NOVOS PARADIGMAS E APLICAÇÕES DA INTELIGÊNCIA ARTIFICIAL*

NEW PARADIGMS AND APLLICATIONS IN (ARTIFICIAL INTELLIGENCE)

Juan Manuel Adán COELLO**

RESUMO

Este artigo apresenta e discute algumas técnicas e paradigmas desenvolvidos na área de Inteligência Artificial (IA) que têm sido usados com sucesso, ou oferecem grande potencial para a construção de aplicações inovadoras. Entre os paradigmas clássicos, discute-se a solução de problemas usando algoritmos de busca heurística e os sistemas especialistas (SEs), enfatizando, neste último caso, a representação de conhecimento e raciocínio baseados em regras (RBR). Dentre os novos paradigmas, destaca-se a indução automática de regras e o raciocínio baseado em casos, oriundos da área de aprendizado de máquina (*Machine Learning*). Os algoritmos de indução permitem a extração automática de conhecimento, como alternativa ao oneroso processo de extração manual de conhecimento, típico de sistemas baseados em regras, considerado o aspecto mais complexo da construção de um SE. O raciocínio baseado em casos (RBC) é também uma alternativa para simplificar e acelerar o processo de aquisição de conhecimento, além de suportar o aprendizado automático e contínuo do sistema. Finalmente, conceitua-se e ilustra-se a construção de agentes inteligentes, um dos paradigmas mais promissores para a construção de novas aplicações, especialmente em ambientes complexos, dinâmicos e ricos em informação como a Internet.

Palavras-chave: inteligência artificial, busca heurística, sistemas especialistas, raciocínio baseado em regras, raciocínio baseado em casos, indução, agentes inteligentes.

ABSTRACT

This paper presents and discusses some techniques and paradigms from the Artificial Intelligence field that have been successfully applied, or provide great potential for innovative applications. Classical and still very useful approaches as search methods and rule based experts systems are reviewed. Among the new and promising techniques, not yet fully explored, this paper focus on the induction of rules from examples and CBR (Case-Based Reasoning). Finally, we introduce Intelligent Agents, a paradigm that should play a major role in the construction of new applications for complex, dynamic and information rich environments, like the Internet.

Key-words: Artificial Intelligence, Heuristic Search, Expert Systems, Case-Based Reasoning, Induction, Intelligent Agents.

(*) Este artigo é derivado de texto publicado nos sinais da 2ª Escola Regional de Informática, da Sociedade Brasileira de Computação, realizada na UNIMED, no período de 2 a 6 de junho de 1997.

(**) Doutor em Engenharia pela UNICAMP é professor do Instituto de Informática da PUC-Campinas.

1. INTRODUÇÃO

A Inteligência Artificial (IA) é um campo multidisciplinar que encontra os seus fundamentos em diversas áreas, incluindo a filosofia, a matemática, a psicologia, a lingüistica e, naturalmente, a ciência da computação. Da mesma forma que a filosofia e a psicologia, a IA procura entender a Inteligência, porém, em adição a isso, procura também construir entidades inteligentes.

Dada a complexidade do problema e as múltiplas áreas envolvidas, não há uma definição universalmente aceita de IA. Nem mesmo há um consenso sobre se a empreitada é ou não possível. Russel e Norvig (1995) enquadram as definições de IA expressas em onze livros-texto recentes em quatro categorias, relacionadas à construção de:

1. Sistemas que pensam como humanos;
2. Sistemas que agem como humanos;
3. Sistemas que pensam racionalmente; e
4. Sistemas que agem racionalmente.

As definições do grupo 1 e 3 estão relacionadas aos processos de pensamento e raciocínio, enquanto que as definições dos grupos 2 e 4 ao comportamento dos sistemas. Por outro lado, as definições dos grupos 1 e 2 medem o sucesso dos sistemas em termos do desempenho humano, enquanto que as definições dos grupos 3 e 4 empregam um conceito idealizado de inteligência, denominado racionalidade (um sistema é racional se "faz a coisa certa").

Embora não tenha sido possível chegar a um paradigma unificador que explique a inteligência e permita construir entidades inteligentes, as pesquisas conduzidas até o momento produziram uma série de técnicas e paradigmas que permitem resolver inúmeros problemas complexos e relevantes. Este artigo tem por objetivo apresentar e discutir algumas dessas técnicas e paradigmas representativos dos resultados obtidos na área.

O artigo obedece à seguinte estrutura. Na seção 2 discute-se a solução de problemas usando algoritmos de busca. Na seção 3 faz-se uma introdução aos sistemas especialistas. Na seção 4 discute-se o raciocínio baseado em regras (RBR), o paradigma mais empregado para representar conhecimento e fazer inferências em sistemas especialistas. Na seção 5 discute-se sucintamente a extração automática de conhecimento através de algoritmos de indução. A seção 6 trata do raciocínio baseado em casos (RBC). A seção 7 conceitua e ilustra a construção de agentes inteligentes. Finalmente, na seção 8 são feitas algumas considerações finais.

2 . SOLUÇÃO DE PROBLEMAS USANDO ALGORITMOS DE BUSCA

O principal objetivo dos primeiros pesquisadores em IA era encontrar mecanismos para resolver problemas cuja solução era difícil empregando as técnicas computacionais convencionais, embora fossem facilmente resolvidos pelo cérebro humano, um dispositivo em muitos sentidos mais limitado que o computador. Para esses problemas não havia soluções algorítmicas, ou estas eram tão complexas que não era praticável implementá-las no computador.

A solução foi desenvolver novas técnicas de solução de problemas, semelhantes àquelas usadas por humanos, e implementá-las num computador. Uma das mais importantes dessas técnicas é a busca.

2.1 Espaço do Problema x Espaço da Solução

As técnicas de busca em IA procuram em um espaço do problema, e não numa estrutura de dados pré-definida, como na computação convencional. O objetivo é encontrar um caminho que ligue a descrição inicial do problema à descrição do estado desejado do problema (a sua solução). Esse caminho, quando encontrado, irá representar os passos dados para resolver o problema. O processo de busca de uma solução desenvolve o espaço da solução, i.e., a porção do espaço do problema que foi examinada.

Ao contrário de uma estrutura de dados, que é predefinida e já existe quando a busca começa, espaços de problemas são geralmente definidos procedimentalmente. Ou seja, o espaço do problema não é totalmente criado e então feita a busca, ele é criado à medida que é explorado. A partir de um dado estado, são usados procedimentos (ou operadores), guiados por uma estratégia de controle para definir os próximos estados passíveis de busca.

Os espaços de solução de problemas são normalmente representados por grafos, ou árvores, onde os nós representam os estados da solução e os arcos

as operações que levaram o problema de um estado a outro.

A eficácia de uma estratégia de busca pode ser avaliada pelos seguintes critérios:

1. **Completeza**: Garante-se que a estratégia encontra uma solução, quando houver uma?
2. **Complexidade Temporal**: Quanto tempo é necessário para encontrar uma solução?
3. **Complexidade Espacial**: Quanta memória é necessária para fazer uma busca?
4. **Otimalidade**: A estratégia encontra a melhor solução quando houver várias?

Podemos avaliar a complexidade de uma estratégia de busca considerando o seu fator de ramificação r, que indica o número de novos estados que podem ser gerados a partir de cada estado e a profundidade p a que se encontra a solução, na árvore de busca.

2.2 Busca Cega

A busca de uma solução no espaço de estados (espaço do problema) pode ser feito usando procedimentos que definem o próximo estado a pesquisar, sem utilizar qualquer conhecimento que indique quão próximo se estará da solução percorrendo esse caminho. A este tipo de busca chama-se busca cega. Dois métodos representativos deste tipo de busca são a busca em largura *(Breadth-first Search)* e a busca em profundidade *(Depth-first Search)*.

Na busca em largura, o nó raiz (estado inicial) é expandido, em seguida são expandidos todos os nós gerados pelo nó raiz, em seguida os seus sucessores, e assim por diante.

Na busca em profundidade, sempre é expandido um dos nós no nível mais profundo da árvore até chegar a um estado solução ou a um beco sem saída (um nó que não representa um estado objetivo e que não pode ser expandido). Ao chegar a um beco sem saída, volta-se ao nível imediatamente anterior onde haja nós ainda não expandidos, e continua-se a busca desse ponto.

