ENTER>
```

```
MATRIZ ENFESTO

posicao[15][15] = 999

tecle <ENTER>
```

FIGURA 26 - Continuação da matriz do encaixe das três primeiras peças.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	0.03	0.04	0.05	0.06	0.11	0.21	0.31	0.41	0.46	0.47	0.48	0.49	0.51	5.00	0	
1	0.03	1	1	1	1	1	901	1	901	1	1	1	1	1	1	999
2	0.20	1	1	1	901	901	901	901	901	901	901	1	1	1	1	999
3	0.25	1	901	901	901	901	901	901	901	901	901	901	901	1	1	999
4	0.75	901	901	901	901	901	901	901	901	901	901	901	901	901	1	999
5	0.78	1	1	1	1	1	902	1	902	1	1	1	1	1	1	999
6	0.95	1	1	1	902	902	902	902	902	902	902	1	1	1	1	999
7	1.00	1	902	902	902	902	902	902	902	902	902	902	902	1	1	999
8	1.50	902	902	902	902	902	902	902	902	902	902	902	902	902	1	999
9	1.56	1	1	1	1	1	911	911	911	1	1	1	1	1	1	999
10	1.61	1	1	1	1	911	911	911	911	911	1	1	1	1	1	999
11	1.71	911	911	911	911	911	911	911	911	911	911	911	911	911	1	999
12	1.76	1	911	911	911	911	911	911	911	911	911	911	911	1	1	999
13	1.81	1	1	911	911	911	911	911	911	911	911	911	1	1	1	999
14	2.00	1	1	1	1	1	1	1	1	1	1	1	1	1	1	999
15	0	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999

FIGURA 27 - Representação da matriz do encaixe das três primeiras peças.

As FIGURAS 28 a 34 mostram a situação de cada uma das 20 peças encaixadas, sendo que a última tela fornece o aproveitamento obtido, que neste teste foi de 44,36%. Justifica-se este baixo valor no aproveitamento pelo fato do comprimento do enfesto ser bem maior do que o necessário, condição necessária para se testar a heurística 2.





PECA No 4	
ENCAIXADA	====> sim
REPRESENTACAO DA PECA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.02 Y2 = 0.74
LOCACAO NO ESPACO	====> X1 = 0.52 X2 = 1.02 Y1 = 0.76 Y2 = 1.50
tecle <ENTER>	

PECA No 5	
ENCAIXADA	====> sim
REPRESENTACAO DA PECA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.02 Y2 = 0.74
LOCACAO NO ESPACO	====> X1 = 1.03 X2 = 1.53 Y1 = 0.03 Y2 = 0.75
tecle <ENTER>	

PECA No 6	
ENCAIXADA	====> sim
REPRESENTACAO DA PECA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.02 Y2 = 0.74
LOCACAO NO ESPACO	====> X1 = 1.03 X2 = 1.53 Y1 = 0.76 Y2 = 1.50
tecle <ENTER>	

FIGURA 29. - Situaçao das peças n.º4, n.º5 e n.º6, no encaixe gerado.

PEÇA No 7	
ENCAIXADA	====> sim
REPRESENTACAO DA PEÇA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.02 Y2 = 0.74
LOCACAO NO ESPACO	====> X1 = 1.54 X2 = 2.04 Y1 = 0.03 Y2 = 0.76
tecle <ENTER>	

PEÇA No 8	
ENCAIXADA	====> sim
REPRESENTACAO DA PEÇA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.02 Y2 = 0.74
LOCACAO NO ESPACO	====> X1 = 1.54 X2 = 2.04 Y1 = 0.76 Y2 = 1.50
tecle <ENTER>	

PEÇA No 9	
ENCAIXADA	====> sim
REPRESENTACAO DA PEÇA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.02 Y2 = 0.74
LOCACAO NO ESPACO	====> X1 = 2.05 X2 = 2.55 Y1 = 0.03 Y2 = 0.75
tecle <ENTER>	

FIGURA 30 - Situação das peças n.º7, n.º8 e n.º9, no encaixe gerado.

PECA No 10	
ENCAIXADA	====> sim
REPRESENTACAO DA PECA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.02 Y2 = 0.74
LOCACAO NO ESPACO	====> X1 = 2.05 X2 = 2.55 Y1 = 0.76 Y2 = 1.50
tecle <ENTER>	

PECA No 11	
ENCAIXADA	====> sim
REPRESENTACAO DA PECA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.05 Y2 = 0.30
LOCACAO NO ESPACO	====> X1 = 0.01 X2 = 0.51 Y1 = 1.51 Y2 = 1.81
tecle <ENTER>	

PECA No 12	
ENCAIXADA	====> sim
REPRESENTACAO DA PECA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.05 Y2 = 0.30
LOCACAO NO ESPACO	====> X1 = 0.52 X2 = 1.02 Y1 = 1.51 Y2 = 1.81
tecle <ENTER>	

FIGURA 31 - Situaçao das peças n.º10, n.º11 e n.º12, no encaixe gerado.

PECA No 13	
ENCAIXADA	====> sim
REPRESENTACAO DA PECA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.05 Y2 = 0.30
LOCACAO NO ESPACO	====> X1 = 1.03 X2 = 1.53 Y1 = 1.51 Y2 = 1.81
tecle <ENTER>	

PECA No 14	
ENCAIXADA	====> sim
REPRESENTACAO DA PECA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.05 Y2 = 0.30
LOCACAO NO ESPACO	====> X1 = 1.54 X2 = 2.04 Y1 = 1.51 Y2 = 1.81
tecle <ENTER>	

PECA No 15	
ENCAIXADA	====> sim
REPRESENTACAO DA PECA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.05 Y2 = 0.30
LOCACAO NO ESPACO	====> X1 = 2.05 X2 = 2.55 Y1 = 1.51 Y2 = 1.81
tecle <ENTER>	

FIGURA 32 - Situaçao das peças n.º13, n.º14 e n.º15, no encaixe gerado.

PEÇA No 16	
ENCAIXADA	====> sim
REPRESENTACAO DA PEÇA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.05 Y2 = 0.30
LOCACAO NO ESPACO	====> X1 = 2.56 X2 = 3.06 Y1 = 0.03 Y2 = 0.33
tecle <ENTER>	

PEÇA No 17	
ENCAIXADA	====> sim
REPRESENTACAO DA PEÇA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.05 Y2 = 0.30
LOCACAO NO ESPACO	====> X1 = 2.56 X2 = 3.06 Y1 = 0.34 Y2 = 0.74
tecle <ENTER>	

PEÇA No 18	
ENCAIXADA	====> sim
REPRESENTACAO DA PEÇA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.05 Y2 = 0.30
LOCACAO NO ESPACO	====> X1 = 2.56 X2 = 3.06 Y1 = 0.75 Y2 = 1.05
tecle <ENTER>	

FIGURA 33 - Situação das peças n.º16, n.º17 e n.º18, no encaixe gerado.



PEÇA No 19	
ENCAIXADA	====> sim
REPRESENTACAO DA PEÇA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.05 Y2 = 0.30
LOCACAO NO ESPACO	====> X1 = 2.56 X2 = 3.06 Y1 = 1.06 Y2 = 1.36
tecle <ENTER>	

PEÇA No 20	
ENCAIXADA	====> sim
REPRESENTACAO DA PEÇA	====> X1 = 0.02 X2 = 0.50 Y1 = 0.05 Y2 = 0.30
LOCACAO NO ESPACO	====> X1 = 2.56 X2 = 3.06 Y1 = 1.37 Y2 = 1.67
tecle <ENTER>	

Aprov. minimo = 70.00 %
Aprov. obtida = 44.36 %
tecle <ENTER>

FIGURA 34. - Situação das peças n.º19 e n.º20, no encaixe gerado e aproveitamento obtido.

#### 4.3 - RESULTADOS

O sistema, quando testado, gerou encaixes satisfatórios, demonstrando que as quatro heurísticas iniciais estão no caminho certo. Embora para casos mais específicos seja possível aumentar o aproveitamento obtido, sua utilização genérica já é um bom começo.

#### 4.4 - LIMITAÇÕES

A maior limitação é sem dúvida nenhuma o fato de ainda não haver uma representação gráfica para a entrada do molde e saída do encaixe obtido. O fato de ter que se "desenhar" a matriz de cada molde antes de entrar no sistema, desestimula seu uso. E se esta fosse a forma final do algoritmo, ele não serviria para nada, pois o trabalho de realizar o encaixe manualmente seria substituído pelo trabalho de fazer a matriz de cada molde e interpretar a matriz resultante do encaixe. E esta substituição de trabalho não seria nem um pouco vantajosa para quem se utilizasse do algoritmo!

No entanto, o que tem que ser levado em consideração é que o objetivo deste trabalho era direcionar o estudo do assunto e iniciar a implementação do algoritmo, e isto foi feito. Mas não quer dizer que este não possa ser melhorado, e muito.

A representação gráfica seria então, o próximo passo a ser realizado, com o objetivo de melhorar o algoritmo, o que é um objetivo bem específico - e que já está sendo feito - para um problema que no início era bastante obscuro, não se sabendo nem por onde começar.

Outra limitação é o que fazer quando a eficiência do encaixe

obtido estiver abaixo da estipulada. Esta é uma limitação que pode ser resolvida através da implementação de novas heurísticas. Uma sugestão seria uma heurística que reordenasse os moldes e que, numa segunda tentativa, levasse em consideração a área de cada molde em vez de seu aproveitamento individual.

## 5 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

1 - Usando-se apenas as técnicas da Pesquisa Operacional, os elevados tempos de processamento, bem como as restrições de equipamentos, normalmente inviabilizam uma solução prática aceitável, obrigando a busca da solução através de outras técnicas.

2 - As técnicas de Inteligência Artificial, principalmente os procedimentos heurísticos, têm conduzido a melhores resultados, mesmo sem a garantia da melhor solução possível.

*conclu  
são*

3 - Embora se saiba da existência de sistemas para a solução do problema de encaixe de peças irregulares, as informações obtidas na literatura especializada são pouco reveladoras.

4 - Para a solução do problema, além da eficiência do encaixe obtido, deve ser analisado o tempo gasto pelo computador para gerá-lo. Nesse sentido, novas heurísticas podem ser testadas; no entanto, deve ser avaliado o quanto elas melhoram o desempenho do sistema e, em contrapartida, quanto tempo de processamento a mais está sendo gasto em função dessa melhora no desempenho.

5 - No caso da aplicação do sistema à indústria do vestuário, deve-se estar atento para o fato de que há diferenças de um tipo de roupa para outro. No caso de calças compridas, camisas e roupões, por exemplo, o aproveitamento é grande. Já no caso de roupas menores, como o da indústria de "lingerie", o aproveitamento possível normalmente é bem menor. Dessa forma, deve-se ava-

liar a necessidade ou não de se adotar procedimentos heurísticos diferenciados.

6 - A automatização do encaixe proporciona uma redução no tempo exigido de cada especialista por encaixe obtido. No entanto, ela não substitui totalmente esse especialista, por necessitar, às vezes, de uma otimização do resultado. A grande vantagem, no caso de um aumento na produção, é que o computador pode trabalhar 24 horas por dia, sendo que a análise dos encaixes, por ele gerados durante a noite, poderá ser feita pelos especialistas durante o dia, paralelamente à realização de encaixes que exijam maior interatividade.

7 - O sistema até aqui desenvolvido encontra-se ainda em fase de testes. A limitação de ainda não haver representação gráfica para a entrada das peças, dificulta tais testes. O que já se tem concluído é a sua capacidade de armazenar e manipular as peças, independentemente de suas geometrias, não havendo a imposição de que a peça seja uma figura convexa.

Sugere-se para trabalhos futuros:

1 - O desenvolvimento de uma "interface" gráfica do sistema aqui desenvolvido.

2 - Avaliação de novas heurísticas, levando-se em consideração o tipo de lote de peças, indicando que associação (grupo de heurísticas/tipo de lote) proporcionará melhores resultados.



3 - O desenvolvimento de um sistema que realize um pré-encaixe, formando envelopes de duplas de peças, cujo aproveitamento deverá ser melhor que o aproveitamento dos envelopes quando separados.

4 - A criação e avaliação de novas heurísticas, por exemplo, uma que rotacione as peças cujos envelopes tenham aproveitamento menor que 50%.

5 - O desenvolvimento de um sistema similar ao que foi aqui proposto, em linguagem PROLOG.

## APÊNDICE A

### INTRODUÇÃO AOS TERMOS DE PESQUISA OPERACIONAL

Para melhor compreensão do capítulo 2, será necessário uma introdução a alguns termos de Pesquisa Operacional.

Os termos serão apresentados numa ordem lógica e não na mesma ordem em que aparecem no capítulo 2. Isto se deve ao fato de que alguns conceitos são pré-requisitos para o entendimento de outros. Por exemplo, no capítulo 2, o problema da mochila é citado antes do que o método simplex. No entanto, para se entender o problema da mochila, são necessárias noções do algoritmo "branch and bound", que utiliza conceitos da programação inteira, que por sua vez parte do pressuposto que o método simplex já é conhecido.

A explicação dos termos, sempre que possível, será feita através de exemplos, o que atingirá melhor o objetivo de elucidar o assunto.

#### A.1 - PESQUISA OPERACIONAL

Durante a 2<sup>a</sup> Guerra Mundial, os militares ingleses contrataram uma equipe de cientistas para estudar problemas estratégicos e táticos associados com a defesa aérea e terrestre do país. O objetivo era decidir a melhor utilização dos limitados recursos militares. As aplicações incluíram, entre outros, estudos sobre como usar o recém inventado radar e a eficácia de novos tipos de bombas. O estabelecimento desta equipe científica marcou a primeira atividade formal em pesquisa operacional.

O nome "pesquisa operacional" (algumas vezes abreviado PO em português ou OR, de "operations research", em inglês) aparentemente foi inventado porque a equipe estava tratando com pesquisa em

operações (militares) [39].

Numerosos sinônimos para pesquisa operacional são de uso comum. Os ingleses gostam de *operational research* (pesquisa operacional), enquanto os americanos usam *operations research* (pesquisa de operações). Um freqüente substituto americano é *management science* (ciência da administração). A popularidade deste nome é alimentada por uma sociedade profissional internacional chamada *Institute of Management Sciences*.

Por conveniência, e com precisão razoável, pode-se simplesmente definir pesquisa operacional como uma abordagem científica à resolução de problemas para administração executiva [43].

Desde seu nascimento, esta nova maneira de se tomar decisões tem sido caracterizada pelo uso do conhecimento científico através dos esforços de equipes interdisciplinares com o propósito de decidir sobre a melhor utilização de recursos limitados.

Os resultados encorajadores obtidos pelas equipes de pesquisa operacional inglesas motivou o governo militar dos Estados Unidos a começar com atividades similares.

Aplicações bem sucedidas das equipes norte americanas incluíram o estudo de problemas de lógica complexa, invenção de novos modelos de vôo, "planning sea mining" e utilização eficaz dos equipamentos eletrônicos.

Com o fim da guerra, o sucesso das equipes militares atraiu a atenção dos administradores industriais que estavam procurando soluções para seus complexos problemas tipo executivos. Estes problemas estavam se tornando mais "agudos" por causa da introdução da especialização funcional nas organizações comerciais.

Apesar do fato de que funções especializadas são estabelecidas primariamente para servir a objetivos de toda a organização,

os objetivos individuais destas funções nem sempre eram compatíveis com as metas da organização. Isto resultou em complexos problemas de decisão, os quais, ultimamente têm forçado organizações comerciais a procurar a utilização das ferramentas eficazes da pesquisa operacional.

Embora à Inglaterra seja creditada a iniciação da PO como uma nova disciplina, a liderança no campo que cresceu rapidamente foi logo tomada pelos Estados Unidos.

A primeira técnica matemática no campo, chamada "método simplex" em programação linear, foi desenvolvida em 1947 pelo matemático americano, George B. Dantzig. Desde então, novas técnicas e aplicações têm sido desenvolvidas através dos esforços e cooperação mútua entre instituições acadêmicas e as indústrias.

O progresso impressionante no campo de PO é devido em grande parte ao desenvolvimento paralelo dos computadores digitais, com suas tremendas capacidades em velocidade computacional bem como em armazenamento e recuperação de informações.

Um grande número de firmas de consultoria gerencial estão geralmente engajadas em atividades de PO. Estas atividades foram além das aplicações militares e comerciais para incluir estudos em hospitais, instituições financeiras, bibliotecas, planejamento de cidades, sistemas de transporte e até investigação de crimes, para mencionar somente alguns [39].

## **A.2 - PROGRAMAÇÃO LINEAR**

O entendimento do conceito de programação linear fica bastante facilitado através do seguinte exemplo simplificado [43]:

Indústria - Processamento de batatas em embalagens de batati-

nha frita, picadinho de batata e flocos para purê.

No início do processo de fabricação, as batatas em bruto são classificadas por comprimento e qualidade e, então, distribuídas a linhas de produção separadas.

As batatas podem ser compradas de duas fontes, que diferem no sortimento dos vários tamanhos e qualidades.

Estas características de produção são mostradas na TABELA A.1:

Produto	Fonte 1	Fonte 2	Limitação de vendas
batatinha frita	0,2	0,3	1,8
picadinho	0,2	0,1	1,2
flocos	0,3	0,3	2,4
lucro por ton.	5	6	

TABELA A.1 - Produções de batata e lucro.

Observe-se que, da Fonte 1, há uma produção de 20% de batatinha frita, uma produção de 20% de picadinho e uma produção de 30% de flocos; os restantes 30% são refugo irre recuperável.

Os números para flocos e refugo são também 30% para as batatas da fonte 2, mas a produção de batatinha frita é relativamente maior.

Quantas toneladas de batata devem ser compradas de cada fonte?

A resposta depende, em parte, das contribuições de lucro das fontes.

Estes números são calculados somando-se as receitas de vendas associadas às produções por tonelada comprada para os produtos separados e subtraindo-se os custos de comprar uma tonelada de bata-

ta, que podem diferir entre as duas fontes.

Suponha-se que a contribuição de lucro por tonelada comprada seja 5 para a fonte 1 e 6 para a fonte 2. Embora a fonte 2 seja mais lucrativa, não segue daí que a companhia deva comprar todas as suas batatas da fonte 2.

A companhia deve também reconhecer seu potencial de vendas. O mercado para os diferentes alimentos congelados é limitado.

Estão indicados na TABELA A.1 valores estimados dos limites superiores das toneladas que podem ser vendidas de cada produto. Os custos de armazenagem são altos demais para permitir a fabricação de qualquer produto para estoque.

Assim, ao se tomar decisões de compra das duas fontes, deve-se tomar cuidado para não comprar quaisquer toneladas de batatas que produzissem produtos em excesso a 1,8 t de batatinha frita, 1,2 t de picadinho e 2,4 t para flocos.

Seja  $P_1$  a quantidade (em toneladas) de batata que será comprada da fonte 1 e  $P_2$  a quantidade da fonte 2.

Então os valores de  $P_1$  e  $P_2$  são restringidos pelas desigualdades lineares:

$$\begin{aligned} 0,2 P_1 + 0,3 P_2 &\leq 1,8 && \text{para batatinha frita} \\ 0,2 P_1 + 0,1 P_2 &\leq 1,2 && \text{para picadinho} \\ 0,3 P_1 + 0,3 P_2 &\leq 2,4 && \text{para flocos} \end{aligned} \quad (1)$$

onde:

$$P_1 \geq 0 \quad P_2 \geq 0$$

As restrições de não negatividade  $P_1 \geq 0$  e  $P_2 \geq 0$  são impostas porque um valor tal como  $P_1 = -4$  não teria significado físico.

Todos os valores de  $P_1$  e  $P_2$  que satisfazem (1) são mostrados na região sombreada da FIGURA A.1(d).



