Pós-Graduação em Ciência da Computação

JOSÉ CARLOS DE MOURA JÚNIOR

"COLETA MULTIAGENTE DE RECURSOS EM JOGOS DE ESTRATÉGIA EM TEMPO REAL"

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

Recife
2009

JOSÉ CARLOS DE MOURA JÚNIOR

"COLETA MULTIAGENTE DE RECURSOS EM JOGOS DE ESTRATÉGIA EM TEMPO REAL"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: Dr. GEBER RAMALHO

Recife
2009

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

M929c Moura Júnior, José Carlos
Coleta multiagente de recursos em jogos de estratégia em tempo real / José Carlos Moura Júnior. – 2009.
53 f.: il., fig., tab.

Orientador: Geber Lisboa Ramalho.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2009.
Inclui referências.

1. Inteligência artificial. 2. Sistemas multiagente. 3. Jogos de estratégia. I. Ramalho, Geber Lisboa (orientador). II. Título.

006.3 CDD (23. ed.) UFPE- MEI 2018-095

Dissertação de Mestrado apresentada por **José Carlos de Moura Júnior** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Coleta Multiagente de Recursos em Jogos de Estratégia em Tempo Real**", orientada pelo **Prof. Geber Lisboa Ramalho** e aprovada pela Banca Examinadora formada pelos professores

_____

Profa. Patricia Cabral de Azevedo Restelli Tedesco
Centro de Infromática/ UFPE

_____

Prof. Clauirton de Albuquerque Siebra
SAMSUNG / CIN

_____

Prof. Geber Lisboa Ramalho
Centro de Infromática/ UFPE

Visto e permitida a impressão.
Recife, 27 de agosto de 2009.

_____

Prof. FRANCISCO DE ASSIS TENÓRIO DE CARVALHO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco

Agradecimentos

Primeiramente, agradeço a Deus por tudo que tenho alcançado, gostaria de agradecer também ao meu anjo da guarda que Deus mandou a terra para cuidar de mim a minha Mãe, minha Lady Laura. Gostaria de agradecer também o meu pai, meu querido, meu velho, meu amigo, que acompanhou todo o meu esforço no desenvolvimento deste trabalho, mas infelizmente Deus resolveu chamá-lo antes que a gente pudesse comemorar a conclusão.

Gostaria de agradecer também a Alice minha namorada, que sempre foi compreensiva e me ajudou muito no desenvolvimento deste trabalho, e as outras mulheres da minha vida, minhas irmãs Adriana, Janaína e Ceça.

Agradeço também a Geber Ramalho, pela oportunidade de ser orientado por ele, e por todo o seu apoio. Obrigado também ao apoio que obtive dos meus colegas de trabalho, Fred, Rui, Vicente, Renan, Leo, Tulio, Allan, Alex, Diego, e um agradecimento especial a Heloíse que fez a revisão do meu trabalho.

Agradeço aos membros da banca Clauirton e Patrícia, nesse momento tão importante para mim é bom ter rostos familiares por perto, e vocês desde a graduação me apoiaram sempre que precisei.

Resumo

Uma tarefa bastante comum em jogos de estratégia de tempo real (RTS) é a coleta de recursos (por exemplo, ouro, minério, madeira ou alimentos).Ela é de grande importância para o desenrolar do jogo, pois a quantidade de recursos coletados vai determinar a velocidade de crescimento e evolução dentro do jogo. Essa tarefa normalmente é executada por vários Non Player Characters (NPC), que são os personagens não controlados pelo jogador, implementados como agentes inteligentes, que após uma designação inicial deve ser capaz de manter-se executando tal ação sem a necessidade de ações do jogador. Tal tarefa se torna complicada, e com isso interessante, devido a necessidade da coordenação entre os agentes, para que eles possam trabalhar de maneira cooperativa e não competitiva. Nesse trabalho, será apresentado um estudo sobre várias arquiteturas de sistemas multiagentes dedicadas a coleta de recursos. Serão apresentadas as vantagens e desvantagens de cada abordagem em relação às demais, segundo os critérios apresentados, e como elas contribuem para o objetivo da cooperação.

Palavras-chave: Sistemas multiagentes. Coleta de recursos. Jogos. RTS.

Abstract

One common task in Real-Time Strategy Games (RTS) is resource gathering (for instance, gold mining or gas extraction). It has a big importance in the development of the strategy, because the amount of resources, and the gathering velocity is going to impact how fast can a society grown, and evolve, since all the tasks in the game are going to spend this resources. This task is usually performed by some Non Player Characters (NPCs), which are implemented as intelligent agents, these agents should be capable of maintain their activities once his role is defined, without the need of actions from the player. In order to maximize resource gathering by a team of agents, coordination is required. In this work, we present an experimental work concerning the study of various architectures of multiagent systems devoted to resource gathering. The pros and cons of each architecture are discussed according to some evaluation criteria we have proposed.

Keywords: Multi Agent System. Resource gathering. Foraging. Games. RTS.

Lista de Figuras

Lista de Tabelas

Principais Abreviaturas

RTS	*Real Time Strategy*
IA	*Inteligêmcia Artificial*

Sumário

1 Introdução

Este capítulo relata as principais motivações para realização deste trabalho, lista os objetivos de pesquisa almejados, e, finalmente, mostra como está estruturado o restante da presente dissertação

1.1 Motivação

Jogos de estratégia de tempo real (RTS) é um gênero muito popular, muito interessante para ser estudado pela Inteligência Artificial, por conta da grande variedade de problemas e desafios que tal tipo de jogo pode oferecer. O gênero RTS foca no planejamento cuidadoso e na habilidade de gerenciar recursos de maneira a alcançar a vitória. Os jogos desse gênero são jogos nos quais a estratégia deve ser montada e executada a cada instante, em tempo real. Os jogos RTS são excelentes aplicações em termos de IA por exigirem coordenação multiagente, tomadas de decisão multi-critério e combinação de decisões estratégicas e planejamento com decisões táticas e reatividade. [Schwab 2004]

Um problema desafiador e interessante dos jogos RTS é a coleta de recursos (gathering), uma tarefa executada por um sistema multiagente que para conseguir seus objetivos necessita uma coordenação, uma vez que todos os agentes compartilham o mesmo ambiente e recursos.

O problema analisado aqui neste trabalho é parte de uma competição que ocorre desde 2006 em AIIDE 2006 (Artificial Intelligence and Interactive Digital Entertainment). Nessa competição os participantes deveriam desenvolver um grupo de agentes que deveriam coletar recursos espalhados em um cenário, e levá-los para um ponto central (o Control Center), exatamente como a coleta de recursos em jogos RTS [ORTS 2003].

A coleta de recursos multiagentes é uma área com muitos estudos e análises, entretanto no contexto de jogos RTS, é um problema ainda em aberto. Apesar dos vários estudos na área de coleta multiagentes, nenhuma das soluções encontrada na literatura pode ser vista como a solução definitiva para este tipo de problema. Em outras palavras, a coleta de recursos de

jogos RTS é um problema que precisa ser mais explorado e estudado dadas as grandes possibilidades existentes.

1.2 Objetivos

O objetivo principal deste trabalho de mestrado é fazer um estudo sistemático e metodológico do problema da coleta multiagente de recursos em jogos de estratégia de tempo real. Em outras palavras, o objetivo desse trabalho não é criar uma solução definitiva para o problema da coleta de recursos, mas sim fazer um estudo das diferentes arquiteturas de sistemas multiagentes e suas vantagens e desvantagens, entendendo dessa maneira as suas propriedades em relação à coleta multiagentes de recursos.

1.3 Metodologia

Para realizar tal estudo, o primeiro passo foi realizar um amplo estudo sobre coleta de recursos, jogos RTS e a aplicação de Inteligência Artificial em seu domínio.

Todo este estudo tinha como objetivo a definição formal do tipo de coleta multiagente de recursos, uma vez que tal problema pode ser formulado de várias maneiras, assim como a definição exata do ambiente onde ele deve ocorrer que pode ter muitas variações.

Com a definição formal do tipo de coleta multiagente de recursos e a descrição exata do ambiente, tínhamos a definição do problema. Então o próximo passo foi estudar as soluções já existentes na literatura, e fazer uma análise delas, e se elas poderiam ser aplicadas no problema.

O próximo passo foi definir os parâmetros a serem avaliados, e quais abordagens seriam propostas para resolver cada um dos subproblemas. Em seguida foram criadas as combinações possíveis das diferentes abordagens, gerando os possíveis agentes a serem analisados. A escolha destas arquiteturas começou das mais simples para as mais complexas, visando mostrar a variação de comportamento à medida que suas complexidades iriam aumentando.

Com as abordagens escolhidas e os agentes definidos, foi feita então a escolha do ambiente onde a análise seria executada, o simulador, desenvolvido em parceria como trabalho de dissertação de mestrado de outro aluno. Em seguida foram implementadas todas as abordagens e definidos os mapas e as regras das simulações. Com o resultado das simulações então foi feita uma análise individual e comparativa dos agentes no ambiente simulado.

Como tal estudo não havia sido realizado este trabalho serve como contribuição original e útil para o desenvolvimento de sistemas multiagentes no domínio da coleta de recursos de jogos RTS.

1.4 Estrutura da dissertação

O restante da dissertação está organizado da seguinte maneira. O capítulo dois visa à definição do tipo da coleta de recursos a ser utilizada, para isso explicando a definição de jogos RTS, a importância da coleta de recursos nesse ambiente. Também é mostrado como a Inteligência Artificial já é utilizada nesse tipo de jogo, e os níveis de decisão que ela pode tomar.

O capítulo 3 utiliza as definições do capítulo anterior para definir o problema e utiliza tal definição para analisar as diferentes soluções existentes no estado da arte apresentado neste capítulo.

O objetivo do capitulo 4 é a criação dos parâmetros a serem analisados e quais implementações iram cobrir cada uma destas abordagens, e posteriormente a apresentação dos cenários, os critérios de avaliação que as abordagens serão submetidas, e uma breve descrição sobre o simulador onde serão implementados os agentes.

O capítulo 5 apresenta os cenários, os agentes gerados a partir das combinações das diferentes abordagens e os resultados das simulações. Posteriormente é feita uma análise individual de cada agente em seguida uma análise comparativa.

O capítulo 6 apresenta as considerações finais deste trabalho, assim como sua importância e contribuição além dos trabalhos futuros.

2 Coleta Multiagente de Recursos

O objetivo deste capítulo é descrever o que é coleta multiagente de recursos e explicar seus subproblemas. No decorrer deste capítulo será visto como ocorre a coleta multiagente de recursos em sistemas multiagentes e os problemas e benefícios da utilização de vários agentes. Posteriormente será feita uma explanação sobre RTS (Jogos de Real Time Strategy), o uso da Inteligência Artificial nesses jogos, como funciona a coleta multiagente de recursos nesses jogos e sua importância estratégica.