Comparando as duas estratégias, podemos observar que a busca em largura, ao contrário da busca em profundidade, não perde tempo explorando "becos sem saída", ou seja caminhos que não levam à solução. Além disso, se houver uma solução para o problema, ela será encontrada. Adicionalmente, se houver várias soluções, a mínima (que requer o menor número de passos) será encontrada. Isto é, a estratégia é ótima e completa. O número máximo de nós a expandir antes de encontrar a solução é:

$$1 + r + r^2 + r^3 + \ldots + r^p$$

portanto, a sua complexidade espacial e temporal é $O(r^p)$. Deve-se observar que todos os nós folhas (na extremidade da árvore) devem ser mantidos na memória. De modo geral, o uso desta estratégia esbarra mais freqüentemente em limitações de memória que de tempo, particularmente para valores pequenos de p. Considere, a título de exemplo, os valores apresentados pela tabela 1, considerando que r=10, em um sistema com capacidade de expandir 1000 nós por segundo, requerendo 100 bytes para armazenar cada nó, valores característicos de muitos problemas do tipo quebra-cabeça ou jogos, executados em microcomputadores modernos.

No busca em profundidade, apenas os nós do caminho corrente devem ser armazenados, isto é ,

Tabela 1 - Custo da busca em largura.

Profundidade (p)	Número de Nós	Tempo requerido	Memória
0	1	1 mili segundo	100 bytes
2	111	0,1 segundo	11 kbytes
4	11.111	11 segundos	1 megabytes
8	10^8	31 horas	11 gigabytes
12	10^{12}	35 anos	111 terabytes
14	10^{14}	3500 anos	11.111 terabytes

apenas os nós ligando o nó raiz a um nó folha, juntamente com os nós irmãos não expandidos de cada nó do caminho são mantidos na memória. Se *p* indicar a profundidade máxima da árvore de busca a complexidade temporal da busca em profundidade será $O(r^p)$ e a complexidade espacial O(r x m). Para *r* e *p* iguais a 10 e a 12, respectivamente, seriam necessários 12 kbytes de memória e não os 111 terabytes necessários para a busca em largura com valores iguais para esses parâmetros. Portanto, um valor 10 bilhões de vezes menor.1 mili segundo

A busca em profundidade não é completa nem ótima, ela pode cair num "poço sem fundo", que não conduz a uma solução, e nada garante que uma solução encontrada seja a melhor (a mais curta). Apesar disso, em muitos casos pode encontrar uma solução sem ter que examinar uma grande porção da árvore de busca. Na busca em largura, todos os nós da árvore até um dado nível n devem ser examinados antes que qualquer nó do nível n+1 possa ser examinado. Este aspecto é particularmente relevante se houver diversas soluções para o problema e não for fundamental encontrar a melhor. A busca em profundidade pode parar quando qualquer das soluções for encontrada.

2.3 Busca Dirigida

Apesar de muito úteis, os métodos de busca cegos, mostram-se inadequados para a solução de inúmeros problemas complexos. Considere, por exemplo, o custo de explorar o espaço de estados associado a um jogo de xadrez, onde o fator médio de ramificação é de cerca de 35, e um jogo envolve cerca de 50 movimentos por parte de cada jogador, de modo que uma árvore de busca teria aproximadamente 35^{100} nós (embora haja somente cerca de 10^{40} posições diferentes válidas). Para este tipo de problemas, podem ser usados procedimentos que sejam capazes de estimar se ao seguir um determinado caminho aproximamo-nos ou afastamo-nos da solução. Este tipo de busca, denominada busca dirigida, heurística, ou baseada em conhecimento, é um dos alicerces da IA clássica.

Entre as inúmeras técnicas de busca dirigida discutiremos as estratégias "subindo a colina" (*Hill-climbing*), *Branch and Bound,* melhor caminho primeiro (*Best-first*) e *A**.

A estratégia *Hill-climbing* explora o espaço de soluções em profundidade, seguindo o caminho que aparenta diminuir mais a distância que falta para atingir o objetivo. Por exemplo, suponha que estamos em Paris tentando chegar à torre Eiffel. Usando esta técnica, cada vez que chegássemos a uma esquina, olharíamos para a torre (supondo que pudéssemos vê-la de qualquer ponto) e seguiríamos pela rua que nos parecesse levar mais diretamente à torre. Um dos problema com este procedimento é que poderíamos chegar a um beco sem saída, e a aplicação direta do método não nos permitiria sair dele, pois, para fazer isso teríamos que aumentar a nossa distância do objetivo.

Enquanto na técnica *hill-climbing*, sempre se segue um caminho a partir do nó mais recentemente examinado, na busca *branch and bound,* segue-se um caminho a partir do nó mais promissor na árvore de soluções, independente da sua localização. Um dos problemas com esta técnica de busca é que uma porção significativa de toda a árvore deve ser desenvolvida até encontrar o menor caminho para a solução.

Para fazer a busca empregando a técnica do menor caminho primeiro (*Best-first*), deve-se dispor de uma função heurística *h(n)*, capaz de estimar o custo do caminho mais barato que leva do estado *n* a um estado objetivo, de modo que a busca sempre prossiga pelo nó não expandido, independente da sua localização, com o menor custo estimado para levar a uma solução. Esta técnica assemelha-se à *hill-climbing*, com a diferença que na *hill-climbing* a busca prossegue sempre a partir da posição (nó) corrente.

Em problemas de roteamento *h(n)* costuma ser a distancia em linha reta até o objetivo. No caso da nossa busca pela Torre Eiffel, quando a técnica *Best-first* é usada, cada interseção de duas ou mais ruas deve ser marcada com uma estimativa da distância que a separa da torre, e sempre se avança pela próximo caminho disponível, a partir da interseção que se estima estar mais perto da torre.

Esta técnica é uma variante da busca em profundidade, não sendo, portanto completa nem ótima. Se *p* for a profundidade da árvore de busca a sua complexidade temporal é $O(b^p)$ e a sua complexidade temporal também é $O(b^p)$, pois todos os nós gerados devem ser mantidos na memória. Dependendo do

problema sendo resolvido, uma boa função heurística pode reduzir substancialmente ambas complexidades.

As técnicas *hill-climbing* e *best-first* não garantem que as soluções encontradas sejam ótimas, isto é, representem o menor caminho. No entanto, o custo para encontrar uma solução com elas é geralmente inferior ao custo para encontrar uma solução (ótima) com a *branch-and-bound*. Um problema com a busca *best-first*, é que ela, ao contrário da *branch and bound,* não leva em conta o custo efetivamente realizado até a chegada a um dado estado (no caso de um problema de busca de rotas, qual o custo do caminho já percorrido)

O algoritmo A* de busca combina a *best-first*, com a *branch and bound*. Ela supõe que temos conhecimento suficiente para estimar o custo para ir de qualquer nó até o objetivo. O algoritmo A* estima o custo total da busca, somando o custo efetivo para chegar ao nó corrente a partir do nó inicial, ao custo estimado para atingir o objetivo a partir desse nó.

Em geral é muito difícil obter estimativas precisas. Para que seja garantido que o A* encontre o caminho ótimo, é necessário que todas as estimativas sejam maiores que zero, porém nunca superiores ao custo real.

2.4 Aplicações de estratégias de busca

Variantes dos algoritmos de busca encontram ampla aplicação na solução de problemas complexos, entre eles, Russel e Norvig (1995), destacam os seguintes:

- **Determinação de rotas:** Ex.: Roteamento de pacotes em redes de computadores e planejamento de viagens aéreas.
- **Planejamento de viagens:** Ex. Visitar cada cidade de um dado mapa, pelo menos uma vez, começando e terminando numa mesma cidade. O problema do caixeiro viajante é uma variante muito conhecida deste problema no meio acadêmico, e bastante complexa. Nesta variante do problema, cada cidade deve ser visitada exatamente uma vez, devendo-se encontrar o caminho mais curto (ou mais barato).
- ***Layout* de células e roteamento de canais**, em circuitos VLSI (que podem ter da ordem de um milhão de portas lógicas). O objetivo é minimizar a área e o comprimento das conexões, maximizando, em conseqüência, a velocidade do circuito.
- **Navegação de Robôs:** Generalização do problema de roteamento. Aqui, em vez de mover-se por um conjunto discreto de rotas, o robô pode mover-se por um espaço contínuo.
- **Seqüenciamento de montagem**: objetiva encontrar uma ordem em que as partes de algum objeto devem ser montadas. O problema é complexo e relevante, devendo ser observado que durante uma montagem se a ordem errada for escolhida, não haverá como adicionar alguma peça posteriormente, sem desfazer parte do trabalho já feito.

3. SISTEMAS ESPECIALISTAS/PERITOS

No início dos anos 70, pesquisadores em IA perceberam as limitações dos métodos gerais de solução de problemas e das técnicas de busca desenvolvidas na década anterior. Eles notaram que para resolver determinados problemas complexos era requerido conhecimento específico sobre o domínio de aplicação de interesse, em lugar de conhecimento amplo e geral que se aplica a inúmeros domínios. Três conceitos fundamentais distinguem os sistemas especialistas, ou baseados em conhecimento, de programas algorítmicos convencionais e de programas genéricos de solução de problemas baseados em busca:

1. A separação do conhecimento de como ele é usado;
2. O uso de conhecimento extremamente específico de um determinado domínio;
3. A natureza heurística, em vez de algorítmica, do conhecimento empregado.