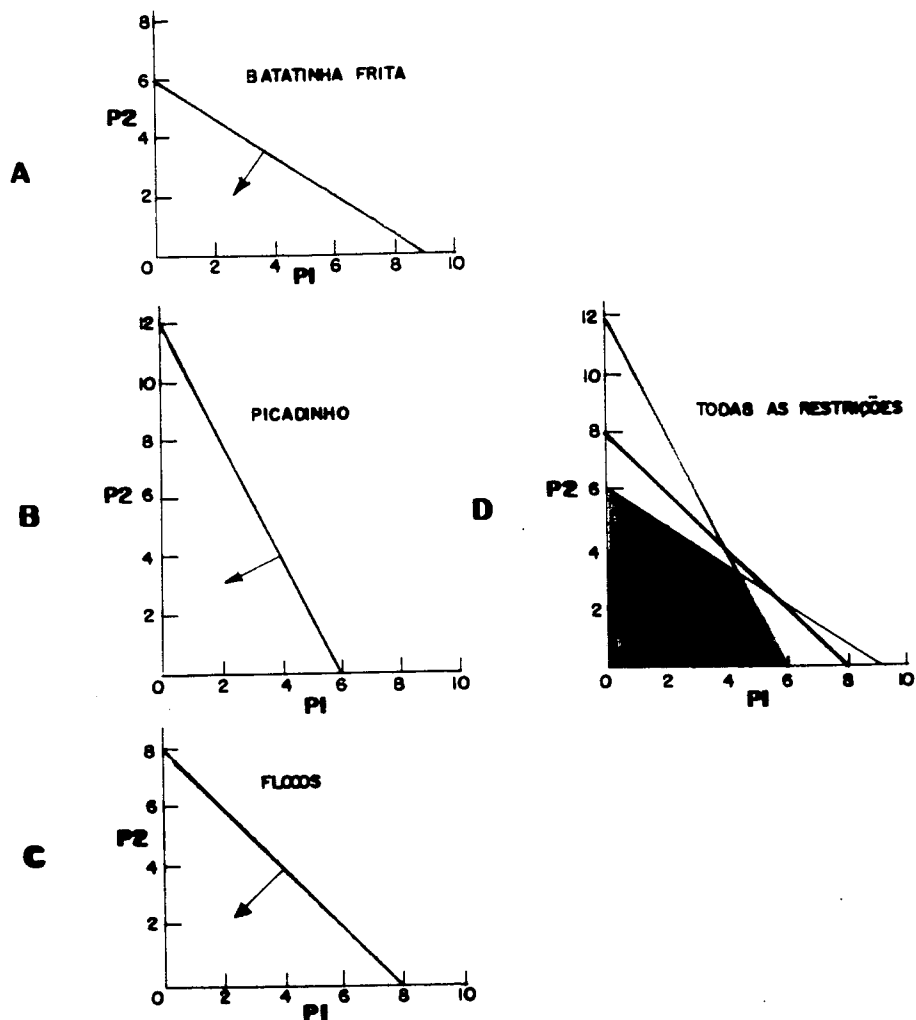


FIGURA A.1 - Políticas praticáveis de compras.

Note-se que cada linha no diagrama é representada por uma restrição em (1) expressa como uma igualdade.

A seta associada a cada linha mostra o sentido indicado pelo sinal de desigualdade em (1).

Observe-se que um par de valores para  $P_1$  e  $P_2$  que satisfaça ambas as restrições de batatinha frita e picadinho também satisfará a restrição para flocos.

Encontram-se valores ótimos para  $P_1$  e  $P_2$  se se tornar a contribuição de lucro tão grande quanto possível, respeitadas as restrições. Portanto, de acordo com os dados da TABELA A.1, o problema de otimização é:

$$\text{Maximize } (5 P_1 + 6 P_2) \quad (2)$$

sujeito a (1).

Neste problema simples, a solução pode ser mostrada graficamente, na FIGURA A.2.

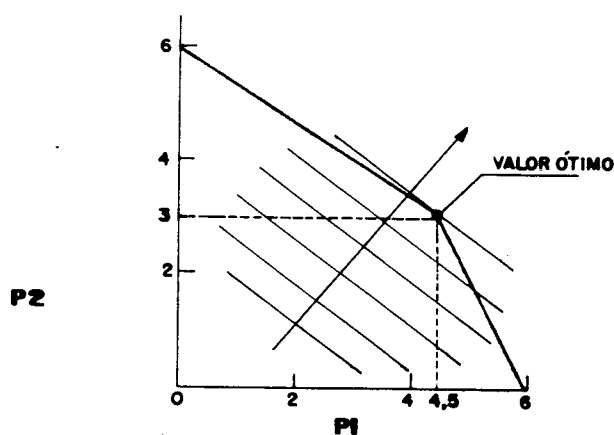


FIGURA A.2 - Máximo lucro.

Cada um dos segmentos de reta paralelos representa diferentes combinações de  $P_1$  e  $P_2$  que dão o mesmo valor para a função objetivo linear  $5 P_1 + 6 P_2$ .

O segmento mais alto de todos que ainda tem um ponto na região de restrição é o valor ótimo da função objetivo, e tal ponto é uma solução ótima.

Como pode se ver na FIGURA A.2, há somente uma solução ótima neste exemplo; ela ocorre na intersecção das restrições de batatinha frita e picadinho.

Conseqüentemente, pode-se calcular os valores ótimos resolvendo as equações lineares simultâneas:

$$0,2 P_1 + 0,3 P_2 = 1,8 \quad \text{para batatinha frita} \quad (3)$$

$$0,2 P_1 + 0,1 P_2 = 1,2 \quad \text{para picadinho.}$$

As respostas ótimas são  $P_1 = 4,5$  e  $P_2 = 3$ , como mostrado na figura 3, dando um valor da função objetivo de 40,5.

Aplicações reais de programação linear comumente envolvem centenas de restrições e milhares de variáveis.

### A.3 - MÉTODO SIMPLEX OU ALGORITMO SIMPLEX DE DANTZING

Uma tradução mais correta de *simplex method* seria *metodo do simplexo*. Simplexo é o nome dado pelos matemáticos a uma configuração espacial de pontos, com certas propriedades. No entanto, o termo *método simplex* já se tornou consagrado em português.

Muitos algoritmos diferentes já foram propostos para resolver problemas de programação linear, mas o algoritmo simplex provou ser o mais eficiente no geral.

O procedimento é o seguinte:

*Passo 1* - Selecione um conjunto de  $m$  variáveis que produzam uma solução tentativa inicial viável. Elimine as  $m$  variáveis escolhidas da função-objetivo.

*Passo 2* - Verifique a função-objetivo para ver se há uma variável que seja igual a zero, na solução tentativa, mas que melhoraria a solução-objetivo se fosse tornada positiva. Se existir uma tal variável, vá ao *Passo 3*. Caso contrário, pare.

*Passo 3* - Determine quão grande pode ser feita a variável encontrada no passo anterior até que uma das  $m$  variáveis da solução tentativa torne-se zero. Elimine Esta última variável e faça o próximo conjunto tentativo conter, em vez disso, a variável recém-encontrada.

*Passo 4* - Resolva para estas  $m$  variáveis e deixe as variáveis restantes iguais a zero na próxima solução tentativa. Retorne ao *Passo 2*.

Depois de ordenar a linguagem destes passos para remover certas ambigüidades, o algoritmo resultante acha de fato uma solução ótima para um modelo de programação linear geral num número finito de iterações.

Será examinado um problema "bem comportado" para a explicação do método Simplex. Neste exemplo, o objetivo será o máximo lucro. Em outros casos os objetivos podem ser outros, tais como minimizar o custo, minimizar a quantidade de matéria prima necessária, etc.

- Suponha que uma Companhia XYZ tenha a opção de usar um, ou mais de um, dentre quatro diferentes tipos de processos de produção.

- Os primeiro e segundo processos produzem unidades de Produto A, e os terceiro e quarto processos produzem unidades de produto B.

- Os insumos para cada processo são o *trabalho*, medido em *homens-semanas*, *quilos de material Y* e *caixas de Material Z*.

- Uma vez que cada processo varia em suas exigências de insumos, as lucratividades dos processos diferem, mesmo para processos produzindo o mesmo produto.

- O fabricante, decidindo sobre o planejamento de produção de uma semana, está limitado na gama de possibilidades pelas quantidades disponíveis de mão-de-obra e de ambas as espécies de matérias-primas.

- A tecnologia completa e as restrições de entrada são dadas na TABELA A.2.

Item	Uma unidade de produto A		Uma unidade de produto B		Disponibilidades totais
	Processo 1	Processo 2	Processo 3	Processo 4	
Homens-semana	1	1	1	1	$\leq 15$
Quilos de Material Y	7	5	3	2	$\leq 120$
Caixas de Material Z	3	5	10	15	$\leq 100$
Lucro Unitário(NCz\$)	4	5	9	11	Maximize
Nível de produção	$x_1$	$x_2$	$x_3$	$x_4$	

TABELA A.2 - Escolha de produtos da Companhia XYZ.

**Axiomas lineares** - Serão feitas duas suposições tecnológicas e econômicas cruciais com relação a como o processo de produção opera:

a) *Divisibilidade* - Para cada atividade, as quantidades totais de cada insumo e o lucro associado são estritamente proporcionais ao nível de produção - isto é, ao nível de atividade. Para ilustrar, o efeito direto de dobrar os insumos de qualquer processo é dobrar a produção e os lucros. Em particular, fabricar 10 unidades pelo Processo 1 ( $x_1 = 10$ ), requer 10 homens-semana, 70 quilos de Material Y e 30 caixas de Material Z destinados à atividade, e os lucros resultantes são NCz\$ 40,00. O postulado da divisibilidade implica, além disso, que se permita que os níveis de atividade assumam valores fracionários bem como valores inteiros. Por exemplo, admite-se a possibilidade tecnológica de  $x_2 = 2,5$  ou  $x_4 = 10/3$ .

b) *Aditividade* - Dados os níveis de atividade para cada uma das variáveis de decisão  $x_j$ , as quantidades totais de cada insumo e o lucro associado são as somas dos insumos e lucro para cada



processo individual. Para ilustrar, na TABELA A.2, uma unidade do Processo 1 ( $x_1 = 1$ ) e uma unidade do Processo 3 ( $x_3 = 1$ ) requerem 2 homens-semana, 10 quilos de Material Y e 13 caixas de Material Z e dá NCz\$ 13,00 de lucro.

As suposições de divisibilidade e aditividade são equivalentes a declarar que o modelo matemático fundamental pode ser formulado em termos de relações lineares.

Em situações reais, os dois postulados acima são válidos apenas aproximadamente, o que é suficiente para permitir aplicações úteis da abordagem linear.

Neste exemplo específico, há uma relação de desigualdade linear para cada restrição de trabalho e material e uma relação linear expressando lucratividade:

$$\text{maximize lucro} \equiv \text{maximize } (4 x_1 + 5 x_2 + 9 x_3 + 11 x_4) \quad (4)$$

sujeita às restrições:

$$\begin{aligned} 1 x_1 + 1 x_2 + 1 x_3 + 1 x_4 &\leq 15 \text{ (homens-semana)} \\ 7 x_1 + 5 x_2 + 3 x_3 + 2 x_4 &\leq 120 \text{ (Material Y)} \\ 3 x_1 + 5 x_2 + 10 x_3 + 15 x_4 &\leq 100 \text{ (Material Z)} \end{aligned} \quad (5)$$

Não há nenhum significado físico para níveis de produção negativos, tais como  $x_1 = -4,2$ , de modo que não será permitido à produção ser *negativa*. Restringe-se então, cada nível de produção incógnito a ser zero ou positivo, isto é, a ser *não-negativo*,

$$x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 \geq 0 \quad x_4 \geq 0 \quad (6)$$

(Uma variável que se permite que assuma *também* valores negativos é dita ser *irrestrita no sinal*).

O problema de gerenciamento é achar valores para todos os níveis de produção incógnitos  $x_j$  que satisfazem as relações (5) e

(6) e também maximizem o lucro (4). Em geral, tais valores não precisam ser únicos. Podem existir soluções ótimas alternativas.

Seja  $x_0$  o valor da função objetivo e acrescenta-se variáveis de folga. Então, escreve-se o sistema como:

$$\begin{array}{rcll}
 1 x_0 - 4 x_1 - 5 x_2 - 9 x_3 - 11 x_4 & = & 0 & \text{Linha 0} \\
 1 x_1 + 1 x_2 + 1 x_3 + 1 x_4 + 1 x_5 & = & 15 & \text{Linha 1} \\
 (7) \quad 7 x_1 + 5 x_2 + 3 x_3 + 2 x_4 + 1 x_6 & = & 120 & \text{Linha 2} \\
 3 x_1 + 5 x_2 + 10 x_3 + 15 x_4 + 1 x_7 & = & 100 & \text{Linha 3.}
 \end{array}$$

onde todas as variáveis devem ser não-negativas. Note-se como a introdução da variável  $x_0$  na Linha 0 permite expressar a função-objetivo em forma de equação.

A tarefa do Passo 1 é achar uma solução viável inicial para (7). Há um grande número de tais soluções, mas é certamente mais conveniente começar com  $x_0 = 0$ ,  $x_5 = 15$ ,  $x_6 = 120$ ,  $x_7 = 100$  e todas as outras variáveis iguais a zero. Em outras palavras, começa-se com uma solução com folga em tudo. Chama-se *uma solução viável básica*, e  $x_0, x_5, x_6$  e  $x_7$  são conhecidos como **variáveis básicas**, algumas vezes simplificados para a **base**. As variáveis restantes são chamadas de **não-básicas**.

A palavra *básica* provém da propriedade matemática de que a solução é representada por quatro variáveis, sendo "quatro" o número de relações lineares, e que os valores destas quatro variáveis são determinados univocamente (e dependentes das constantes do lado direito das equações). O método simplex emprega somente soluções *básicas* tentativas. A despeito da restrição a esta classe de soluções, o método simplex determina, não obstante, uma solução ótima, desde que exista uma.

Se  $x_0$  for interpretado como "lucros", então a solução corrente certamente não é muito lucrativa. Indubitavelmente pode ser melhorada. Uma reflexão de momento leva a examinar os coeficientes na Linha 0 das variáveis não-básicas. Especificamente, os coeficientes de  $x_1$ ,  $x_2$ ,  $x_3$  e  $x_4$ . Cada coeficiente negativo representa quanto  $x_0$  aumentará ao se fazer a variável correspondente igual a 1. Esta conclusão em relação à interpretação dos coeficientes na Linha 0 permanece válida durante todos os cálculos tentativos, e registra-se aqui o princípio geral.

INTERPRETAÇÃO DOS COEFICIENTES NA LINHA 0. Cada coeficiente representa o aumento (para coeficientes negativos) ou diminuição (para coeficientes positivos) em  $x_0$  com um aumento unitário da variável não-básica associada.

Para o *Passo 2* o método simplex adota a seguinte regra, fácil de aplicar para decidir qual a variável que deve entrar na base tentativa seguinte.

CRITÉRIO DO SIMPLEX I (MAXIMIZAÇÃO). Se houverem variáveis não-básicas que tenham coeficiente negativo na Linha 0, escolha uma dessas variáveis com o coeficiente mais negativo de todos, isto é, o melhor ganho potencial por unidade. Se todas as variáveis não-básicas tiverem coeficientes positivos ou zero na Linha 0, já foi obtida uma solução ótima.

De acordo com o *Crítério I*, deve-se introduzir  $x_4$ . Cada unidade de  $x_4$  adicionada produz um aumento de 11 em  $x_0$ . Evidentemente, quanto mais se aumentar  $x_4$ , mais aumentará  $x_0$ .

Ao se examinar as restrições (7), observa-se que, à medida

que se aumenta  $x_4$ , deve-se diminuir cada variável básica corrente correspondente a uma linha na qual  $x_4$  tenha um coeficiente positivo. Especificamente, se  $x_4$  for elevado ao nível 1, então verifica-se em (7) que:

- (i)  $x_5$  deve diminuir de 1 de modo que a restrição da linha 1 permaneça satisfeita,
- (ii)  $x_6$  deve diminuir de 2 de modo que a restrição da linha 2 permaneça satisfeita,
- (iii)  $x_7$  deve diminuir de 15 de modo que a restrição da linha 3 permaneça satisfeita.

Quão grande pode tornar-se  $x_4$  antes que uma das variáveis correntes básicas atinja seu limite inferior 0?

Verifica-se que este número é  $x_4 = 100/15 = 6,66$ , para cujo valor se acha  $x_7 = 0$ . Sendo tal o caso, introduz-se  $x_4$  na base e retira-se  $x_7$ .

Resume-se a discussão anterior pela seguinte regra para o Passo 3.

#### CRITÉRIO DO SIMPLEX II.

- (a) Tomam-se as relações do lado corrente da direita pelos coeficientes das variáveis de entrada  $x_j$  (ignoram-se as relações com número zero ou negativo no denominador).
- (b) Escolhe-se a relação mínima - essa relação será igual ao valor de  $x_j$  na próxima solução tentativa. A relação mínima ocorre para uma variável  $x_k$  na solução presente; fixa-se  $x_k = 0$  na nova solução.

Os cálculos indicados pelo *Crítério II* são executados na

TABELA A.3.

Variáveis básicas	Solução corrente $\div$	Coefs. de $x_4$	= Relações	Mín.	Solução seguinte
$x_0$	0	-11	- -		
$x_5$	15	1	15		
$x_6$	120	2	60		
$x_7$	100	15	6,66	6,66	$x_4 = 6,66, x_7 = 0$

TABELA A.3 - Iteração 1: Critério II para  $x_4$  introduzido na base.

Agora que já é sabido que  $x_4$  deve substituir  $x_7$  na base tentativa, executa-se o Passo 4. Reescreve-se as relações (7) de modo que  $x_4$  tenha um coeficiente 1 na Linha 3 e coeficientes 0 nas Linhas 0, 1 e 2, tal como acontece com  $x_7$  em (7). O processo pelo qual isto é conseguido é conhecido como um cálculo com **mudança de base** ou uma **operação-pivô**. Primeiro divide-se a Linha 3 por 15, o coeficiente de  $x_4$ .

$$\begin{array}{rcll}
 1 x_0 - 4 x_1 - 5 x_2 - 9 x_3 - 11 x_4 & = & 0 & \text{Linha 0} \\
 1 x_1 + 1 x_2 + 1 x_3 + 1 x_4 + 1 x_5 & = & 15 & \text{Linha 1} \\
 (8) \quad 7 x_1 + 5 x_2 + 3 x_3 + 2 x_4 + 1 x_6 & = & 120 & \text{Linha 2} \\
 1/5 x_1 + 1/3 x_2 + 2/3 x_3 + 1 x_4 + 1/15 x_7 & = & 20/3 & \text{Linha 3}
 \end{array}$$

Pivotando sobre 15, criou-se o coeficiente de 1 para  $x_4$  na Linha 3. (A manipulação matemática é legítima, pois tudo que se fez foi dividir iguais por iguais, isto é, ambos os lados da Linha 3 por 15). Cria-se coeficientes iguais a 0 nas Linhas 0, 1 e 2 como se segue:

- . Linha 0: multiplica-se a Linha 3 por 11 e soma-se à Linha 0,
- . Linha 1: multiplica-se a Linha 3 por -1 e soma-se à Linha 1,
- . Linha 2: multiplica-se a Linha 3 por -2 e soma-se à Linha 2.

