2
Agentes Inteligentes e Sistemas Multi-Agentes

2.1
Agentes Inteligentes

O processamento baseado em agentes pode ser caracterizado, em um nível alto de abstração, por um estado de entrada que, após processado, produz uma saída. O estado de entrada é determinado pelo ambiente em que o agente atua e a saída é determinada pelos efeitos que o agente causa em seu ambiente. A despeito de existirem muitas similaridades entre a teoria de agentes e a teoria de controle, agentes encontram sua melhor aplicação em situações específicas, como ambientes complexos e dinâmicos ou quando as informações sobre o ambiente são incompletas (Brenner et. al, 1998; Müller, 1996).

Agentes podem ser classificados segundo várias características. As principais são autonomia e inteligência.

Autonomia: uma das mais importantes diferenças entre agentes e programas tradicionais é a capacidade que um agente possui de seguir seus objetivos automaticamente, isto é, sem comandos externos. Para permitir este comportamento autônomo, o agente deve ter controle sobre suas ações e percepção sobre seu estado e o ambiente. O grau de autonomia deve ser especificado pelo usuário/programador, uma vez que podem existir situações nas quais não se deseja que o agente tenha completa autonomia no processo de decisão, como, por exemplo, em sistemas relacionados à área financeira ou em operações que envolvam a vida humana (Brenner et. al. 1998).

Inteligência: a característica inteligência de um agente é formada por um ou mais dos seguintes componentes:

- a base de conhecimento;
- a capacidade de raciocínio baseado em sua base de conhecimento;

PUC-Rio - Certificação Digital Nº 0610800/CA

- a capacidade de aprender ou de se adaptar às mudanças ocorridas no ambiente.

A capacidade de raciocínio permite ao agente observar o ambiente e tomar decisões específicas quando mudanças ocorrem no mesmo. A capacidade de aprendizado e adaptação ao ambiente amplia a capacidade autônoma do agente. O aprendizado é, portanto, o componente mais importante, e também mais complexo, da inteligência de um agente (Nwana, 1996; Brenner et. al, 1998; Russel e Norvig, 2010).

Além dos aspectos de autonomia e inteligência, os agentes podem ser também denominados: a) **estáticos:** todo o seu processamento está baseado em um único computador; b) **móveis**: têm habilidade de se mover através de algum tipo de rede de computadores; c) **reativos:** não têm qualquer tipo de modelo do ambiente, agindo em resposta a estímulos do ambiente; d) **deliberativos:** possuem internamente um modelo de ambiente, planejando e negociando com o objetivo de coordenarem suas atividades; e) **sociáveis**: é a capacidade de um agente interagir com outros agentes (ou usuários) para satisfazer seus objetivos; f) **pró-ativos:** o agente não reage apenas a mudanças no ambiente, mas possui iniciativa própria sob certas circunstâncias (Müller, 1996; Nwana e Azarmi, 1997; Brenner et. al, 1998; Russel e Norvig, 2010).

Os agentes podem acumular duas ou mais características citadas anteriormente. Muitas dessas características ainda apresentam um conceito bastante vago, permitindo muitas interpretações, porém a inteligência é a que mais apresenta conceitos abstratos.

2.2
Inteligência

Um agente autônomo deve interagir com o ambiente para atingir seus objetivos. Isso é possível quando o agente é capaz de reunir informações sobre o seu ambiente, tomar decisões baseadas nestas informações, e iniciar ações específicas baseadas nas decisões.

PUC-Rio - Certificação Digital Nº 0610800/CA

O agente inteligente deve ser capaz de se adaptar ao ambiente em que está inserido e de aprender, sem que seja necessário dar ao agente qualquer conhecimento explícito sobre o ambiente (Monsieurs, 2002).

Alguns autores abordam a característica inteligência de formas diferentes. Brooks (1991), por exemplo, diz que a inteligência não existe em ambientes fechados, como no caso dos agentes deliberativos. Por outro lado, nos agentes reativos, a inteligência existe e cresce apenas através da interação contínua com o sistema. Agentes reativos são simples e podem se comunicar de forma básica, contudo, padrões de comportamento complexos podem emergir da interação entre os diversos agentes e/ou entre este(s) e o ambiente.

Segundo Nwana e Azarmi (1997), a capacidade do agente de se adaptar ao ambiente (aprendizado), pré-requisito para a inteligência, é uma forma de aumentar o desempenho de sistemas baseados em agentes. Maes (1995) também afirma que, muitas vezes, o uso do termo "agentes inteligentes" não é adequado a uma aplicação, porque agentes verdadeiramente inteligentes ainda não existem. Nwana e Azarmi (1997) e Brenner et. al. (1998) concordam com o fato de que, para os agentes serem autônomos, eles devem ser capazes de aprender.

Sendo assim, pode-se concluir que, para que um programa possa ser classificado como agente, é essencial que ele seja autônomo. Para que seja autônomo, deve possuir, segundo alguns pesquisadores, a característica inteligência, especificamente a capacidade de aprender.

Nas próximas seções serão abordadas as partes que compõem a inteligência, isto é, base de conhecimento, raciocínio e aprendizado.

PUC-Rio - Certificação Digital Nº 0610800/CA

2.2.1
Base de Conhecimento

Existem muitas formas de conhecimento e diferentes maneiras de se armazenar e representar o conhecimento. Os programadores usam a forma simbólica (números ou letras) chamada de representação interna, pois esta é facilmente processada por computadores. A forma mais confortável é a linguagem natural, que, no entanto, apresenta muitas ambigüidades. Por este motivo, linguagens formais matemáticas e lógicas foram desenvolvidas. Existem diferentes tipos de conhecimento e muitas maneiras diferentes de representá-los:

fatos simples ou relacionamentos complexos, fórmulas matemáticas ou regras de sintaxe de linguagem natural, etc. A escolha da forma de representar o conhecimento está diretamente relacionada às necessidades da aplicação e do computador. A base de conhecimento é um repositório de informações, central ou não, contendo fatos que são conhecidos sobre objetos e suas relações. O processo de mapear um conjunto de conhecimentos no domínio de um problema e convertê-lo em base de conhecimento é chamado de engenharia do conhecimento ou aquisição do conhecimento. Maiores detalhes podem ser encontrados em (Bigus e Bigus, 1998; Russel e Norvig, 2010).

2.2.2
Raciocínio

Regras do tipo *if-then-else* tornaram-se uma forma muito popular para representar o conhecimento. Isto é motivado pelo fato de ser mais confortável ler regras na forma em que normalmente as concebemos do que na forma de lógica de predicados (linguagem formal). Cada regra é vista como uma unidade do conhecimento que pode ser alterada pela modificação ou criação de novas regras, sem gerar problemas para o sistema.

Normalmente, as regras são compostas por dois componentes: antecedentes e conseqüentes. Se todas as cláusulas dos antecedentes forem verdadeiras, a regra é disparada. Quando isto acontece, o conseqüente da regra é tomado como verdade e um novo fato pode ser adicionado à base de conhecimento. O raciocínio é classificado como monotônico quando novos fatos podem ser adicionados a *work memory*, ou como não-monotônico quando fatos podem ser cancelados (Bigus e Bigus, 1998; Brenner et. al, 1998).

2.2.3
Aprendizado

Para alguns pesquisadores (Brenner et. al., 1998), a autonomia está diretamente relacionada à capacidade de se aprender um comportamento.

Embora muitos pesquisadores explorem ainda hoje as áreas relativas à base de conhecimento e raciocínio, o aprendizado é, no entanto, o que pode dotar

PUC-Rio - Certificação Digital Nº 0610800/CA

os agentes de verdadeira capacidade de desenvolver sua autonomia, pois com o aprendizado o agente está sempre apto a se adaptar aos ambientes.

O processo de aprendizado pode ser classificado segundo a quantidade de esforço empregado, como pode ser visto abaixo (Weiss, 1999):

- Aprendizado simples: implantação do conhecimento e da capacidade de aprendizado de forma direta, sem necessitar de inferência ou transformação por parte do agente;
- Aprendizado a partir de instrução ou por conselho: operacionalização de novas informações que não são diretamente executadas pelo agente, isto é, são transformadas para uma representação interna e integradas ao conhecimento ou capacidade existente;
- Aprendizado a partir de exemplos e pela prática: extração e refinamento do conhecimento; unificação de padrões a partir de exemplos positivos e negativos ou a partir de exemplos práticos;
- Aprendizado por analogia: transformação do conhecimento a partir de um problema resolvido ou similar;
- Aprendizado através de descobertas: novos conhecimentos podem ser adquiridos através de observações e/ou experimentos, gerando e testando hipóteses ou teorias.