O uso de heurísticas em sistemas baseados em conhecimento é sensivelmente diferente das funções heurísticas usadas em algoritmos de busca. As funções heurísticas representam conhecimento genérico usado para guiar a busca. A heurística empregada em sistemas baseados em conhecimento, refere-se ao conhecimento heurístico usado por um especialista na solução de um problema específico.

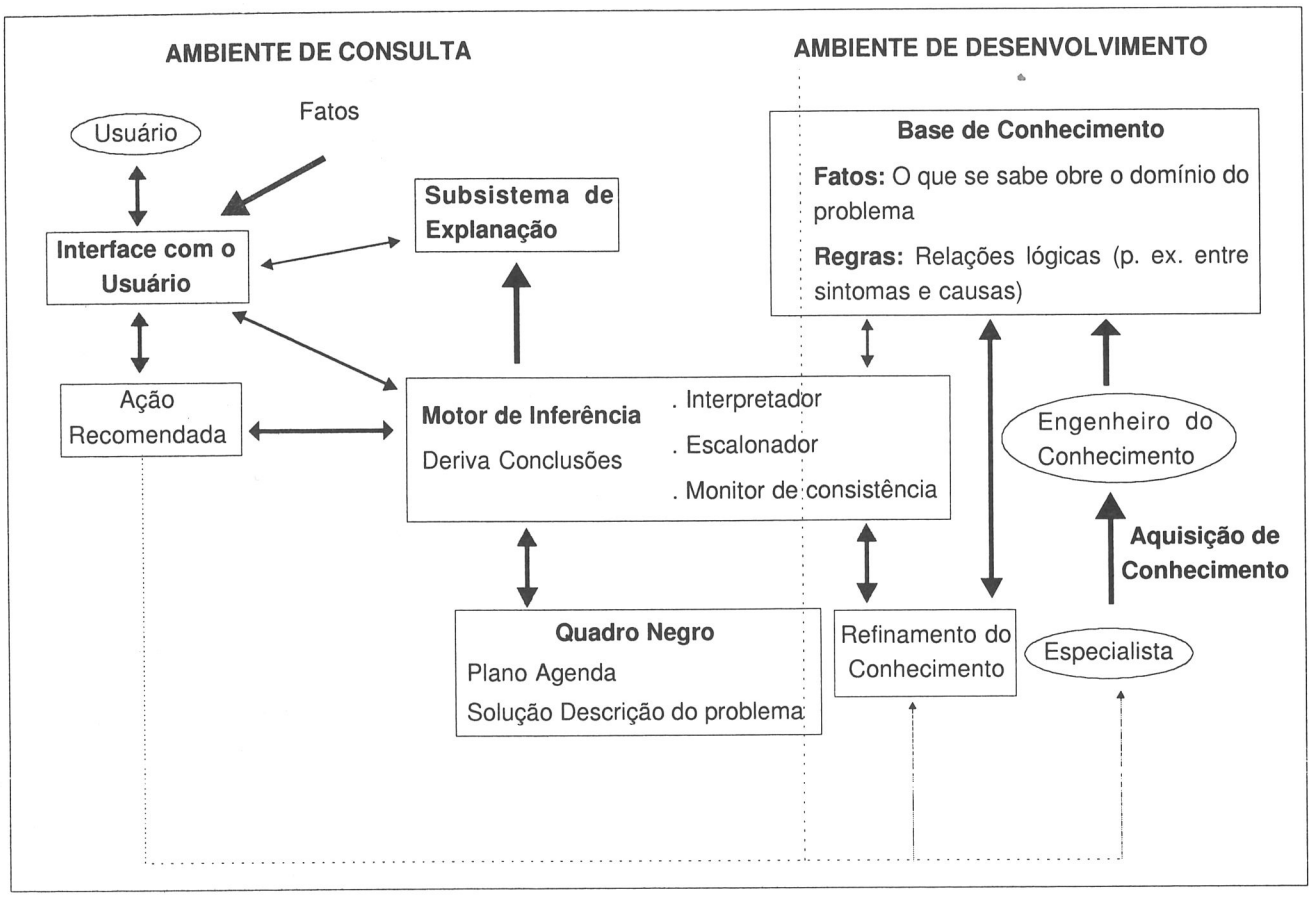

Figura 1 - Estrutura de um sistema especialista (Fonte: Turban, 1995).

3.1 Estrutura dos Sistemas Especialistas

A estrutura de um sistema especialista é representado na figura 1. A Base de Conhecimento (BC) contém o conhecimento necessário ao entendimento, formulação e solução de problemas. A base de conhecimento é incorporada a um programa de computador através de técnicas para a representação de conhecimento. A aquisição de conhecimento, a partir de especialistas e de outras fontes, como livros, relatórios e figuras, é uma tarefa complexa que freqüentemente cria o principal "gargalo" na construção de um SE. Geralmente é necessário um Engenheiro de Conhecimento (ECO) para interagir com um ou mais peritos humanos a fim de construir uma base de conhecimento.

O motor de inferência é o "cérebro" de um SE. Ele implementa as técnicas de raciocínio a partir do conhecimento e fatos presentes na base de conhecimento e no "quadro negro". Quando uma consulta é feita, ele desenvolve uma agenda que organiza e controla os passos necessários para resolver um problema.

O quadro-negro *(Blackboard)* é uma área de trabalho empregada em alguns sistemas para manter a descrição do problema a ser resolvido, de acordo com uma especificação presente em dados de entrada, e para o armazenamento de resultados intermediários. No quadro-negro pode-se registrar hipóteses e decisões intermediários, tais como: 1) **plano** para abordar o problema, 2)**agenda** de ações aguardando execução e 3) **solução:** hipóteses e cursos alternativos de ação gerados pelo sistema até um dado momento. O uso do quadro-negro é particularmente popular para aplicações em que vários especialistas devem trabalhar em conjunto para resolver um problema.

O Subsistema de Explanação é responsável por justificar as conclusões tiradas pelo SE. Ele permite explicar ao usuário porque determinada questão foi formulada pelo SE, como uma conclusão foi obtida, por que determinada alternativa foi rejeitada e qual é o plano a ser seguido para alcançar a solução, indicando, por exemplo, o que falta ser estabelecido antes que um diagnóstico final seja determinado.

Especialistas humanos têm um sistema de refinamento de conhecimento que lhes permite analisar o seu desempenho, aprender a partir dele, e refiná-lo para consultas futuras. De modo semelhante, este componente é necessário para que um SE possa ser capaz de analisar as razões para suas falhas e sucessos, i.e. para aprender. Isto lhe permite melhorar a sua BC e a eficácia das suas inferências. Estes subsistemas são ainda, de modo geral, objeto de pesquisa em universidades e centros de pesquisa.

3.2 Aquisição de Conhecimento

É a tarefa mais crítica e que mais consome tempo durante o processo de desenvolvimento incremental de um sistema baseado em conhecimento, é conhecida com o "gargalo dos sistemas baseados em conhecimento". A aquisição de conhecimento compreende em duas atividades principais: 1) elicitação, ou extração propriamente dita, do conhecimento; e 2) a representação do conhecimento (usando regras ou outro formalismo adequado).

O principal mecanismo empregado na extração do conhecimento é a discussão face a face entre o perito e o engenheiro de conhecimento que lhe faz perguntas e o observa resolvendo problemas. Este costuma ser um processo penoso e demorado, por diversos motivos, alguns associados ao especialista, como a sua incapacidade de verbalizar, ou mesmo reconhecer, o processo de solução de problemas que emprega, ou o emprego de mecanismos de defesa cognitiva, usualmente subconscientes, e alguns associados ao ECO, como a sua falta de conhecimento do domínio do problema.

Para fazer frente às dificuldades associadas aos métodos tradicionais de aquisição de conhecimento, baseadas em entrevistas, foram desenvolvidos métodos alternativos, como a análise da grade de repertório (*Repertory Grid Analysis - RGA)* e o raciocínio indutivo, que podem ser usados em lugar das técnicas, ou em combinação com elas.

A RGA foi proposta por Kelly (1955) para a implementação da sua Teoria da Construção Pessoal, destinada a aumentar a eficácia de sessões de psicologia clínica. O RGA foi adaptado para auxiliar na extração de conhecimento e automatizado por diversos sistemas. Sistemas que implementam o RGA, interrogam o usuário (o especialista ou o engenheiro do conhecimento) a respeito do domínio, visando ajudá-lo na estruturação do conhecimento.