O resultado é:

$$\begin{array}{rcl}
 1 \ x_0 - 9/5 \ x_1 - 4/3 \ x_2 - 5/3 \ x_3 & + 11/15 \ x_7 = & 220/3 \quad \text{Linha 0} \\
 4/5 \ x_1 + 2/3 \ x_2 + 1/3 \ x_3 & + 1 \ x_5 - 1/15 \ x_7 = & 25/3 \quad \text{Linha 1} \\
 (9) \ 33/5 \ x_1 + 13/3 \ x_2 + 5/3 \ x_3 & + 1 \ x_6 - 2/15 \ x_7 = & 320/3 \quad \text{Linha 2} \\
 1/5 \ x_1 + 1/3 \ x_2 + 2/3 \ x_3 + 1 \ x_4 & + 1/15 \ x_7 = & 20/3 \quad \text{Linha 3}
 \end{array}$$

Como antes, as manipulações matemáticas são legítimas, pois tudo que se fez foi multiplicar iguais por iguais (ambos os lados de cada linha por uma constante) e, então, somar iguais a iguais (uma linha a outra linha). Conseqüentemente, as relações (9), embora parecendo ser de forma diferente de (7), são equivalentes a (7). A utilidade da forma de (9) é que, fazendo  $x_1 = x_2 = x_3 = x_7 = 0$ , vê-se imediatamente os valores para a nova solução básica tentativa (TABELA A.4).

$x_0$	220/3
$x_5$	25/3
$x_6$	320/3
$x_4$	20/3

TABELA A.4 - Segunda solução tentativa básica.

**Iteração 2.** Neste ponto, a primeira iteração do método simplex foi completada. O Critério I, que examina as variáveis



não-básicas, indica que parece existir uma solução ainda melhor. Poderia-se introduzir com proveito  $x_1$ ,  $x_2$  ou  $x_3$  na base. O Critério I escolhe  $x_1$ , uma vez que ele promete o maior ganho por unidade de aumento. Em seguida, executa-se os cálculos do Passo 3, usando o Critério II como mostrado na TABELA A.5.

Variáveis básicas	Solução corrente ÷	Coefs. de $x_1$	= Relações	Mín.	Solução seguinte
$x_0$	220/3	- 9/5	- -		
$x_5$	25/3	4/5	125/12	125/12	$x_1 = 125/12, x_5 = 0$
$x_6$	320/3	33/5	1600/99		
$x_4$	20/3	1/5	100/3		

TABELA A.5 - Iteração 2: Critério II para  $x_1$  introduzido na base.

Da TABELA A.5, observa-se que  $x_1$  substituirá  $x_5$  na solução tentativa seguinte. Deve-se reescrever (9) para considerar a substituição. Para fazer os cálculos de mudança de base no Passo 4, primeiro divide-se pelo pivô 4/5 na Linha 1, fazendo com que o coeficiente 1 seja retirado de  $x_5$  e apareça para  $x_1$ :

$$\begin{array}{rcl}
 1 x_0 - 9/5 x_1 - 4/3 x_2 - 5/3 x_3 & +11/15 x_7 = & 220/3 \quad \text{Linha 0} \\
 1 x_1 + 5/6 x_2 + 5/12 x_3 & +5/4 x_5 - 1/12 x_7 = & 125/12 \quad \text{Linha 1} \\
 (10) 33/5 x_1 + 13/3 x_2 + 5/3 x_3 & +1 x_6 - 2/15 x_7 = & 320/3 \quad \text{Linha 2} \\
 1/5 x_1 + 1/3 x_2 + 2/3 x_3 + 1 x_4 & + 1/15 x_7 = & 20/3 \quad \text{Linha 3}
 \end{array}$$

Então, para completar a operação-pivô, cria-se coeficientes iguais a zero para  $x_1$  nas Linhas 0, 2 e 3 como se segue:

- . Linha 0: multiplica-se a Linha 1 por  $9/5$  e soma-se à Linha 0,
- . Linha 2: multiplica-se a Linha 1 por  $-33/5$  e soma-se à Linha 2,
- . Linha 3: multiplica-se a Linha 1 por  $-1/5$  e soma-se à Linha 3.

O resultado é:

$$\begin{array}{rcll}
 1 \ x_0 & +1/6 \ x_2 - 11/12 \ x_3 & +9/4 \ x_5 & +7/12 \ x_7 = 1105/12 & \text{Linha 0} \\
 1 \ x_1 & +5/6 \ x_2 + 5/12 \ x_3 & +5/4 \ x_5 & -1/12 \ x_7 = 125/12 & \text{Linha 1} \\
 (11) & -7/6 \ x_2 - 13/12 \ x_3 & -33/4 \ x_5 + 1 \ x_6 + 5/12 \ x_7 & = 455/12 & \text{Linha 2} \\
 & 1/6 \ x_2 + 7/12 \ x_3 + 1 \ x_4 - 1/4 \ x_5 & & +1/12 \ x_7 = 55/12 & \text{Linha 3}
 \end{array}$$

A terceira solução tentativa básica aparece na TABELA A.6. Observe-se que, a cada iteração, os coeficientes de qualquer variável que não está na solução tentativa, têm a interpretação de que, se uma unidade desta variável for introduzida, as variáveis da base corrente serão alteradas por estas quantidades. Em (9), por exemplo, introduzindo cada unidade de  $x_1$ ,  $x_5$  diminuirá de  $4/5$ ,  $x_6$  de  $33/5$  e  $x_4$  de  $1/5$ .

$x_0$	1105/12
$x_1$	125/12
$x_6$	455/12
$x_4$	55/12

TABELA A.6 - Terceira solução tentativa básica.

**Iteração 3.** Tendo completado a segunda iteração do simplex, examina-se mais uma vez os coeficientes na Linha 0 para verificar se já se descobriu uma solução ótima. Parece agora favorável introduzir  $x_3$ . Os cálculos para determinar que variável deixa a ba-

se são dados na TABELA A.7. A partir destes cálculos,  $x_4$ , que foi introduzido na primeira iteração, deve agora ser retirado da base. Frequentemente, em aplicações do método simplex, uma variável é introduzida na solução numa iteração e é removida numa iteração posterior. É esta possibilidade que proíbe enunciar um limite superior útil sobre o número de iterações do simplex necessário a resolver qualquer problema de programação linear.

Variáveis básicas	Solução corrente <sup>†</sup>	Coefs. de $x_3$	= Relação	Mín.	Solução seguinte
$x_0$	1105/12	-11/12	- -		
$x_1$	125/12	5/12	25		
$x_6$	455/12	-13/12	- -		
$x_4$	55/12	7/12	55/7	55/7	$x_3 = 55/7, x_4 = 0$

TABELA A.7 - Iteração 3: Critério II para  $x_3$  introduzido na base.

Assim como nas iterações anteriores, cria-se um coeficiente 1 para a variável introduzida  $x_3$  na Linha 3, dividindo a Linha 3 por 7/12. Isto dá:

$$\begin{array}{rcl}
 1 \ x_0 & +1/6 \ x_2 - 11/12 \ x_3 & +9/4 \ x_5 \quad +7/12 \ x_7 = 1105/12 \quad \text{Linha 0} \\
 1 \ x_1 & +5/6 \ x_2 + 5/12 \ x_3 & +5/4 \ x_5 \quad -1/12 \ x_7 = 125/12 \quad \text{Linha 1} \\
 (12) & -7/6 \ x_2 - 13/12 \ x_3 & -33/4 \ x_5 + 1 \ x_6 + 5/12 \ x_7 = 455/12 \quad \text{Linha 2} \\
 & 2/7 \ x_2 + & 1 \ x_3 + 12/7 \ x_4 - 3/7 \ x_5 \quad + 1/7 \ x_7 = 55/7 \quad \text{Linha 3}
 \end{array}$$

Cria-se agora coeficientes iguais a 0 para  $x_3$  nas linhas restantes como se segue:

- . Linha 0: multiplica-se a Linha 3 por 11/12 e soma-se à Linha 0,
- . Linha 1: multiplica-se a Linha 3 por -5/12 e soma-se à Linha 1,
- . Linha 2: multiplica-se a Linha 3 por 13/12 e soma-se à Linha 2.

O resultado é:

$$\begin{array}{rcll}
 1 \ x_0 & +3/7 \ x_2 & +11/7 \ x_4 + 13/7 \ x_5 & +5/7 \ x_7 = 695/7 & \text{Linha 0} \\
 1 \ x_1 & +5/7 \ x_2 & -5/7 \ x_4 + 10/7 \ x_5 & -1/7 \ x_7 = 50/7 & \text{Linha 1} \\
 (13) & -6/7 \ x_2 & +13/7 \ x_4 - 61/7 \ x_5 + 1 \ x_6 & +4/7 \ x_7 = 325/7 & \text{Linha 2} \\
 & 2/7 \ x_2 + 1 \ x_3 & +12/7 \ x_4 - 3/7 \ x_5 & +1/7 \ x_7 = 55/7 & \text{Linha 3}
 \end{array}$$

**Iteração 4.** Todos os coeficientes na Linha 0 de (13) são não-negativos, e, conseqüentemente, o *Critério I* afirma que se encontrou uma solução ótima (TABELA A.8). Assim os cálculos são determinados no *Passo 2*.

$x_0$	695/7
$x_1$	50/7
$x_6$	325/7
$x_3$	55/7

TABELA A.8 - Quarta tentativa e solução básica ótima.

A interpretação do resultado é a seguinte:

- O lucro máximo será de 695/7;
- Para se obter este máximo lucro, deverão ser produzidos 50/7 unidades do Produto A, através do processo 1 e 55/7 unidades do Produto B, através do processo 3.

- Nenhuma unidade do Produto A será feita pelo processo 2 e nenhuma unidade do Produto B será feita pelo processo 4, pois

$$x_2 = x_4 = 0.$$

- Sobrarão  $325/7$  caixas de material Y, pois  $x_6 = 325/7$ .

**Sumário.** Em síntese, o método simplex consiste em

**Passo 1.** Escolha uma base inicial.

**Passo 2.** Aplique o *Critério do Simplex I*. Se a solução tentativa não for ótima, prossiga ao *Passo 3*. Caso contrário, pare.

**Passo 3.** Aplique o *Critério do Simplex II*.

**Passo 4.** Faça a mudança de base e retorne ao *Passo 2*.

Para se demonstrar que, de fato, se encontrou uma solução ótima, supõe-se que se começou com (13) e rearranja-se levemente a Linha 0, podendo-se escrever função objetivo como:

$$(14) \quad \text{maximize } x_0 = 695/7 - 3/7 x_2 - 11/7 x_4 - 13/7 x_5 - 5/7 x_7$$

Se as variáveis não básicas  $x_2$ ,  $x_4$ ,  $x_5$  ou  $x_7$  tiverem qualquer valor viável que não seja zero, então a relação acima mostra que  $x_0$  será menor que seu valor presente de  $695/7$ .

Os coeficientes finais da Linha 0 das variáveis originais são algumas vezes referidos como **custos-relativos** ou **custos-sombra**. Eles representam a diminuição no valor ótimo da função objetivo resultante de um aumento unitário numa variável não básica, supondo que a base final permaneça viável. Por exemplo, suponha que se decida fixar  $x_2 = 1$ ; então o valor de  $x_0$  seria diminuído de  $3/7$ .

Os valores das variáveis básicas também variariam, e pode-se

examinar este efeito do mesmo modo que se fez para  $x_0$ .  
Considere-se a variável básica  $x_1$  na Linha 1 de (13):

$$(15) \quad x_1 = 50/7 - 5/7 x_2 + 5/7 x_4 - 10/7 x_5 + 1/7 x_7$$

Fazendo  $x_2 = 1$ , diminui-se o valor de  $x_1$  pela quantidade de  $5/7$ .

**REPRESENTAÇÃO TABULAR** - Ao prosseguir de uma solução tentativa à próxima, é tanto enfadonho quanto desnecessário escrever  $x_1, x_2, \dots, x_n$  como foi feito anteriormente. Somente os *coeficientes* das variáveis são necessários nas computações. Uma vez que se tenha entendido a lógica direta das iterações do simplex, pode-se economizar considerável esforço de escrita organizando as computações numa forma tabular conveniente chamada de um **quadro do simplex**.

A TABELA A.9 representa uma abordagem para o exemplo visto anteriormente.



Iteração	Base	Valores correntes	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	Linha
1	$x_0$	0	-4	-5	-9	-11				0
	$x_5$	15	1	1	1	1	1			1
	$x_6$	120	7	5	3	2		1		2
	$x_7$	100	3	5	10	<u>15</u>			1	3
2	$x_0$	220/3	-9/5	-4/3	-5/3				11/15	0
	$x_5$	25/3	<u>4/5</u>	2/3	1/3		1		-1/15	1
	$x_6$	320/3	33/5	13/3	5/3			1	-2/15	2
	$x_4$	20/3	1/5	1/3	2/3	1			1/15	3
3	$x_0$	1105/12		1/6	-11/12		9/4		7/12	0
	$x_1$	125/12	1	5/6	5/12		5/4		-1/12	1
	$x_6$	455/12		-7/6	-13/12		-33/4	1	5/12	2
	$x_4$	55/12		1/6	<u>7/12</u>	1	-1/4		1/12	3
4	$x_0$	695/7		3/7		11/7	13/7		5/7	0
	$x_1$	50/7	1	5/7		-5/7	10/7		-1/7	1
	$x_6$	325/7		-6/7		13/7	-61/7	1	4/7	2
	$x_3$	55/7		2/7	1	12/7	-3/7		1/7	3

TABELA A.9 - Quadro do simplex para o problema da escolha de produtos da companhia XYZ.

Concluindo, é honesto dizer que a solução ótima mesmo para este pequeno problema não era óbvia de início. Em modelos maiores, é muito importante se poder abordar uma solução ótima de uma maneira sistemática.

#### A.4 - PROGRAMAÇÃO INTEIRA

No caso de problemas grandes, com muitas variáveis e restrições, as dificuldades de computação são enormes, mesmo quando se utilizam computadores digitais de grande capacidade. Já se dedicou muita atenção ao desenvolvimento de métodos especiais que se aproveitam de características particulares das restrições a fim de resolver problemas grandes quando as disponibilidades do computador são limitadas.

Algumas vezes ocorre que, além de outras restrições, determinadas variáveis só podem assumir valores inteiros. No problema geral de programação linear a solução pode assumir qualquer valor e devem-se usar métodos especiais quando se exigem valores inteiros. A não ser quando as variáveis assumem valores grandes (possivelmente da ordem das centenas), arredondar a solução para o inteiro mais próximo nem sempre produz a melhor solução inteira e poderá até levar a um resultado muito afastado do ótimo. Como exemplo de uma situação na qual se exige um resultado em inteiros, citemos o problema do número de vezes que os anúncios devam ser colocados em um conjunto de revistas. Já se procurou formular este problema empregando-se uma função objetivo linear e restrições lineares (em uma revista mensal não se podem publicar mais de doze anúncios por ano). Não estamos discutindo as hipóteses sobre a linearidade (será que dois anúncios produzem duas vezes mais resultados do que um?), mas é evidente que o número de vezes que se anuncia terá que ser um número inteiro (103).

No estudo do Método Simplex foram vistas duas suposições chamadas *divisibilidade* e *aditividade*. Aqui estas duas suposições são

atenuadas. Deve-se lembrar que a combinação destes postulados implicaram em linearidade tanto na função objetivo como nas restrições e, também, permitiram que as variáveis tomassem valores fracionários, tais como 2,5 ou 10/3. Aqui abandona-se a **suposição da divisibilidade** e trata-se de problemas nos quais algumas ou todas as variáveis podem tomar somente valores (ou números) inteiros.

Considere-se o modelo:

$$(16) \quad \text{otimize} \quad \sum_{j=1}^n c_j x_j,$$

sujeito a:

$$(17) \quad \sum_{j=1}^n a_{ij} x_j \leq b_i \quad \text{para } i = 1, 2, \dots, m$$

$$(18) \quad x_j \geq 0 \quad \text{para } j = 1, 2, \dots, n$$

$$(19) \quad x_j \text{ de valor inteiro para } j = 1, 2, \dots, p (\leq n).$$

Este tipo de modelo de otimização é conhecido como um **problema de programação inteira** (ou **diofantina** ou, ainda, **discreta**). Quando  $p = n$ , de modo que todas as variáveis devam ter valores inteiros, o modelo é chamado de um problema de programação inteira **pura**; caso contrário, é chamado de um problema de programação inteira **mista** [43].

Dependendo da aplicação particular, o sentido de otimização na função-objetivo (16) pode ser maximização ou minimização. Além disto, um problema de programação inteira pode incluir igualdades e desigualdades ( $\geq$ ).

As estipulações de valores discretos (19) é que distinguem um problema de programação inteira de um de programação linear. Em geral, impor (19) é restritivo, de modo que o valor máximo de uma função-objetivo para um problema de programação inteira é comumente menor que para o problema de programação linear correspondente.

**Importância dos problemas de programação inteira.** A maioria das aplicações industriais de modelos de programação de grande escala são orientados para decisões de planejamento em situações complexas. Há várias circunstâncias de ocorrência freqüente que levam a modelos de planejamento contendo variáveis de valores inteiros.

1. *Utilização de equipamento.* Pode-se definir uma variável  $x_j$  como as unidades de equipamento que devem operar durante o horizonte de planejamento do modelo. Se cada unidade de equipamento fornecer uma grande capacidade e for cara - por exemplo, uma máquina automática de fazer parafusos, um navio petroleiro ou uma máquina corrugadora de papel de dupla-faca de 3,75m - então um valor fracionário para  $x_j$ , como  $10/3$ , pode não ter significado (não ser realizável) no contexto da decisão real. Neste caso,  $x_j$  teria que ser restringido a valores inteiros.

2. *Custos de preparação.* Pode-se querer considerar uma atividade que esteja sujeita a um chamado custo fixo (ou *custo de preparação*)  $C_j$  sempre que o nível correspondente  $x_j > 0$ , onde  $C_j$  é independente do nível real de  $x_j$ . Por exemplo, se  $x_j$  representar a utilização horária de uma fornalha numa usina de aço, então  $C_j$  representa o custo de pôr para funcionar a fornalha e aquecê-la até a temperatura necessária. Neste caso, serão englobados os custos de preparação num modelo de programação linear introduzindo variáveis de valor inteiro.

3. *Tamanhos de lotes.* Em algumas situações de planejamento de produção, pode-se querer restringir o nível de  $x_j$  a: ou  $x_j = 0$  ou  $x_j \geq L_j$ . Por exemplo,  $x_j$  pode ser a quantidade de um produto especial a ser fabricado durante o período  $t$ , e  $L_j$  pode representar o tamanho mínimo possível de lote de produção para o item. Es-

ta estipulação é um exemplo de uma restrição "ou-ou" e pode ser formulada introduzindo variáveis inteiras.

4. *Decisões "sim-ou-não"*. Pode-se querer especificar outros tipos de situações "ou-ou". Para fazer isto, pode-se restringir os níveis de  $x_j$  a: ou  $x_j = 1$  ou  $x_j = 0$ , representando as decisões "sim" ou "não". Como exemplo, pode-se fazer  $x_j = 1$  corresponder a construir uma nova fábrica, abrir um território de vendas, adquirir um outro negócio ou vender um bem possuído no presente. Frequentemente, estas espécies de alternativas são classificadas como decisões de *orçamento de capital*, por que elas requerem grandes gastos de capitais e recursos. Esta é a razão principal por que a programação inteira é tão importante para decisões administrativas. Uma solução ótima para um problema de orçamento de capital pode dar lucro consideravelmente maior a uma firma do que dará uma solução aproximada ou conjecturada. Por exemplo, uma firma fabricante de cimento com 25 fábricas pode ser capaz de aumentar substancialmente os lucros reduzindo a 20 fábricas ou menos, desde que esta redução seja planejada otimamente. O "overhead" (despesas gerais) diminuído com menos fábricas pode facilmente sobrepor qualquer aumento conseqüente nos custos de transporte, se a nova configuração das fábricas for ótima.

Para um algoritmo ser de uso geral na solução de problemas de programação inteira, ele deve evitar enumerar *explicitamente* todas as possibilidades. O que se quer são técnicas que enumerem *parcialmente* um número tratável de possibilidades e enumerem *implicitamente* todo o resto. Deve-se lembrar que o método simplex é exatamente uma técnica assim para resolver problemas comuns de programação linear - ele examina sistematicamente somente um nú-

mero pequeno de todas as soluções básicas possíveis. O sucesso destes métodos de otimização parcialmente enumerativos motiva a busca para achar abordagens semelhantes para resolver problemas de programação inteira.