2.1 Coleta Multiagente de Recursos

Coleta multiagente de recursos, também conhecido como *foraging* ou *gathering*, consiste em procurar por recursos e coletá-los dentro de um determinado ambiente. O foraging tem várias características que o descrevem. Abaixo vão ser descritas as principais características que definem o tipo dele:

- Mono Agente X Multiagente: no mono-agente, como o próprio nome diz, um único agente é responsável pela coleta, enquanto no Multiagente um grupo de agentes é responsável pela coleta e o foraging pode ser feito em menor tempo, já que a coleta pode ser paralelizada. Entretanto é preciso haver coordenação entre os agentes para evitar que um atrapalhe o outro.
- Depósito Central X Vários Depósitos: refere-se à quantidade de pontos de depósito à disposição dos agentes. No caso do central, tudo que for coletado precisa ser levado para um único ponto. Se os agentes tiverem mais de um ponto de depósito, então possuem vários depósitos.
- Recorrente X Único: único, siginifica que o agente só precisa visitar um determinado recurso uma única vez, enquanto no recorrente o mesmo ponto tem que ser visitado várias vezes.

Além de descrever a coleta é preciso também definir o ambiente onde os agentes estarão inseridos. Para isso tomou-se como base as classificações vistas por Russel e Norvig [Russel 2003]

- Discreto X Contínuo: em relação à quantidade de ações e percepções de um agente, no ambiente discreto a quantidade é limitada.
- Completamente observável X Parcialmente observável: no completamente observável o agente tem informação perfeita do mundo, no parcialmente algumas informações do mundo não são acessíveis.
- Determinístico X Estocástico: se o próximo estado do mundo é completamente determinado pelo estado atual e as ações do agente. Estocástico quando o próximo estado do mundo for determinado por algo além do estado atual e as ações do agente.
- Episódico X Seqüencial: episódico refere-se ao ambiente em que a tarefa pode ser divido em tarefas menores e mais simples.
- Estático X Dinâmico: no ambiente dinâmico o ambiente pode mudar enquanto o agente raciocina, no ambiente estático o ambiente espera o agente.

2.2 Motivação

A coleta multiagente de recursos possui vários subproblemas interessantes de serem estudados pela Inteligência Artificial, como alocação de recursos, pathfinding e coordenação multiagente. É uma série de problemas que já possuem soluções isoladas na literatura, entretanto quando todos são parte de um problema maior, a solução de um passa a impactar na outra solução, de maneira que as melhores soluções individuais não necessariamente compõem a melhor solução do problema maior.

Suas aplicações vão além de jogos de estratégia de tempo real, a coleta de recursos pode acontecer fora do mundo dos jogos, por exemplo, um grupo de robôs que deve coletar objetos em um ambiente inseguro para seres humanos, coletar amostras de rocha em Marte. Outro exemplo seria um grupo de robôs encarregado de manter um ambiente limpo, coletando as sujeiras que possam aparecer e levar para o lixeiro.

2.3 RTS

Estratégia de tempo real (RTS, do inglês Real Time Strategy), é um gênero de jogo, normalmente caracterizados por serem jogos de guerra, em que a disputa acontece em tempo real. [RTS 2009]

Nesse tipo de jogo o jogador tem controle total sobre suas unidades e ele é responsável direto pela tática e estratégia a ser seguida. Através dessas unidades é que o jogador vai poder interagir com o mundo Tais unidades podem ser um prédio, um tanque de guerra, ou um mero aldeão. As principais ações a serem executadas pelo jogador que é comum em todos os jogos RTS são as seguintes:

- Coletar Recursos: o jogador necessita de unidades especiais que fazem a coleta de recurso, e precisa que essa coleta seja feita de forma otimizada, pois tais recursos serão consumidos para ele executar as tarefas de construir edificações, treinar unidades e desenvolver tecnologia. A variedade de recursos pode variar de jogo para jogo. Em Command & Conquer, por exemplo, o usuário coleta um único tipo de recurso (minério), enquanto em Age Of Empires II, ele necessita coletar 4 tipos de recursos (ouro, pedra, madeira e comida).

- Construir edificações: para treinar as suas unidades e desenvolver tecnologias, o jogador precisa construir edificações, onde tais ações serão executadas. Cada tipo de edificação tem um propósito diferente. Uma serve para treinar unidades do tipo A, outra para treinar unidades do tipo B, uma terceira edificação serve para evoluir a unidade do tipo B para o tipo C.

- Treinar unidades: o jogador precisa treinar unidades para combater o exército inimigo, e como foi visto anteriormente cada uma dela é treinada em uma edificação diferente. Unidades podem ser navios, canhões, trabalhadores, entre outros, e cada unidade tem características diferentes e por isso finalidades diferentes. Uma, por exemplo, coleta madeira, enquanto outra atira flechas. Cabe ao jogador decidir qual unidade treinar em função de sua tática/estratégia.

- Desenvolver tecnologia: Também conhecida como evolução, ela é necessária nesse tipo de jogo para que o jogador tenha maiores chances no combate com o seu inimigo, pois o desenvolvimento de tais tecnologias vai tornar as suas unidades melhores. Ele pode evoluir uma unidade para que ela tenha um poder de destruição maior, ou para que ela se movimente mais rápido. É possível até evoluir edificações para que elas tenham maior resistência contra ataques inimigos.

- Combater exército inimigo: Todas as ações anteriores são feitas visando o combate com o inimigo, pois o objetivo principal desse gênero de jogo é destruir todas as unidades inimigas. No combate o jogador faz decisões tanto do nível

estratégico, como decidir se vai atacar ou não, quanto no nível tático, onde ele decide que tipo de formação dar ao seu exército ou onde cada unidade deve atacar.

Além de todas essas tarefas, o jogador precisa tomar decisões no nível estratégico, como decidir se é melhor atacar naquele momento, ou se é melhor se defender e coletar mais recursos para treinar mais unidades e fazer um ataque mais forte posteriormente. Ele precisa saber também como gerenciar os recursos da melhor maneira possível, saber se naquele momento é mais importante construir 10 unidades do tipo A, ou construir apenas 5 e utilizar o restante dos recursos para evoluí-las, tornando-as mais fortes.

Em um jogo RTS o jogador precisa também tomar decisões em um nível mais baixo, no nível tático. Por exemplo, no nível estratégico ele toma decisões como atacar ou defender, enquanto no nível tático, ele vai definir como vai atacar, qual a formação que as suas unidades vão ter, se vão em bando, ou isolados, decisões deste tipo.

2.4 IA EM RTS

Inteligência Artificial em jogos é qualquer implementação que passe uma noção de inteligência em algum nível, tornando o jogo mais imersivo, desafiador e o mais importante, divertido. A IA é utilizada em duas situações. Na primeira ele executa alguma tarefa para o jogador humano, como calcular a rota das suas unidades. Na segunda situação a IA controla os personagens que não são controlados pelo jogador humano, simulando um jogador adversário por exemplo. Dessa maneira a indústria de jogos vem evoluindo suas técnicas e utilizado-as em seus jogos, como uma maneira de atrair mais jogadores. A tabela a seguir mostra como a IA é utilizada em jogos desde muito tempo atrás.

Ano	**Descrição**	**IA utilizada**
1962	Primeiro jogo de computador, Spacewar, para 2 jogadores.	Nenhuma
1972	Lançamento do jogo Pong, para 2 jogadores.	Nenhuma
1974	Jogadores tinham que atirar em alvos móveis em Pursuit e Qwak.	Padrões de Movimento
1975	Gun Fight lançado, personagens com movimentos aleatórios.	Padrões de movimento

1978	Space Invaders contém inimigos com movimentos padronizados, mas também atiram contra o jogador.	Padrões de movimento
1980	O jogo Pac-man conta com movimentos padronizados dos inimigos, porém cada fantasma (inimigo) tem uma "personalidade" sobre o modo em que caça o jogador.	Padrões de movimento
1990	O primeiro jogo de estratégia em tempo real, Herzog Wei, é lançado. Junto, os jogadores puderam noticiar uma péssima busca de caminho.	Máquina de estados
1993	Doom é lançado como primeiro jogo de tiro em primeira pessoa.	Máquina de estados
1996	BattleCruiser: 3000AD é publicado como o primeiro jogo a utilizar redes neurais em um jogo comercial	Redes neurais
1998	Half-Life é lançado e analisado como a melhor IA em jogos até a época, porém, o jogo utiliza IA baseada em scripts.	Máquina de estados / Script
2001	O jogo Black & White é alvo da mídia a respeito de como as criaturas do jogo aprendem com as decisões feitas pelo jogador. Utiliza redes neurais, reinforcement e observational learning.	Diversos

Tabela 1 - Historia da IA em jogos

Em jogos de estratégia de tempo real (RTS), é possível distinguir diversos módulos de Inteligência Artificial. Um dos mais básicos pode ser um sistema de path-finding, que vai muito além de descobrir um caminho entre o ponto A e o B. Este sistema precisa ser eficiente ao ponto de calcular as rotas de varias unidades em uma fração de segundos, conseguindo obter a melhor rota e evitar colisões entre os caminhos.

Existem também sistemas de IA que tratam de problemas em um nível mais alto, como módulos responsáveis pela economia, desenvolvimento, ou até um módulo responsável pela análise de terreno, presente em praticamente todos os jogos RTS, que vai auxiliar em decisões como onde construir edificações específicas, de maneira que elas não fiquem vulneráveis. Baseado em um mapa de influência gerado pela análise de terreno, um módulo de IA responsável pelo combate pode determinar qual a melhor formação do exército.

Os sistemas de IA de um jogo RTS podem resolver problemas específicos como coleta de recurso, considerando tendo informação perfeita do mundo e um grupo de agentes autônomos qual a melhor maneira deles se organizarem e coletarem a maior quantidade de recurso no menor tempo possível.

Os jogos *Dark Reign: The Future Of War*, *Battlezone* e *Civilization: Call To Power* são bons exemplos de como a IA pode ser dividida em módulos. Nesses jogos a IA é dividida em duas tarefas principais, IA tática e IA estratégica, que posteriormente vão ser divididas em três partes: Análise, Alocação de Recursos e IA de Alto Nível (personalidade). Cada um desses módulo pode ser implementado utilizando maquinas de estado, sistemas baseados em regra ou lógica fuzzy. [Davis 1999]

IA Tática, entre várias tarefas, define, por exemplo, que cada unidade vai verificar se existe alguma unidade inimiga nas redondezas, e quando encontrar cada unidade escolhe qual a melhor unidade para atacar e o faz. A melhor unidade a ser atacada é definida maximizando quanto de poder de fogo do inimigo eu posso retirar o mais rapidamente. Desta maneira as unidades mais fortes acabam destruindo as unidades mais fracas primeiramente, e dessa maneira é garantido que unidades que não sejam de combate só vão ser atacadas quando não houver mais unidades de combate.

IA Estratégica, como o próprio nome diz, toma decisões de nível estratégico com o uso de informações adquiridas através de outros módulos como de análise, alocação de recurso e módulo de alto nível.

Análise: O objetivo do modulo de analise é definir os objetivos estratégicos atuais e classificá-los por prioridades. Os objetivos estratégicos pode ser exploração, reconhecimento, construção de base, objetivos defensivos e objetivos ofensivos.