3.3 Benefícios, Problemas e Limitações dos SE

Da mesma forma que outras aplicações computadorizadas, os sistemas especialistas costumam estar associados a aumento de produtividade, na medida em que são mais rápidos que seres humanos, e a um padrão de qualidade uniforme, pois fornecem sugestões consistentes e baixa taxa de erros.

Um benefício normalmente destacado dos SEs é a captura de perícia escassa, evidente em diversas situações, por exemplo:

- há um número insuficiente de peritos
- um perito está a ponto de aposentar-se ou de mudar de emprego
- um perito é necessário numa grande área geográfica
- é necessária a transferência de conhecimento para localidades remotas, ou de difícil acesso, por exemplo, conhecimento médico, para áreas distantes assoladas por algum tipo de epidemia

Entre as promessas futuras dos SEs está a capacidade de resolver problemas cuja complexidade excede as habilidades humanas. Isto ocorre, por exemplo, quando o escopo do conhecimento necessário é muito maior que aquele detido por qualquer indivíduo isoladamente.

Se os sistemas especialistas oferecem diversas vantagens, o seu uso esbarra também com algumas dificuldades e limitações. Por um lado, as metodologias e ferramentas para a construção de SE não são tão simples e efetivas como deveriam ser, mesmo para as classes de aplicações apresentadas anteriormente. Alem disso, uma série de dificuldades têm reduzido o ritmo de difusão comercial de SE, entre elas, podemos destacar:

- a perícia/conhecimento é difícil de extrair de seres humanos
- diferentes peritos podem ter diferentes visões (corretas) de uma mesma situação
- usuários de SE têm limitações cognitivas naturais

- SE funcionam bem apenas em domínios estreitos (problemas específicos)
- a maioria dos especialistas não têm meios independentes de verificar se as suas conclusões são razoáveis
- o vocabulário que os peritos usam para expressar fatos e relações é, em geral, de difícil compreensão para leigos
- em geral é necessária a ajuda de engenheiros de conhecimento, raros e caros
- falta de confiança dos usuários finais
- SE podem não ser capazes de chegar a uma conclusão
- SE, assim como especialistas humanos, podem fazer recomendações erradas

3.4 Áreas de Aplicação de SEs

Os SEs foram aplicados com sucesso a diversas classes de aplicações entre as quais Turbam (1995) destaca as seguintes:

- **Interpretação:** Os SEs inferem descrições de situações a partir de observações. Este tipo de aplicação inclui sistemas para entendimento de fala, análise de imagens e interpretação de sinais.
- **Previsão**: Os SEs inferem conseqüências prováveis a partir de uma dada situação. Este tipo de aplicação inclui sistemas para previsão do tempo, previsões demográficas, previsões econômicas e estimativas de colheitas e de mercado.
- **Diagnóstico:** Os SEs inferem disfunções de um sistema a partir de observações. Tipicamente relacionam irregularidades de comportamento às suas causas. Este tipo de aplicação inclui sistemas de diagnóstico médico, eletrônico, mecânico e de *software*.
- **Projeto:** Os SEs desenvolvem configurações de objetos que satisfazem as restrições de um projeto. Este tipo de aplicação inclui projeto de *layout* de circuitos, projeto de construções e plantas.
- **Planejamento:** Os SEs desenvolvem planos para atingir a objetivos dados. Este tipo de aplicação inclui programação automática, planejamento de projetos, roteamento, comunicações, desenvolvimento de produtos, aplicações militares e planejamento financeiro.
- **Monitoração:** Os SEs comparam observações com planos, sinalizando exceções. Este tipo de aplicação inclui a monitoração de processos industriais, de tráfego aéreo e a monitoração fiscal.
- **Depuração:**Os SEs, a partir de capacidades de planejamento, projeto e previsão criam especificações ou recomendações para corrigir um problema diagnosticado.
- **Reparação:** Os SEs desenvolvem e executam planos para reparar um problema diagnosticado. Estes sistemas incorporam capacidades de depuração, planejamento e execução.
- **Instrução:**Os SEs diagnosticam, depuram e corrigem o desempenho de estudantes. Estes sistemas, em geral, de início, constróem uma descrição hipotética do conhecimento do estudante. A seguir, eles diagnosticam deficiências e identificam as maneiras de superar estas deficiências. Finalmente, eles planejam uma interação que permita ministrar o conhecimento necessário ao estudante.
- **Controle:** Os SEs interpretam a situação corrente de um sistema, prevêem o futuro, diagnosticam causas de problemas antecipados, formulam planos de solução e monitoram a sua execução.

4. RACIOCÍNIO BASEADO EM REGRAS (RBR)

Conforme discutido anteriormente, num sistema especialista o conhecimento é armazenado numa base de conhecimento, a partir da qual, um motor de inferência resolve problemas específicos (raciocina) a partir de fatos fornecidos pelo usuário, ou por qualquer outro meio. O conhecimento na maioria dos SEs é representado usando regras de produção da forma:

SE	antecedente condição premissa	ENTÃO	conseqüente (1) ação (2) conclusão (3)

Por exemplo:

(1) SE X é um gato ENTÃO X é um animal

(2) SE você estiver dirigindo e uma ambulância com a sirene ligada se aproximar ENTÃO reduza a velocidade, coloque-se à direita e deixe a ambulância passar

(3) SE a sua temperatura for superior a 37°C ENTÃO você está febril

As regras são particularmente importantes porque quando peritos são questionados sobre o processo que empregam para solucionar problemas, normalmente respondem na forma de regras. De modo geral, as pessoas acham natural expressar conhecimento usando regras, especialmente para conhecimento acumulado ao longo do tempo e que resultou em associações empíricas internalizadas (conhecimento heurístico). Por outro lado, sistemas baseados em regras são relativamente fáceis de implementar e estender uma vez implementados.

Em sistemas baseados em regras, a progressão de um conjunto de dados, ou fatos, para uma resposta, ou conclusão, pode dar-se dar partindo dos dados e progredindo até uma conclusão, num processo de solução guiado pelos dados (*data driven*), também denominado encadeamento para a frente (*forward chaining*), ou selecionando uma possível conclusão e tentando provar a sua validade através da busca de evidências que a suportem, num processo de solução dirigido pelo objetivo (*goal driven*), também denominado de encadeamento para trás (*backward chaining*).

As duas estratégias costumam ser empregadas e alguns sistemas, inclusive, permitem combiná-las para resolver um mesmo problema. De modo geral, no entanto, o encadeamento para a frente é indicado para problemas envolvendo síntese (projeto, configuração, planejamento e escalonamento). Nesse tipo de problema os dados dirigem a solução, pois costumam ser requeridos poucos dados para resolver o problema, havendo, no entanto muitas soluções diferentes, porém igualmente aceitáveis. O encadeamento para trás, por seu lado, é indicado para problemas de diagnóstico, onde há muitos dados disponíveis, porém poucos relevantes. Por ex., no diagnóstico médico, utiliza-se os sintomas do paciente, em vez de todos os seus dados que poderiam implicar em boa ou má saúde. Nesse tipo de sistemas, normalmente há poucas soluções (conclusões) válidas.

A codificação de conhecimento usando regras apresenta vários aspectos positivos. Regras são fáceis de entender e comunicar porque são uma forma natural de expressar conhecimento. Regras permitem também construir sistemas modulares de manutenção relativamente simples. Regras são usualmente independentes entre si, e a BC pode ser construída e testada por partes. A geração de explanações é simples, consistindo, fundamentalmente na apresentação das regras usadas para derivar uma conclusão, ou na apresentação de regras sendo avaliadas. Finalmente, mecanismos para expressar e processar incerteza são fáceis de combinar com regras [Zadeh, 1979].

Naturalmente, os sistemas baseados em regras também apresentam alguns inconvenientes. Sistemas onde conhecimento complexo deve ser expresso podem requerer um grande número de regras (milhares, às vezes), o que pode acarretar sérias dificuldades para a construção e manutenção da BC. Por outro lado, alguns tipos de conhecimento, por exemplo que envolvem representação de objetos estruturados, são muito difíceis de codificar usando regras, porém os construtores de sistemas especialistas costumam gostar de regras, levando-os a procurar representar todo o conhecimento do domínio sob essa forma, em vez de buscar representações eventualmente mais adequadas. Não se deve esquecer também o ditado popular que diz que toda regra tem uma exceção, e a expressão de exceções usando regras não é trivial. Também não é trivial examinar uma BC já construída e determinar com precisão quando cada possível ação vai ocorrer (opacidade).

5. INDUÇÃO

A indução automática de regras ou árvores de decisão, advinda da área de aprendizado de máquina (*machine learning*) é uma técnica de aprendizado a partir de exemplos. A indução é um processo de raciocínio que parte do específico para o geral.