Especialistas técnicos têm concentrado seus esforços nesta área e o progresso em resolver problemas reais de programação inteira está aumentando rapidamente. Quando estes algoritmos se tornarem perfeitos, os analistas de pesquisa operacional, equipados com computadores de alta velocidade, devem de fato ser reconhecidos como tendo criado uma pedra filosofal [43].

#### A.5 - ALGORITMO "BRANCH-AND-BOUND"

A abordagem mais amplamente adotada para resolver problemas de programação inteira usa um método de busca em árvore, algumas vezes referido como um *algoritmo de retrocesso* [43].

É uma técnica de ramificação e avaliação progressiva e é chamada, em inglês, de "*Branch-and-Bound*" e pode ser tanto aplicada a problemas de programação inteira pura ou de programação mista. Consiste em observar que, se após encontrar o ponto ótimo de um problema, introduzir-se restrições adicionais, o novo ótimo será *não melhor* que o anterior sem as restrições adicionais [11].

Para melhor entender, o processo será exemplificado:

$$\text{Max } Z = x_1 + 4 x_2$$

Sujeito a

$$2 x_1 + 4 x_2 \leq 7$$

$$10 x_1 + 3 x_2 \leq 15$$

$$x_1 \geq 0, x_2 \geq 0, \text{ inteiros.}$$

Seja o problema  $t_0$  idêntico ao acima e para o qual desiste-se da restrição de inteireza. Resulta:

$$t_0 \rightarrow \begin{cases} x_1 = 0 \\ x_2 = 7/4 \\ Z = 7 \end{cases}$$

Sabe-se, então, que ao se considerar as restrições de inteireza nunca se poderá obter um valor maior do que 7 para Z. Chamando de  $Z_M$  o maior valor para o problema que também satisfaz as restrições de inteireza, pode-se inicializar em  $Z_M = 0$ , já que  $x_1 = 0$ ,  $x_2 = 0$  satisfazem.

O problema  $t_0$  subdivide-se em dois outros com restrições adicionais.

$$t_1 \rightarrow x_2 \leq 1$$

$$t_2 \rightarrow x_2 \geq 2$$

O problema  $t_1$  resulta em:

$$t_1 \rightarrow \begin{cases} x_1 = 1,2 \\ x_2 = 1 \\ Z = 5,2 \end{cases}$$

E como não satisfaz as restrições de inteireza continua-se com  $Z_M = 0$ .

O problema  $t_1$  origina dois outros:

$$t_3 \rightarrow x_2 \leq 1, x_1 \leq 1$$

$$t_4 \rightarrow x_2 \leq 1, x_1 \geq 2$$

Cujos resultados são:

$$t_3 \rightarrow \begin{cases} x_1 = 1 \\ x_2 = 1 \\ Z = 5, \text{ de modo que o novo } Z_M = 5. \end{cases}$$

$$t_4 \rightarrow \text{não viável.}$$

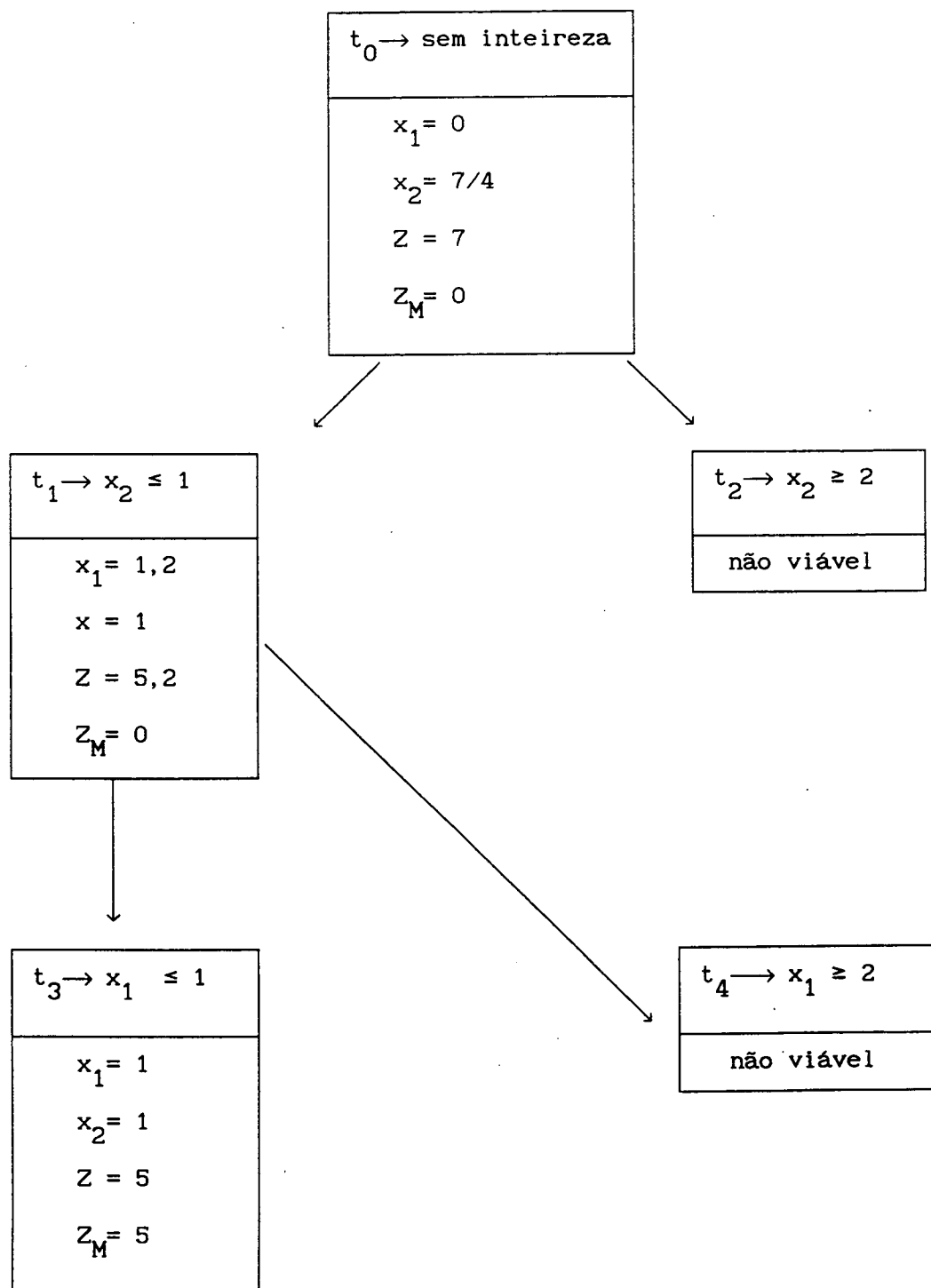
O problema  $t_2$  resulta:

$$t_2 \rightarrow \text{não viável.}$$



Terminou o processo onde  $t_3$  resultou no desejado.

Esquemmatizando:



Vê-se que cada vez que uma variável resulta não inteira, ramifica-se com duas opções de restrição adicional - o inteiro logo acima e o inteiro logo abaixo. A cada restrição adicional, o valor de  $Z$  decresce.

Sempre que um novo ramo resulta num problema não viável, podem-se abandonar todas as suas possíveis descendências já que a imposição de restrições adicionais não poderá tornar o problema viável.

Sempre que em duas soluções dos problemas de maximização  $t_i$  e  $t_j$ , oriundos de um mesmo problema anterior e nos quais, para  $t_i$  a restrição adicional foi  $x_l \leq I$  e para  $t_j$  foi  $x_l \geq I + 1$ , obtiver-se os máximos  $Z_{M_i}$  e  $Z_{M_j}$ , aquele que resultar no menor  $Z_M$  pode ser abandonado.

Quando as variáveis  $x$  só podem assumir valores 0 ou 1, tem-se o caso de programação binária. O trabalho fundamental neste campo surgiu em 1963 e é de autoria de Egon Balas. É o chamado algoritmo aditivo por só necessitar de operações aditivas [11].

#### A.6 - PROBLEMA DA MOCHILA

A equipe de astronautas está se preparando para um voo extenso em torno da Terra num laboratório orbital no espaço. Os astronautas podem levar consigo, para sua leitura pessoal,  $b$  quilos de livros e eles escolherão entre  $n$  livros. O peso do Livro  $j$  é  $a_j$ , e a equipe avalia conjuntamente que o prazer de leitura do Livro  $j$  seja  $c_j$ , onde o número  $c_j$  especifica o índice relativo de satisfação deles. Os astronautas fazem sua escolha resolvendo:

$$(20) \quad \text{maximize} \quad \sum_{j=1}^n c_j x_j$$

sujeito a

$$(21) \quad \sum_{j=1}^n a_j x_j \leq b$$

$$(22) \quad x_j = 0, 1 \text{ para } j = 1, 2, \dots, n.$$

Neste assim chamado problema da mochila 0-1, (20), (21) e (22), cada variável de decisão  $x_j$  é zero-um, correspondendo a rejeitar ou levar o livro  $j$ . A função-objetivo (20) é simplesmente a soma dos valores de satisfação individuais para os livros escolhidos, e a restrição (21) é a estipulação total de peso.

O modelo da mochila 0-1 é um problema de programação inteira pura e satisfaz às suposições para o algoritmo "branch-and-bound" [43].

#### A.7 - PROGRAMAÇÃO DINÂMICA

A matemática clássica não tem se mostrado muito adequada para o tratamento de problemas de otimização que envolvem grande número de decisões variáveis e/ou grande número de inequações de restrição. Até há algum tempo atrás, estes problemas podiam ser resolvidos somente tentando-se encontrar valores ótimos para todas as decisões simultaneamente, solução esta que conduz rapidamente a cálculos não-econômicos, ou até mesmo impraticáveis, dentro dos limites do equipamento de computação disponível. Existe, evidentemente, a tentação de dividir um grande problema em sub-problemas, cada qual envolvendo apenas poucas variáveis. É isto que faz a programação dinâmica [01].

Programação dinâmica é uma técnica muito empregada em problemas que envolvem a otimização de problemas que podem ser modelados

por uma seqüência de estados. Pode ser aplicada indiferentemente tanto a problemas lineares como a problemas não lineares. Sua aplicabilidade é bastante geral, isto é, os tipos de problemas de programação solúveis por esta técnica são muitos, incluindo distribuição, estoque e substituição, embora o método não seja sempre o mais fácil nem o mais eficiente.

Esse método, que envolve relações algébricas de recorrência desenvolvidas principalmente por Richard Bellman (1957), evoluiu como resultado do estudo de problemas de programação, nos quais as decisões são tomadas ao longo do tempo, razão pela qual foi denominada de "*programação dinâmica*". No entanto, a técnica pode ser aplicada a problemas estáticos, isto é, que se referem a um instante determinado. Nesse caso, o tempo passa a ser irrelevante. Por isso, seria talvez preferível que essa técnica tivesse um outro nome, tal como "*programação por estágios*", mas o termo "*programação dinâmica*" já está tão arraigado que mudá-lo agora seria difícil [11].

Uma descrição geral das características da *programação dinâmica* e dos tipos de problemas que podem ser formulados e resolvidos com essa técnica teria que ser extensa e, ao mesmo tempo, um tanto vaga e abstrata, já que existe uma variedade imensa de tipos muito diversos de problemas solúveis com a *programação dinâmica* [39].

A capacidade da *programação dinâmica* poder resolver uma variedade muito grande de problemas de programação freqüentemente revela-se teórica, porque, devido à sua relativa ineficiência computacional, ela apresenta, na prática, restrições sérias quanto aos tipos de problemas e ao tamanho dos mesmos, que permitem uma resolução numérica econômica.

---

O ponto crucial, ao se iniciar a solução de um problema por *programação dinâmica*, reside na definição dos *estágios* e dos *estados*. Uma definição inteligente pode diminuir enormemente o tempo envolvido nos cálculos.

A cada oportunidade de se tomar uma decisão associa-se um estágio. Um estágio compreende muitos estados.

Uma seqüência de decisões, uma para cada estágio, constitui uma política. O objetivo da otimização é determinar a política ótima que otimize a função objetivo global do sistema em seus  $n$  estágios. Esta é a política ótima ou seqüência ótima.

O princípio da otimalidade de Richard Bellman especifica que, considerando-se um problema com  $n$  estágios e tendo-se determinado a política ótima - isto é, a seqüência de decisões que redundam num roteiro ótimo, passando por uma seqüência de estados - e re-analisando-se o problema para uma seqüência tomada no meio do caminho, a partir de um estado que pertencia ao roteiro ótimo do problema de  $n$  estágios, então a seqüência de decisões até o fim será a mesma para os dois problemas.

O princípio da otimalidade garante que a política ótima a ser adotada num determinado estágio depende, daí por diante, tão-somente do estado no qual se encontre naquele estágio e não de como se chegou àquele estado. Em outras palavras, dado um certo estado de um estágio, a política ótima para os estágios remanescentes é independente da adotada nos estágios previamente analisados.

Como primeiro exemplo, deseja-se minimizar, na FIGURA A.3, o custo do caminho entre os nós 1 e 6. Com este exemplo serão vistas as primeiras idéias de *Programação Dinâmica* [11].

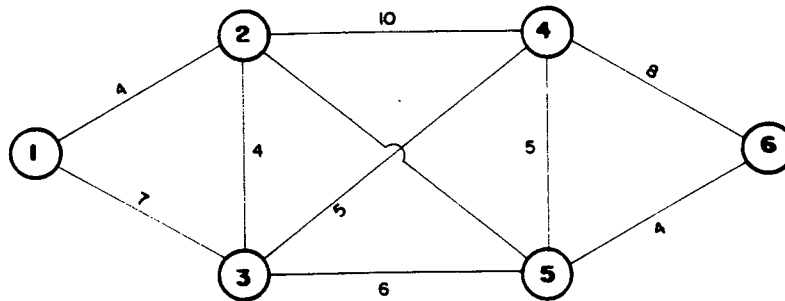


FIGURA A.3 - Caminho e seus custos.

Primeiramente, será procurado um algoritmo. Cada nó corresponde a um estado ao qual se associa um valor.

Seja  $c_{ij}$  o custo associado ao movimento do nó  $i$  ao nó  $j$ . Seja  $A_i$  o custo total mínimo para mover-se do nó  $i$  ao nó final, que é o nó 6. Segue-se então:

$$A_5 = \text{mínimo} (c_{56} + A_6) = 4 + 0 = 4$$

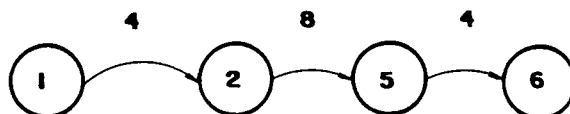
$$A_4 = \text{mínimo} [(c_{45} + A_5); (c_{46} + A_6)] = \\ = \text{mínimo} [(5+4); (8+0)] = 8$$

$$A_3 = \text{mínimo} [(c_{34} + A_4); (c_{35} + A_5)] = \\ = \text{mínimo} [(5+8); (6+4)] = 10$$

$$A_2 = \text{mínimo} [(c_{23} + A_3); (c_{24} + A_4); (c_{25} + A_5)] = \\ = \text{mínimo} [(4+10); (10+8); (8+4)] = 12$$

$$A_1 = \text{mínimo} [(c_{12} + A_2); (c_{13} + A_3)] = \\ = \text{mínimo} [(4+12); (7+10)] = 16$$

O caminho com o custo mínimo resulta:



O algoritmo expressa-se de forma geral pelo cálculo de:

$$A_i = \text{mínimo} (c_{ij} + A_j)$$

Todo o conceito se encontra na movimentação de nó para nó. Os pontos de referência são os nós. Estando-se em um nó qualquer, a solução do problema mostra a melhor política para se atingir o nó final e o custo correspondente. Para cada nó só se guarda a política ótima deste nó e não todas as combinações possíveis.

Como segundo exemplo será visto um problema que visa encontrar uma estratégia ótima de decisão, resolvido por um processo particular da *programação dinâmica*, conhecido como *indução reversa* [43].

A diretora executiva de uma firma de análise e previsões econômicas, deseja empregar uma nova secretária executiva e pretende pedir a uma agência de empregos que mande pessoas qualificadas para entrevistar. Ela descobriu, por experiência passada, que pode determinar, a partir de uma entrevista, se a candidata, caso seja escolhida, se tornará excelente, boa ou apenas regular. Ela dá um valor relativo 3 para uma secretária excelente, 2 para uma boa e 1 para uma regular. Sua experiência anterior também a leva a crer que há uma chance de 0,2 de entrevistar uma candidata que será uma



excelente secretária, uma chance de 0,5 que a candidata será boa e uma chance de 0,3 que a candidata será regular.

Ela deseja entrevistar três candidatas no máximo. Infelizmente, se ela não empregar uma secretária imediatamente depois de uma entrevista, a pessoa aceitará outro emprego, portanto ela tem que decidir imediatamente.

Se a primeira secretária que ela entrevistar for excelente, ela empregará a pessoa de imediato, naturalmente. E, se a candidata for regular, ela não tem nada a perder entrevistando uma segunda candidata. Mas, se a primeira candidata parecer boa, então ela não está certa do que fazer. Se deixar passar a pessoa, ela pode terminar com uma secretária apenas regular. Todavia, se ela empregar a secretária, ela renuncia à chance de encontrar uma excelente. Do mesmo modo, se escolher entrevistar uma segunda candidata, estará de novo em face de uma difícil decisão na eventualidade de que a entrevistada venha a ser boa.

O problema da seleção pode ser mostrado convenientemente por uma assim chamada *árvore de decisão*, mostrada na FIGURA A.4. Os nós com círculo representam as candidatas entrevistadas, e os ramos a partir destes nós mostram os eventos incertos e suas probabilidades. Os quadrados indicam onde a decisão deve ser tomada, e o número no fim de um ramo representa o valor de parar o processo de decisão naquele ponto.

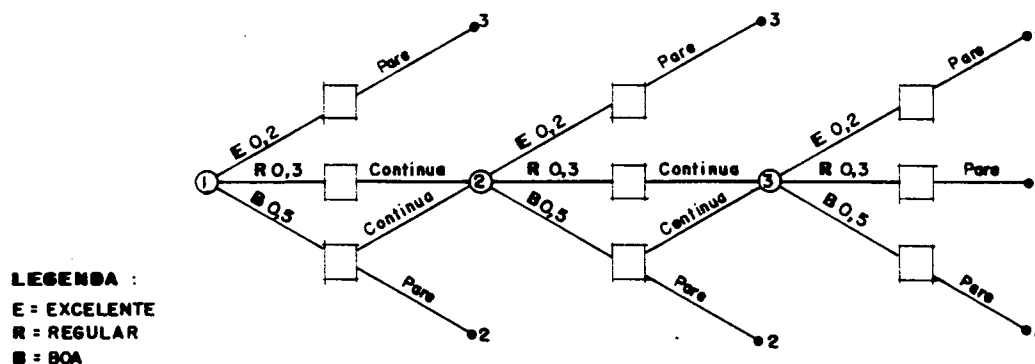


FIGURA A.4 - Árvore de decisão para o problema da secretária.

Supondo que a diretora executiva acabe mesmo entrevistando, e daí empregando, uma terceira candidata. Então o valor esperado associado ao evento incerto é:

$$(23) \quad 3 (0,2) + 2 (0,5) + 1 (0,3) = 1,9$$

Em outras palavras, o valor médio de uma secretária selecionada ao acaso para uma entrevista é 1,9. Supõe-se que esta expectativa representa fielmente a avaliação feita pela empregadora, do evento incerto. Marca-se o número 1,9 acima do Nó 3 na FIGURA A.4.