Outra forma de classificar o aprendizado é segundo a forma de realimentação (Weiss, 1999):

- Aprendizado supervisionado: a realimentação especifica a atividade desejada pelo agente e o objetivo do aprendizado é tornar a ação executada e a ação desejada tão próximas quanto possível;
- Aprendizado não supervisionado: nenhuma realimentação explícita é fornecida e o objetivo é descobrir atividades úteis e desejadas através de tentativa-e-erro e processos auto-organizáveis;
- Aprendizado por reforço: a realimentação é um valor ou sinal que especifica a utilidade da atividade ou ação realizada pelo agente quando ele se encontra em um determinado estado. O objetivo do aprendizado é maximizar esta utilidade para todos os estados.

PUC-Rio - Certificação Digital Nº 0610800/CA

Agentes veem uma ação de realimentação como um mestre no caso de aprendizado supervisionado, como um crítico no caso de aprendizado por reforço ou como um observador passivo no caso de aprendizado não supervisionado.

A próxima seção é destinada ao detalhamento do funcionamento de métodos de aprendizado por reforço, que serve de base para os algoritmos utilizados neste trabalho.

2.3
Aprendizado por Reforço (Reinforcement Learning - RL)

O aprendizado por reforço é geralmente usado com o objetivo de controlar processos. Quando aplicado a agentes autônomos, possui a finalidade de fazer com que os agentes aprendam um determinado comportamento. De uma maneira geral, o algoritmo de aprendizado por reforço recebe do ambiente um sinal referente à avaliação do desempenho atual (reforço); baseado neste reforço, o algoritmo modifica o controlador tentando evoluir ações que melhor determinem o comportamento esperado.

Este tipo de modelo de aprendizado tem inspiração biológica: quando se realiza algo corretamente ou benéfico, recebe-se uma recompensa, e quando se realiza algo incorretamente ou prejudicial, recebe-se uma punição. A tendência é repetir a ações que estão associadas a recompensas positivas e evitar ações associadas a recomensas negativas, sempre que as condições forem semelhantes.

Reinforcement Learning consiste, portanto, em determinar o comportamento de um agente através da interação com o seu ambiente. As ações executadas têm como conseqüência um sinal de reforço, porém não existe conhecimento prévio da ação correta para cada possível estado. Cabe ao agente descobrir quais ações levam a um melhor desempenho do sistema. O modelo de *Reinforcement Learning* padrão pode ser visto na Figura 1.

PUC-Rio - Certificação Digital Nº 0610800/CA

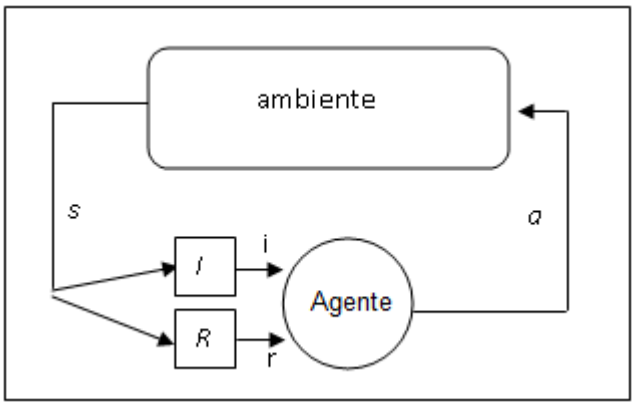

Figura 1: Esquema do modelo Reinforcement Learning.

A cada instante de tempo, o agente examina, através da sua percepção, (sensores) o estado *S(I)*. A função *I* define a observabilidade do ambiente. Esta função será considerada como a função identidade, ou seja, os estados apresentam observabilidade total. O agente em um estado *s* deve escolher, baseado em suas observações, uma ação *a* a partir de um conjunto *A* discreto de ações disponíveis, isto é $a \in A$. Após a execução desta ação, o ambiente passa a um estado *s´* com probabilidade que depende da ação *a*, e um sinal de reforço *r* é recebido do ambiente. Este sinal, denominado de reforço primário, depende do estado anterior *s* e da ação *a*. O agente deve determinar qual ação executar quando está em um estado específico. A função de reforço qualifica uma tarefa realizada pelo agente através de recompensas e punições. Para sistemas que não recebem sinais de reforços primários a cada iteração, premiar ou punir ações passadas que tenham contribuído para o sucesso ou falha do sistema torna-se um problema (*delayed reinforcement)*. Neste tipo de aplicação o sinal de reforço não é conhecido até que o agente atinja o objetivo ou chegue ao final da tarefa. Algumas vezes pode ser difícil chegar ou identificar o final da tarefa, além de consumir grande quantidade de memória. Este problema é denominado de associação temporal de crédito (*temporal credit-assignment)* (Sutton & Barto, 1998; Kaelbling et. al., 1996; Jouffe, 1998). Para contornar este problema, o modelo matemático usado é o Processo de Decisão de Markov (*Markovian Decision Problems* – MDP).

Dado um completo e preciso modelo MDP, a Programação Dinâmica oferece uma ferramenta eficiente para o aprendizado *off-line* do comportamento do agente.

PUC-Rio - Certificação Digital Nº 0610800/CA

Programação Dinâmica (Bellman, 1957), para ser computacionalmente eficiente, exige uma modelagem precisa do ambiente, como um Processo de Markov.

Como as necessidades de Programação Dinâmica muitas vezes são inviáveis, como, por exemplo, a necessidade do conhecimento do modelo do ambiente, a função de Programação Dinâmica tem sido a de servir de inspiração para outros modelos que visam a adaptar estas necessidades.

O método de Monte Carlo possibilita o aprendizado a partir da experiência obtida através de seqüências de estados, ações e reforços recebidos pela interação agente-ambiente. Esse processo de aprendizagem a partir de experiência *on-line* é interessante pois não requer conhecimento a priori da dinâmica do ambiente, mesmo assim consegue encontrar uma política ótima (Sutton e Barto, 1998).

Dessa forma, os métodos de Monte Carlo (Rubinstein, 1981) não precisam da modelagem do ambiente e se apresentam de forma simples em termos conceituais. Não são viáveis quando a solução do problema é possível apenas de forma incremental (utilizando somente o estado atual e os estados imediatos), pois para se atualizar a função de valor os métodos de Monte Carlo exigem que o estado final seja alcançado no processo.

Os métodos de Diferença Temporal são uma combinação de características dos métodos de Monte Carlo com as idéias da Programação Dinâmica, no sentido de que buscam estimar valores de utilidade para cada estado do ambiente através dos ganhos de transições e de valores de estados sucessivos. Nos métodos de Diferença Temporal o ambiente não precisa ser modelado completamente (é vantagem sobre Programação Dinâmica), não é necessário chegar ao estado final (é vantagem sobre o Monte Carlo). Os dois métodos mais representativos na literatura são Q-Learning (Watkins, 1992) e SARSA (Rummery e Niranjan, 1994; Sutton, 1998).

Vale ressaltar que a principal idéia de Reinforcement Learning é o uso de função de valores para a descoberta de boas políticas. Maiores detalhes podem ser encontrados no em Kaelbling et. al. (1996), Sutton & Barto (1998) e Bellman (1957).

PUC-Rio - Certificação Digital Nº 0610800/CA

2.4
Sistemas Multi-Agentes (SMA)

2.4.1
Introdução

Os Sistemas Multi-Agentes formam uma área emergente da Inteligência Artificial que fornece os princípios para construção de sistemas complexos envolvendo múltiplos agentes e os mecanismos de coordenação entre eles (Stone, Veloso, 2000). Um SMA pode ser definido como um grupo de agentes autônomos, interagindo entre si e compartilhando um mesmo ambiente, que é percebido através de sensores, onde eles agem realizando ações (Vlassis, 2003). Este tipo de sistema tem uma grande variedade de aplicação, como controle distribuído (Riedmiller et al., 2000; Stephan et al., 2000; Wiering, 2000; Bakker et al., 2005), times robóticos (Balch, 1997; Luke et al., 1997; Luke, 1998; Fernandez e Parker, 2001; Stone e Veloso, 2000; Hladek, 2007), controle de navegação (Bowling e Veloso, 2002; Hu e Wellman, 2003; Calvo e Romero, 2007), controle e planejamento de rotas (Boyan e Littman, 1993; Choi e Yeung, 1995; Tillotson et al., 2004), programação de elevadores (Crites e Barto, 1998), balanceamento de carga (Schaerf et al., 1995), negociação (*trading*) automática entre agentes (Wellman et al., 2003; Hsu e Soo, 2001; Lee e Oo, 2002; Oo et al., 2002; Tesauro e Kephart, 2002; Raju et al., 2003), precificação dinâmica de produtos de varejo (Raju et al. 2003), etc.