Um sistema de indução de regras recebe exemplos de um problema cuja solução é conhecida, ou conjunto de treinamento, e induz as regras que poderão ser usadas para resolver outros problemas cuja solução não é conhecida.

Tabela 2 - Classificação de aviões, a partir de seus atributos.

PosAsas	Turbinas	Nariz	EntAr	Fuselagem	Classificação
meio	1	achatado	nariz	charuto	russo
meio	1	achatado	nariz	arredondada	russo
baixo	1	arrebitado	nariz	arredondada	russo
alto	2	pontudo	corpo	angulosa	EUA
Alto	1	pontudo	corpo	angulosa	EUA

O coração de um sistema de indução de regras é um algoritmo de indução. Este, a partir de uma matriz (grade) de conhecimento de atributos, valores e seleções, gera as regras correspondentes. Diversos algoritmos com esse propósito existem, sendo populares o ID3 [Quinlan, 1986] e sua versão mais recente o C4.5 [Quinlan, 1993].

O princípio do ID3 é a geração de uma árvore de decisão a partir de uma matriz de conhecimento. A aparência final da árvore de decisão depende da ordem em que os atributos forem escolhidos para fins de seleção. Essa aparência irá afetar as regras resultantes. O ID3 é capaz de derivar uma árvore de seleção mínima que, por sua vez, irá minimizar a complexidade das regras, pois o número de testes necessários para fazer uma classificação será o menor possível.

Por exemplo, considere que dadas imagens de aviões deseje-se identificar se eles são de fabricação americana ou russa. Supondo que um sistema de pré-processamento de imagens é capaz de extrair as informações mostradas na tabela 2, um sistema indutivo que não otimiza-se a árvore de decisão poderia gerar, facilmente, as regras mostradas na figura 2. No entanto, um sistema usando o ID3 com otimização iria gerar a base de regras, em formato de árvore de decisão, mostrada na figura 3

A indução de regras ou árvores de decisão apresenta uma série de aspectos positivos para a construção de sistemas baseados em conhecimento. Diversos pesquisadores da área acreditam que as técnicas manuais de aquisição de conhecimento são apropriadas apenas para domínios onde o conhecimento é bem determinado, pouco volumoso, fracamente acoplado, ou modular. Em domínios onde o conhecimento requerido é vasto e complexo, os especialistas passam a ter dificuldades para explicar como procedem. Entretanto, eles são capazes de oferecer ao ECO exemplos de problemas significativos e a sua solução.

R1: SE PosAsas = meio E Turbinas = 1 E nariz = chato E EntAr = nariz E Fuselagem = charuto
ENTÃO Classificação = russo.

R2: SE PosAsas = meio E Turbinas = 1 E nariz = chato E EntAr = nariz E Fuselagem = arredondada
ENTÃO Classificação = russo.

R3: SE PosAsas = baixo E Turbinas = 1 E nariz = arrebitado E EntAr = nariz E Fuselagem = arredondada
ENTÃO Classificação = russo.

R4: SE PosAsas = alto E Turbinas = 1 E nariz = pontudo E EntAr = corpo E Fuselagem = angulosa
ENTÃO Classificação = EUA.

R5: SE PosAsas = alto E Turbinas = 1 E nariz = pontudo E EntAr = corpo E Fuselagem = angulosa
ENTÃO Classificação = EUA.

Figura 2 - Base de regras, não otimizada, induzida a partir da tabela 3.

Em sistemas pequenos, o uso de algoritmos de indução permite que o construtor da BC não precisa ser necessariamente um ECO. Em sistemas médios e grandes (de maior interesse comercial) a criação das tabelas de indução pode ser bastante complexa, sendo necessária a participação de um ECO. As regras geradas por um sistema de indução devem sempre ser revisadas pelo especialista, o que com freqüência contribui para que ele aumente a compreensão do processo de raciocínio que emprega para resolver problemas e para que ele estruture melhor o seu conhecimento.

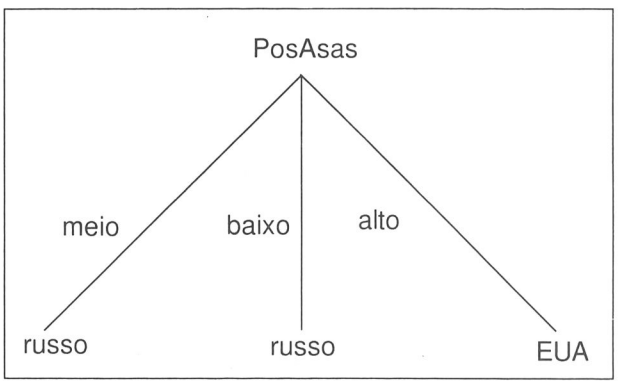

Figura 3 - Árvore de decisão, otimizada, gerada usando o algoritmo ID3.

A Indução automática oferece a possibilidade de deduzir conhecimento novo, pois pode-se listar os fatores que influenciam uma decisão, sem entender os seus impactos, e induzir regras que funcionem adequadamente e possam ser usadas em situações novas.

Apesar de seu grande potencial, o uso de sistemas de indução de regras depara-se com algumas dificuldades e requer certos cuidados. Alguns sistemas, por exemplo, podem gerar regras que não são facilmente entendidas por humanos, pois o modo empregado pelos programas para classificar os atributos e propriedades do problema podem não estar de acordo com o modo empregado por humanos. Por outro lado, sistemas de indução de regras não selecionam os atributos. Um especialista deve especificar quais são os atributos significativos, para um dado domínio. O número destes atributos deve, sempre que possível, ser pequeno, pois em caso contrário o custo computacional de executar o algoritmo tende a ser muito elevado, podendo mesmo requerer o uso de computadores de grande desempenho. Finalmente, uma questão importante é saber quantos exemplos são necessários para fazer a indução. De modo geral, este número é elevado, sendo, no entanto, de difícil dimensionamento.

Os algoritmos de indução têm sido usados com sucesso para a extração de conhecimento em inúmeras aplicações de interesse acadêmico e comercial, incluindo sistemas de apoio à decisão para concessão de crédito, diagnóstico de falhas em dispositivos mecânicos, classificação automática de objetos celestes, aumento da separação de gás e petróleo durante a sua extração [Langley e Simon, 1995].

6. RACIOCÍNIO BASEADO EM CASOS - RBC (*"CASE-BASED REASONING" - CBR)*

Kolodner (1993) define RBC como sendo um modelo de raciocínio que envolve o entendimento de situações, a solução de problemas e o aprendizado, integrando esses aspectos com processos de memória.

O RBC pode ser visto como um modelo cognitivo e uma metodologia para a construção de Sistemas Especialistas. Há muitas evidências de que as pessoas usam RBC como parte do seu processo de raciocínio. Psicólogos verificaram que as pessoas sentem-se confortáveis usando casos para tomar decisões, entretanto, elas têm dificuldades para lembrar-se dos melhores casos.

O entendimento do processo empregados por seres humanos para raciocinar a partir de casos potencializa novas e interessantes aplicações, como sistemas de apoio à decisão que ajudem as pessoas a melhor recuperar casos (aumentem a sua memória) e estratégias e ferramentas para ensinar a partir de bons exemplos.

A qualidade das decisões e análises feitas por um raciocinador baseado em casos depende de cinco aspectos:

1. As experiências que teve;
2. A sua habilidade para entender novas situações em termos de experiências conhecidas. Isto pressupõe relembrar experiências prévias e ser capaz de interpretar a nova situação em termos das experiências relembradas;
3. A sua capacidade de adaptar soluções;

4. A sua capacidade de avaliar e reparar soluções, baseado em resultados de casos similares, realimentação ou simulações;