Considera-se, em seguida, o que acontece se for entrevistada uma segunda candidata, que resulta ser boa. Se decidir parar, obtém então um valor 2. Mas, se continuar, então pode esperar obter somente o valor 1,9. Assim, deveria parar quando a segunda candidata parecer boa. Põe-se um X sobre o ramo indicando "continue"

quando a segunda candidata for boa; isto significa não tomar aquela ação.

Agora já se pode determinar a decisão correta se a primeira candidata parecer boa. Parando, seria obtido o valor 2. Mas, continuando, então o valor esperado associado ao resultado probabilístico da segunda entrevista, e possivelmente da terceira, é:

$$(24) \quad 3 (0,2) + 2 (0,5) + 1,9 (0,3) = 2,17$$

O primeiro termo em (24) é para o evento de entrevistar uma secretária excelente, que será escolhida; o segundo termo é para o evento de entrevistar uma boa secretária, que será escolhida, como já se determinou no parágrafo anterior; e o terceiro termo é para o evento de entrevistar uma secretária regular e, conseqüentemente, continuar ao terceiro evento incerto que tem um valor de 1,9, dado em (23). Uma vez que 2,17 é maior que 2, deve-se desistir da primeira secretária se a candidata resultar ser boa. Marca-se 2,17 acima do Nó 2 na FIGURA A.4 e põe-se um X sobre o ramo indicando "pare" quando a primeira candidata for boa.

Resumindo, a política ótima é parar depois da primeira entrevista somente se a candidata for excelente e continuar depois da segunda entrevista somente se a candidata for regular. O valor esperado global do processo de entrevista, dado que se aja otimamente, é:

$$(25) \quad 3 (0,2) + 2,17 (0,5) + 2,17 (0,3) = 2,336$$

Marca-se este número acima do Nó 1 na FIGURA A.4. Uma vez que o valor 1,9 calculado em (23), também representa o valor esperado

se for entrevistada uma única secretária e for empregada esta candidata, a diferença ( $2,336 - 1,9 = 0,436$ ) é o valor incremental de entrevistar até mais duas pessoas.

## APÊNDICE B

### INTELIGÊNCIA ARTIFICIAL

#### B.1 - ASPECTOS DA INTELIGÊNCIA

Ao se tentar encontrar uma definição pura e simples da IA como uma ciência, depara-se com uma série de questões filosóficas onde dominam as tentativas de se definir o significado tanto de *inteligência* como de *artificial* [32].

Analisando-se o termo Inteligência Artificial, não há dúvidas sobre o significado de "*artificial*"; porém "*inteligência*" sempre foi um atributo humano e a própria maneira que se usa para "*medi-la*" é inútil na área de inteligência computacional, pois o que se observa é que qualquer processo de medição linear da inteligência é inadequado, sendo possível programar um computador que responderia muito bem às questões complicadas de um "teste de inteligência" [23].

Tratando-se de um ser humano, presume-se que ele é inteligente e esta inteligência pode ser "medida". Num programa de computador, antes de mais nada, a inteligência precisa ser "detectada".

Para se determinar se uma máquina possui inteligência ao "nível humano" existe o clássico teste de Turing [25].

Neste teste, dois seres humanos A e B e um computador C são colocados em um ambiente de forma que não haja comunicação entre A, B e C, a não ser através de um terminal de computador (FIGURA B.1).

O humano A representa o papel de interrogador, e seu objetivo é descobrir, analisando as respostas de B e C, qual deles é o computador.

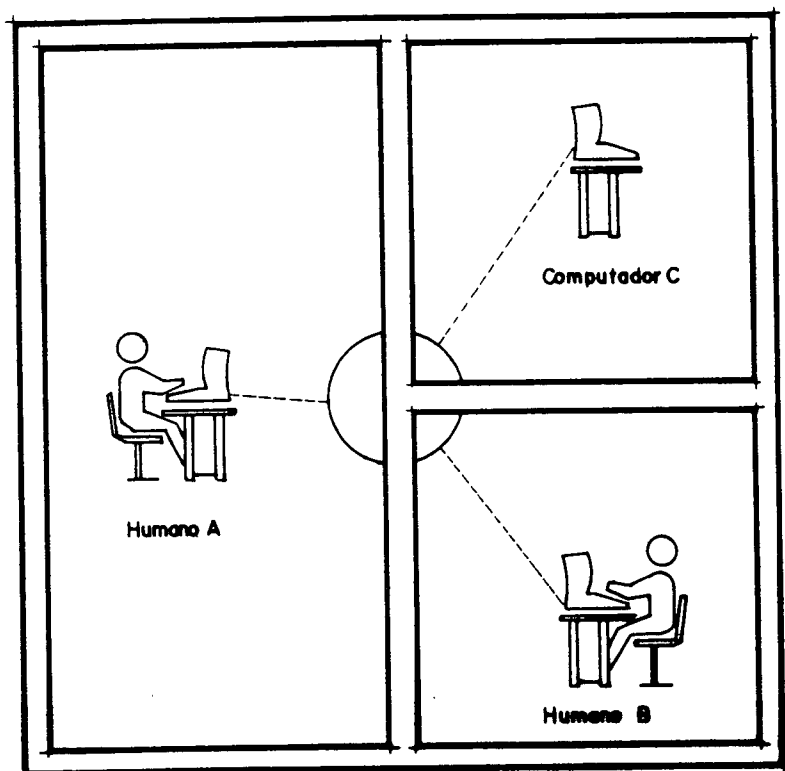


Fig. B.1 - Teste de Turing.

Caso o interrogador não consiga determinar, com um mínimo de 50% de precisão, qual dos dois (B ou C) é o outro humano, e tal resultado for confirmado por outras pessoas representando o papel de A e B, diz-se que o computador C passou pelo teste Turing e, portanto, que a máquina simula a inteligência humana.

E o quociente de inteligência da máquina seria a média dos QI's das pessoas que foram "enganadas" pelo computador [23].

No entanto, esta simulação de inteligência humana é relativa,

pois muito do que normalmente está associado à ela, como sabedoria, emoção, estética, ética e consciência entre outros, não pode, e na verdade não deve, aparecer numa simulação de inteligência de um sistema técnico [37].

Segundo Herbert Schildt [36], a questão é: se vai se fazer um computador agir como uma pessoa, então como fazer para que ele aparente ter emoções humanas? Tudo o que o computador tem que fazer é agir como se tivesse emoções humanas - ele não precisa realmente senti-las. Em suma, não importa o que o computador sente - só importa a maneira como ele age. Será suficiente se um programa simplesmente revelar um comportamento que sugira a existência de emoções. O programa não precisa senti-las da mesma maneira que uma pessoa o faria.

As maiores dificuldades para que as máquinas de hoje passem pelo teste de Turing são:

- a) sua inabilidade de comunicação em linguagem natural, e
- b) seu restrito campo de conhecimento, quando comparado ao da mente humana [31].

Analisando-se as "atitudes inteligentes", conclui-se que elas contém, no mínimo, três estágios. O primeiro, seria o de aprendizagem; o segundo, uma busca de regras, entre as que já se aprendeu, para se resolver o problema; e o terceiro, a tomada de decisão.

A busca de regras para a solução do problema, encontrada no segundo estágio, vem a ser o componente conhecido como "raciocínio".

O raciocínio humano não se parece com o raciocínio de uma máquina, uma vez que baseia-se na memória e a organização da memória humana é muito diferente da de acesso aleatório normal (RAM).



Ao tentar se lembrar de alguma coisa, existe uma grande chance de também se recordar outras coisas a ela relacionadas. Diz-se que a memória humana é "associativa" porque um fato é memorizado num conjunto de fatos a ele associado. Uma coleção de fatos armazenada em um computador parece-se mais com a maneira pela qual uma enciclopédia "se lembra" das coisas do que com a maneira pela qual os humanos se lembram.

O cérebro humano é um aparelho muito complicado, e é bem capaz de executar um grande número de operações simples simultaneamente, sendo, portanto, perfeitamente possível que o raciocínio humano seja o resultado de um grande número de operações simples executadas simultânea e repetidamente. Para o estudo de inteligência artificial, isto pode significar que, mesmo que algum dia seja descoberto o funcionamento exato do cérebro humano, não será possível usar este conhecimento para escrever programas, pois os computadores não poderiam executá-los com a mesma velocidade que um humano o faz, pelo menos com a tecnologia hoje disponível. Em outras palavras, a escolha pode ser entre fazer exatamente como os humanos e não conseguir um programa prático, ou usar os recursos especiais dos computadores para resolver os problemas de maneiras não humanas [23].

Existem dois modos de se abordar a solução de um problema e são conhecidos como "*performance mode*" e "*simulation mode*" [31]. O primeiro consiste na construção de máquinas de comportamento "inteligente", independentemente de serem os seus métodos similares aos dos seres humanos. Na segunda abordagem, há uma tentativa de imitar a maneira pela qual o ser humano realiza tarefas que exijam inteligência.

Para se entender melhor esta distinção entre os dois modos de

abordagem, pode-se traçar uma analogia com o desenvolvimento da aviação. Antes da descoberta do avião, o homem sempre teve o sonho de voar e começou a observar como as aves o faziam. Analisando-se as primeiras máquinas mais pesadas que o ar para voar, nota-se que em muitas características elas procuravam imitar algum tipo de pássaro. E algumas dessas máquinas tinham, inclusive, movimento de asas similar ao dos pássaros. Esse tipo de abordagem pertence ao segundo caso - "*simulation mode*".

Após várias tentativas para a construção dessas máquinas, e com o desenvolvimento de outras ciências como a física, a mecânica e a aerodinâmica, o homem obteve o mesmo resultado final - VOAR - por outros processos, bastante diferentes de como os pássaros o fazem. Neste caso, a solução do problema "voar" foi apresentada pela primeira abordagem "*performance mode*". E o que se observa hoje é que um moderno avião a jato soluciona o problema proposto de voar de maneira muito mais eficiente que qualquer pássaro, melhorando em muito a "performance" inicial esperada.

Se por um lado o progresso da neurofisiologia, que é a parte da biologia que investiga as funções orgânicas, processos e atividades vitais do sistema nervoso, tem dado informações, que, pelo menos atualmente, explicam com grandes detalhes como funcionam pequenas coleções de neurônios, toda vez que se tenta usar estas informações como uma maneira de implementar sistemas de IA, sempre se conclui que há maneiras melhores e mais simples de se obter os mesmos resultados [23].

Comparações entre o homem e as máquinas mostram que os sistemas baseados em IA podem apenas imitar - e não copiar - certas funções da inteligência humana [37].

Da mesma forma que os programas de xadrez não jogam da manei-

ra que um humano faria, muitos programas de IA parecem alcançar aproximadamente os mesmos resultados que os humanos, usando, entretanto, métodos que parecem completamente diferentes. Na prática o que se tem feito é a mistura de alguns métodos "humanos" com outros métodos adaptados para computadores, ou seja, usa-se um pouco de estratégia misturada com grande capacidade de armazenamento e velocidade.

Num programa de xadrez, usam-se alguns dos métodos que os humanos usam: avaliação dos movimentos e exame de algumas alternativas possíveis, entre outros. Entretanto, para tornar o programa de xadrez prático, usam-se as exclusivas vantagens do computador digital de pesquisar rapidamente várias alternativas possíveis, até um determinado nível [23].

No que diz respeito à inteligência das máquinas, existem conceitos implícitos, segundo os quais quando uma máquina tem algoritmos que simplifiquem a sua própria operação, ela já possui algum grau de inteligência [31].

O "raciocínio" de um programa de computador utiliza regras do tipo:

SE (condição)

ENTÃO (conclusão),

e isto pode parecer muito natural e promissor, até que se comece a pensar no tipo de julgamento que os humanos fazem. Quase nunca uma decisão pode ser alcançada com certeza absoluta, sendo mais comum chegar-se a conclusões do tipo "acho que o problema poderia ser..." ou "pode ser que...", do que "tenho a certeza de que..." ou "é...".

Um importante componente do raciocínio humano que não pode ser desconsiderado é a incerteza.

Há dois tipos diferentes de incerteza presentes no raciocínio, sendo o primeiro simplesmente não ter muita certeza da condição. Por exemplo, um animal em movimento pode ser observado por um instante tão pequeno, que não dê para ver se tem cauda ou não. Este tipo consiste, portanto, em não se ter certeza da evidência e é fácil de tratar nos programas. O segundo tipo de incerteza é aquele no qual a evidência é clara, ou seja, existe absoluta certeza dos fatos, mas não existe uma conexão clara entre estes e a conclusão que se pretende tirar. Esta forma de incerteza é conhecida por "incerteza de inferência" [23].

Uma das maneiras que se pode tratar a incerteza é baseando-se em probabilidades. Embora possa ser a maneira adequada para exprimir a incerteza sobre alguns eventos no mundo, não é necessariamente o melhor modo de exprimir a maneira pela qual os humanos se sentem "incertos" sobre algo.

Existe uma segunda maneira de se tratar as incertezas, que é a lógica nebulosa [23].

A lógica tradicional usa os números "1" para indicar verdadeiro e "0" para falso, sem valores intermediários. Na lógica nebulosa, pode-se usar "graus de verdade" entre "0" e "1", novamente com "0" indicando falso e "1" indicando verdadeiro, sendo que os valores intermediários representam algo "mais ou menos verdadeiro ou falso".

Um ponto importante a observar é que a aplicação de algumas regras muito simples dá a impressão de inteligência. Os seres humanos são inteligentes e, se sua expectativa for elevada a um nível adequado, tentarão interpretar qualquer coisa como indício de inteligência em alguma outra pessoa ou coisa.

No caso dos computadores, alguns filmes de ficção científica elevaram muito a expectativa do público, de maneira que é muito fácil convencer a um observador inocente de que algum programa trivial é a quintessência da inteligência do universo, pois os humanos têm uma grande tendência a atribuir, mesmo às máquinas mais simples, um pouco de sua inteligência ou de sua personalidade. Esta tendência a acreditar na inteligência dos computadores tem dois importantes aspectos. Primeiro, significa que é possível alcançar resultados práticos sem muito esforço, simplesmente emprestando um pouco da inteligência do usuário. Em segundo lugar, é uma advertência de que não é razoável acreditar muito rapidamente em tudo que se vê! [23].

## **B.2 - Busca Heurística**

A heurística é uma técnica que melhora a eficiência de um processo de busca ao sacrificar idéias de perfeição. Ela é como uma guia de turismo: boa quando aponta para direções importantes; ruim se leva a becos sem saída.

A finalidade de uma função heurística é orientar o processo de busca na direção da solução, sugerindo qual caminho seguir, quando mais de um estiver disponível.

Algumas técnicas heurísticas ajudam a orientar o processo de busca sem sacrificar qualquer idéia de perfeição que o processo possa ter tido anteriormente. Outras poderão, ocasionalmente, fazer com que um caminho excelente não seja explorado, mas, em média, elas melhoram a qualidade dos caminhos percorridos [32].

Embora as aproximações conseguidas pelas técnicas heurísticas possam não ser muito boas num pior caso, casos piores raramente surgem no mundo real.

É possível que uma pessoa resolva um problema sem saber COMO ele foi resolvido. Mas, para poder escrever um programa que solucione um problema, não é suficiente ser capaz de solucioná-lo, mas também é necessário ser capaz de descrever como chegar à solução.

A situação atual é que existem muitos problemas que as pessoas solucionam, e que são muito difíceis - se não impossíveis - de serem reduzidos à aplicação de um conjunto de regras que garantam sua solução [23].

Num extremo de perfeição, a função heurística seria tão boa que essencialmente nenhuma busca seria necessária. O sistema se deslocaria diretamente para uma solução. Mas, para muitos programas, o custo de calcular o valor de tal função seria muito maior que o dispendido no processo de busca. Em geral, há um ajuste entre o custo de avaliar uma função heurística e a economia de tempo de busca que a função oferece.

É necessário que se encontre uma regra cuja aplicação repetida tenda a levar o processo para mais perto da solução. Embora isto pareça fácil, muitas vezes é difícil compreender o problema com suficiente clareza para encontrar e aplicar uma heurística. Por exemplo, nem sempre é possível medir a distância da situação atual para a solução. É possível concluir que uma solução foi alcançada, mas saber se um movimento conduzirá para mais perto ou mais longe de uma solução é algo que freqüentemente é difícil de se conseguir.

O tipo de heurística natural que as pessoas utilizam é normalmente difícil de descobrir e expressar. O que tem sido feito é um estudo no sentido de se encontrar heurísticas que funcionem eficientemente quando executadas por um computador e computadores trabalham muito depressa. Sendo assim, é mais fácil encontrar heu-

rísticas simples e deixar que os computadores as apliquem repetidas vezes.

Pode-se pensar que uma heurística simples, aplicada muitas vezes seria, provavelmente, tão boa quanto uma heurística complexa aplicada poucas vezes, no entanto, tais generalizações são perigosas, pois é importante que se tenha o cuidado de utilizar a heurística correta [23].

### B.3 - COMPARAÇÃO ENTRE A PROGRAMAÇÃO CONVENCIONAL E A PROGRAMAÇÃO EM INTELIGÊNCIA ARTIFICIAL

As técnicas da inteligência artificial permitem a construção de um programa no qual cada parte representa uma etapa altamente independente e identificável em direção à solução de um problema ou de um conjunto de problemas. Cada parte do programa é como uma informação na mente de uma pessoa. Se aquela informação é contestada, a mente pode automaticamente ajustar seu pensamento para acomodar um novo conjunto de fatos. Não é preciso se dar ao trabalho de reconsiderar cada informação que já se aprendeu, mas apenas as partes que são relevantes àquela determinada mudança.

Um programa de IA possui uma característica notável, equivalente a uma característica vital da inteligência humana. Cada parte minúscula pode ser modificada sem afetar a estrutura do programa inteiro. Essa flexibilidade permite maior eficiência e compreensibilidade na programação - em uma palavra, inteligência.

Os programas computacionais com os quais a IA está relacionada são, primariamente, processos simbólicos, envolvendo complexidade, incerteza e ambigüidade [31].

Usualmente estes processos são de características não numéricas, onde não existem soluções algorítmicas. Isto torna necessário

que se pesquise a solução em um universo de alternativas bastante grande. E é desta maneira que se apresentam os problemas típicos do ser humano no dia-a-dia.

Tal forma de solução de problemas difere da forma usual de cálculos científicos e de engenharia, que são primariamente de natureza numérica, onde as soluções são conhecidas e produzem respostas precisas.

Em IA os programas lidam com palavras e conceitos pré-determinados, e que nem sempre garantem a solução correta, isto é, algumas respostas erradas são toleradas.

Um dos aspectos mais característicos dos programas de Inteligência Artificial é possuírem estruturas de controle separadas do domínio do conhecimento.

O conhecimento vem a ser o conjunto de informações sobre as quais o computador é capaz de raciocinar.

A inteligência depende fundamentalmente do conhecimento e o mesmo deverá estar disponível para ser usado durante a solução de um problema.

Em inteligência artificial, é comum separar-se o conhecimento do mecanismo que controla a busca da solução, de forma a permitir trocas de informações, bastando trocar a base de dados (conhecimento).

Em programas convencionais, as trocas de bases de dados geralmente acarretam em grande impacto para o programa, sendo normalmente necessário fazerem-se grandes modificações em sua estrutura básica, ou mesmo programas diferentes para acessar bases de dados diferentes.

Um fato que não pode ser contestado é o de que "inteligência requer conhecimento" [32] e o conhecimento além de volumoso, é di-



fícil de ser caracterizado com precisão e está em constante mutação.

A técnica de inteligência artificial explora o conhecimento que deverá ser representado de um modo que possa:

- ser generalizado, ou seja, não é necessário que ele represente separadamente cada situação individual. Ao contrário, situações que partilhem propriedades importantes deverão ser agrupadas. Se o conhecimento não tiver essa propriedade, será necessário mais memória do que se dispõe para representá-lo e mais tempo do que se tem para mantê-lo atualizado.

- ser facilmente modificado para corrigir erros e para refletir mudanças no mundo e na visão do mundo.

- ser utilizado em muitas situações, mesmo quando não totalmente preciso ou completo.