Os benefícios trazidos pela aplicação de SMA são diversos. Em primeiro lugar, através da computação paralela, vários agentes podem trabalhar em conjunto para melhor explorar a estrutura descentralizada de uma determinada tarefa e acelerar sua conclusão (Lagoudakis, 2002; Kok et al., 2005; Fitch et al., 2005). Além disso, agentes podem trocar experiências se comunicando (Tan, 1993), podem apenas observar os mais habilidosos e aprender (Price e Boutilier, 2003), os mais inteligentes podem servir de professores para outros agentes (Clouse, 1995), etc. Os SMA também podem fornecer alto grau de escalabilidade, através da inclusão de novos agentes quando necessário, e ainda fazer com que agentes assumam as atividades de outros agentes em casos de falha (Busoniu et al., 2008). Gerhard e Sen (1996) também destacam a utilização de SMA no estudo

PUC-Rio - Certificação Digital Nº 0610800/CA

e elucidação da inteligência. Como a inteligência está profundamente acoplada à interação, a melhor forma de criar máquinas inteligentes pode ser através da construção de "redes sociais" de máquinas.

Shoham et al. (2003) apresentam quatro grandes subáreas de pesquisa dentro do aprendizado de sistemas multi-agentes: uma descritiva e 3 prescritivas. A descritiva, inspirada na Psicologia, explica como humanos aprendem no contexto de outros humanos. As três linhas de pesquisa prescritivas trabalham como os agentes devem aprender. A primeira delas, Inteligência Artificial Distribuída ("*Distributed AI*"), estuda problemas de controle distribuído. Geralmente, um agente central controla múltiplos agentes, mas não define a "política ótima" para eles individualmente. Define apenas um procedimento adaptativo que converge para uma "política ótima". A segunda ("*Equilibrium Agenda*") analisa o equilíbrio entre as estratégias de diferentes agentes. Na Teoria dos Jogos não-cooperativos, a noção de "estratégia ótima" perde o sentido, sendo substituída pelo conceito de "melhor resposta" individual, que gera um equilíbrio e faz o sistema convergir. A terceira e última ("*AI Agenda*") estuda a escolha da melhor estratégia para um agente, considerando uma classe fixa de agentes no jogo, ou a definição de um agente ótimo para um determinado ambiente. Alguns não consideram esse tipo de aplicação como um problema verdadeiramente multi-agente, pois não utiliza totalmente alguns de seus conceitos básicos, como a coordenação entre os agentes.

Já Stone e Veloso (2000) colocam os SMA como um sub-campo da Inteligência Artificial Distribuída (DAI), que por sua vez, nasce da interseção de Computação Distribuída e Inteligência Artificial. Na Computação Distribuída, diversos processadores compartilham dados (mas não o controle) com foco em uma paralelização de baixo-nível ou em aspectos de sincronização. Na DAI, além dos dados, o controle também é distribuído, com foco na solução de problemas, coordenação e comunicação. Ainda, a DAI consiste em duas grandes linhas de pesquisa, havendo uma interseção entre elas. Na primeira, *Solução de Problemas Distribuídos* (SPD), a ênfase é na decomposição, compartilhamento e combinação de tarefas e dos resultados. A linha tracejada dentro de SPD demonstrando a existência do subcampo de SPD Cooperativos. Na segunda, os *Sistemas Multi-Agentes* (SMA), o foco é na coordenação do comportamento dos agentes. De uma forma resumida, enquanto SPD lida com o gerenciamento da informação, os SMA

PUC-Rio - Certificação Digital Nº 0610800/CA

estudam como gerenciar o comportamento. A interseção ocorre em problemas envolvendo múltiplos agentes que gerenciam informações.

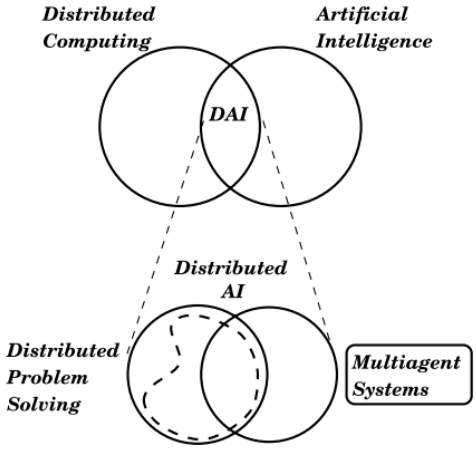

Figura 2: Taxonomia das áreas de pesquisa dentro da Inteligência Artificial Distribuída (DAI), de acordo com Stone e Veloso (2000).

Embora o comportamento dos agentes em um ambiente multi-agente possa ser pré-programado, na grande maioria das situações é necessário algum tipo de aprendizado on-line, que faça com que o desempenho dos agentes melhore de maneira gradativa (Weiss e Sen, 1999; Stone e Veloso, 2000).

O aprendizado por reforço pode ser usado para aprender um determinado processo de controle, que pode ser aplicado a agentes com o objetivo de obter o comportamento desejado. Diversos algoritmos de aprendizado por reforço, que apresentam propriedades consistentes, já vêm sendo usados em aplicações que funcionam através de um único agente. Porém, existem novos desafios na utilização de algoritmos de aprendizado por reforço em problemas englobando múltiplos agentes. Entre as principais dificuldades estão: a definição de um objetivo para vários agentes, a coordenação entre os comportamentos dos agentes e a escalabilidade dos algoritmos, que já é inclusive um problema em aplicações com apenas um agente.

Esta seção teve como objetivo fornecer uma revisão dos pontos mais importantes na área de SMA, com ênfase no funcionamento de técnicas e algoritmos baseados em aprendizado por reforço. As próximas seções estão organizadas da seguinte forma: a seção 2.4.2 explora os principais conceitos e taxonomias ligados aos SMA; a seção 2.4.3 explica o funcionamento de alguns dos algoritmos de *Reinforcement Learning* para múltiplos agentes mais usados e

PUC-Rio - Certificação Digital Nº 0610800/CA

seus desafios; e, finalmente, na seção 2.4.4, aplicações com múltiplos agentes são descritas.

2.4.2
Conceitos e Taxonomia

Existem diversos fatores que determinam a categoria de uma aplicação como SMA. Eles podem variar de acordo com o tipo da tarefa, ambiente, grau de homogeneidade do aprendizado, conhecimento a priori dos agentes, política de divisão do reforço, etc.

O ambiente de uma aplicação SMA pode ser estático, quando o ambiente permanece inalterado, ou dinâmico, onde mudanças no estado podem ocorrer ao longo do processo de interação entre os agentes e dos agentes com o estado.

Quanto ao tipo da tarefa envolvida no problema, existem três categorias principais (Busoniu, 2008). A primeira é a das aplicações *totalmente competitivas*, que sempre têm um vencedor e um perdedor no final. Neste tipo de problema, um dos princípios que pode ser aplicado é o do *minimax* (Von Neumann, 1920). O *minimax* é um método da teoria da decisão tradicional para "minimizar a máxima perda possível". Também pode ser visto como a estratégia de punir o outro jogador, minimizando o máximo que ele pode obter. Littman (1994) criou o algoritmo *minimax-Q*, que aplica o princípio do *minimax* para calcular as estratégias a cada estágio do aprendizado, e propaga os valores aos pares de estado-ação usando o algoritmo de *Reinforcement Learning* denominado *Q-learning (Watkins, 1989)*.

A segunda categoria é composta pelas aplicações *totalmente cooperativas*. Neste caso, o objetivo é maximizar o retorno comum dos múltiplos agentes. Neste tipo de aplicação, os retornos são positivamente correlatos. Ou seja, quando um agente recebe reforço positivo, os demais também recebem.

Finalmente, a terceira categoria é composta de problemas com tarefas mistas, onde nenhuma restrição é imposta aos retornos dos agentes. Este grupo ainda pode ser subdividido em algoritmos que não implementam a coordenação entre os agentes (*coordination-free*), como o *Team Q-learning* (Littman, 2001) e o *Distributed Q-learning* (Lauer e Riedmiller, 2000); os que são baseados em um

PUC-Rio - Certificação Digital Nº 0610800/CA

mecanismo de coordenação; e os que possuem uma forma de coordenação indireta.