5. A sua habilidade para integrar apropriadamente novas experiências à sua memória.

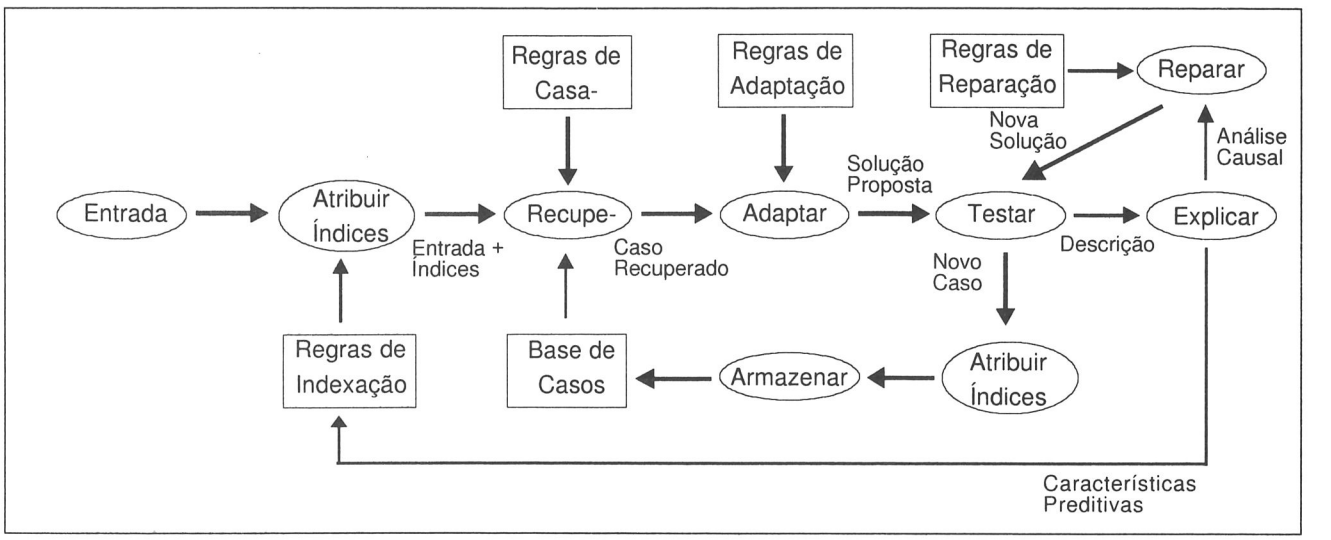

Figura 4 - O processo de Raciocínio Baseado em Casos

A figura 4, ilustra o processo de raciocínio baseado em casos segundo Riesbeck e Schank (1989).

Um caso pode ser definido como uma peça de conhecimento contextualizado, representando uma experiência que ensina uma lição fundamental para alcançar os objetivos do raciocinador [Kolodner,1993]. Um caso representa conhecimento específico ligado a um contexto (índices). Ele registra conhecimento ao nível operacional. Casos podem ter muitas formas e tamanhos, cobrindo intervalos de tempo grandes e pequenos, associando soluções com problemas, resultados com situações ou ambos. Casos registram experiências que são diferentes do esperado. Entretanto, nem todas as diferenças são importantes o bastante para registrar. Casos que vale a pena registrar são aqueles que ensinam uma lição útil, ou seja, lições que têm o potencial de ajudar o raciocinador a atingir um objetivo ou conjunto de objetivos, advertir sobre a possibilidade de falha, ou indicar um problema não percebido.

Em termos de sua aplicação, podemos distinguir duas grandes categorias, ou estilos, de sistemas usando RBC. De um lado temos os sistemas voltados à solução de problemas, onde se tende a enfatizar o uso de casos para propor soluções, de outro lado, temos os sistemas voltados à interpretação de situações, onde se tende a usar casos para gerar críticas e justificativas. Em termos da automação do processo descrito acima, há uma grande gama de sistemas que se pode construir, indo dos totalmente automáticos, aos que se restringem à recuperação de casos.

Os sistemas totalmente automáticos, resolvem problemas sozinhos e têm algum meio de interagir com o mundo para receber realimentação relativa às decisões tomadas, já os sistemas apenas de recuperação interagem com uma pessoa para resolver problemas. Este tipo de sistema é o que mais tem sido implementado. Ele tem como principal objetivo aumentar a memória da pessoa, fornecendo-lhe casos que ela pode não conhecer, ou não lembrar, a fim de que ela possa tomar as decisões (raciocinar) a partir dos casos.

O RBC oferece uma série de vantagens relativas a outros métodos. Entre elas podemos destacar:

- O RBC permite propor soluções rapidamente, evitando o tempo necessário para derivá-las a partir do início, como quando se usam os algoritmos de busca ou o raciocínio baseado em regras (RBR).;
- A criação e manutenção de bases de conhecimento baseadas em casos costuma ser muito mais fácil e rápida;
- O RBC permite propor soluções em domínios que não são completamente entendidos

com base no que funcionou no passado (p. ex. economia, onde muito depende da imprevisibilidade do comportamento humano). Nesse tipo de problemas é muito difícil formalizar o conhecimento na forma de regras, por exemplo;

- O RBC fornece elementos para avaliar soluções quando não há métodos algorítmicos disponíveis;
- Casos são úteis na interpretação de conceitos/situações em aberto e mal definidos (muito usado, por exemplo, por advogados).
- Relembrar experiências é particularmente útil para trazer à tona problemas que ocorreram no passado, alertando para a necessidade de ações que impeçam a repetição de equívocos passados;
- Casos ajudam a focalizar a atenção em elementos importantes de um problema.
- O que foi importante para um sucesso ou fracasso numa situação passada tende a sê-lo numa nova situação.

O raciocínio a partir de casos apresenta também alguns riscos, entre eles pode-se destacar:

- Um raciocinador RBC pode ser tentado a usar casos passados cegamente, confiando em experiências prévias sem validá-las na nova situação;
- Casos podem direcionar excessivamente um raciocinador na solução de um novo problema;
- Freqüentemente pessoas, especialmente novatos, não se lembram dos casos mais apropriados quando estão raciocinando.

6.1 Exemplo de sistemas RBC: Clavier

O RBC têm sido usado com sucesso em diversas aplicações, incluindo, auxilio ao projeto conceitual em arquitetura [Pearce, et al. 1992], diagnóstico de doenças cardíacas [Koton, 1989], construção de *help-desks* e suporte técnico a clientes [Simoudis, 1992; Nguyen et al., 1993], detecção de fraudes em transações com cartões de crédito [Reategui e Campbell, 1994], suporte a analistas de investimentos [Madhavan, 1994].

Vamos, a título de exemplo, discutir o sistema Clavier [Hinkle e Toomey. (1995)], desenvolvido pela Lockheed, para auxiliar na determinação de cargas eficientes de materiais compostos de fibras de grafite, destinadas à indústria aeroespacial, em fornos pressurizados de ventilação forçada, denominados autoclaves.

O processo empregado para a produção desses peças envolve duas etapas principais. Na primeira, é feita a deposição de várias camadas de materiais compostos de fibra de vidro e grafite em moldes. Esta etapa requer de 2 a 7 dias, dependendo das peças a fabricar. A segunda etapa consiste da cura, isto é no endurecimento das peças por calor e pressão, feita nos autoclaves. Esta etapa demora de 6 a 8 horas.

Há vários motivos para a melhoria do processo. Vejamos alguns. As peças que durante o processo de cura saírem de um determinado perfil termodinâmico, devem ser inspecionadas, a um custo unitário de US$ 1.000, se forem encontradas falhas durante a inspeção, a perda pode chegar a US$ 50.000, por peça.

O objetivo básico dos operadores dos autoclaves e dos engenheiros de processo no momento de determinar uma carga é maximizar o número de peças devidamente curadas. Para isso, devem decidir que partes (peças) carregar em cada sessão e onde posicioná-las. O problema é difícil pois deve-se assegurar que no início do processo de cura seja mantida uma diferença de temperatura entre as partes (DT) menor que 30 °F e que a taxa de aumento da temperatura seja maior que 1 grau/minuto. Num intervalo de 40 a 80 minutos, do inicio do processo, deve-se alcançar um DT < 20 °F, que deve ser mantido à medida que o autoclave é pressurizado.

Como a Lockheed dispõe de apenas dois autoclaves na sua planta de Sunnyvale, não se pode restringir cada carga a peças de um único tipo, devem ser feitas cargas mistas. Essas cargas são muito difíceis de fazer pois a) cada molde tem um perfil de aquecimento distinto, afetado pelo seu tamanho, forma e espessura; b) os autoclaves têm regiões mais quentes que outras: um deslocamento de 30 cm pode estragar uma peça; c) as regiões quentes de cada autoclave localizam-se em diferentes pontos; e d) as partes carregadas afetam o fluxo de ar e a temperatura no autoclave.

Antes de empregar RBC a Lockheed tentou utilizar sistemas especialistas baseados em regras, modelagem termodinâmica e indução, nenhuma dessas abordagens funcionou. No caso dos sistemas baseados em regras, o principal obstáculo foi que os especialistas não eram capazes de fornecer regras, o seu raciocínio concentra-se na carga como um todo, e não no posicionamento individual dos moldes. A modelagem termodinâmica, embora bastante empregada em problemas semelhantes, torna-se inviável quando é preciso considerar os efeitos de partes adjacentes no fluxo de ar. A indução não foi bem sucedida pois o espaço de busca (o conjunto de todas as cargas possíveis) é muito grande e o espaço de exemplos (cargas realizadas) muito pequeno.

O RBC foi considerado pois os engenheiros do conhecimento envolvidos na construção do sistema verificaram que os operadores do forno consultavam suas anotações, onde documentavam cargas bem e mal sucedidas, à busca de idéias para novas cargas, logo o RBC mostrou-se a solução natural para o problema.