- ser utilizado para superar seu próprio volume, ao ajudar a diminuir a faixa de possibilidades que deverão normalmente ser consideradas.

Embora as técnicas de IA devam ser projetadas tendo em vista as restrições impostas pelos problemas de IA, há certo grau de independência entre os problemas e as suas técnicas de resolução. Tanto é possível resolver problemas de IA sem utilizar as técnicas de IA como aplicar as técnicas de IA à solução de problemas que não sejam da IA. É mais provável que isto seja proveitoso para os problemas que possuam muitas características comuns à IA.

Resumidamente, alguns dos problemas que se enquadram dentro do escopo da inteligência artificial são [32]:

- jogos;
- prova de teoremas;
- resolução de problemas gerais;
- percepção
  - visão;
  - fala;
- compreensão de linguagem natural;
- resolução de problemas especializados
  - matemática simbólica;
  - diagnose médica;
  - análise química;
  - projetos de engenharia.

#### **B.4 - FORMAS DE REPRESENTAÇÃO DO CONHECIMENTO**

A representação do conhecimento vem a ser a formalização deste num sistema, e é feita através de métodos específicos da IA. Ela pode ser imaginada como um conjunto de convenções para se descrever objetos, fatos e situações [31].

Os pesquisadores de inteligência artificial descobriram que é mais importante o conhecimento que se possui do que os métodos usados para se raciocinar sobre ele. Os métodos são necessários apenas para que esse conhecimento seja eficientemente modelado e colocado pronto para ser acessado [31]. Ao se construir um programa em IA, o problema maior não é decidir qual a linguagem de programação a ser utilizada e sim qual a melhor forma para se representar este conhecimento, de modo que se possa recuperá-lo, manipulá-lo e processá-lo com facilidade [09].

Em muitas aplicações, o conhecimento a ser codificado, na base de conhecimentos do sistema, é oriundo de relatórios descritivos que são difíceis de serem modelados [28].

Um mesmo objeto pode ter diversas representações mas a habilidade de um sistema resolver problemas, para o qual foi projetado, é altamente influenciada pela escolha da representação do conhecimento, e esta escolha dependerá da natureza do conhecimento e da preferência do programador [36].

As dificuldades encontradas ao se representar um espectro de conhecimentos, são:

- Estruturar o conhecimento explícito de forma conveniente ao tipo de problema e de acordo com a natureza do conhecimento.

- Avaliar heurísticas para manipular e inferir sobre o conhecimento.

- Extrair as informações necessárias de um especialista humano para transformá-las em conhecimento.

- Lidar com um novo conhecimento, atualizando um anterior.

- Controlar inferência e semântica.

O conhecimento ainda precisa ser melhor depurado e formalizado; há uma carência muito grande de trabalhos nesta área e deve-se estar aberto a novas idéias. Por outro lado, ao se verificar que nenhuma das formas tradicionais atende à representação que se busca, é necessário usar a criatividade e procurar novas formas [31].

Existem diversos métodos de se armazenar o conhecimento, entre eles:

- a) fatos e regras;
- b) redes semânticas;
- c) "frames";
- d) lógica proposicional;

e) lógica de predicados.

a) Fatos e regras

Na mente humana o conhecimento está relacionado a uma lista de objetos e idéias [25]. Estes conhecimentos são aplicados não só para que se atinja um determinado objetivo mas também de forma que haja um aprendizado contínuo com as novas experiências.

A inteligência pode ser dividida em duas partes:

- a) uma coleção de fatos;
- b) um meio de se utilizar estes fatos para se alcançar determinados objetivos.

Este meio vem a ser, em parte, a formulação de um conjunto de regras relacionadas a todos os fatos armazenados.

Um exemplo desta relação fatos/regras é:

FATO: A água conduz eletricidade.

REGRA: SE eu estiver na água e em contacto com energia ENTÃO eu vou levar um choque elétrico.

Esta regra foi expressa numa relação SE - ENTÃO, ou relação condicional.

Ou seja, SE uma certa condição existe, ENTÃO ocorrerá uma ação ou outra resposta.

A base de conhecimento dá as características de funcionamento do sistema e é o local onde se armazenam fatos e regras.

Na FIGURA B.2 está representada a estrutura de uma base de conhecimento com seus fatos e regras.

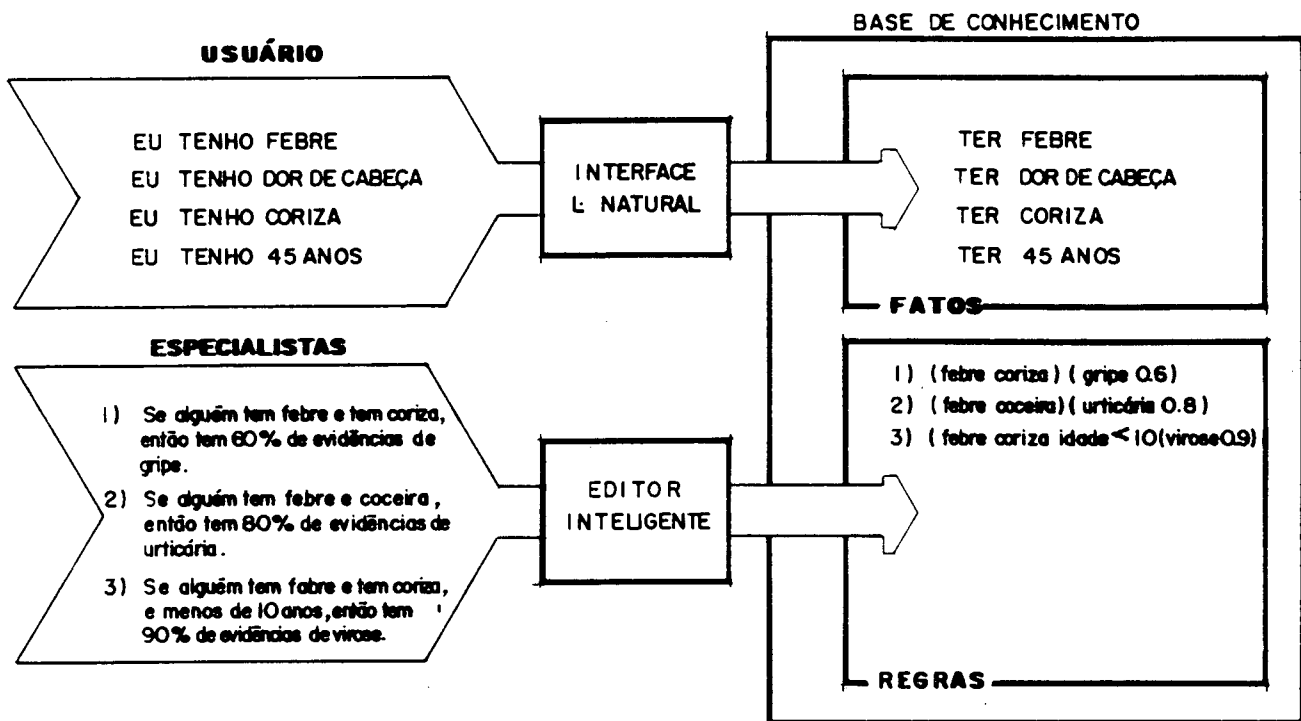


FIGURA B.2 - Estrutura e atualização da base de conhecimento.

Um novo fato pode modificar todo um processo de *inferência* de acordo com as regras que estão sendo aplicadas e também sobre os novos fatos gerados pela avaliação destas regras.

A técnica utilizada para o armazenamento do conhecimento irá influir no processo de conduzir a inferência, mas, de uma maneira geral, é um mecanismo que compara partes de "*strings*" com padrões, e dependendo do resultado desta comparação, uma atitude será tomada. Esta atitude poderá ser a busca de novas regras, ou outro encaminhamento até que se atinja um determinado objetivo.

Considerando-se a seguinte informação:

- a) Os pais de Lúcia são João e Maria;
- b) Os pais de Renato são João e Maria.

O objetivo é determinar o parentesco entre Lúcia e Renato. A base de conhecimento possui a seguinte regra:

SE duas pessoas têm os mesmos pais, ENTÃO elas são irmãs.

No processo de alcançar o objetivo, um novo fato é deduzido: Lúcia e Renato são irmãos. É o mecanismo de inferência que ajuda a chegar a este novo fato. Ele é central na habilidade de se aprender com a experiência porque permite gerar novos fatos a partir dos já existentes, que poderão ser utilizados na solução de outros problemas.

O mecanismo de inferência também ajuda a detectar erros no pensamento, e permite modificar e aprimorar as regras usadas para alcançar determinados objetivos [25].

Assume-se que o programa está sempre tentando identificar algum objetivo e para isto ele recupera todas as regras necessárias para a obtenção de um destes objetivos. Algumas vezes é necessário decompor um objetivo em submetas e, para atendê-las, serão buscadas e avaliadas novas regras, determinando-se novas submetas, e assim sucessivamente.

Isto irá gerar um processo de avaliação que dependerá dos fatos que encontra.

As estratégias de busca e avaliação de regras dependem do tipo de representação do conhecimento e da arquitetura das próprias regras [31].

Alguns fatos são obviamente mais complicados que outros e algumas regras dizem respeito a mais de um fato. Em geral, os seres humanos têm a capacidade de relacionar conjuntos de fatos e regras muito complexos na tentativa de alcançar objetivos mais difíceis.

Quando a mente humana parte para a solução de um problema, mesmo que seja um problema simples, a quantidade de dados a serem acessados é muito grande. Por exemplo, ir para o trabalho andando.

Ao se chegar na esquina, o objetivo imediato será o de atravessar a rua. Enquanto se aguarda o melhor momento para começar a atravessá-la, o cérebro estará sendo "bombardeado" com uma série de dados e suas inter-relações, como a existência ou não de semáforo no cruzamento, a velocidade e o volume do tráfego, a distância até a outra calçada, entre outros. Todos estes fatores têm que ser analisados antes que se faça qualquer movimento. Além disto, existe uma grande quantidade de impressões sensoriais que são irrelevantes para o objetivo imediato mas que estarão sendo processadas pelo cérebro, como as condições do tempo, a cor e o modelo dos carros que estão passando, o ambiente ao redor, placas de anúncios, pessoas, etc...Pode ser que também se esteja pensando onde se está indo, com que rapidez se deseja chegar lá, com quem vai encontrar quando chegar, o que vai fazer, etc...

Se houvesse a necessidade de se processar os fatos diretamente relacionados, os indiretamente relacionados e também os que não têm relação com o objetivo imediato de se atravessar a rua, o tempo consumido para que se decidisse começar a atravessar a rua seria muito grande.

Se na realidade existem tantos fatos e tantas regras a eles aplicados, como a mente humana extrai o conjunto certo de regras para se adaptar a uma determinada situação?

Quando a meta é um determinado objetivo, serão analisados os fatos e as regras que dizem respeito àquele contexto.

Isto leva a crer que existe um sistema sofisticado que guia a seleção de uma resposta adequada a uma situação específica. Esta técnica é chamada de poda ("pruning") [25]. Ela elimina os caminhos de pensamento que não são relevantes para o objetivo imediato de se alcançar uma meta.

O mecanismo de poda que atua no cérebro humano afasta qualquer fato e regra que não leve ao cumprimento do objetivo.

Quando a mente se confronta com uma situação, o mecanismo de poda guia o processo de pensamento apenas para as regras que tenham relação com a solução do problema imediato.

Uma visão geral de como a poda ajuda o cérebro a distinguir um conjunto de regras de outro é mostrada na FIGURA B.3. Neste caso a poda ajuda a decidir se o caminho a ser tomado deve ser o A ou o B. Ela faz isto com um teste SE - ENTÃO de uma condição. Se a condição for A, toma o caminho A; se for B, toma o caminho B.

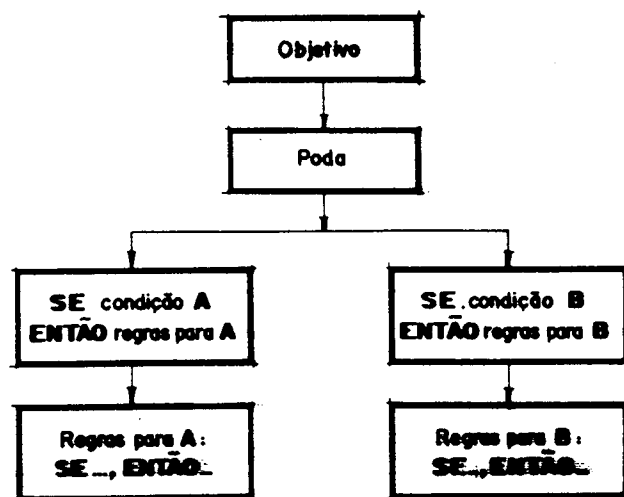


FIGURA B.3 - Poda em ação.



Tudo o que se precisa saber para tomar decisões pode estar presente no cérebro o tempo todo, mas se o dado correto não for obtido no momento exato, todo o conhecimento não será mais útil.

A poda estabelece uma ordem vital no pensamento.

No computador, ela permite que se pule ou se processe qualquer parte da base de conhecimentos, de acordo com sua relevância para um determinado objetivo. Dependendo dos fatos, o mecanismo de poda guiará o programa no sentido de processar ou passar por cima de uma seção. Isto elimina o processamento de percursos que não ajudarão a alcançar o objetivo [25].

#### b) Redes Semânticas

Numa rede semântica, a informação é representada como um conjunto de nós ligados um ao outro por um conjunto de arcos rotulados, que representam relações entre os nós.

Um fragmento de uma rede semântica típica é apresentado na FIGURA B.4.

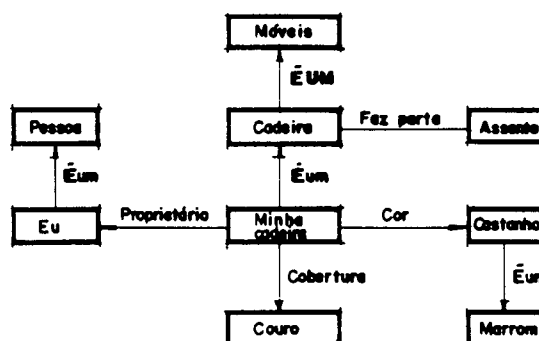


FIGURA B.4 - Uma rede semântica.

Esta técnica de representação do conhecimento foi originalmente projetada como um meio de representar o significado de pala-

vras inglesas [32] e baseou-se no modelo de representação do conhecimento da memória humana, acabando por se tornar uma valiosa ferramenta para a IA [29].

O conhecimento em rede é representado por nós num diagrama não hierárquico, onde um nó representa um conceito, um objeto ou uma situação dentro de um contexto. Ao contrário de uma árvore, numa rede todos os nós têm a mesma importância e qualquer um dos nós poderia ser usado como um ponto inicial.

Uma vantagem do método de redes semânticas para representação do conhecimento é que tanto o conhecimento hierárquico como o não hierárquico podem ser fácil e eficientemente manuseados.

Basicamente existem quatro tipos diferentes de nós:

a) Nós concepção

São parâmetros constantes da aplicação em enfoque;

b) Nós eventos

São as ações do fato que está sendo representado;

c) Nós características

São nós que descrevem propriedades de uma concepção;

d) Nós valores

São nós que correspondem aos domínios de valores da característica em consideração.

Estas quatro categorias podem ser visualizadas na figura

B.5.



Uma rede semântica ilustrando a relação entre a forma de um pássaro e a de um avião aparece na FIGURA B.6.

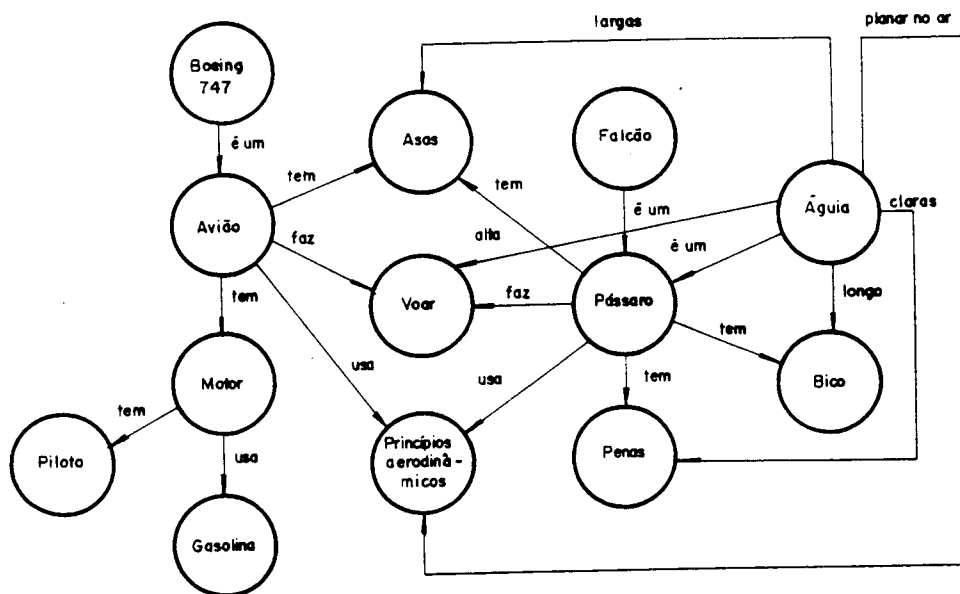


FIGURA B.6 - Rede semântica mostrando a relação entre as partes de um pássaro e as de um avião.

A partir da rede semântica pode ser criada uma base de conhecimentos que permitirá, de acordo com o exemplo da FIGURA 3.6, identificar se um objeto voador é um pássaro ou um avião. A seguir tem-se algumas das regras que podem ser parte do banco de conhecimentos pássaro-avião:

SE ATRIBUTO-OBJETO = BICO

ENTÃO OBJETO-VOADOR = PÁSSARO

SE ATRIBUTO-OBJETO = ASAS

ENTÃO OBJETO-VOADOR = PÁSSARO OU

OBJETO-VOADOR = AVIÃO

SE ATRIBUTO-OBJETO = MOTOR

ENTÃO OBJETO-VOADOR = AVIÃO

SE OBJETO-VOADOR = PÁSSARO

ENTÃO ATRIBUTO-OBJETO = PENAS

Um resumo de alguns dos princípios das redes semânticas é o seguinte:

- 1 - As redes semânticas descrevem as relações existentes entre o que é representado pelos nós.
- 2 - Os nós são círculos que têm nomes.
- 3 - As relações entre os nós são representadas por arcos que conectam os círculos.
- 4 - Uma rede semântica pode ser usada para gerar estruturas e objetos.
- 5 - Uma rede semântica pode ser usada para gerar regras para um banco de conhecimentos. [25].

Muitos pesquisadores da IA acreditam que o modelo de redes são os que mais se assemelham com o método de representação do conhecimento humano. Mas isto ainda não está provado de uma forma conclusiva [36].

c) "Frames" (Quadros)

Nos programas em IA, o conhecimento também é coletado e classificado. Nesta classificação, o conhecimento seria "dividido" e estas divisões são chamadas de "slots". Os títulos dos "slots" são chamados "atributos" e o conhecimento colocado nestes "slots" é

chamado de "valor" [32].

Um "frame" é uma estrutura que representa características de uma situação ou objeto, cujas descrições são feitas através dos "slots" [31].

Quando os valores são colocados nesses "slots", o objeto real começa a existir [32].

À semelhança das redes semânticas, os "frames" são também representados na forma de redes e relações, organizados hierarquicamente, onde os nós de mais alto nível representam conceitos gerais e os de baixo nível, conceitos mais específicos [29].

Esta forma de representação do conhecimento é útil porque, às vezes, "o que fazer e quando" não é o único objetivo em interesse, sendo também importante uma representação detalhada dos conceitos ou dos objetos físicos do contexto, para que se possa raciocinar sobre suas propriedades e inter-relações existentes [22].