Um dos pontos mais importantes dos SMA é a homogeneidade do aprendizado. O aprendizado dos múltiplos agentes pode ocorrer de três formas (Panait e Luke, 2005): homogêneo, com todos aprendendo da mesma forma; heterogêneo, onde cada agente aprende de uma forma distinta, aumentando o espaço de busca (Good, 2000); e híbrido, onde cada grupo de agentes aprende de uma forma homogênea, por exemplo, um time de futebol de robôs (Luke et al., 1997 ; Luke, 1998), sendo cada grupo de jogadores com uma especialização.

Existem também diferenças em relação às observações de um agente em relação aos demais: um agente pode precisar observar as ações de outros agentes; pode precisar observar as ações e recompensas de outros agentes; ou não precisar observar nem as ações nem as recompensas de outros agentes. Mesmo não havendo nenhum tipo de comunicação ou compartilhamento entre os agentes, é possível haver coordenação entre eles (Weiss e Sen, 1996).

Após a avaliação das necessidades de observação, outro fator de extrema relevância a ser definido é o grau de comunicação entre os agentes. De uma maneira geral, existem duas diferentes formas de comunicação entre os agentes: direta e indireta. Sendo realizada indiretamente, um agente central pode ter acesso a tudo e repassar aos demais; as informações podem ser colocadas em um "*blackboard*" que todos acessem; ou pode haver uma transferência implícita de informação, por exemplo, um agente deixando "pegadas" ou sinais que outro agente pode usar no aprendizado. As próprias ações realizadas pelos agentes também podem ser utilizadas para uma comunicação implícita de informações e a coordenação entre os agentes, como proposto por Tummolini et al. (2005) no modelo BIC (*Behavioural Implicit Communication*). Ocorrendo comunicação direta, os agentes trocam mensagens entre si, não havendo a necessidade de troca de informações implícitas ou através de "*blackboards*".

Combinando o grau de comunicação e a homogeneidade do aprendizado, Stone e Veloso (2000) sugerem uma definição dos principais grupos e o nível de complexidade de um SMA. Conforme mostra a figura 3, quanto maior a heterogeneidade (ou maior especialização) e o grau de comunicação, maior é a complexidade do sistema.

PUC-Rio - Certificação Digital Nº 0610800/CA

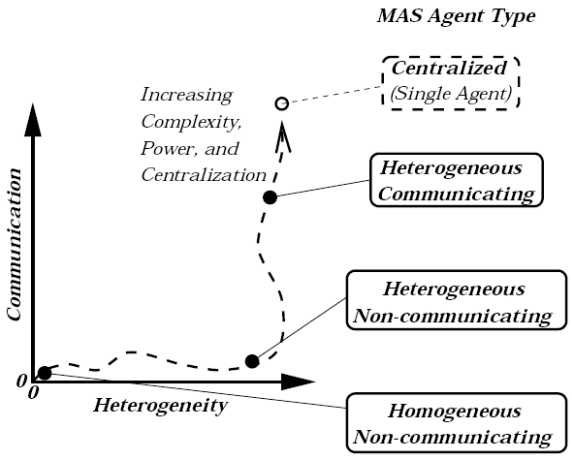

Figura 3: Categorias de SMA de acordo com o nível de comunicação e heterogeneidade dos agentes (Stone e Veloso, 2000).

Em relação ao nível de homogeneidade do aprendizado, o aumento da complexidade com a maior especialização ocorre porque cada agente ou grupo de agentes aprenderá utilizando uma dinâmica diferente.

Já, na parte de comunicação, quanto maior a quantidade de informações que o agente recebe sobre os demais agentes, maior é o seu espaço de busca. Para evitar o crescimento exponencial do estado de busca, a comunicação pode ser simplificada, por exemplo, através da criação de níveis de percepção sobre as ações e localizações dos demais agentes (Touzet, 2000), de aproximações de funções (Guestrin, 2002), etc.

Havendo a comunicação total das informações, um SMA pode chegar a ser equivalente a um sistema de grande complexidade, onde um agente único, ao invés de vários, toma todas as decisões e controla os demais como se fossem "escravos". Com isso, não há nenhum tipo de distribuição ou decomposição de tarefas, visando simplificar o trabalho dos agentes.

Para cada categoria, existem diferentes definições e técnicas de aprendizado que devem ser escolhidas na construção de um sistema com múltiplos agentes. Nos parágrafos seguintes, alguns aspectos gerais de sistemas multi-agentes serão destacados. Em seguida, as características de cada tipo de sistema, de acordo com o grau de comunicação e homogeneidade do aprendizado, serão analisadas separadamente.

PUC-Rio - Certificação Digital Nº 0610800/CA

2.4.2.1
Aspectos Gerais a todos os Sistemas Multi-Agentes

- Benevolência/Cooperação x Competitividade: um dos pontos básicos de definição do comportamento dos agentes é se eles irão ajudar um ao outro para chegar ao objetivo (benevolentes ou cooperativos) ou se irão competir entre si, considerando apenas seus próprios objetivos (competitivos). No extremo, podem ocorrer situações de "soma-zero", onde os agentes deverão se opor às estratégias dos demais para conquistar seus objetivos.

- Estabilidade x Evolução: também é necessário definir se os agentes são estáveis ou se evoluem suas estratégias. A evolução deve ocorrer apenas na presença de ambientes dinâmicos. Em aplicações onde há a competição com evolução, podem ocorrer uma corrida sem fim pela melhor estratégia, nunca permitindo a estabilização em um bom comportamento, ou ainda o problema do "*credit/blame assignment*", onde não é possível afirmar se o agente melhorou por causa de seu comportamento ou por causa do comportamento negativo de outro agente.

- Conhecimento a priori dos agentes: os agentes podem possuir ou não algum conhecimento a priori do modelo das tarefas que devem ser executadas. Considerando esse aspecto, Busoniu et al. (2008) classifica os algoritmos em *model-based learning*, quando o modelo de tarefa está disponível, e *model-free learning*, quando não há a disponibilização de modelo de tarefa a priori para os agentes.

- Divisão do Reforço (*Credit Assignment* ou *Reinforcement Sharing*): Em determinados problemas, geralmente de natureza cooperativa, o ambiente da aplicação não é suficientemente inteligente para definir individualmente o retorno de cada agente que faz parte de um mesmo time. Logo, se faz necessário a utilização de algum mecanismo de divisão do reforço (*Credit Assignment* ou *Reinforcement Sharing*). Entre as principais formas de divisão de reforço podem ser citados:

PUC-Rio - Certificação Digital Nº 0610800/CA

- *Global reinforcement* – reforço do "time" é dividido igualmente, devido às ações tomadas pelos agentes em conjunto, como se fosse uma inteligência coletiva (Wolpert e Tumer, 1999 e 2001).

- *Local reinforcement* – atualização de reforço feita somente com base no desempenho do próprio agente, com o objetivo de "desencorajar" os preguiçosos. Porém, isso faz com que os agentes não sejam incentivados a ajudar os demais.

- *Local reinforcement* + *social reinforcement*: reforço local pode ser melhorado com alguns tipos de "*reforço social*" (Mataric, 1994). Por exemplo:

→ *observational reinforcement* - pode ser obtido observando e imitando outros agentes;

→ *vicarious reinforcement* – recompensa indireta é obtida por um agente quando os outros são recompensados diretamente ("experiência de vida *indireta"*).

Harati et al. (2007) propõem o método KEBCA (*Knowledge evaluation-based credit assignment*), onde se utiliza a própria história e conhecimento do agente, através de medidas de "certeza", para a atualização do reforço. Essas medidas têm como objetivo avaliar e julgar as ações de cada agente para assinalar o crédito a elas da melhor maneira possível.

Abul et al. (2000) sugere uma simples combinação entre o retorno global e o retorno do agente em ambientes colaborativos. Na prática, cada agente recebe um retorno proporcional a sua contribuição para o retorno global.

$$r_{coop} = \alpha\, r + (1 - \alpha)\, \frac{\sum r_{\alpha}}{|\alpha|} \qquad (1)$$

Na equação 1, r é o retorno do agente, r_a é o retorno dos agentes observados e α ($0 < \alpha < 1$) é responsável pelo ajuste da relevância dos retornos.

PUC-Rio - Certificação Digital Nº 0610800/CA

Em Miyazaki e Kobayashi (2001), é proposto o modelo *Rationality Theorem of Profit Sharing*. Neste, quando um agente recebe uma recompensa direta, os demais agentes recebem uma recompensa indireta. Em Zhou e Hong (2006), a preocupação maior é com a racionalização do uso de memória. Eles sugerem um modelo de *On-line profit sharing algorithm*, que consiste na atualização baseada no conceito de *eligibility trace* (Sutton e Barto, 1998), com o objetivo de consumir menos memória.