Alocação de Recursos: Cada recurso pode ser um trabalhador, um arqueiro, um prédio, ou uma quantidade de ouro por exemplo. Um objetivo pode ser treinar uma quantidade grande de uma unidade. Como mostra a figura abaixo, cada tarefa pode necessitar de mais de um recurso, e cada tarefa tem uma prioridade diferente. Então elas são organizadas de maneira decrescente e as tarefas são realizadas primeiramente.

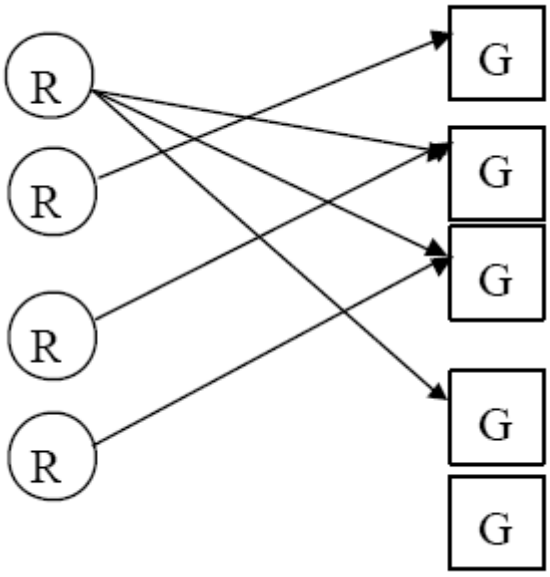

Ilustração 1 - Alocação de Recursos

Alto Nível: Como o próprio nome diz esse módulo têm uma visão mais alto nível do problema, podendo coordenar as tropas como uma espécie de general. Ela pode tomar decisões do tipo formar alianças.

Como foi dito anteriormente, o objetivo da IA em jogos RTS é tornar o jogo mais imersivo e divertido. Porém, alguns jogos para impor um nível de dificuldade mais alto para o jogador acabam 'roubando', ou seja, suas unidades coletam ouro mais rápido que as unidades do jogador, ou eles têm informações que se fosse um jogador humano não teria. Quando isso transparece para o jogador, o usuário final, o jogo acaba perdendo a diversão (Davis 1999). A construção de técnicas eficientes de IA tem por objetivos conseguir um desempenho excelente sem infringir as regras.

2.5 Coleta multiagente de recursos em RTS

Numa partida de RTS o primeiro problema encontrado pelo jogador é o foraging, ou a coleta de recursos. O jogador precisa coletar recursos para poder construir edificações, evoluir tecnologicamente, treinar novas unidades, etc. Equipes com uma coleta de recurso mais eficiente vão ter conseqüentemente maior quantidade de recursos e maior vantagem na batalha.

A coleta de recurso é a base da economia de uma sociedade em um jogo RTS, tanto que algumas táticas de combate se baseiam no foraging seja para angariar recursos o mais rápido possível para si, ou para fragilizar o inimigo impedindo que ele colete recursos.

O *Raiding*, por exemplo, é uma tática de combate que visa um ataque rápido aos pontos de coleta de recursos do inimigo, uma vez que as unidades que coletam recurso têm pouco poder de combate. Raramente ganha a batalha, mas desequilibra a economia do inimigo, tornando o mais frágil, facilitando um futuro ataque a sua base.

O *Fast-Heroic* é uma tática que se baseia na evolução tecnológica mais rápida possível, pois, quanto mais evoluída for uma sociedade, mais fortes serão suas unidades. Para conseguir os recursos necessários para a evolução rápida, não são construídas unidades de combate. Somente as unidades de coleta são construídas de maneira que elas possam coletar os recursos necessários o mais rápido possível permitindo assim a evolução.

O foraging é um problema complexo, pois ele é composto de vários subproblemas que apesar de possuírem soluções individuais excelentes, podem possuir soluções conflitantes no problema maior. Por exemplo, a alocação de recursos pode dizer que o agente deve coletar o recurso mais próximo do control center. Tal solução funciona de maneira excelente em um ambiente monoagente, mas em um ambiente multiagente, se todos forem nos recursos mais próximos um agente vai começar a atrapalhar o outro. Além disso, é necessário utilizar a

capacidade máxima dos agentes, fazendo com que eles carreguem o máximo de recursos. Deste modo, é preciso coletar o máximo de recurso no menor tempo possível.

O foraging que ocorre nos jogos RTS, e também o usado na competição, pode ser descrito como Multiagente Depósito Central Recorrente Foraging e possui um ambiente completamente observável, determinístico, seqüencial, dinâmico e contínuo. Multiagente, pois o fato de serem vários agentes é crucial para a determinação da solução, pois é preciso fazer o a coordenação entre eles, fazendo que o trabalho do grupo seja maior que a soma do trabalho das partes. Central Place, pois os recursos coletados devem ser levados para um ponto central e lá serem depositados. Recorrent, porque o agente precisa ir à mina várias vezes coletar recursos, se abastecer e levar para a base, até que a mina se esgote. O ambiente é completamente observável uma vez que o agente possui todas as informações sobre o mundo, determinístico, pois o próximo estado do mundo depende unicamente do estado atual do mundo e das suas ações, seqüencial, pois a coleta de recursos é uma tarefa que não pode ser quebrada em episódios, tal ambiente é dinâmico, pois o ambiente está sempre se atualizando não esperando pelo raciocínio do agente e contínuo, pois a velocidade dos agentes e o espaço onde o jogo ocorre são contínuos.

O problema para o qual esse trabalho se propõe a desenvolver uma abordagem foi visto em uma competição que ocorreu em Junho de 2006 na AIIDE 2006 (Artificial Intelligence and Interactive Digital Entertainment). Na competição o usuário precisava programar um grupo de agentes que tinham de coletar recursos espalhados por um cenário e levá-los para um centro, simulando o problema da coleta de recurso dos jogos RTS.

A coleta de recursos pode ser dividida em dois subproblemas: Alocação de Recursos (“Quem vai aonde?”) e Path Finding (“Como cada um chega lá?”), que posteriormente podem ser divididos em subproblemas menores e variações.

A alocação de recursos vai definir que recurso cada agente deve coletar, e tal decisão pode ser definida com base em vários parâmetros, por exemplo:

- Custo para chegar até o recurso: a que distância do ponto de depósito está o recurso, os mais interessantes são aqueles mais próximos ao ponto de depósito visto que levaria menos tempo para poder ir ao recurso, coletar e voltar ao ponto. Porém não se pode considerar apenas a distância em linha reta, pois o caminho disponível entre eles pode não ser o mais curto. Além disso, o caminho mais curto nem sempre pode ser o menos custoso, é preciso verificar se ele possui obstáculos, se possui gargalos e quantos agentes estão utilizando aquele caminho, pois se muitos estiverem usando vão haver colisões entre eles.
- O recurso deve estar disponível: é necessário verificar se existe uma rota entre o recurso e o ponto de depósito, pois o recurso pode estar sendo bloqueado por algum obstáculo, por outro recurso, ou por outro agente.

- Quantos agentes estão naquela área: é importante verificar a quantidade de agentes que já estão coletando naquela região de modo a evitar que um atrapalhe o caminho do outro.

O pathfinding é o mais popular, potencialmente frustrante, problema encontrado na inteligência artificial dos jogos [Bourg 2004]. O pathfinding consiste em encontrar um caminho entre o ponto A e o ponto B, evitando obstáculos. É desejável que a rota encontrada seja a menos custosa possível, tanto no tempo que vai ser levado para calcular ela, quanto para poder executá-la (encontrar o menor caminho). De pouco adianta conseguir o menor caminho, mas demorar um tempo muito grande processando pra poder encontrá-lo. Do mesmo modo, um pathfinding que me retorna um caminho rapidamente, mas tal rota é muito longa passando por pontos desnecessários, se torna desinteressante. Para tal decisão de que pathfinding escolher é preciso certo equilíbrio entre qualidade da solução e o custo de processamento para consegui-la.

Um problema decorrente da utilização de vários agentes é a coordenação entre os mesmos, uma vez que vários agentes irão coletar os recursos, é necessário que haja uma sincronia entre estes agentes de maneira que um não atrapalhe o outro. É preciso ter em mente que cada agente no momento da tomada de decisão leve em conta o que é melhor para o grupo, que ação ele deve escolher de maneira que o resultado seja o melhor para o grupo, mesmo que tal ação não seja a melhor para ele individualmente. Por exemplo, é preferível para o bem comum do time, que um agente vá coletar em um recurso mais afastado dele, e deixe o recurso mais próximo para outro agente coletar.

Outro aspecto da coleta, que visa colaborar com a coordenação, é a comunicação Os agentes precisam se comunicar de maneira eficiente de maneira que cada agente possa informar aos outros participantes de seu time que ações ele vai tomar, uma vez que a decisão de um pode influenciar a tomada de decisão de todos os outros. Por exemplo, se um coletor decide coletar um determinado recurso, ele precisa avisar aos outros, para que os outros saibam que aquele recurso já vai ser coletado, evitando assim que outro membro do time faça uma viagem desnecessária até aquele ponto.

2.6 Critérios de Avaliação

A melhor solução da coleta multiagente de recursos vai ser aquele em que coletar a maior quantidade de recursos em um determinado período de tempo, em diferentes tipos de cenários, com configurações com quantidades diferentes de recursos e agentes. As soluções do capítulo a seguir serão avaliadas de acordo com a sua capacidade de se adequar ao

problema descrito nesse capítulo: Multiagente Depósito Central Recorrente Foraging, com um único tipo de recurso em um ambiente completamente observável, determinístico, seqüencial, dinâmico e contínuo. Além do que precisa respeitar as regras de um jogo de RTS em que os recursos só podem ser depositados no control center.

3 Estado da arte

O objetivo deste capítulo é apresentar o estado da arte para o problema da coleta de recursos, e fazer uma breve análise de como as soluções existentes na literatura se adéquam ao problema, em que tipo de ambiente eles ocorrem e qual o tipo de coleta de recursos multiagente eles fazem.

3.1 Emergent Bucket Brigading

Na abordagem denominada de Bucket Brigading [Ostergaard 2005], um grupo de agentes deve coletar vários objetos espalhados dentro de algumas áreas e levá-los para outras áreas de depósito. Nesse tipo de coleta multiagente os recursos são únicos, necessitando apenas uma visita única para coletá-lo. Nessa abordagem pode haver vários locais de depósito.