O processo de aquisição do conhecimento foi simples pois toda a informação necessária já era armazenada como parte normal da rotina de trabalho. Casos em Clavier representam os moldes usados, as mesas usadas para suportar os moldes, a configuração espacial dos moldes e mesas (em coordenadas 2-D) no forno e o resultado da carga (bem sucedido ou mal sucedido, com respectivas análises).

Dado um problema (um conjunto de peças a curar), o processo RBC em Clavier compreende a recuperação de casos prévios (cargas semelhantes), a adaptação do caso mais próximo (carga) recuperado, validação da nova carga sugerida e aprendizado. A recuperação de casos prévios relevantes e feita hierarquizando os casos disponíveis de modo a encontrar o "vizinho mais próximo" do problema atual. O vizinho mais próximo (a carga passada a recuperar) será aquele que maximize o número de peças a curar, minimize o número de peças da carga anterior que não casam com as peças a curar e maximize a qualidade da carga (determinada pela compatibilidade das peças). Embora o sistema possua um módulo de adaptação automático, a experiência mostrou que isto deve ser feito pelo operador que sugere como modificar o caso recuperado. Clavier prevê, então, se a nova carga sugerida será bem sucedida ou não (através da comparação com cargas anteriores). Finalmente, o sistema aprende através do armazenamento da descrição da nova carga e da análise dos seus resultados na base de casos.

O desenvolvimento de Clavier foi iniciado em março de 1989, tendo a versão 1.0 sido concluída em setembro de 1990 e a versão 2.0 em novembro de 1991. A implementação foi feita em Commom Lisp e roda em um microcomputador Macintosh com 8MB de memória. O tempo total de desenvolvimento foi estimado em duas pessoas-ano.

Clavier tem sido usado para 2 ou 3 cargas diárias, desde 1990. Inicialmente, com uma pequena base de casos que foi aumentando com o uso. Depois da implantação de Clavier os problemas de carga do autoclave, e os custos decorrentes, foram virtualmente eliminadas. A Lockheed está usando em várias instalações e negociando o seu licenciamento para outras companhias do setor aeroespacial.

Clavier é um sistema que fornece sugestões e advertências a especialistas humanos que tomam as decisões finais. Clavier trata de algumas questões chave relacionadas à construção de sistemas baseados em conhecimento. O sistema ilustra como casos podem ser usados para capturar o conhecimento de um domínio pouco entendido (superação de teorias imperfeitas), como o processo de aquisição de conhecimento pode ser simplificado, na medida em que uma base inicial de casos esteja disponível, como é possível reduzir o custo computacional (relativamente, por exemplo, a estratégias clássicas de busca) através da reutilização de casos disponíveis.

7. AGENTES INTELIGENTES

Nos últimos tempos, muito tem sido dito sobre agentes. Fala-se de agentes autônomos, agentes móveis, agentes de *software*, de busca, de compras, de recuperação de informações, e assim por diante. Da mesma forma são dadas diversas definições que procuram caracterizar cada um deles [Heilmann et alli,1996; Franklin e Graesser, 1996]. De modo geral, esses tipos de agentes apresentam facetas ou são exemplos de uma categoria mais abrangente de agente denominado Agente Inteligente.

Norvig e Russel (1995), definem um agente com sendo qualquer coisa que percebe o seu ambiente através de sensores e age sobre ele através de

atuadores. Dependendo de como se entende o que venha a ser ambiente, sensores, atuadores, perceber e agir, essa definição é tão abrangente que permite incluir praticamente qualquer programa. Como apontam Franklin e Graesser, se definirmos ambiente como qualquer coisa que fornece uma entrada ao programa e recebe uma saída, e considerarmos que receber a entrada é perceber o ambiente e produzir uma saída é atuar sobre o ambiente, qualquer programa é um agente.

No entanto, para Norvig e Russel agentes devem também ser racionais, no sentido empregado na seção 1. Logo, um agente racional é aquele que faz a coisa certa, isto é, que atua para ser bem sucedido. Assim, para cada possível seqüência de percepções, um agente racional ideal deve executar a ação que procure maximizar a sua medida de desempenho (segundo critérios dependentes de contexto), com base nas evidências fornecidas pela seqüência percebida e no conhecimento que o agente tem. Uma característica importante de tais agentes é a autonomia. Um agente é autônomo se seu comportamento for determinado pela sua própria experiência, em lugar de pelo conhecimento do ambiente que lhe foi incorporado pelo projetista.

A definição acima é coerente com as seguintes propriedades que Wooldridge e Jennings (1995) atribuem a um agente:

- autonomia: agentes operam sem a direta intervenção de humanos ou outros, e tem algum tipo de controle sobre as suas ações e estado interno;
- habilidade social: agentes interagem com outros agentes (e possivelmente com humanos) através de algum tipo de linguagem de comunicação de agentes;
- reatividade: agentes percebem o seu ambiente e respondem a mudanças nele observadas, observando as restrições temporais associadas ao ambiente.
- pró-atividade: agentes não agem simplesmente em resposta ao seu ambiente, eles são capazes de tomar iniciativas, de modo a exibir um comportamento dirigido pelos seus objetivos.

Mecanismos de raciocínio e aprendizado são, portanto, centrais à implementação de agentes inteligentes. Nesse sentido as abordagens discutidas anteriormente, busca, RBR, indução e RBC podem ser, e tem sido, combinadas para a implementação de agentes inteligentes.

Para ilustrar a implementação de agentes inteligentes vamos discutir na próxima seção os agentes de interface propostos por Maes (1994).

7.1 Exemplo de agente inteligente: Agente de Interface

O paradigma dominante de interação com o computador, atualmente, é denominado de manipulação direta, que requer que o usuário inicie explicitamente todas as tarefas a realizar e monitore todos os eventos decorrentes. Maes propõe o uso de agentes de interface para suportar um estilo de interação complementar denominado *manipulação indireta*, segundo o qual, o usuário envolve-se num processo cooperativo em que tanto pessoas como agentes podem iniciar a comunicação, monitorar eventos e realizar tarefas. Os agentes de interface são construídos segundo o modelo do assistente pessoal que colabora com o usuário num determinado ambiente de trabalho, tornando-se progressivamente mais efetivo à medida que aprende os interesses, hábitos e preferências do usuário e de sua comunidade.

Para a construção de agentes de interface Maes destaca dois problemas importantes a resolver. O primeiro, é o problema da competência, que consiste em saber como um agente adquire o conhecimento necessário para decidir quando, com o que, e como ajudar ao usuário. O segundo, é o problema da confiança, que consiste em garantir que o usuário se sinta seguro para delegar tarefas a um agente. Três abordagens poderiam ser adotadas para procurar resolver esses problemas: 1) agentes programados pelo usuário; 2) agentes baseados em conhecimento; e 3) agentes com capacidade de aprendizado.

Na primeira abordagem, o usuário final programa o agente, ele poderia, por exemplo, definir as regras a empregar para processar mensagens de correio eletrônico recebidas, isso feito o agente pode processar as regras sem a intervenção direta do usuário. Nesta abordagem a solução do problema da competência requer muito conhecimento e esforço por parte do usuário para criar e manter o conhecimento do agente. A solução do problema da confian-

ça depende da confiança que o usuário tem nas suas habilidades para programar.

Na abordagem do agente baseado em conhecimento, o agente contêm um modelo do domínio e do usuário e, em tempo de execução, ele usa o seu conhecimento para reconhecer os planos do usuário e para encontrar oportunidades de contribuir com eles. A construção de um agente competente, nesta abordagem, requer muito trabalho por parte de um engenheiro do conhecimento, havendo baixo reaproveitamento do conhecimento e estrutura de controle na construção de agentes para outras aplicações. Neste caso, o conhecimento do agente é fixo, sendo necessário um processo de manutenção constante da base de conhecimento que reflita mudanças do ambiente e do usuário (por exemplo, mudanças de hábitos e preferências). Em adição a isso deve-se considerar que a representação de conhecimento que permita compreender sempre as ações, às vezes imprevisíveis, do usuário é um problema, em si, muito complicado. Em termos da confiança que o usuário deposita no agente, deve-se notar que agentes muito sofisticados, competentes e autônomos desde o início, podem transmitir ao usuário a impressão de perda de controle e entendimento, pois como o agente foi programado por outra pessoa, o usuário pode não ter um bom modelo das suas limitações, modo de funcionamento, etc.