2.4.2.2
Características dos Sistemas Homogêneos

2.4.2.2.1
Sistemas Homogêneos com Comunicação (com "Agente Central")

PUC-Rio - Certificação Digital Nº 0610800/CA

As aplicações onde os agentes possuem as mesmas características (homogêneos) e podem se comunicar diretamente e sem restrições muitas vezes são consideradas sistemas de um agente único. Neste tipo de sistema, um agente central armazena todas as informações e coordena as ações dos demais agentes, que não possuem nenhuma inteligência. Embora possa parecer que sistemas com apenas um agente inteligente sejam mais fáceis de compreender, a distribuição do controle permite a criação de agentes simples, o que pode ajudar na solução de problemas complexos.

Existem dois aspectos relevantes a serem considerados neste tipo de SMA:

- Nível de paralelismo desejado: um dos principais objetivos a serem alcançados neste tipo de SMA é a exploração paralela do ambiente. O nível de exploração em paralelo é definido pela quantidade de agentes no ambiente. Quanto mais agentes, maior será o volume de tarefas executadas em paralelo.

- Grau de Controle do Agente Central: o grau de controle do agente central sobre os demais agentes também pode variar. Na maioria das vezes, neste tipo de SMA a relação dos agentes com o agente central é do tipo mestre-escravo. Ou seja, o agente central coordena todas as ações e informações dos demais agentes.

Em algumas situações, apesar do controle do agente central, os demais agentes podem possuir algum tipo de inteligência própria.

2.4.2.2.2
Sistemas Homogêneos sem Comunicação

Diversos fatores precisam ser estudados e definidos em Sistemas Multi-Agente homogêneos sem comunicação. Entre os principais, estão:

- Agentes reativos x deliberativos: é necessário definir o grau de "raciocínio" dos agentes. Quando reativos, os agentes se limitam apenas a recuperar comportamentos já existentes de acordo com seus reflexos. Agentes deste tipo são aqueles que precisam fugir de obstáculos e, para isso, convertem os dados obtidos através de seu sensor (posições dos obstáculos e de outros agentes) em vetores de movimento (Balch e Arkin, 1995). Já, se deliberativos, os agentes assumem que cada um escolherá suas ações para concluir seus objetivos, podendo usar técnicas oriundas da teoria dos jogos para encontrar pontos de equilíbrio e decidir como agir (Levy e Rosenschein, 1992). Existe também a possibilidade de combinar "pensamentos" reativos e deliberativos.

- Perspectiva Local x Global: é importante definir o quanto de informação os agentes percebem do ambiente através de seus sensores. Em algumas situações, caso todos os agentes tenham uma perspectiva global e usem o mecanismo de aprendizagem, isso pode levá-los a escolher sempre as mesmas ações. Eventualmente, com menos informação, melhores resultados podem ser obtidos ("*A Ignorância é uma Benção*") (Durfee, 1995).

- Modelagem dos estados dos outros agentes: muitas vezes é necessário modelar o estado interno de outros agentes com objetivo de prever suas ações. Mesmo que os agentes conheçam os objetivos e estruturas dos demais, eles podem não conhecer suas futuras ações. A informação faltante é o estado interno (se deliberativo) ou os sensores de entrada dos outros agentes.

PUC-Rio - Certificação Digital Nº 0610800/CA

- Como afetar os outros agentes: Sem a possibilidade de comunicação, os agentes podem afetar os demais de forma direta ou indireta. Diretamente, eles podem ser percebidos pelos sensores alheios ou alterar o estado dos outros agentes. Indiretamente, eles podem afetar o resto do grupo através de dois tipos de *Estigmergia* (Holland, 1996). Na Estigmergia ativa, o agente altera o ambiente de forma que as mudanças sejam captadas pelos outros sensores. Na passiva, a alteração no ambiente ocorre com a finalidade de mudar o efeito da ação do outro agente. Por exemplo, se um agente desliga um quadro central de energia de um prédio, o efeito da ação de outro agente ligar a luz de um quarto no prédio é alterado indiretamente, uma vez que a luz não acenderá, já que o quadro de luz, que faz parte do ambiente, está desligado.

2.4.2.3
Características dos Sistemas Heterogêneos

De uma maneira geral, nos sistemas de natureza heterogênea, onde agentes podem possuir diferentes mecanismos de aprendizado, é importante a análise dos papéis e funções que os agentes deverão assumir, e de aspectos ligados a gestão de recursos.

- Papéis e Funções: quando os agentes possuem diferentes habilidades e mesmos objetivos, eles podem e devem ser organizados como um time, onde cada um desempenha suas funções específicas. Isso vale para sistemas com ou sem comunicação entre os agentes. Em ambientes dinâmicos, pode ainda ser necessário que cada agente tenha um determinado papel dependendo da situação em que se encontra. Uma das subáreas de pesquisa dentro de SMA é justamente como um agente pode assumir novas funções em ambientes em constante evolução.

- Gerenciamento de recursos: em um grande número de aplicações, agentes possuem recursos limitados para serem compartilhados. Por exemplo, em um problema de balanceamento de carga, vários computadores ou usuários possuem uma quantidade limitada de processamento que pode ser compartilhada (Schaerf et al., 1995). Havendo comunicação entre os agentes, também são

PUC-Rio - Certificação Digital Nº 0610800/CA

possíveis a coordenação do tempo das atividades e o controle de cronogramas (Decker e Lesser, 1995).

2.4.2.3.1
Sistemas Heterogêneos sem Comunicação

Não havendo a comunicação entre agentes em sistemas heterogêneos, algumas características precisam ser definidas durante a construção do SMA, como, por exemplo :

- Modelagem dos objetivos, ações e conhecimento dos outros agentes: como os agentes não são homogêneos, não basta modelar seus estados internos para prever suas ações. Com agentes heterogêneos, a previsão das ações só pode ser feita modelando os objetivos, ações e conhecimento dos demais, tornando o problema extremamente mais complexo e caro. A alternativa mais viável é tentar deduzir as ações dos agentes observando seus movimentos (Huber e Durfee, 1995; Wang, 1996).

- "Convenções sociais": mesmo não havendo comunicação entre os agentes, existem formas de se chegar a "acordos" ou escolhas coincidentes. O método do Ponto Focal (Fenster et al., 1995) explora o fenômeno cultural de preferências fazendo com que os agentes "se encontrem" sem nenhuma comunicação. Um exemplo prático foi dado por R. Vohra no "AAAI95 Fall Symposium on Active Learning". Ao perguntar para 40 pessoas onde eles se encontrariam se estivessem em Paris a procura de um amigo, 75% dos entrevistados disseram que iriam para a Torre Eiffel ao meio-dia.

2.4.2.3.2
Sistemas Heterogêneos com Comunicação

Por sua vez, nos sistemas heterogêneos com comunicação entre os agentes, precisam ser analisados os seguintes fatores :

PUC-Rio - Certificação Digital Nº 0610800/CA

- Padrões de Comunicação: obviamente, existe a necessidade de definição do protocolo que será usada na comunicação entre os agentes. Esses padrões para troca de informações podem ser criados especificamente para uma aplicação ou ser pré-definidos. Um determinado protocolo deve contemplar o conteúdo da informação (Genesereth e Fikes, 1992), formato das mensagens (Finin et al., 1994) e mecanismos de coordenação (Barbuceanu e Fox, 1995).

- Planejamento das ações de comunicação: é possível considerar que uma comunicação seja feita da mesma forma que uma ação qualquer. Ou seja, cada transmissão de informação pode possuir pré-condições e efeitos como uma ação teria. Em um SMA, o efeito da comunicação pode ter como objetivo a alteração da crença de um agente em relação ao estado de outro(s) agente(s). Além disso, Stone e Veloso (2000) também destacam que um agente pode fazer intencionalmente uma comunicação errada a outro agente, com o objetivo de atingir um objetivo. Ou seja, um agente pode simplesmente estar "blefando". Nestes casos, também é importante que o receptor da informação defina quando irá ou não assumir que uma determinada informação é verdadeira.