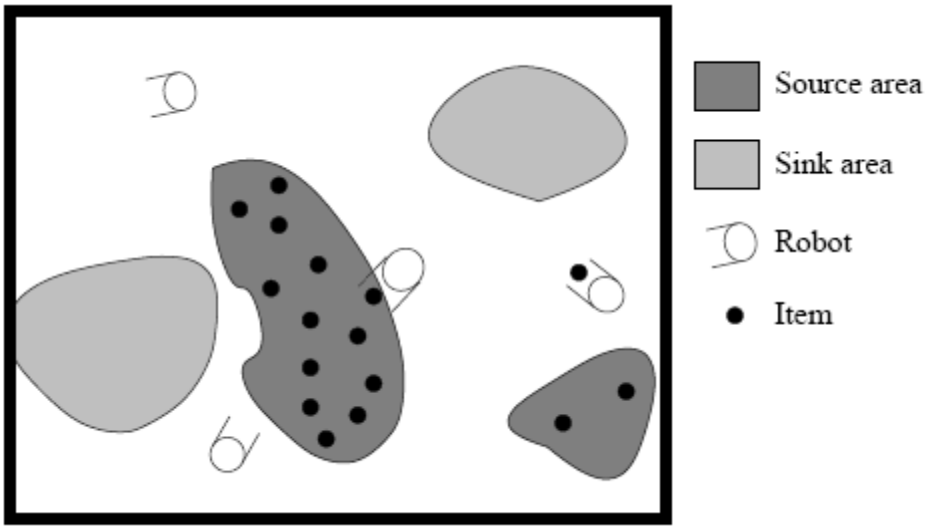

Ilustração 2 - Emergent Bucket Brigading

Os agentes não fazem qualquer tipo de planejamento, (quem vai aonde? E como vão chegar lá?), eles funcionam de maneira reativa. Se ele não estiver carregando recursos ele vai para as áreas de “source” onde estão os recursos inicialmente, enquanto ele estiver na área de “source”, ele navega randomicamente até encontrar um recurso, quando encontra ele coleta e vai em direção a uma área de “sink”, ao chegar a uma área de sink ele deposita o recurso.

Infelizmente o artigo não fornece detalhes da implementação, como os agentes encontram cada uma das áreas, ou que área ele escolhe.

3.2 Numerical Potential Fields with reactive agents

O objetivo deste artigo é utilizar campos potenciais para resolver o problema da coleta de recursos [Simonin 2005]. Em um ambiente parcialmente observável, com apenas uma área de depósito, um grupo de agentes deve vasculhar uma área e coletar os recursos nela existentes.

O mundo é dividido em uma grid, e os agentes vão se movendo randomicamente pelo mundo marcando a distância até o Control Center formando um gradiente a partir da fonte. Quando um agente encontra um recurso, ele coleta e vai "descendo" no gradiente, marcando a trilha, quando um agente encontra essa trilha e esta sem recurso, ele "sobe" no gradiente, pois sabe que no final ele encontrará um recurso.

Nesse cenário o ambiente é diferente do descrito no problema, pois o ambiente não é completamente observável.

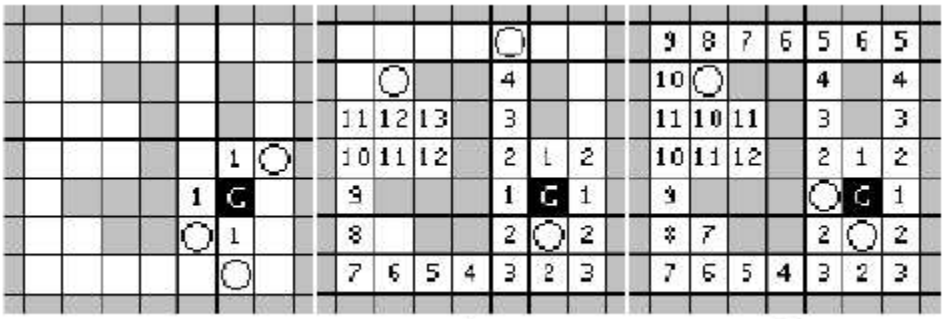

Ilustração 3 - Três etapas da construção do campo potencial

3.3 On Foraging strategies for large-scale multi-robot systems

Essa estratégia proposta por Shell é um pouco mais evoluída, pois ela faz um planejamento prévio [Shell 2006]. Os agentes dividem a área total de busca, pela quantidade total de agentes, criando subáreas, e cada agente têm um comportamento reativo semelhante ao Bucket Brigading [Ostergaard 2005]. No primeiro momento os agentes fazem um pequeno planejamento decidindo quem vai aonde, mas não há informações de como eles vão. Dado que os ambientes não possuem obstáculos, provavelmente não há nenhuma estratégia definida para definir o caminho. Os recursos são de visitas únicas, ou seja, eles só precisam ir uma única vez em cada para coletá-lo.

Nesse artigo, a solução proposta, além de violar a regra de recursos só poderem ser depositados no control center, possui um ambiente apenas parcialmente observável.

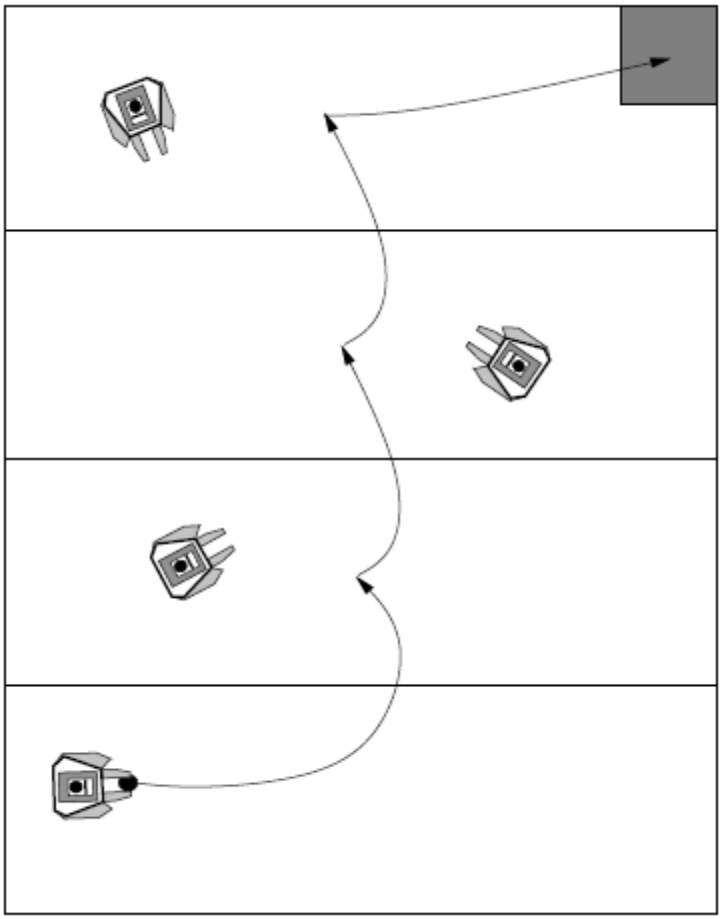

Ilustração 4 - On Foraging strategies for large-scale multi-robot systems

3.4 Multi-Agent Foraging – Taking a Step Further Q-Learning with Search

Essa abordagem visa ambientes com diferentes tipos de recursos [Hayat 2005], em um ambiente parcialmente observável. Nesse artigo existem dois tipos de recursos: A e B. Do tipo A pode ser carregado por um único agente, enquanto do tipo B são necessários 2 agentes.

Os agentes exploram randomicamente a região a procura de comida. Quando um agente encontra uma fonte do tipo A, se ele não estiver carregando nada ele coleta, se estiver ocupado ele manda um aviso, informando aos outros agentes.

Quando um agente encontra uma fonte do tipo B, e ele estiver desocupado, ele manda um alerta e fica esperando um agente vir ajuda-lo. Se ele estiver ocupado ele manda o aviso informando que é do tipo B e continua seu caminho pra home.

No problema abordado existe apenas um tipo de recurso, os agentes podem carrega-los de maneira isolada, não necessitando de ajuda, e o ambiente é completamente observável, que não são cobertas por essa solução.

3.5 Relation between Task-based diversity and efficiency in Multi-Robot Foraging

Nesse artigo é analisada a utilização de dois tipos de agentes: Homo-Algotithm e Hetero-Algorithm [Wang 2002].

No Homo-algorithm os agentes se movem randomicamente até encontrar um recurso, quando encontra, verifica se ele está “marcado”, se não estiver marca, para evitar que outro agente vá em busca do mesmo. Depois de marcar vai até o objeto e coleta. Em seguida move para casa, e deposita o objeto, e volta para o primeiro estágio. Ele não faz qualquer tipo de planejamento, quem vai aonde, ou como vai, os agentes apenas movem-se de maneira randômica.

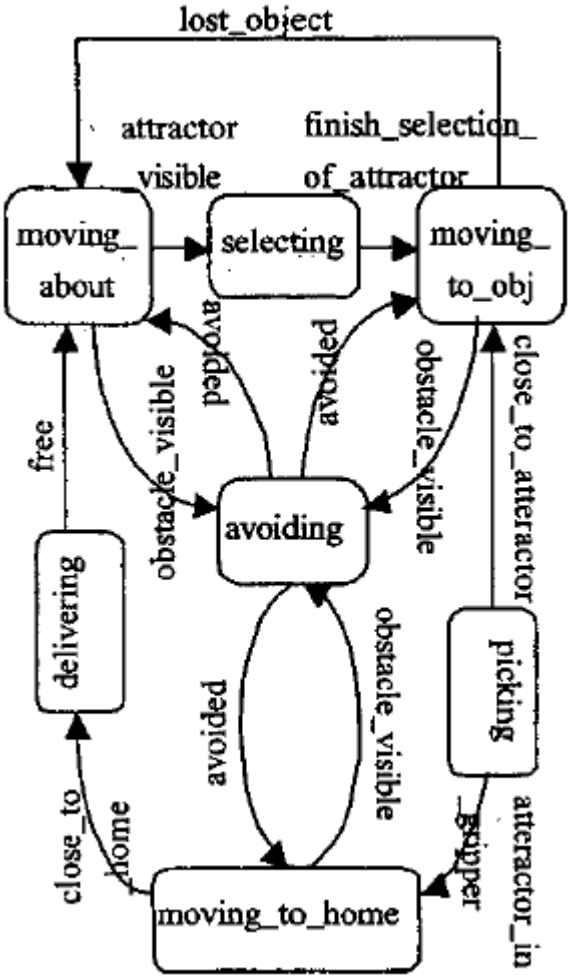

Ilustração 5 – Homo-algorithm

No Hetero-Algorithm o agente varia entre dois papéis, o explorador e o coletor.

O explorador move-se randomicamente pelo cenário procurando por objetos, quando encontra um, avisa ao resto do grupo. Quando um agente passa muito tempo sem encontrar um objeto, ele muda de papel para ser coletor.

O coletor escolhe um dos objetos já encontrados e não selecionados, coleta e traz para o home. Quando um agente passa muito tempo sem encontrar nenhum objeto não selecionado ele passa para tarefa de explorador.

No hetero-algorithm o artigo não cita nenhuma estratégia de definição de qual agente deve ir a cada recurso encontrado pelo coletor.