Ligados a cada "frame" podem existir várias classes de informação, entre elas:

- descrição;
- ação e
- controle.

Na classe de controle tem-se o conhecimento sobre o conhecimento, chamado "meta-conhecimento".

Os "frames" podem não ocorrer sozinhos e sim em conjuntos relacionados, ligados em árvores de "frames".

Quando um "frame" é acessado, alguns de seus "slots" podem já estar preenchidos - são valores "default". Estes "slots" poderão ser instanciados de acordo com a situação que se apresenta. Para melhor compreensão, suponha-se o seguinte fato:

Ao se entrar numa sala, imagina-se logo um paralelepípedo com

portas e janelas nas paredes. Estas são as informações padrão. Agora, os detalhes a respeito do tapete, móveis, adornos e outros são completados com as informações necessárias para descrever corretamente a sala. Isto está indicado na FIGURA B.7.

Nem só situações e objetos podem ser representados pelos "frames". Eles também podem ser utilizados para expressar o significado de ações. Cada um dos "frames" contém os elementos essenciais para caracterizar a ação em pauta [31].

Na FIGURA B.8 tem-se a representação de uma ação e os efeitos dela decorrentes.

Um "frame" contém descrições de várias tarefas e suas respectivas ações, num contexto de ação, que ficam prontas para serem disparadas tão logo um "slot" seja acessado, chamadas "demônios".

Demônios, conceitualmente, são procedimentos que observam, até que alguma condição se torne verdadeira, e então, ativam um processo associado [32].





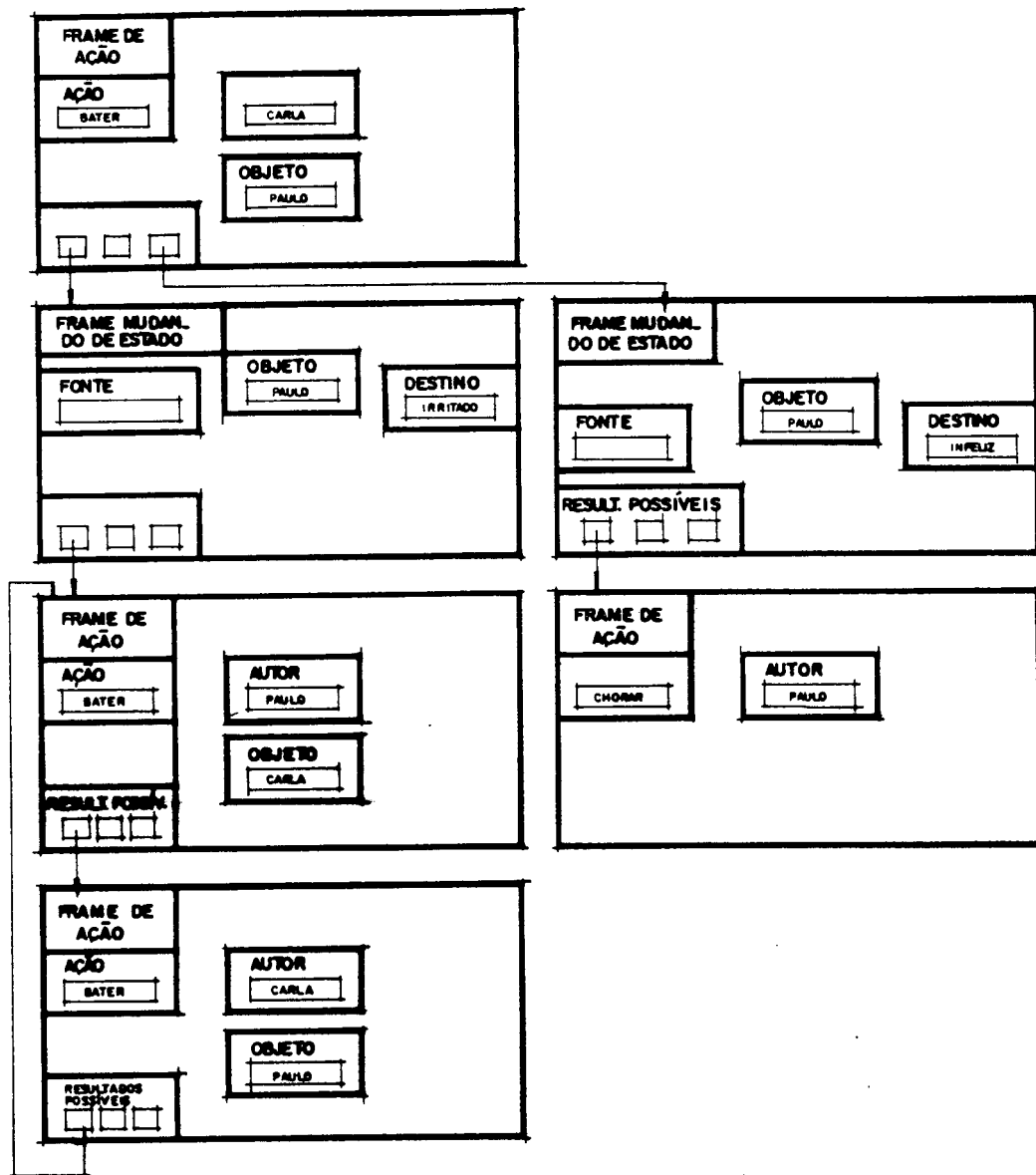


FIGURA B.8 - Representação de uma ação completa com resultados.

d) Lógica Proposicional (Programação e Lógica)

A lógica proposicional trata da operação sobre proposições, ou seja, frases que podem assumir ou não assumir valores verdadeiros ou falsos, e expressões lógicas deduzidas destas frases formadas através do uso de conectivos como:

- ~ (negação);
- ^ (e lógico);
- ∨ (ou lógico);
- ⇒ (implicação).

Alguns fatos simples da lógica proposicional [31]:

Está chovendo.

CHOVENDO

Está ensolarado.

ENSOLARADO

Está ventando.

VENTANDO

Se estiver chovendo não estará ensolarado.

CHOVENDO = ⇒ ~ ENSOLARADO

A lógica proposicional possui um poder expressivo pequeno, pois não permite a expressão de frases onde exista alguma informação indefinida e não cria novas informações [29].

e) Lógica de Predicados (Regras de Produção)

A lógica de predicados é citada como uma poderosa ferramenta para o modelamento do mundo, de uma maneira formal. Isto se deve ao fato de ter sido desenvolvido por matemáticos, como um meio de formalizar teorias e de ter uma notação simples, capaz de traduzir situações da vida cotidiana de forma bem definida. A lógica de

predicados permite ainda que, com regras de inferência, possam ser feitos questionamentos a respeito dos fatos.

Seus componentes elementares são [29]:

- Símbolos predicativos;
- símbolos funcionais;
- símbolos variáveis;
- símbolos constantes e
- símbolos especiais ou conectivos.

Símbolos predicativos são usados para representar uma relação no domínio do conhecimento. Por exemplo: CASADO (JOÃO, MARIA), onde JOÃO E MARIA são símbolos constantes e CASADO é um símbolo predicativo que relaciona JOÃO com MARIA.

Símbolos funcionais são usados para denotar funções no domínio do conhecimento. Por exemplo: pai (JOÃO) e mãe (JOSÉ), representam o nome do pai de JOÃO e da mãe de JOSÉ.

Símbolos variáveis são representações para informações não conhecidas, através dos quais o objetivo vai agir. Por exemplo: CASADO (JOSÉ, x) ou mãe (x).

Os blocos elementares na construção de expressões, na linguagem de predicados, são chamados de "cláusulas".

Para se representar uma sentença por uma cláusula é necessário que se verifique qual a relação e quais as entidades que se relacionam.

Uma cláusula pode ser representada por mais de uma maneira.

Por exemplo:

A cor do carro de José é vermelha.

COR (carro (JOSÉ), VERMELHA)

COR-CARRO (JOSÉ, VERMELHA)

VALOR (cor (carro, (JOSÉ)), VERMELHA)

Fórmulas mais complexas podem ser representadas com a utilização da negação ( $\sim$ ) e dos conectivos "e" ( $\wedge$ ), "ou" ( $\vee$ ) e "implicação" ( $\Rightarrow$ ). Estes conectivos tornam possível a composição de expressões da forma SE-ENTÃO. Por exemplo:

João joga bola e gosta de natação.

João joga futebol ou volibol.

Se João joga futebol então não joga volibol.

São representadas por:

JOGA (JOÃO, BOLA)  $\wedge$  GOSTA (JOÃO, NATAÇÃO)

JOGA (JOÃO, FUTEBOL)  $\vee$  JOGA (JOÃO, VOLIBOL)

JOGA (JOÃO, FUTEBOL)  $\Rightarrow \sim$  JOGA (JOÃO, VOLIBOL)

Na lógica de predicados a cada fórmula é dada uma interpretação, ou seja, um significado que depende do contexto do conhecimento sobre o qual se está trabalhando. Diz-se que uma cláusula tem valor V (verdadeiro) se sua interpretação é verdadeira e F (falsa) se sua interpretação é falsa.

Na lógica de predicados mais geral, pode haver uma cláusula  $P(x)$  que tenha um valor verdadeiro para qualquer valor que  $x$  possa receber, e outras vezes, uma cláusula  $P(x)$  que tenha um valor verdadeiro para pelo menos um valor de  $x$ . Para se representar estes fatos, usam-se os símbolos quantificadores "para todo" ( $\forall$ ) e "existe" ( $\exists$ ) [29].

Exemplos:

Todos os homens são pessoas.

$$\forall x \text{ homem } (x) \Rightarrow \text{pessoa } (x)$$

Algumas pessoas têm o cabelo loiro.

$$(\exists x) [\text{pessoa } (x) \Rightarrow (\text{COR-CABELO}(x, \text{LOIRO}))].$$

### B.5 - SISTEMAS ESPECIALISTAS

Sistemas especialistas e sistemas baseados em conhecimento diferem levemente um do outro [03]. O desenvolvimento de um típico sistema especialista inicia com um engenheiro do conhecimento que exaustivamente entrevista uma reconhecida autoridade em um campo particular e codifica a especialidade obtida em fatos e regras. Depois de representado simbolicamente, o conhecimento extraído é transportado para um computador. Um procedimento similar caracteriza o desenvolvimento de sistemas com base em conhecimento. Contudo, estes sistemas derivam seu particular conhecimento de fontes outras que especialistas humanos, e incorporam tópicos de assuntos que não requeiram aptidão ou educação especiais, como aqueles encontrados em livros ou manuais.

Virtualmente todos os sistemas especialistas são sistemas baseados em conhecimento, não sendo o inverso necessariamente verdadeiro. Um programa de inteligência artificial para o "jogo da velha" não pode ser considerado um sistema especialista, ainda que possua o conhecimento especializado separado do resto do programa.

Antes de se decidir se computadores podem tomar decisões de especialistas, deve-se primeiro delinear os atributos de um especialista. Randall Davis, do M.I.T., fornece a seguinte definição [38]:

O especialista deve ser capaz de:

- a) resolver problemas;
- b) explicar os resultados;
- c) aprender com a experiência;
- d) reestruturar seu conhecimento;
- e) quebrar as regras quando necessário;
- f) determinar a validade dos fatos;
- g) diminuir o desempenho gradativamente quando se alcançam os limites de seu conhecimento.

Atualmente os computadores são capazes de executar os três primeiros itens, mas não têm a capacidade humana de se reprogramar sozinho ou quebrar regras quando necessário [26].

Embora sistemas especialistas e especialistas humanos possam desempenhar tarefas idênticas em alguns casos, as características de ambos são criticamente diversas. A TABELA B.1 resume esta comparação [03]:

CONHECIMENTO HUMANO	CONHECIMENTO ARTIFICIAL
<p>perecível            difícil de transferir            difícil de documentar            imprevisível            caro            discriminatório            individualizado            criativo            adaptável            enfoque amplo            baseado em senso comum</p>	<p>permanente            fácil de transferir            fácil de documentar            consistente            de preço razoável            imparcial            social            sem inspiração            inflexível            enfoque restrito            técnico</p>

TABELA B.1 - Comparação entre conhecimento especializado humano e artificial.

Os seres humanos - especialistas ou não - possuem o conhecimento advindo do senso comum, que se constitui num grande conheci-

mento sobre o mundo, acumulado durante toda sua vida e que permeia todas as suas decisões. Devido à enorme quantidade de conhecimento de senso comum torna-se difícil construir um programa inteligente, particularmente um sistema especialista.

Senso comum inclui conhecimento sobre o que não se sabe assim como o que se conhece. Por exemplo, se for perguntado como se apresentava a temperatura da cidade em que se vive, há um ano atrás, puxa-se pela memória para recuperar a informação. Se a pergunta fosse sobre uma cidade desconhecida, imediatamente responde-se que não se sabe. Da mesma forma, se for perguntado que carro D. Pedro I dirigia ao proclamar a Independência, sabe-se que isto não ocorreu pois não havia carro naquela época. Quando um S.E. é questionado sobre informações que não disponha ou não existam, ele não detecta esta situação por não possuir senso comum. Então, iniciará exaustiva pesquisa nos seus fatos e regras para procurar a solução e quando esta não for encontrada, pode julgar que é porque seu conhecimento está incompleto e solicita informação adicional para completar sua base de conhecimento. Por isto, sistemas especialistas são freqüentemente utilizados como uma ajuda ou consultoria para que um especialista humano ou um iniciante possa tomar uma decisão em algum problema [03].

Apesar de algumas limitações, existem muitas vantagens em se ter um especialista no computador. Ao contrário de um especialista humano, que precisa dormir, comer, descansar, tirar férias e assim por diante, um sistema especialista é útil 24 horas por dia, todos os dias do ano. Também poderão ser criados muitos sistemas especialistas enquanto o número de especialistas humanos tende a ser limitado. Além disto, o especialista computadorizado nunca morre -

levando o conhecimento consigo! O conhecimento num S.E. pode ser facilmente copiado e armazenado, tornando rara a perda permanente do conhecimento especializado.

Outra vantagem dos S.E. sobre os humanos é que eles sempre estão no pico do desempenho. Quando um especialista humano fica cansado, diminui a confiança nas suas informações. E o S.E. gerará sempre a melhor opinião possível - dentro dos limites do seu conhecimento, é claro.

Uma vantagem menos importante de um S.E. é a sua falta de personalidade. Como se sabe, nem sempre as personalidades são compatíveis. Se não se der bem com o especialista, pode-se ficar relutante em se utilizar o seu conhecimento. A situação reversa também é verdade. Se o especialista não gostar da pessoa que necessita da informação, talvez não seja capaz de entregar uma informação digna de confiança. Entretanto, um S.E. não tem (ou pelo menos não deve ter) uma personalidade, eliminando-se este problema.

Uma vantagem final dos S.E. sobre os humanos é que depois que existe um S.E., pode-se criar um novo "especialista" simplesmente copiando-se o programa, enquanto um humano precisa de longo tempo para se tornar especialista num assunto, o que torna difícil se adquirir novos especialistas humanos [36].

Os sistemas especialistas podem ser classificados de acordo com as áreas de aplicação ou pelas categorias de problemas que pretendem resolver.



As áreas de aplicação são [29]:

- Ciências básicas como:

matemática;

física e

química.

- Ciências aplicadas como:

ciência da computação;

eletrônica;

engenharia;

geologia e

meteorologia.

- Áreas de atuação como:

agricultura;

manufatura;

gerenciamento;

atividades militares;

direito;

medicina;

controle de processos e

tecnologia espacial.

Quanto às categorias de problemas pode-se classificar os S.E.

em:

a) Interpretação - inferem descrição de situações a partir da análise de dados nem sempre corretos e normalmente incompletos.

b) Diagnóstico - inferem causas de funcionamento imperfeito de sistemas a partir de sintomas não bem definidos. O sistema é capaz de relatar os sintomas de uma falha específica.

c) Reparo - normalmente agregados aos sistemas de diagnóstico, eles propõem ações para o reparo das falhas.

d) Monitoração - interpretação continua de sinais, bem como comparação destes sinais com observações pré-estabelecidas. Tocam alarmes quando algo não corre bem ou disparam ações previstas.

e) Previsão - inferem conseqüências a partir de situações dadas.

f) Controle - conjugação de vários problemas como interpretação, monitoração e reparo em sistema de tempo real.

g) Projeto - ferramenta auxiliar no desenvolvimento de projetos, funcionando como assistente do projetista.

Se não fosse por um sistema especialista chamado MYCIN, os S.E. talvez tivessem permanecido nos laboratórios de pesquisa de IA e não teriam progredido para o mundo comercial.

Um dos maiores problemas da imagem da IA foi o de que muitas pessoas, incluindo outros programadores, achavam que as técnicas de IA funcionavam apenas para problemas simples que só necessitassem de uma série de regras e suposições. Estas pessoas acreditavam que a IA nunca seria usada para resolver problemas difíceis.

O MYCIN mudou tais idéias. Ele foi o primeiro S.E. bem sucedido do mundo. Desenvolvido na Universidade de Stanford, no meio da década de 70, foi projetado para ajudar os médicos no diagnóstico de certas doenças bacterianas.

O diagnóstico de doenças é uma tarefa que essencialmente compara os sintomas apresentados pelo paciente com os sintomas característicos das doenças, até que se encontrem pontos em comum. O problema é que é difícil para um médico diagnosticar rápida e con-

fiavelmente, uma vez que é muito grande o número de doenças existentes. O MYCIN ajuda confirmando o diagnóstico.

Outro exemplo de um S.E. comercialmente viável é o PROSPECTOR, que foi criado em 1978 por Richard Duda, Peter Hard e Rene Reboh. É um S.E. em geologia. Prognostica a probabilidade de se encontrarem depósitos de certos minerais em determinadas regiões. Existem muitas variações deste programa, incluindo programas que podem ajudar na descoberta de petróleo, gás natural e hélio [36].

Waterman [44] apresenta mais de 200 sistemas especialistas nas diversas áreas e diferentes categorias.

#### **B.6 - LINGUAGENS**

Desde o início da era dos computadores tem havido discussão sobre linguagens de programação. Muitas delas se aglutinaram em alguns tipos básicos, que são chamados de paradigmas de linguagem. O paradigma maior, ou mais influente, tem sido o da programação imperativa, onde o programa baseia-se em lógica e controle, tendo como base a seqüência de instruções que o programador deve fornecer ao computador para que este a execute cegamente. Neste princípio, os programas descrevem exatamente, passo a passo, o que deve ser feito para solucionar o problema. Atualmente, as linguagens como BASIC, PASCAL e FORTRAN, são exemplos de linguagens usadas em programação imperativa. Este paradigma de programação tem sido muito útil pela própria arquitetura da máquina, derivada ainda da arquitetura inicial de Von Newman, que facilita a execução de programas seqüenciais.

Com o desenvolvimento tecnológico, estão sendo criados novos tipos de arquiteturas e com elas novos paradigmas de linguagens de

programação.

O projeto japonês de quinta geração prevê uma máquina com diversos recursos novos que facilitem o uso dos computadores. Paralelamente ao desenvolvimento de novas arquiteturas, foram sendo desenvolvidas também novas linguagens de programação que introduziram novas facilidades para a confecção dos programas [29]. No novo paradigma de programação que está sendo criado, o programa baseia-se em conhecimento e estratégia.

Embora seja formalmente possível escrever qualquer programa em qualquer linguagem, o processo de construir sistemas de IA pode ser facilitado consideravelmente pela utilização de uma linguagem de programação que forneça apoio a uma variedade de estruturas comuns, tanto para dados como para controle [36].