- Acordos e Compromissos: havendo comunicação entre os agentes, eles podem realizar acordos de cooperação com o objetivo de chegar a objetivos comuns. Os acordos e compromissos podem ser de vários tipos. Castelfranchi (1995) sugere 3 tipos: compromissos internos (o próprio agente se compromete a fazer algo), compromisso com outro agente e compromissos coletivos, onde o agente se dispõe a desempenhar uma determinada função. Agentes também podem usar os meios para se chegar a certos fins, de forma a planejar oportunidades de compromissos. Isso é contemplado no modelo BDI ("*Belief/Desire/Intention*") (Haddadi, 1995), que trata os possíveis "estados de comprometimento" como estados de planejamento, podendo ser uma cooperação potencial, um pré-compromisso ou um compromisso. Outro possível tipo de acordo é entre grupos de agentes, ao invés de acordos gerais ou entre dois agentes. Nestes casos, os agentes agem como se estivessem formando diversas coalizões (Zlotkin e Rosenschein, 1994).

PUC-Rio - Certificação Digital Nº 0610800/CA

A seção seguinte aborda os principais desafios e as famílias de algoritmos dentro da área de SMA, que funcionam a partir de técnicas de aprendizado por reforço.

2.4.3
Sistemas Multi-Agentes baseados em Reinforcement Learning

2.4.3.1
Principais Desafios

A aplicação de métodos de *Reinforcement Learning* tradicionais em "grandes ambientes", utilizando *lookup table* (tabela de armazenamento das funções de valor para espaço de estados pequeno ou discreto) é muitas vezes inviável, devido à grande dimensão do espaço de estados. Conforme já descrito na seção 1.1, esta limitação é conhecida como "*curse of dimensionality*". Em SMA, o problema do armazenamento dos valores para cada possível estado-ação é ainda maior, já que a memória necessária passa a crescer também exponencialmente com a quantidade de agentes envolvidos na aplicação (devem ser representados os valores dos pares de estado-ação de todos os agentes em conjunto).

A estabilidade na dinâmica de aprendizado do agente e a coordenação entre os agentes, enquanto ocorre a adaptação nas mudanças de outros agentes, também é um grande desafio. Ao longo da evolução do comportamento dos demais agentes, alguns valores de reforço de determinados pares de estados e ações podem ser alterados, gerando sinais confusos para o aprendizado do agente. Na prática, em um SMA, cada agente tenta aprender um objetivo "móvel", já que sua melhor política é alterada, conforme a política dos demais agentes muda. Isso pode fazer com que o aprendizado se torne não estacionário.

2.4.3.2
Técnicas de RL usadas em SMA

As seções seguintes exploram os métodos de Diferença Temporal e o Gradiente Descendente, que deram origem a diversos algoritmos de aprendizado

PUC-Rio - Certificação Digital Nº 0610800/CA

para multi-agentes. Em seguida, são relacionados os principais algoritmos SMA já desenvolvidos, suas características e suas origens.

2.4.3.2.1
Diferença Temporal

Um grande número de algoritmos de RL para multi-agentes deriva de métodos de Diferença Temporal (TD) (Sutton e Barto, 1998), sendo a grande maioria do algoritmo *Q-learning* (Busoniu, 2008). O *Q-learning* foi o primeiro método RL a possuir provas de convergência (Jaakkola et. al., 1994). É uma técnica muito simples que calcula diretamente as ações sem avaliações intermediárias e sem a necessidade de definições prévias. Nele, cada par estado-ação <*s,a*> é relacionado a um valor-Q, *Q(s,a),* que é o valor estimado para o estado *s* quando a ação *a* é executada, assumindo que a ação executada será uma ação ótima para todos os estados subseqüentes.

A função de valor-Q pode ser estimada on-line sem necessidade de aprender a política. O *Q-learning* é um método iterativo para aprendizado de ações. Ele é baseado no valor da ação (ou Q) e é definido por:

$$Q(s_t, a_t) = E[r(s_t, a_t) + \gamma V^*(s_{t+1})] \quad (2)$$

$$= r(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} | s_t, a) V^*(s_{t+1}) \quad (3)$$

Na equação acima, *r* é o retorno instantâneo recebido do ambiente, $V^*(s_{t+1})$ é o valor estimado para a função de valor do próximo estado, e γ é o fator de desconto que define a influência do valor associado ao estado s_{t+1} no valor atual estimado.

O valor Q da ação representa o valor esperado ao se executar a ação a_t quando o agente está no estado s_t (Ribeiro, 1999). A partir desta definição e como uma conseqüência do princípio de optimalidade de Bellman (1957), tem-se:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t [r(s_t, a_t) + \gamma \hat{V}(s_{t+1}) - Q(s_t, a_t)] \quad (4)$$

PUC-Rio - Certificação Digital Nº 0610800/CA

α é a taxa de aprendizado variando entre 0 e 1. A forma mais simples e intuitiva de transformar o algoritmo *Q-learning* em uma versão multi-agentes é evoluindo a equação acima para considerar *n* agentes, que conhecem as ações dos demais agentes, agindo ao mesmo tempo com as ações a_1, ... a_n. Considerando isso, a equação de atualização dos valores de Q para o i-ésimo agente passa a ser definida por:

$$Q^i(s_t, a_t^1, ..., a_t^n) = Q^i(s_t, a_t^1, ..., a_t^n + \alpha_t[r^i(s_t, a_t^1, ..., a_t^n) + \gamma \hat{V}(s_{t+1}) - Q^i(s_t, a_t^1, ..., a_t^n)] \quad (5)$$

Um problema que surge, neste caso, é a forma como *V* deve ser atualizado, dada a complexidade de um SMA (Claus e Boutilier, 1998; Yang e Gu, 2004). Para solucionar isso, deve ser escolhido um algoritmo que busque equilíbrio e convergência do sistema, entre eles, Nash-Q (Hu e Wellman, 1998), Minimax-Q (Littman, 1994), Hyper-Q (Tesauro, 2003), Team-Q (Littman, 2001), Distributed-Q (Lauer e Riedmiller, 2000), CE-Q (Greenwald e Hall, 2003), Asymetric-Q (Kononen, 2003), etc.

Por exemplo, supondo um jogo competitivo de soma-zero com 2 agentes A e B, executando passos alternados, o objetivo de A é aprender uma política que maximize o valor esperado dos reforços. Conforme dito no parágrafo anterior, esse aprendizado é difícil, pois depende das ações do agente B. No caso do algoritmo Minimax-Q (Littman, 1994), a solução proposta é avaliar cada política em relação à ação do outro jogador que a torna pior. Para viabilizar isso, a atualização do valor ótimo para o jogador A ocorre de acordo com as equações abaixo:

$$V^*(s) = \max_\pi \min_b Q(s, a, b) \quad (6)$$

$$\Delta Q_t(s_t, a_t, b_t) = \alpha[r_t + \gamma V_t(s_{t+1}) - Q_t(s_t, a_t, b_t)] \quad (7)$$

$$\Delta Q_t(s_t,\ a_t,\ b_t) = Q_{t+1}(s_t,\ a_t,\ b_t) - Q_t(s_t,\ a_t,\ b_t) \quad (8)$$

Nas equações, *r* é a recompensa resultante do jogador A tomar a ação *a* e B tomar a ação *b* no estado *s*, e γ é o fator de desconto.

PUC-Rio - Certificação Digital Nº 0610800/CA

Uma das estratégias mais usadas na maioria dos algoritmos para convergência é o Equilíbrio de Nash (Nash, 1950). Estratégias que fazem parte desse equilíbrio são simultaneamente as melhores respostas para todos os jogadores. Logo, não há incentivo para nenhum jogador desviar do equilíbrio, unilateralmente. Para saber se é equilíbrio de Nash, basta perguntar a cada jogador separadamente: mudando a sua estratégia você ficaria melhor? Quando todas as respostas de todos os jogadores forem negativas, então é um Equilíbrio de Nash. O grande desafio dos algoritmos é encontrar estratégias que estejam em Equilíbrio ao longo do jogo.

Um dos algoritmos que usa o conceito do Equilíbrio de Nash para a convergência é o Nash-Q (Hu e Wellman, 1998). Em seu trabalho, Hu e Wellman (1998) utilizam programação quadrática com este objetivo. Alguns pesquisadores questionam a usabilidade do Equilíbrio de Nash na convergência do aprendizado de múltiplos agentes. Por exemplo, Shoham et al. (2003) destacam a falta de clareza na ligação entre o estágio de conhecimento da convergência para equilíbrio de Nash e o desempenho na dinâmica do jogo. Já Bowling e Veloso (2001 e 2002) apresentam outra forma de convergência entres os agentes. Nela, um agente converge para a melhor resposta enquanto os outros agentes permanecem estacionários. Com isso, a convergência para Equilíbrio de Nash não é explícita. Porém, ela surge naturalmente se todos os agentes são 'racionais'.

Em Bowling (2004), é proposto o conceito de "no-regret" (sem arrependimento), que é uma forma de convergência alternativa à racionalidade. Usando este conceito, o agente que está aprendendo deve buscar retorno pelo menos tão bom quanto o retorno de qualquer estratégia estacionária. Assim, ele previne ser explorado por outros agentes.