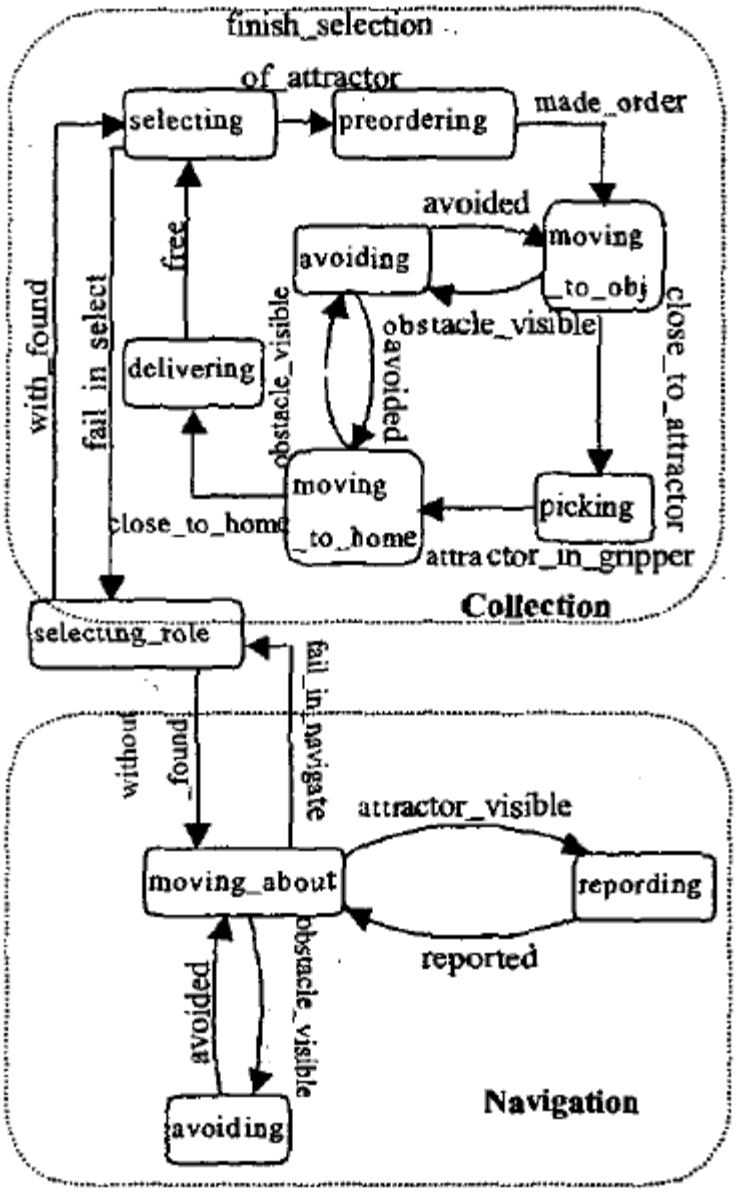

Ilustração 6 - Hetero-algorithm

Nesse artigo, o ambiente é apenas parcialmente observável e o foraging não é recorrente, impossibilitando uma comparação com o problema original.

3.6 Foraging Behavior of Multi-Robot System and Emergence of Swarm Intelligence

Essa abordagem não visa coletar os recursos e levar para um depósito único. Aqui o objetivo é agrupar os recursos em pequenos clusters [Sugawara 2000].

Agentes funcionam de maneira reativa. Eles se movem randomicamente pelo cenário até encontrar os pucks. Se um agente encontra um puck ele pega. E continua se movendo randomicamente. Se um agente encontra um puck e ele já está carregando um puck, todos os agentes que já estão carregando pucks vão para o mesmo lugar. No final os pucks ficam agrupados em vários clusters.

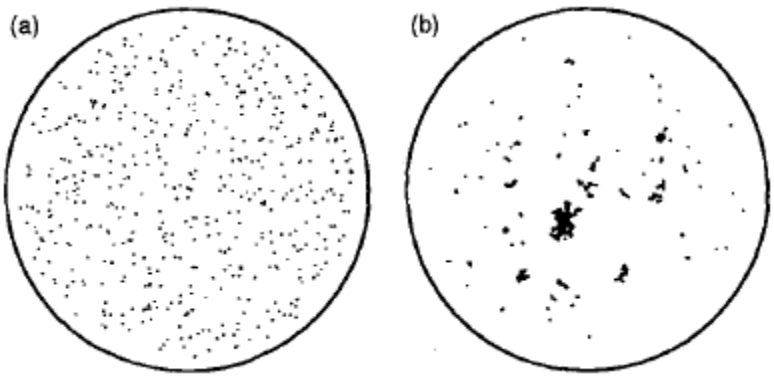

Ilustração 7 - Estado inicial (A) e final (B) do ambiente

O foraging aqui abordado é do tipo único, cada recurso só precisa ser visitado uma vez para ser coletado, não possui um depósito central e o ambiente é apenas parcialmente observável. Os agentes não fazem qualquer tipo de planejamento que defina quem vai aonde e de que maneira vai. Eles apenas navegam randomicamente.

3.7 Swarming Robots – Foraging Behavior of Simple Multi-Robot System

Essa abordagem proposta de Swarming Robots [Sugawara 2002] acontece em um cenário sem obstáculos, onde os agentes devem coletar uma serie de recursos espalhados pelo cenário. Os agentes se movem randomicamente pelo cenário até encontrar os recursos (Recorrente). Se um agente encontra um ponto de coleta ele pega um objeto, faz um broadcast para os outros agentes irem lá coletar e leva o seu objeto até a “home”.

O ambiente aqui descrito é apenas parcialmente observável, e discreto, possuem um único ponto de depósito, entretanto não é feito nenhum planejamento que defina aonde vai cada agente e como vai lá. Eles apenas andam randomicamente até encontrarem um recurso, ou receber um aviso que algum outro agente encontrou um recurso.

3.8 ZigWorker

Solução proposta por [Sette 2008], acontece no mesmo ambiente descrito aqui nos capítulos anteriores. Uma série de agentes autônomos devem coletar os recursos de diferentes lugares, num ambiente determinístico e completamente observável.

Para definir onde cada agente deve ir, é utilizado uma função de utilidade complexa que leva em consideração acessibilidade da mina, a quantidade de agentes naquela região e a disponibilidade da mina, quantos agente planejam utilizá-la. Para definir as rotas é utilizado o algoritmo de pathfinding A* com reserva de caminho.

4 Metodologia

O objetivo deste capítulo é descrever a metodologia para a construção de um sistema multiagente, para a tarefa de coleta de recursos no simulador RTSCup. Nesse capítulo vai ser realizado um estudo, não encontrado na literatura, de diferentes arquiteturas e abordagens para resolver tal problema. Nesse capítulo também será definido os critérios de avaliação das diferentes abordagens e o tipo de cenário onde ocorrerá a avaliação.

4.1 Arquiteturas de Sistemas Multiagentes Propostas

Para definir as arquiteturas dos sistemas a serem estudadas e avaliadas no problema da coleta multiagente de recursos, o problema principal foi dividido em subproblemas, melhor descritos abaixo, como mostra a tabela a seguir.

	Arquitetura	Estratégia de Coordenação	Comunicação	Path-Finding	Alocação
worker0	Reativo	N/A	Sim	N/A	N/A
worker1	Cognitivo	Central	Sim	A-Star	Elitist
worker2	Cognitivo	Central	Sim	A-Star	Auction
worker3	Cognitivo	Individual	Sim	A-Star	Decentralized Communicated
worker4	Cognitivo	Individual	Não	A-Star	Decentralized Without Communication
worker5	Cognitivo	Central	Sim	LRA	Elitist
worker6	Cognitivo	Central	Sim	LRA	Auction
worker7	Cognitivo	Individual	Sim	LRA	Decentralized Communicated
worker8	Cognitivo	Individual	Não	LRA	Decentralized Without Communication

Tabela 2 - Arquiteturas Propostas

O objetivo dessa escolha foi partir de arquiteturas mais simples e evoluí-las, de modo que pudesse investigar o aumento de desempenho ou não, à medida que as soluções fossem se tornando mais complexas. Fazendo combinações de diferentes soluções para os subproblemas, esperasse encontrar quais soluções melhor se combinam.

Antes de criar as diferentes estratégias para resolução dos subproblemas, “Quem vai aonde?” e “Como vai?”, é necessário responder uma pergunta anterior, “Vai haver planejamento?”.

Primeiramente será explanada a diferença entre agentes que possuem planejamento, e os que não possuem. Posteriormente é explicado o problema da coordenação, e as diferenças entre coordenação centralizada e descentralizada, em seguida é explicado o problema da alocação de recursos, e como são as arquiteturas propostas, tanto no caso da decisão descentralizada, quanto na centralizada. Em seguida é abordado o subproblema da comunicação entre os agentes, utilizada no caso da alocação descentralizada. Por último são apresentados os critérios de avaliação, assim como cenários e apresentação do simulador.

4.2 Arquitetura do agente

A tarefa de criar uma seqüência de ações, para atingir um objetivo é chamada planejamento. [Russel 2003]

O primeiro parâmetro considerado para a definição do agente foi a diferença entre os agentes reativos, e os agentes cognitivos. Os agentes reativos operam unicamente baseado em sua percepção atual, enquanto os agentes cognitivos podem perseguir um objetivo, um planejamento. Diferente dos agentes cognitivos, os agentes reativos possuem uma visão imediatista, enquanto os cognitivos podem considerar acontecimentos eventuais futuros ou um objetivo que está mais a frente, que faz parte de um plano.

4.3 Coordenação

Quando vários agentes estão trabalhando juntos, é necessário manter uma série de atividades suplementares que não são diretamente produtivas, mas servem para melhorar os resultados da atividade inicial [Ferber]. Essas atividades suplementares são parte do sistema, e são chamadas atividades de coordenação. Elas são indispensáveis a partir do momento que nós temos um grupo de agentes perseguindo objetivos próprios, e sem as atividades de coordenação os objetivos podem não ser cumpridos.

Por exemplo, imagine um aeroporto que recebe apenas alguns aviões por dia, o tráfego aéreo é praticamente inexistente, o controle aéreo praticamente não existe. Enquanto em um aeroporto de uma grande cidade como São Paulo, ou Nova York, onde um grande número de aviões está pousando ou decolando a cada momento, a atividade de controle aéreo se torna de importância fundamental, desenvolvidas por especialistas treinados a evitar que os aviões colidam. As atividades de coordenação passam a ser fundamentais e impões restrições no problema inicial, transporte de passageiros, limitando a quantidade de aviões em circulação. Se

a quantidade de aviões diminuir demais, então toda a atividade de coordenação perde o sentido. Se a quantidade aumentar, problemas de coordenação ficam ainda mais cruciais.

Em relação à cooperação entre agentes, a coordenação pode ser definida como a articulação de ações individuais a serem cumpridas por cada agente de maneira que toda a operação seja coerente de alta performance [Ferber]. É uma questão de organizar as atividades dos agentes ao longo do espaço e tempo, de maneira que a ação do grupo é melhorada, seja por melhora de performance ou através da redução de conflitos. Em resumo, a coordenação é um dos principais métodos de conseguir a cooperação entre os agentes, atividade fundamental para a coleta multiagente.

A principal necessidade da coordenação na coleta multiagente de recursos é porque os recursos são limitados, não só os recursos a serem coletados, mas também o espaço ocupado por cada agente, que deve ser considerado um recurso a ser compartilhado com os outros agentes. O local onde ele está pode fazer parte de uma rota de outro agente, e os recursos a serem coletados possuem uma quantidade limitada de espaços ao redor dele que podem ser ocupados pelos agentes que querem lhe coletar. A coordenação cresce de importância à medida que os recursos diminuem, se mais agentes são adicionados ao ambiente, mais complexa será a coordenação.