Na terceira abordagem, agentes com capacidade de aprender, o agente recebe um conhecimento inicial (*background*) mínimo e aprende o comportamento apropriado interagindo com o usuário e com outros agentes. Para que esta abordagem seja adequada é necessário que a tarefa a realizar envolva uma quantidade substancial de comportamento repetitivo e que este comportamento seja potencialmente diferente para usuários distintos. Se a primeira condição não se verifica o agente não será capaz de aprender, já que não será possível observar padrões (regularidades) no comportamento do usuário. Se a segunda condição não se verificar, a abordagem anterior, construção de um agente baseado em conhecimento, seria a mais adequada. Um agente aprendiz inicia o seu trabalho com uma capacidade muito reduzida de ajudar ao usuário, a sua competência aumenta progressivamente, à medida que ele observa o usuário e este o instrui e fornece uma realimentação relativa ao seu desempenho. Isto contribui para que o usuário tenha tempo de gradualmente desenvolver um modelo de como o agente toma decisões, permitindo-lhe adquirir confiança nele, contribuindo para isso o fato de o agente poder dar explicações para o seu raciocínio e comportamento em termos de casos passados semelhantes à nova situação. Um agente com capacidade de aprender requer menos esforço por parte do usuário final e do desenvolvedor da aplicação, e pode mais facilmente adaptar-se ao longo do tempo a mudanças nas preferências individuais e organizacionais.

Maxims [Maes, 1994] é um dos agentes com capacidade de aprender baseados no modelo do assistente pessoal desenvolvido por Pattie Maes. Maxims aprende a priorizar, apagar, redirecionar, classificar e armazenar mensagens de correio eletrônico de um usuário. Ele foi implementado em Common Lisp para Macintosh e comunica-se com o pacote Eudora, usando Apple Events. A técnica de aprendizado usada foi o raciocínio baseado em memória - um subtipo de RBC que foca o processo de recuperação.

Maxims observa continuamente o usuário e, à medida que este executa ações, memoriza os pares situação-ação gerados. A descrição de uma situação inclui o emissor e o receptor da mensagem, a lista *c:*, as palavras chave na linha *Subject:*, se a mensagem foi lida ou não, se se trata de uma resposta a uma mensagem anterior, etc.

Quando uma nova situação ocorre, o agente tenta prever as ações do usuário, baseado nos exemplos registrados na sua memória. A semelhança entre situações é determinada por uma soma ponderada das diferenças entre as características que representam uma situação. Sendo a importância relativa das características determinada pelo agente. Ocasionalmente o agente analisa a sua memória e determina as correlações existentes entre características e ações tomadas. Se o usuário não quiser esperar que o agente detecte um determinado padrão, ele pode instruir o agente criando uma situação hipotética e mostrando ao agente o que deve ser feito.

Maxims determina também um grau de confiança para as suas previsões. Este grau de confiança determina como ele usa a sua previsão, se o grau for alto ele age sem consultar o usuário se o grau de confiança for médio ele faz uma sugestão ao usuário, e se o grau for baixo, ele simplesmente observa a atitude do usuário. Quando o agente está muito inseguro com relação ao que fazer numa dada situa-

ção, ele pede ajuda a agentes assistindo a outros usuários. Os agentes descobrem gradualmente quais outros agentes são fontes confiáveis de informação para cada tipo de problema.

8. CONCLUSÃO

É provável que ainda estejamos muito longe de alcançar os propósitos finais da IA: entender o que é inteligência e construir entidades (máquinas) inteligentes. A IA é considerada por muitos uma disciplina em crise pela ausência de um paradigma unificador, que possa servir de base aos pesquisadores e praticantes desejando construir sistemas inteligentes. Isso é devido, em parte, a que o termo inteligência tende a colocar como referencial para medir o sucesso da área a sua manifestação mais desafiadora, a inteligência humana, ainda em larga medida incompreendida. No entanto, os trabalhos realizados nas últimas décadas sedimentaram um expressivo número de técnicas, metodologias e paradigmas que permitem a solução de um amplo espectro de problemas de relevância científica e comercial. Este artigo apresentou algumas técnicas e paradigmas representativos, sem a intenção de esgotar o assunto. Dadas as limitações de espaço, a escolha dos temas a tratar foi pautada não somente pela sua importância como também pela familiaridade do autor com eles.

Cada uma das tecnologias estudadas está assentada em teorias que procuram explicar como representar conhecimento e realizar inferências a partir dele. Cada uma delas permite que isto seja realizado com sucesso em alguns tipos de problemas, não ocorrendo o mesmo em outros. A sua combinação, porém, oferece um grande potencial, ainda não devidamente explorado. As possibilidades são muitas. Por exemplo, o raciocínio baseado em casos pode ser combinado com algoritmos de busca e com raciocínio baseado em regras para reduzir os recursos e tempo necessário à solução de problemas já tratados no passado, bem como para aumentar a sua qualidade [Koton, 1989]. Algoritmos de inferência automática de regras podem ser usados para a descoberta de conhecimento em grandes bases de dados [Fayyad, 1996], o que seria extremamente oneroso, ou mesmo virtualmente impossível, dependendo do domínio e da massa de dados a tratar, se realizado por métodos convencionais conduzidos por engenheiros de conhecimento e especialistas. A lógica difusa [Zadeh, 1979] pode ser combinada com RBR para representar e fazer inferências a partir de conhecimento incerto.

A grande difusão dos computadores, o aumento constante de sua capacidade de processamento e armazenamento, aliados à popularização das redes de computadores, particularmente a Internet, está gerando um crescimento vertiginoso de informação (e lixo) acessível. As tecnologias de IA foram desenvolvidas para lidar com problemas complexos, como aqueles que se apresentam nesse ciberespaço em construção (roteamento, acesso e filtragem de informação, etc.) devendo vir a desempenhar um papel importante no desenvolvimento de novas e inovadoras aplicações, muitas delas sob a denominação de agentes.

REFERÊNCIAS

FAYYAD, U.M. (1996). "Data Mining and Knowledge Discovery: Making Sense Out of Data". *IEEE Expert*, october, pp. 20-25.

FRANKLIN, S. and A. Graesser, (1996). "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents". *Proc. Third International Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag.

GONZALEZ, A.J., D. D. Dankel, The Engineering of Knowledge-based Systems: Theory and Practice, Prentice-Hall, 1993.

HEILMANN, K. et al. (1996). "Intelligent Agents: A Technology and Busines Application Analysis", http:://haas.berkeley.edu/~heimann/agents/.

HINKLE, D. and C. Toomey. (1995) ."Applying Case-Based Reasoning to Manufacturing". *AI Magazine*, spring, pp. 65-73.

MAES, P. (1994). "Agents that Reduce Work and Information Overload". *Comm. ACM*, July, pp. 30-39.

KELLY, G. A. (1955). **The Psychology of Personal Constructs, V.1 - A Theory of Personality**, New York, NY, W.W. Norton.

KOLODNER, J. (1993). **Case-based Reasoning**, Morgan Kaufmann Publishers.

KOTON, P. (1989). "Using experience in learning and problem solvin." Ph.D. diss., Dept. of Computer Science, MIT.

LANGLELY, P. and H. A. Simon. (1995). "Applications of Machine Learning and Rule Induction.", Comm. of the ACM, vol. 38, no.11, pp. 54-64..

MADHAVAN, R. D. (1994). "Goal-Based Reasoning for Securities Analysis". *AI Expert*, feb., pp. 23-29

NGUYEN, T. M. Czerwinski and D. Lee. (1993) ."Compaq Quichsource: Providing the Consumer with the Power of AI". *AI Magazine*. fall, pp. 50-60.

PEARCE, et. al. (1992). "Case-Based Design Support: A Case Study in Architectural Design", *IEEE Expert*, october, pp. 14-20.

QUINLAN, J.R.(1986). "Induction of decision trees." *Machine Learning*, 1:81-106.

QUINLAN, J.R. (1993). **C4.5: Programs for machine learning**. Morgan-Kaufmann, San Francisco, CA.

REATEGUI, E. B. and J. Campbell, (1994). "A Classification System for Credit Card Transactions.". European Workshop on CBR.

RIESBECK, C.K., R.C. Schank, (1989). **Inside Case-Based Reasoning**, Lawrence Erlbaum.

RUSSEL, S. and P. Norvig, (1995). **Artificial Intelligence: A Modern Approach**. Prentice-Hall.

SIMOUDIS, E. (1992). "Using Case-Based Retrieval for Customer Techincal Support". IEEE Expert, october, pp. 7-12.

TURBAN, E., **Decision Support and Expert Systems: Management Support Systems, 4**rd. Ed. MacMillan, 1995.

WOOLDRIDGE, M. and N. R. Jennings (1995), "Agent Theories, Architectures, and Languages", in. **Intelligent Agents**, Springer-Verlag, Berlin, 1-22.

ZADEH, L.A. (1979). "A theory of approximate reasoning." *Machine Intelligence 9*.