Outra alternativa na busca pela convergência é o uso da predição e da racionalidade relacionadas à estabilidade e adaptação, respectivamente (Chalkiadakis, 2003). Predição pode ser considerada como a capacidade de aprender com certa precisão os modelos dos demais agentes. E um agente é considerado racional quando ele busca sempre maximizar seus retornos esperados, dado os modelos dos demais agentes. Ou seja, aprendendo o modelo usado pelos demais agentes, o agente ganha estabilidade. E sendo 'racional' ou buscando maximizar seus retornos, o agente se adapta ao ambiente.

PUC-Rio - Certificação Digital Nº 0610800/CA

2.4.3.2.2
Busca Direta – Métodos de Gradiente

Métodos que funcionam através da atualização de gradientes também vêm sendo usados em SMA. Um dos principais algoritmos, que serviu como base para o desenvolvimento de vários outros, é o IGA - *Infinitesimal Gradient Ascent* (Singh et al. 2000). No IGA, os agentes adaptam suas estratégias através da atualização de um gradiente ascendente, com uma taxa de aprendizado η decrescente.

O grande mérito do trabalho de Singh et al. (2000) foi mostrar que, se 2 jogadores jogam com o algoritmo IGA, com $\eta \rightarrow 0$, as estratégias convergem para o Equilíbrio de Nash ou a média dos *payoffs* no tempo converge para os *payoffs* esperados de um Equilíbrio de Nash.

Um dos vários algoritmos derivados do IGA é WoLF-IGA – *Win or Learn Fast IGA* (Bowling e Veloso, 2002 e 2003). Este algoritmo introduz o conceito de *Variable Learning Rate*, fazendo com que a taxa de aprendizado varie para aumentar ou reduzir a velocidade do aprendizado. De forma resumida, o objetivo é fazer com que o aprendizado seja mais rápido quando o agente está perdendo o jogo, e mais lento e cauteloso quando o agente está ganhando.

2.4.3.2.3
Resumo dos Algoritmos de RL para SMA

No quadro seguinte, alguns dos principais algoritmos multi-agentes RL estão relacionados ao tipo de tarefa que abordam e o ambiente.

PUC-Rio - Certificação Digital Nº 0610800/CA

Tabela 2: Principais algoritmos multi-agentes de RL por Tipo de Tarefa e Ambiente.

Algoritmo	Referência	Tipo da Tarefa	Ambiente
JAL	Claus e Boutilier (1998)	Cooperativa	Estático
FMQ	Kapetanakis e Kudenko (2002)	Cooperativa	Estático
Team-Q	Littman (2001)	Cooperativa	Dinâmico
Distributed-Q	Lauer e Riedmiller (2000)	Cooperativa	Dinâmico
OAL	Wang e Sandholm (2002)	Cooperativa	Dinâmico
Minimax-Q	Littman (1994)	Competitivo	Dinâmico
Ficticious Play	Brown (1951)	Misto	Estático
MetaStrategy	Powers e Shoham (2004)	Misto	Estático
IGA	Singh et al. (2000)	Misto	Estático
WoLF-IGA	Bowling e Veloso (2002)	Misto	Estático
GIGA	Zinkevich (2003)	Misto	Estático
GIGA-WoLF	Bowling (2004)	Misto	Estático
AWESOME	Conitzer e Sandholm (2003)	Misto	Estático
Hyper-Q	Tesauro (2003)	Misto	Estático
Sigle-Agent RL	Sen et al. (1994)	Misto	Dinâmico
Nash-Q	Hu e Wellman (1998)	Misto	Dinâmico
CE-Q	Greenwald e Hall (2003)	Misto	Dinâmico
Asymetric-Q	Kononen (2003)	Misto	Dinâmico
NSCP	Weinberg e Rosenschein (2004)	Misto	Dinâmico
WoLF-PHC	Bowling e Veloso (2002)	Misto	Dinâmico
PD-WoLF	Banerjee e Peng (2003)	Misto	Dinâmico
EXORL	Suematsu e Hayashi (2002)	Misto	Dinâmico

De uma maneira geral, os algoritmos podem utilizar técnicas de Diferença Temporal, busca direta de políticas ótimas e Teoria dos Jogos, podendo também funcionar através da combinação de mais de uma técnica. Por exemplo, os algoritmos Nash-Q (Hu e Wellman, 1998) e Minimax-Q (Littman, 1994) se originaram do algoritmo *Q-learning*, e incorporaram os conceitos relacionados à Teoria dos Jogos de Equilíbrio de Nash (Nash, 1950) e *minimax* (Von Neumann, 1920), respectivamente.

A figura abaixo mostra os algoritmos agrupados de acordo com o método utilizado e técnicas utilizadas por cada um (Busoniu, 2008).

PUC-Rio - Certificação Digital Nº 0610800/CA

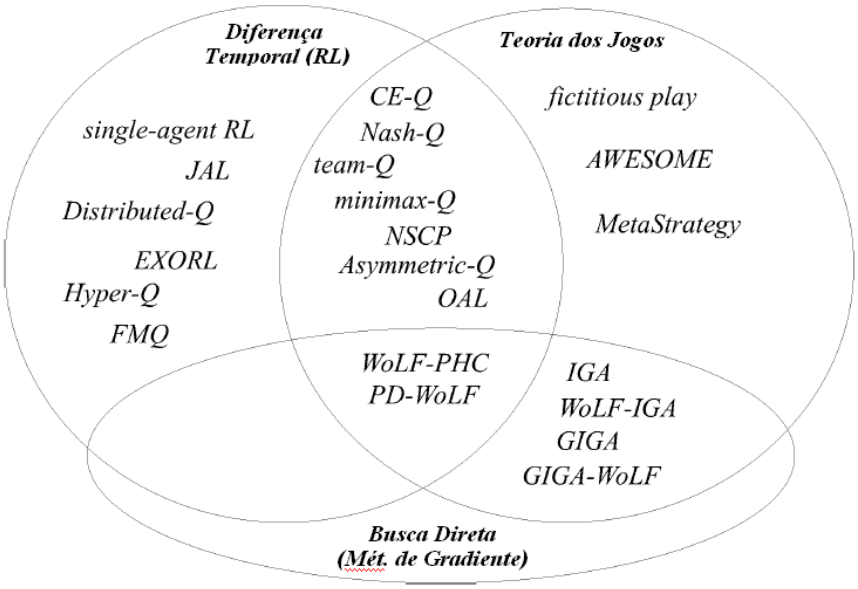

Figura 4: Resumo de algoritmos multi-agentes de RL e suas técnicas (Busoniu, 2008).

PUC-Rio - Certificação Digital Nº 0610800/CA

2.4.4
Aplicações de Sistemas Multi-Agentes

Conforme já apresentado na introdução deste capítulo, os sistemas Multi-Agentes vêm encontrando uma grande variedade de aplicações. A seguir, os principais campos de pesquisa são explorados com o objetivo de fornecer uma visão geral do potencial dos SMA.

2.4.4.1
Controle Distribuído

Em aplicações de controle distribuído, um conjunto de agentes autônomos interage em paralelo, através de ações de controle, com um mesmo objetivo. De maneira geral, sempre que o ambiente é um processo de controle e os agentes são os controladores deste processo, tem-se um problema de controle distribuído. São tipicamente aplicações que envolvem a cooperação entre os agentes, dado que o objetivo é o mesmo. Entre os principais problemas abordados estão: controle de processos (Stephan et al., 2000), controle de sinais de trânsito (Wiering, 2000; Bakker et al., 2005) e controle elétrico de redes de energia (Riedmiller et al., 2000).

2.4.4.2
Times Robóticos

A criação de times robóticos é considerada uma das aplicações mais exploradas dos SMA, por ser talvez a mais natural, instintiva e de compreensão geral. Além disso, vários pesquisadores que atuam na área de SMA também são ativos no campo da robótica. De maneira geral, o ambiente de jogos para os times robóticos possuem duas dimensões, podendo ser real ou simulado. Os agentes podem usar qualquer tipo de algoritmo para aprender a melhor forma de se comportar e adquirir habilidades.

Uma aplicação largamente explorada como benchmark é o jogo da presa/predador ou o "*pursuit-game*" (Benda et al., 1986). Neste jogo, vários agentes predadores devem capturar a presa convergindo ao seu redor (Ishiwaka et al., 2003; Kok et al., 2005; Xiao e Tan, 2007). Este jogo será mais detalhado posteriormente no capítulo de Estudo de Casos.