Como foi visto anteriormente, a coordenação tem uma importância muito grande para a coleta multiagente. Para realizar tal coordenação utilizamos dois métodos bem diferentes. No primeiro colocamos um coordenador central que determinava qual seria o próximo objetivo de cada agente, que por sua vez executava a tarefa da maneira que melhor o conviesse, planejando como a tarefa será executada. No outro método de coordenação, a mesma será feita de forma descentralizada emergindo da interação entre os agentes.

4.4 Alocação de Recursos

Alocação de recursos em Sistemas Multiagentes é o processo de distribuição de uma quantidade de recursos a uma quantidade de agentes. Usamos aqui recursos para nos referir aos itens a serem distribuídos, que no nosso caso vão ser os lugares ao redor das minas, propícios para a coleta. Os agentes são as entidades que recebem os recursos na alocação.

Um fator que complica ainda mais o problema, e acaba tornando o mais interessante, é que os recursos são indivisíveis, isso quer dizer que o mesmo recurso não pode ser compartilhado entre vários agentes ao mesmo tempo, logo ele só pode pertencer a no máximo um agente.

Uma distribuição particular dos recursos entre os agentes é chamada de alocação. Os agentes podem ter uma preferência maior por alguns recursos que eles recebem. A distinção de preferência pode ocorrer de 4 formas diferentes, segundo [Chevaleyre 2006]:

- Cardinal: a preferência pode ser explícita de maneira cardinal, onde uma função de utilidade vai retornar um valor, que vai dizer o quão satisfeito está o agente com aquela alocação.
- Ordinal: a preferência nesse caso é feita apenas numa relação binária, o agente informa se ele prefere a alocação A ou B (A>B). Os agentes podem ordenar quais são as melhores alocações, mas não podem dizer o quão melhor.
- Binária: na preferência sendo exposta de maneira binária os agentes só podem dizer se uma alocação é boa ou ruim, o agente não expressa que uma alocação é melhor que outra ele pode apenas expressar se gosta ou não.
- Fuzzy: nesse caso uma função iria comparar duas alocações diferentes e retornar o grau de preferência de uma alocação sobre a outra.

Avaliando as 4 formas diferentes de expressar a preferência decidiu-se utilizar a preferência cardinal por seus resultados serem mais completos.

O processo utilizado para encontrar a alocação de recursos mais adequada pode ser centralizado ou distribuído.

4.4.1 Alocação Centralizada

No caso da decisão centralizada, uma entidade única decide a alocação final dos recursos entre os agentes, possivelmente levando em consideração as preferências particulares de cada agente. Um exemplo típico são os leilões combinatoriais, nesse caso, a entidade central é o leiloeiro, e as preferências dos agentes seriam os lances. Neste trabalho propomos dois tipos de alocação descentralizada: Auction e Elitit Social Welfare.

4.4.2 Auction

Nesse caso o coordenador central define qual o melhor recurso para o grupo, através de uma função de utilidade que vai indicar sua preferência, e faz um leilão entre os agentes para ver quem dá o melhor lance [Chevaleyre 2006], ou seja, quem teria maior preferência por aquele recurso, e o agente que deu o maior lance é alocado para aquele recurso. Como foi visto anteriormente cada recurso e indivisível, então após cada alocação, o coordenador central, "o leiloeiro", escolhe o próximo recurso disponível que ele tenha maior preferência, e o leiloa, o processo é repetido até que todos os agentes estejam alocados.

4.4.3 Elitist Social Welfare

A alocação de recursos centralizada, Elitist Social Welfare [Chevaleyre 2006], possui um funcionamento semelhante ao Auction, levando em consideração todas as preferências dos agentes em relação a todos os recursos disponíveis. Então cada agente informa ao

coordenador central o quanto ele tem preferência por cada recurso, através de sua função de utilidade. Em seguida de posse de todos os valores das funções de preferência, o controlador central aloca cada agente com o seu recurso preferencial, se dois agentes desejarem o mesmo recurso, ele vai ser alocado para aquele que tiver um grau de preferência maior, definido pela função de utilidade.

4.4.4 Alocação descentralizada

No caso da decisão distribuída a alocação emerge da interação entre os agentes, podendo haver ou não uma negociação.

Os agentes de alocação descentralizada foram baseados no trabalho de [Moura 2006]. Na alocação individual cada agente decide para onde deve ir através de sua função de decisão. Se não houver comunicação cada agente vai para o recurso onde possui o melhor resultado da função de decisão independentemente da decisão dos outros agentes, isso pode ocasionar vários problemas caso o mesmo recurso seja escolhido por diferentes agentes. Esses problemas podem ser resolvidos caso os agentes utilizem comunicação, nesse caso um *BlackBoard* para informar aos outros que recurso cada um escolheu e por quanto vai fazer uso deste recurso. Dessa maneira essa informação vai ser levada em conta, além da função de decisão, na escolha de que recurso deve ser escolhido.

4.5 Comunicação

Em sistemas multiagentes, assim como seres humanos, comunicação é a base para interações e organização social [Ferber]. Sem comunicação, os agentes são apenas indivíduos isolados, surdos para os outros agentes, fechados em um ciclo de percepção-raciocínio-ação. É graças à comunicação entre os agentes que pode haver cooperação, coordenação entre as ações, desenvolvimento de tarefas conjuntas e se tornarem parte de uma sociedade.

Imagine, por exemplo, um grupo de macacos em uma árvore e que um deles vê um tigre. O macaco vai gritar e todos os outros vão entender que aquele grito está indicando perigo, ou seja, a aproximação de um tigre Então todos os outros macacos vão subir para o topo da árvore, salvos graça a comunicação. Em outras palavras, a comunicação permite aprendizado baseado na observação de outro agente.

Em um ambiente com coordenação descentralizada, os agentes podem ou não se comunicar através da comunicação. Os agentes podem informar aos outros agentes o que estão fazendo, seus planos suas visões e o que for importante para os outros agentes tomarem suas decisões. A comunicação pode ocorrer de diferentes maneiras, mensagens diretas,

marcas no ambiente ou através de um blackboard. No caso dos agentes desenvolvidos nesse trabalho será usado somente o blackboard, onde todos os agentes podem ler e escrever informações importantes para os outros.

4.6 PathFinding

Encontrar o caminho entre dois pontos é um problema comum, talvez até o mais comum, em jogos de computadores. Embora o problema possa ser bastante simples para os humanos, uma surpreendente quantidade do tempo total de processamento em muitos jogos comerciais é gasta resolvendo problemas de path finding. Entre as razões para isso ocorrer estão a crescente complexidade do ambiente dos jogos e o numero de entidades que devem ser calculadas. O problema pode ser ainda maior se o ambiente mudar dinamicamente, uma rota antiga pode ser bloqueada e uma nova ser criada. Deste modo, os caminhos não podem ser calculados logo de início, e sim precisam ser calculados dinamicamente durante o jogo.

A definição de um problema de pathfinding é simples: dado um ponto inicial S (start), e um ponto final G (goal), encontrar um caminho de S para G minimizando um dado critério. Normalmente a função de custo é o tempo para percorrer todo o caminho, que pode depender da distancia, o tipo de terreno etc. Num jogo de RTS pode se desejar encontrar o caminho mais seguro, para algumas unidades, que evite encontrar algum inimigo, sem se importar com o tempo que ele vai demorar pra percorrer. O problema da busca pelo melhor caminho pode ser visto como uma busca em um grafo, no qual os nós vão ser os pontos onde o agente tem acesso e as arestas são as ligações entre os pontos e seus vizinhos.

No mundo ideal, o path finding encontraria a melhor rota em um ambiente continuo que vai de S para G. Infelizmente, esta é raramente uma opção realística, pois os espaços de busca se tornam muito complexos. Isso pode ser resolvido discretizando o mundo, e conseqüentemente o espaço da busca, em uma quantidade finita de waypoints, e reduzindo os caminhos a conexões entre eles. Em outras palavras, é construído um grafo em que os vértices são waypoints e as arestas são as conexões entre eles. Com isso o problema inicial foi reduzido a encontrar o caminho em um grafo. A idéia é inspirada na vida real: ir para o waypoint mais próximo (aeroporto, parada de ônibus, estação de trem, etc.), ir através dos waypoints até aquele que seja mais próximo do ponto desejado, e seguir até o destino. [Bourg 2004]

Resumindo a abordagem em três passos: primeiramente é visto como o mundo pode ser discretizado da melhor maneira, baseado na discretização, é formado um grafo e o problema do path-finding é transformado em um de encontrar o menor caminho em um grafo. Existem várias maneiras de resolver esse tipo de problema. Algumas delas vão ser vistas

adiante nesse capitulo. Finalmente, quando o caminho é encontrado no grafo, ele precisa ser transformado em movimentações no mundo real.

O primeiro passo para resolver um problema de pathfinding em um ambiente contínuo é discretizá-lo. Uma vez estabelecido os waypoints, é preciso definir se existe uma conexão entre eles baseado na geometria do ambiente. A conexão pode ser associada com o custo baseado na distância ou o tipo de ambiente, e esse custo pode ser usado como o peso dessa aresta.

Uma grid pode ser criada, para definir os waypoints, que é um conjunto de polígonos, sobre o ambiente. Para simplificar, considera-se apenas grids em que cada tile compartilha no máximo uma aresta com o tile vizinho. Dessa maneira, o centro do tile representa um waypoint, e seus vizinhos, compostos pelos tiles adjacentes, formam as possíveis conexões para outros waypoints. O ambiente dentro de um tile define se ele deve ser considerado com um waypoint (ou se ele é um obstáculo) e quais são as suas conexões com os outros waypoints.

A desvantagem dessa abordagem é que ela se importa com a geometria atual do ambiente. Por exemplo, algumas partes do mundo podem ficar desconectadas se a granularidade do mundo não for pequena o suficiente. E uma granularidade menor do que a necessária pode exigir uma quantidade de memória e processamento muito grande para poder usá-la.

Existem três tipos de polígonos que podem ser usados para formar esse grid [Smed 2006]. A malha pode ser formada por triângulos eqüiláteros, quadrados ou hexágonos regulares. Nesse trabalho será considerada uma malha de quadrados para simplificar a implementação.

Uma vez o mundo discretizado, e as conexões entre os tiles vizinhos efetuadas, o problema de path-finding se torna um problema de busca. Para resolver o problema de busca utilizaremos dois algoritmos o A* (A-Star) e uma variação do mesmo visando sistemas multiagentes.

O algoritmo A* (A-Star) é o algoritmo de path-finding mais utilizado, pois é garantido de encontrar o melhor caminho entre dois pontos, assumindo, é claro, que ele existe. [Bourg 2004] O melhor caminho não necessariamente é o mais curto, mas sim o menos custoso. Um caminho mais curto por um terreno de lama pode ser mais custoso que um caminho mais longo no asfalto.

O A* funciona perfeitamente para um ambiente mono-agente, entretanto em um ambiente multiagente o A* puro, não é suficiente, pois os agentes podem colidir e acabarem entrando em dead-lock, se um estiver ocupando o tile que o outro deseja ocupar. Visando esses casos na próxima seção veremos o LRA*.