Como principais recursos desejáveis de uma linguagem de IA têm-se [32]:

- manipulação de listas;
- decomponibilidade em módulos fáceis de alterar;
- estruturas de controle flexíveis;
- interatividade;
- eficiência;
- possibilidade de vinculação do tamanho de uma estrutura de dados, ou do tipo de um objeto em que se vai operar, à medida que o programa estiver sendo executado;
- dedução automática;
- estruturação do conhecimento;
- comportamento direcionado ao objetivo;
- capacidade de intercalar procedimentos e dados;
- flexibilidade quanto à inferência e ao tipo de resultado esperado.

Nenhuma linguagem existente fornece todos estes recursos. Geralmente alguns são fornecidos em detrimento de outros.

Entre as linguagens desenvolvidas para a pesquisa da inteligência artificial, tem-se entre outras [32]:

- IPL..... Uma linguagem preliminar de processamento de listas.
- LISP..... A linguagem de IA mais amplamente utilizada, nos Estados Unidos, em que a principal estrutura de dados é uma lista.
- INTERLISP..... Um dialeto relativamente recente de LISP, maior que o LISP puro e que fornece uma gama mais ampla de capacitações.
- SAIL..... Um derivado do ALGOL, com diversos recursos adicionais, incluindo apoio para uma memória associativa.
- PLANNER..... Uma linguagem que facilita o processamento direcionado ao objetivo.
- KRL..... Uma linguagem que apóia estruturas complexas semelhantes a "frames".
- PROLOG..... Uma linguagem baseada em regras construídas sobre um provador de teorema de lógica de predicados.

Na FIGURA B.9 [32] tem-se uma genealogia de linguagens de programação de IA e como elas se relacionam entre si. Note-se que dois pontos de partida são apresentados: IPL e ALGOL.

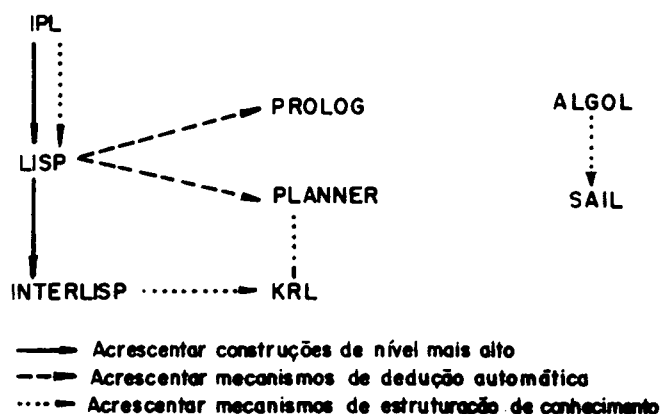


FIGURA B.9 - Genealogia de linguagens de programação de IA.

### B.6.1 - LISP

O nascimento das pesquisas em inteligência artificial pode ser associado à criação da linguagem LISP - a primeira linguagem específica para pesquisa em IA - feita em 1960, por John McCarthy, do M.I.T. [36].

LISP é a abreviação de LIST Processing (processamento de listas). Ela é uma linguagem de computador usada em muitos aplicativos em inteligência artificial.

Uma de suas maiores qualidades é que ela pode manipular listas facilmente. Uma lista pode ser constituída pelas palavras de uma frase, os nomes das pessoas em uma empresa ou qualquer seqüência de palavras colocadas em qualquer ordem.

As listas não precisam ser de tamanho fixo e o conjunto de propriedades conhecidas pode mudar drasticamente à medida que o programa é executado, sendo que suas partes podem ser retiradas do contexto recursivamente, de forma que várias operações podem ser nelas efetuadas [25].

O fato de que tanto dados como procedimentos são representa-

dos como listas, torna possível integrar conhecimento declarativo e procedimental em uma única estrutura, como a lista de propriedades. Também torna possível para um programa construir um procedimento e depois executá-lo.

A maioria dos sistemas LISP processa interativamente, o que facilita o desenvolvimento de todos os tipos de programas. Isto muitas vezes é importante em áreas de aplicação da IA, onde os problemas são complexos demais para serem resolvidos por um programa, sem a intervenção humana [32].

Qualquer programa em LISP corresponde a uma função, que também pode ser definida pelo usuário. Uma definição de uma função é composta de um nome, uma lista de variáveis e um corpo. A lista de variáveis corresponde aos argumentos da função e o corpo corresponde a como a função será avaliada.

Numa programação funcional, os programas são escritos em nível mais alto, são claros e é mais simples de se verificar se estão corretos [29].

Há diversas razões para se utilizar a linguagem LISP para construir sistemas de IA [32]:

a) Sua estrutura de dados principal é uma lista, muito útil para representar boa parte do conhecimento utilizado em programas de IA;

b) Uma coleção de fatos a respeito de um objeto individual pode ser facilmente representada na lista de propriedades que está associada ao "átomo" (variável), representando o conceito. A lista de propriedades é simplesmente uma lista de pares de atributo-valor;

c) A estrutura de controle mais natural é a recursão, apropriada para muitas tarefas de solução de problemas.

Para programas que requeiram um amplo processamento de listas e recursividade, é preferível uma linguagem como LISP [45].

Existem muitos dialetos do LISP, variando desde os nomes das funções padrões e a ordem de seus argumentos, até questões substantivas, envolvendo tipos de recursos supridos [32].

Grande parte da comunidade da inteligência artificial, principalmente a norte-americana, usa a linguagem LISP para desenvolver seus sistemas de inteligência artificial. Muitas pessoas acham que a capacidade da LISP, para computação simbólica, faz dela a linguagem preferida. Não obstante, um programa escrito em LISP não é, por definição, um sistema especialista. O usuário de um sistema especialista está interessado no que o sistema faz e não na linguagem em que ele foi escrito.

Embora a LISP seja particularmente poderosa para processamento simbólico e vários ambientes de programação de grande capacidade baseados em LISP tenham sido desenvolvidos pela comunidade de IA, outras linguagens podem ser consideradas, pois se um programador já estiver mais familiarizado com outra linguagem, sua eficiência de programação será melhor nesta outra linguagem.

O uso destas outras linguagens pode requerer um esforço adicional de programação, mas é totalmente possível implantar um sistema especialista em uma linguagem que não seja LISP.

Os engenheiros do conhecimento e os programadores, que entendam as idéias essenciais do projeto de um sistema especialista, podem escolher entre muitas linguagens de alto nível.

Uma abordagem recente é o uso da PROLOG, uma linguagem de programação lógica [45].



### B.6.2 - PROLOG

Na década de 70, um dos eventos mais importantes na área da IA não foi reparado nos Estados Unidos até o início dos anos 80. Este evento foi a criação da linguagem PROLOG, em 1972, por Alain Colmerauer em Marselha - França [36].

PROLOG é a abreviação de PROgramming in LOGic (Programação em Lógica). Ela é uma linguagem que foi criada especialmente para responder perguntas sobre uma base de conhecimentos que consiste em fatos e regras [25].

Ao contrário da LISP, possui algumas características especiais, como a possibilidade de se trabalhar mais diretamente com a base de conhecimentos, além de ter uma sintaxe mais simples.

Em 1980, podia-se dizer que a LISP era a linguagem de IA escolhida nos Estados Unidos enquanto PROLOG ocupava a mesma posição na Europa.

Entretanto, em 1981, esta situação mudou com o anúncio por parte dos japoneses de que eles pretendiam usar a linguagem PROLOG como a linguagem básica para seus computadores de quinta geração.

O que faz a PROLOG tão importante na história da IA é que ela possibilita um entendimento mais profundo do processo de pensamento, se comparada com a linguagem LISP. Por exemplo, a PROLOG contém facilidades para que se trabalhe direto na base de conhecimentos e rotina de "backtracking", ambas necessárias para a resolução de problemas em IA [36].

É bem fácil imaginar uma linguagem na qual fosse somente descrito o problema (de maneira geral e formal) e a máquina encontrasse a solução a partir desta descrição. Para as linguagens que usam esta maneira de "descrever o problema", denomina-se o paradigma de linguagem declarativa. Um exemplo de um programa

declarativo:

- 1 - Quem tem asma deve evitar lugares poluídos.
- 2 - Cubatão é um lugar poluído.
- 3 - João sofre de asma.

Estas são declarações suficientes para a solução do problema:

- Que lugares João deve evitar?

E o computador deve ser capaz de responder Cubatão. Note-se que esta foi uma dedução lógica a partir dos fatos (2) e (3) e da regra (1).

É neste contexto que surge a linguagem PROLOG e está aquém do ideal de linguagem declarativa, mas já é um considerável avanço [29].

Um programa PROLOG pode ser confundido com uma base de dados em que estão descritos os fatos de um universo e as regras que regem o mecanismo de funcionamento e relacionamento deste universo. Para obter soluções deste programa, são feitas perguntas ("*query*", "*queries*"), que gerarão respostas a partir dos fatos e regras, e de um mecanismo de inferência próprio da PROLOG [29].

Cada relação no programa PROLOG é composta de um conjunto de cláusulas. O interpretador PROLOG tenta encontrar provas da verdade de cada relação especificada. Normalmente, a relação conterá variáveis, e parte do processo de prova envolverá encontrar encadeamentos lógicos de variáveis que tornem verdadeira a relação. O retrocesso que poderá ser necessário para encontrar tal conjunto de encadeamentos lógicos, é tratado pelo interpretador PROLOG.

O desenvolvimento e a utilização do PROLOG levantou interesse na idéia da programação lógica. Mas também há vantagens para o ambiente de programação tradicional do LISP. Como resultado, diversos sistemas de programação lógica foram desenvolvidos para pro-

cessar dentro do LISP. Utilizando tais sistemas, é possível escrever programas que combinem os métodos de busca embutidos do PROLOG com técnicas específicas, definidas pelo programador, codificadas em LISP [32].

Embora desde 1981 a linguagem PROLOG tenha ganho popularidade nos Estados Unidos, não se pode afirmar se ela vai ou não se tornar a principal linguagem norte-americana de pesquisa em IA [36].

### B.6.3 - A LINGUAGEM C PARA INTELIGÊNCIA ARTIFICIAL

C é uma das linguagens de programação mais populares em uso. Ela proporciona um programa estruturado que não limita a criatividade do programador, enquanto os compiladores C produzem programas extremamente rápidos e eficientemente executados. Por estas e outras razões, muitos pacotes de "software" são escritos em C [36].

Existe um número muito maior de programadores que sabem a linguagem C do que os que sabem as linguagens específicas para IA. Portanto, parece lógico que as técnicas de IA também sejam codificadas em C. Enquanto as linguagens que são usadas para pesquisa em IA são mais eficientes para realizar buscas na base de conhecimentos, elas geralmente são mal adaptadas para programação em geral. Por exemplo, a linguagem PROLOG necessita alguma capacidade em programação procedural, uma deficiência que pode tornar certas tarefas - como somar números numa lista - mais difíceis do que elas precisam ser. Além disto, se a aplicação não requer a técnica de retrocesso ou as facilidades de se trabalhar com a base de conhecimentos, elas são simplesmente recursos adicionais desnecessários.

Enquanto a maior parte da pesquisa em IA é feita em linguagens específicas para ela, como LISP e PROLOG, a maioria das apli-

cações do mundo real, como pacotes de contabilidade ou processadores de texto, são escritos em linguagens de utilização geral, como C.

Não tem sido fácil transportar conceitos de uma linguagem específica da IA para uma linguagem convencional. Complicando ainda mais o processo, existe o fato de que a maior parte das linguagens de IA (incluindo LISP e PROLOG) são *declarativas*, enquanto linguagens de utilização geral são *procedurais*.

Numa linguagem declarativa, o programador diz ao computador O QUÊ fazer; numa linguagem procedural, o programador diz ao computador COMO fazê-lo.

Desta forma, linguagens de IA e linguagens de utilização geral tendem a ocupar os lados opostos do espectro de linguagens de programação.

Não existe uma única técnica de IA que não possa ser implementada usando-se uma linguagem procedural como C. Existem apenas razões históricas porque as linguagens especiais de IA foram inventadas em primeiro lugar [36].

Aprender a implementar várias estruturas da IA através da linguagem C é importante porque então estes elementos podem ser aplicados a toda uma gama de situações existentes. Muitas aplicações têm sido, e continuarão a ser, escritas em C.

Embora às vezes seja possível unir um módulo que foi escrito em C a outro que tenha sido escrito numa linguagem de IA (como a "Turbo-PROLOG"), isto pode não ser a solução mais desejável. Primeiro, usar duas linguagens separadas exige mais administração e coordenação dos problemas porque será necessário manter dois projetos separados. Segundo, usar uma linguagem de IA separada significa também que os programadores originais do sistema terão que

aprender a linguagem ou terão que ser contratados outros programadores.

O melhor caminho é implementar tudo o que a técnica em IA necessitar na linguagem original do pacote de aplicação [36]. No entanto, deve ser levado em consideração que, conforme o caso, uma só linguagem não dispõe de todos os recursos necessários, sendo que a união de módulos escritos em linguagens diferentes pode ser uma solução, pois reúne vantagens de ambos os lados.

## REFERÊNCIAS BIBLIOGRÁFICAS

- [01] - ACKOFF, L./SASIENI, Maurice W. - Pesquisa Operacional.  
Coleção Universitária de Administração.
- [02] - ADAMOWICZ, M./ALBANO, A. - A solution of the rectangular cutting-stock problem.  
*IEEE Transactions on Systems, Man and Cybernetics* - Vol. SMC - 6 Number 4 - 1976.
- [03] - CARVALHO, Roberto Lins de/ Grupo SAFO - Processamento de Conhecimento.  
I Escola Brasileiro-Argentina de Informática - 1987.
- [04] - CHRISTOFIDES, Nicos/WHITLOCK, Charles - An algorithm for two-dimensional cutting problems.  
*Operations Research* - Vol. 25 Number 1 - 1977.
- [05] - DORI, Dov/BEN-BASSAT, Moshe - Efficient Nesting of Congruent Convex Figures.
- [06] - DYCKHOFF, H./KRUSE, H.J./ABEL, D./GAL, T.- Trim loss and related problems.  
*OMEGA, The Int. Jl. of Management Science* - Vol. 13 - 1985.
- [07] - EASTMAN, Charles M. - Representations for Space Planning.  
*Communications of the ACM* - Vol. 13 Number 4 - 1970.
- [08] - ———. - Heuristic Algorithms for automated space planning.  
*Proc. Second International Conf. on Artificial Intelligence* - British Computer Society, London - 1971.
- [09] - ———. - Preliminary Report on a System for General Space Planning.  
*Communications of the ACM* - Vol. 15 Number 2 - 1972.

- [10] - ———. - Automated Space Planning.  
*Artificial Intelligence* - Vol. 4 - 1973.
- [11] - EHRLICH, Pierre Jacques - Pesquisa Operacional - Curso In-  
trodutório.  
Atlas.
- [12] - FERREIRA, Áureo C./SCHRAMM, Eduardo S./FERNANDES, Rosinete -  
Introdução ao CAD/CAM e sua utilização nas áreas de cal-  
çados, vestuário e afins.  
Anais Sobracon.
- [13] - GILMORE, P.C./GOMORY, R.E - A linear programming approach  
to the cuttingstock problem - part I.  
*Operations Research* - Vol. 9 - 1961.
- [14] - ———. - A linear programming approach to the cutting  
stock problem - part II.  
*Operations Research* - Vol. 11 - 1963.
- [15] - ———. - Multi-stage cutting-stock problems of two and more  
dimensions.  
*Operations Research* - Vol. 13 - 1965.
- [16] - ———. - The theory and computation of knapsack functions.  
*Operations Research* - Vol. 14 - 1966.
- [17] - GOLDEN, Bruce L. - Approaches to the cutting stock problem.  
*AIIE Transactions* - Vol. 8 Number 2 - 1976.
- [18] - HAESSLER, Robert W. - A heuristic programming solution to a  
nonlinear cutting stock problem.  
*Management Science* - Vol. 17 Number 2 - 1971.
- [19] - ———. - Selection and design of heuristic procedures for  
solving roll trim problems.  
*Management Science* - Vol. 34 Number 12 - December 1988.

- [20] - HAYES-ROTH, Frederick - Knowledge based expert systems.  
*IEEE Computer* - Oct. 1984.
- [21] - HINXMAN, A.I. - The trim loss and assortment problems: a survey.  
*Eur. Jl. Operational Research* - Vol. 5 - 1980.
- [22] - JACKSON, P. - Review of knowledge representation tools and techniques.  
*IEEE Proceedings* - Vol 134 Number 4 - 1987.
- [23] - JAMES, Mike - Inteligência Artificial em Basic.  
Editora Campus Ltda. - 1986.
- [24] - KING JR., Ralph - Made in the U.S.A - Technological change in the clothing industry.  
*Forbes* - Vol. 141 Number 11 - 1988.
- [25] - LEVINE, Robert I./DRANG, Diane E./EDELSON, Barry - Inteligência Artificial e Sistemas Especialistas - Aplicações e exemplos práticos.  
Editora McGraw-Hill, Ltda. - 1986.
- [26] - McNALLY, Phil/INAYATULLAH, Sohail - The rights of robots.  
*Futures* - Butterworth & Co (Publishers) Ltd. -April 1988.
- [27] - MINSKY, Marvin - A Framework for representing knowledge.  
Mind Design - Bradford Books, 1981.
- [28] - NILSSON, Nils J. - Principles of Artificial Intelligence.  
Tioga Publishing Co. - 1980.
- [29] - NIZAM, Omar - Inteligência Artificial e Sistemas Especialistas.  
ABACE - 1987.



- [30] - QU, Weishuang/SANDERS, Jerry L. - A nesting algorithm for irregular parts and factors affecting trim losses.  
*International Journal of Production Research* - Vol. 25  
Number 3 - 1987.
- [31] - RIBEIRO, Horácio da Cunha e Souza - Introdução aos Sistemas Especialistas.  
Livros Técnicos e Científicos Editora S.A. - 1987.
- [32] - RICH, Elaine - Inteligência Artificial.  
Editora McGraw-Hill, Ltda. - 1983.
- [33] - ROBERTS, S.A. - Application of heuristic techniques to the cutting-stock problem for worktops.  
*Journal of Operational Research Society* - Vol. 35 - 1984
- [34] - RODE, Mathias/ROSENBERG, Otto - An analysis of Heuristic Trim Loss Algorithms.  
*Engineering Cost & Production Economics* - Vol. 12 - 1987
- [35] - SANDERS, J.L./BRNSOILER, A. - Performance testing of irregular parts nesting systems for flame cutting and other industrial applications.  
*Transactions of ASME - NAMRC* - XI - 1983.
- [36] - SCHILDT, Herbert - Artificial Intelligence using C.  
Osborne McGraw-Hill - 1987.
- [37] - SCHWÄRTZEL, Heinz - Intelligence without awareness.  
*Siemens Magazine* - Com 2/89.
- [38] - SHIRKIN, Joel - The expert system: the practical face of AI.  
*Technology Review* - Nov.Dec. 1983.
- [39] - TAHA, Hamdy A. - Introduction to Operations Research.  
MacMillan Publishing CO., INC.

- [40] - TEIXEIRA JR, Aluísio/MEGALE, Amarílis/SAKAMOTO, Felício H./  
MARTINS, Flávio P.R./NETO, Vicente A.R - Sistema para  
planejamento de processos de usinagem.  
Anais do 1º Simpósio sobre Tecnologia da Usinagem -  
SOBRACON - 1989.
- [41] - TOKUYAMA, H./UENO, N. - The cutting stock problem for large  
sections in the iron and steel industries.  
*European Journal of Operational Research* - Vol 22 - 1985
- [42] - VASKO, Francis J. - A computational improvement to Wang's  
two dimensional cutting-stock algorithm.  
*Computers & Industrial Engineering* - Vol. 16 Number 1 -  
1989.
- [43] - WAGNER, Harvey M. - Pesquisa Operacional.  
Prentice/Hall do Brasil.
- [44] - WATERMAN, Donald A. - A guide to expert systems.  
Addison-Wesley - 1986.
- [45] - WEISS, Sholom M./KULINOWSKI, Casimir A. - Guia prático para  
projetar sistemas especialistas.  
Livros Técnicos e Científicos Editora S.A. - 1988.