Há também problemas de controle de navegação, onde cada robô deve encontrar seu caminho a partir de um determinado ponto de origem, até o objetivo de destino fixo ou móvel, passando por obstáculos e evitando que a interferência de outros robôs o prejudique (Bowling e Veloso, 2002; Hu e Wellman, 2003; Calvo e Romero, 2007). Entre outras aplicações relacionadas à navegação estão: a localização e busca de objetos, cobertura de superfícies da forma mais ampla possível, e exploração de ambientes com o objetivo de captar o máximo de imagens através de sensores (Mataric, 1997).

Um dos jogos mais populares para SMAs e que merece uma atenção especial é o futebol robótico. Com o objetivo de incentivar e acelerar a pesquisa em SMA, foi criada a RoboCup (Kitano, 1995), ou Copa do Mundo de Futebol de Robôs, onde os jogos ocorrem através do *SoccerServer*, desenvolvido originalmente por Noda (1995). O grande desafio lançado por Kitano (1998) foi a "criação de um time de robôs capaz de derrotar um time real campeão do mundo" até o ano de 2050. Após algumas evoluções na versão original do SoccerServer, foi realizada em 1996, a preRoboCup-96, em Osaka, Japão, e, posteriormente, a RoboCup-97 em Nagoya, também no Japão. Desde então, a competição ocorre todos os anos com crescente interesse de pesquisadores e do público em geral. Atualmente, a competição possui diferentes categorias que permitem jogos em

PUC-Rio - Certificação Digital Nº 0610800/CA

ambientes simulados, através do *SoccerServer*, ou reais, através de robôs. A figura abaixo ilustra o ambiente usado para jogos simulados entre times de robôs.

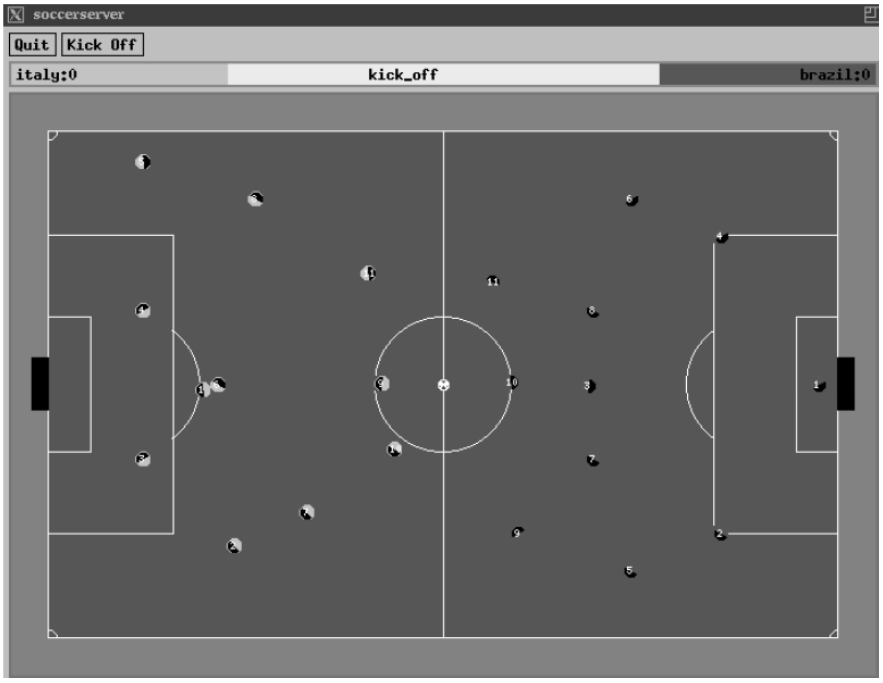

Figura 5: SoccerServer: servidor usado para jogos simulados entre times de robôs.

O simulador dos jogos é realista em diversos conceitos: a visão dos jogadores é limitada, os jogadores podem se comunicar através de mensagens colocadas em um "*blackboard*" visível a todos os demais jogadores, cada jogador é controlado por um processo distinto, cada time possui 11 jogadores, os jogadores possuem quantidades limitadas de energia, os sensores e ações têm ruídos, e o jogo ocorre em tempo real. As principais vantagens desta aplicação na avaliação de SMAs são (Stone e Veloso; 2000):

- Alta complexidade, o que o deixa bem próximo da realidade;
- Fácil acesso;
- Contempla a grande maioria dos conceitos envolvidos nos SMA, tanto para agentes homogêneos ou heterogêneos;
- Permite comparação direta entre times de robôs, independente da técnica utilizada por cada time.

PUC-Rio - Certificação Digital Nº 0610800/CA

2.4.4.3
Sistemas de Negociação e Trading

Existem diversos mercados eletrônicos que permitem a comercialização de produtos entre os agentes, seja através de negociações ou leilões. Um exemplo de *benchmark* para este tipo de aplicação é o *Trading Agent Competition*, concurso onde os agentes devem organizar viagens negociando bilhetes de passagens e reservas de hotel (Wellman et al., 2003).

Em algumas aplicações, os agentes, representando uma determinada empresa ou indivíduo, apenas cooperam em tarefas distintas do processo de negociação (Lee e Oo, 2002; Oo et al., 2002; Lee et al., 2007). A figura 6 ilustra um sistema de negociação multi-agente proposto por Lee et al. (2007), com quatro agentes, que desenvolvem capacidades específicas – gerar o sinal de compra, definir o preço de compra, gerar o sinal de venda e definir o preço de venda.

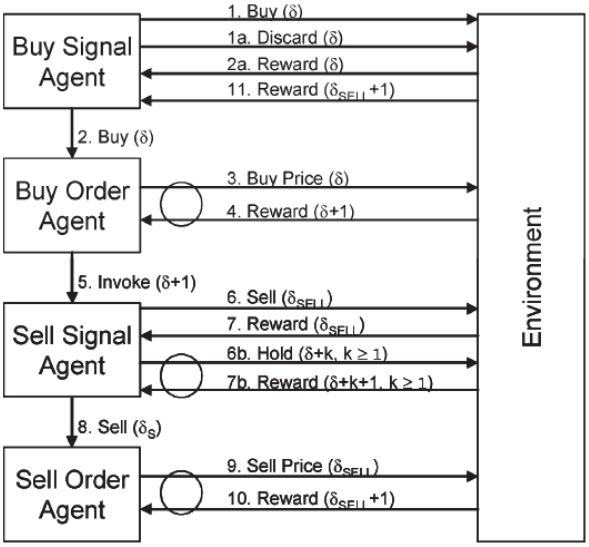

Figura 6: SMA de Negociação, onde cada agente possui uma função (Lee et al., 2007).

Em outros casos, os agentes autônomos buscam maximizar seus retornos interagindo em paralelo com os mercados (Hsu e Soo, 2001; Tesauro e Kephart, 2002; Raju et al., 2003).

PUC-Rio - Certificação Digital Nº 0610800/CA

2.4.4.4
Gerenciamento de Recursos

Quando possuem o objetivo de gerenciar recursos coletivos, os agentes sempre agem de forma cooperativa e possuem basicamente 2 tipos de comportamento (Busoniu, 2008):

- *gerentes dos recursos* - cada agente gerencia um recurso e aprende a melhor forma de responder a solicitações de uso deste recurso, sempre buscando otimizar uma determinada medida de desempenho, como em problemas de programação de elevadores (Crites e Barto, 1998);
- *clientes dos recursos:* os agentes aprendem a selecionar os recursos de maneira ótima, dada uma determinada medida de desempenho. Como, por exemplo, em aplicações de balanceamento de carga (Schaerf et al., 1995).

Exemplos de medidas de desempenho são: tempo de processamento de rotinas, tempo de espera por recursos, utilização e disponibilidade de recursos, distribuição justa de recursos aos clientes, etc.

Além das aplicações de programação de elevadores e balanceamento de carga, um dos campos de pesquisa mais popular no gerenciamento de recursos através de múltiplos agentes é o controle e planejamento de rotas (Boyan e Littman, 1993; Choi e Yeung, 1995; Tillotson et al., 2004).

No próximo capítulo, será apresentado o modelo *Reinforcement Learning Neuro-Fuzzy Hierárquico Politree* (RL-NFHP), proposto originalmente em Figueiredo (2003), que foi usado como base para o desenvolvimento deste trabalho. Serão explorados a célula básica, a arquitetura, as regras hierárquicas associadas e o algoritmo de aprendizado utilizado pelo modelo.

PUC-Rio - Certificação Digital Nº 0610800/CA