*4.6.1 LRA**

Local Repair A* descreve uma família de algoritmos amplamente utilizados na indústria de jogos [Silver 2005]. Cada agente busca o caminho ao seu destino, ignorando todos os outros agentes exceto os seus vizinhos atuais. Então ele segue o seu caminho até ter uma colisão iminente. Quando o agente vai se mover para um tile ocupado, ele para e recalcula o caminho até o destino partindo do seu ponto atual.

4.7 Cenários

Para testar a capacidade dos agentes frente a diferentes adversidades ocorreram simulações em diferentes ambientes. O cenário onde vai ocorrer a simulação é um ambiente fechado com um control center, e um grupo de agentes posicionados ao redor dele inicialmente. Espalhados pelo cenário há vários recursos que devem ser coletados pelos agentes e levados para o control center. Cada recurso só pode ser coletado pelos agentes que estiverem próximos dele, sendo que os agentes possuem uma capacidade máxima. Uma vez que esse recurso esgota, os agentes devem procurar outro recurso para coletar.

Os agentes serão testados em duas variações de ambiente, um mapa com muitas opções de recursos para serem coletados, e o mesmo mapa, porém com uma quantidade de recursos muito menor visando analisar a capacidade de cada arquitetura de coordenação.

Abaixo temos o cenário 1 de simulação. Ele é o mais simples dos cenários com os recursos bem afastados e os obstáculos de fácil transposição sem formar paredes.

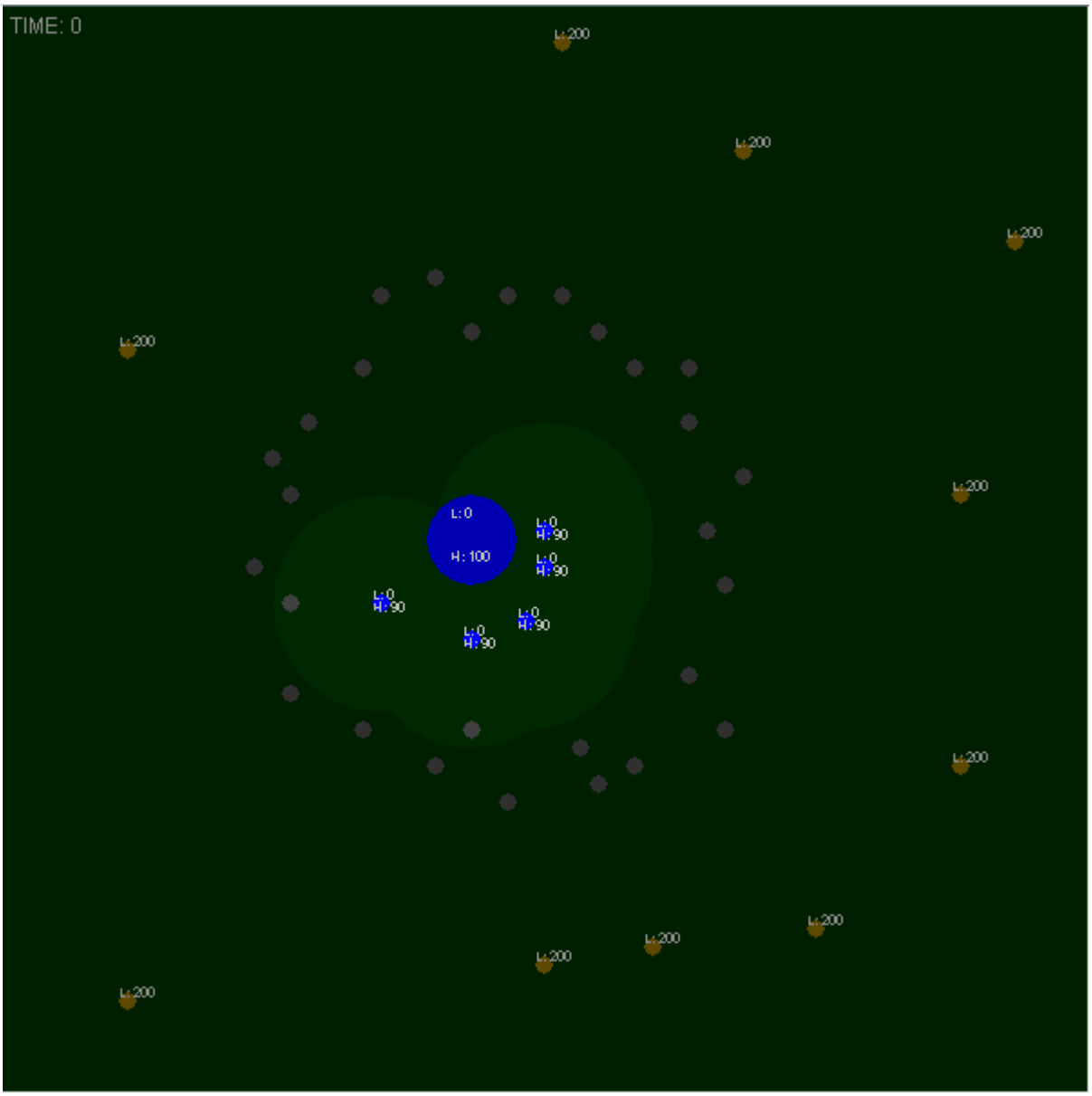

Ilustração 8 - Cenário de simulação 1

A seguir temos o cenário 2, semelhante ao cenário 1, porém com uma quantidade menor de recursos visando testar a capacidade de coordenação dos agentes, de como os agentes conseguem dividir recursos escassos.

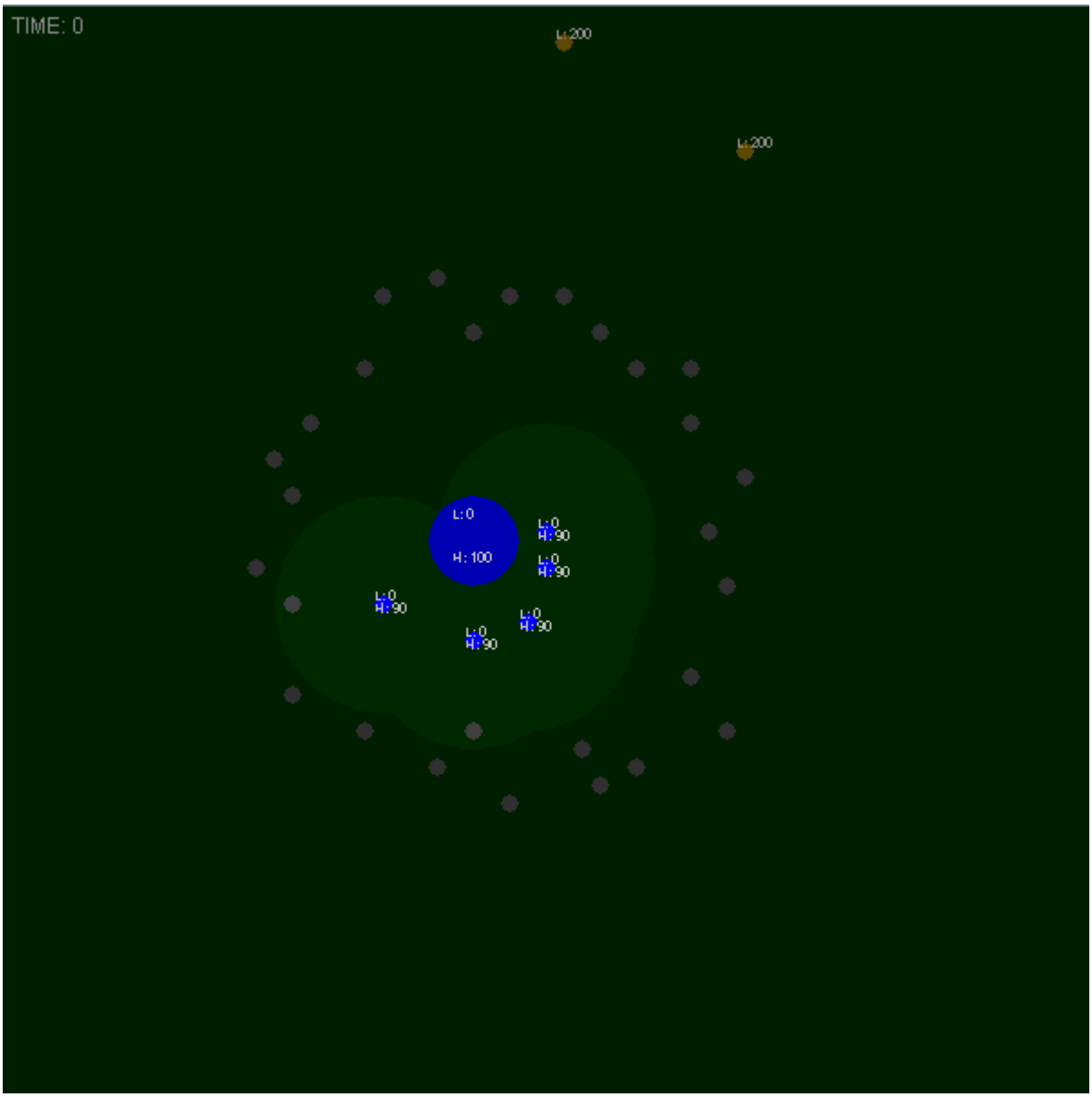

Ilustração 9 - Cenário de Simulação 2

4.8 Critérios de Avaliação

O objetivo de um sistema multiagente de coleta de recursos é coletar o máximo de recursos na menor quantidade de tempo, dito isto, cada sistema multiagente deve utilizar o máximo da capacidade de cada integrante do grupo, evitando a ociosidade deles, e mantendo uma coordenação de maneira que cada um possa executar suas tarefas sem ser atrapalhado.

Os agentes criados serão testados e validados no RTSCup, descrito na próxima sessão. No ambiente cada agente terá 5 minutos de simulação em cada mapa, e eles serão avaliados pela quantidade de recurso que eles conseguirem coletar ao final deste tempo. E uma análise será feita baseada nos seus resultados obtidos, e no comportamento de seus agentes.

4.9 RTS CUP

Para a validação e análise das diferentes abordagens criadas, foi necessário um simulador que servisse como benchmark e que antedesse alguns requisitos como facilidade e praticidade no uso. Por essas razões decidiu-se usar o RTSCup.

O RTSCup [Vieira 2007] foi desenvolvido no Centro de Informática da UFPE com o objetivo de ser utilizado pelos alunos para criar e testar novas técnicas de IA, servindo como uma ferramenta para medir o desempenho e comparar os resultados de cada abordagem.

Ele é um ambiente de simulação de jogos de estratégia em tempo real. E foi escolhido para ser utilizado nesse trabalho, pois ele possui uma arquitetura modularizada que visa manter o desenvolvedor com o foco na Inteligência Artificial e pode servir como benchmark entre diferentes implementações.

Sua utilização é bastante simples, e possui uma interface visual que permite acompanhar a simulação em tempo real, o que facilitou muito o desenvolvimento.

5 Implementação

O objetivo deste capítulo é discutir a implementação das diferentes abordagens e detalhes de sua arquitetura. Posteriormente dentro desse capítulo serão apresentados os resultados e será feita uma análise sobre eles, tanto individual quanto coletiva.

5.1 Desenvolvimento

Como foi visto no capítulo anterior, antes das decisões de planejamento “Quem vai aonde?” e “Como chegar até lá?”, era preciso decidir se ia haver planejamento ou não. Por isso foi criado dois tipos de agentes, o agente reativo e o agente cognitivo, ambos descritos nas seções a seguir.

5.1.1 Agente Reativo

O agente reativo aqui utilizado é baseado em [Simonin 2005]. Os agentes constroem um campo potencial partindo do Control Center e se espalhando por todo o cenário. Durante a construção do campo potencial quando um agente encontra uma mina ele coleta. Em seguida ele desce no gradiente até chegar ao Control Center e entregar o recurso. Durante este percurso ele marca o caminho. Um próximo agente que encontrar essa trilha, vai subir no gradiente seguindo a trilha, pois ele sabe que no final vai haver um recurso.

5.1.2 Agentes Cognitivos

Cada agente possui acesso a uma classe que irá fazer a sua alocação com uma mina (Iallocation), a uma classe (GridManager) que possui o grid criado e a posição das minas, sendo também responsável pelo pathfinding. Os agentes cognitivos funcionam com uma máquina de estados interna com 4 estados descritas a seguir:

- IDLE: É o estado inicial do agente: neste estado o agente procura um local para minerar, então ele pede a classe responsável pela alocação (Iallocation). Quando o local é definido ele pede ao GridManager a rota que ele deve seguir e seu estado é setado para GOING TO MINE;
- GOING TO MINE: neste estado ele vai percorrer seu caminho até chegar ao seu destino. Quando chegar ao local seu estado vai ser setado como MINING;

- MINING: neste estado ele vai minerar até a mina esvair, ou até ele coletar o máximo de recursos que ele pode carregar. Se ele coletar o máximo de recurso possível seu estado é setado como GOING TO DELIVER. Se a mina esvair antes dele preencher sua capacidade máxima ele vai para o estado IDLE;
- GOING TO DELIVER: neste estado o agente vai utilizar a mesma rota que ele usou para chegar à mina. Ao chegar ao control center, o agente deposita todo o recurso que ele esta carregando. Em seguida seu estado é setado como IDLE.

Abaixo podemos ver uma máquina de estados finita que representa os estados dos agentes e suas transições:

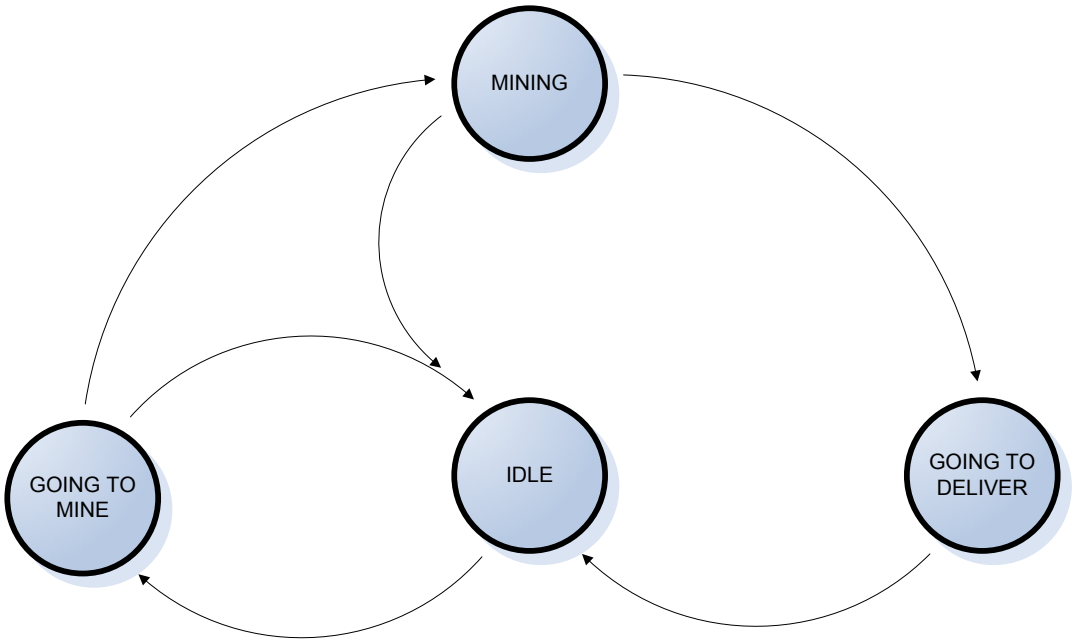

Ilustração 10 - Maquina de Estados do Agente

Para facilitar a decisão de que local ir para minerar, decisão tomada no estado IDLE, existe uma lista, acessível por todos os agentes, com todas as posições de mineração (uma posição é considerada de mineração se ela for vizinha a uma mina).

A seguir temos o diagrama de classes dos Agentes.

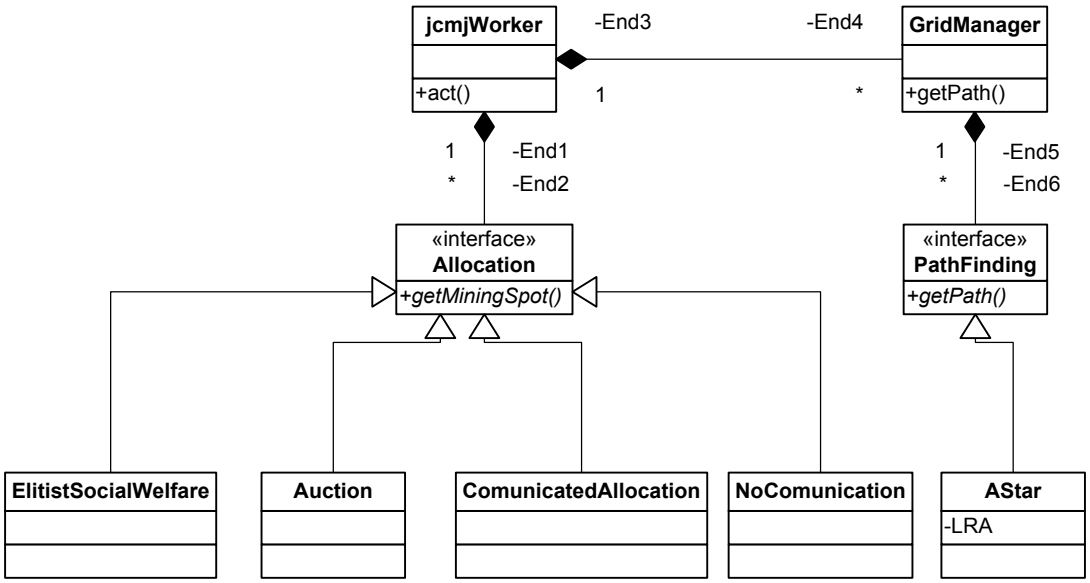

Ilustração 11 - Diagrama de classe dos agentes cognitivos

As classes que implementam a interface Allocation são as classes responsáveis pela alocação dos recursos. Elas definem onde cada agente deve ir e que recurso coletar. A classe jcmjWorker representa o Worker. Ela possui a máquina de estados mencionada anteriormente, e cada agente vai ser uma instância da mesma.

5.2 Resultados

A seguir apresento os resultados das simulações nos dois ambientes vistos no capítulo anterior.

Mapa 1:

Agente	Recursos
Worker0	120
Worker1	440
Worker2	120
Worker3	340
Worker4	310
Worker5	550
Worker6	490
Worker7	450
Worker8	430

Tabela 3 - Resultados do Mapa 1

Mapa 2:

Agente	Recursos
Worker0	20
Worker1	40
Worker2	100
Worker3	20
Worker4	0
Worker5	180
Worker6	150
Worker7	190
Worker8	10

Tabela 4 - Resultados do Mapa 2

5.3 Análise

O objetivo deste capítulo é fazer uma análise dos resultados obtidos, tentando identificar os pontos fortes e fracos de cada abordagem. A análise é feita em duas etapas, primeiramente é

feita uma análise individual de cada agente, e de seu comportamento na simulação. Posteriormente é feita uma análise comparativa entre as diferentes abordagens, mostrando em quais situações alguns agentes levaram vantagens.

5.3.1 Análise Individual

Worker 0

Os agentes passaram um longo tempo criando o gradiente e vagando randomicamente, mas quando um agente encontrava uma mina ele conseguia coletar o máximo possível dela, e conseguia voltar para o Control Center sem dificuldades, e ir novamente ao mesmo recurso utilizando a trilha criada por ele mesmo de maneira bastante eficiente. Possui uma implementação simples e é o que exige menos tempo de processamento, fator importante para jogos RTS.

Worker 1:

Os agentes se dividiram de maneira uniforme pelo cenário evitando colisões entre eles, e eles conseguiram coletar de diferentes minas aproveitando da melhor maneira as minas. A alocação de recursos precisou de um pouco mais de tempo para criar todas as combinações possíveis (mina x trabalhador), para poder escolher a melhor alocação para todos. Alguns agentes colidiram e ficaram presos na proximidade do Control Center no momento que iam entregar os recursos.

Worker 2:

Os agents foram alocados para locais de mineração muito próximo um dos outros, o que acabou gerando certo engarrafamento de agentes nas proximidades das minas, muitos agentes colidiram e ficaram presos passando a maior parte da simulação parados. A alocação foi um pouco custosa também.

Worker 3:

Os agentes conseguiram se dividir bem ao redor das minas para fazer as coletas e todos conseguiam um espaço para coletar. As alocações foram feitas de maneira eficiente necessitando muito pouco tempo de processamento.

Worker 4:

Devido à falta de comunicação, muitos agentes escolhiam a mesma mina então eles ficavam parados esperando o outro terminar de coletar para poder coletar. A alocação foi feita de maneira muito rápida. Quando a quantidade de recurso foi reduzida teve um comportamente muito inferior aos outros, pois vários agentes escolhiam o mesmo recurso e acabavam se prendendo.

Worker 5:

Uma variação do Worker1. Conseguiu o melhor resultado, pois os agentes que haviam colidido nas proximidades do Control Center tiveram a possibilidade de recalcular as suas rotas evitando dessa maneira que eles ficassem presos.

Worker 6:

Variação do Worker2. As proximidades das minas ainda estavam muito povoadas, mas agora os agentes tiveram a possibilidade de recalcular suas rotas ainda assim muito tempo se gastou, pois as colisões ainda sim eram muito freqüentes.

Worker 7:

Variação do Worker3. Os agentes conseguiam se dividir bem nas minas e recalcular seus caminhos de volta. Conseguiu bons resultados sem necessidade de muito tempo de processamento, tanto para calcular o pathfinding quanto para calcular a alocação.

Worker 8:

Variação do Worker4. Mesmo com os agentes podendo recalcular suas rotas, o fato de vários agentes escolherem o mesmo local para minerar acabou decaindo a performance do time, pois muitos agentes passavam muito tempo ociosos. Teve o mesmo problema do worker4 so que de maneira ainda mais grave quando a quantidade de recurso foi reduzida, pois vários agentes escolhiam o mesmo recurso e acabavam se prendendo.

5.3.2 Análise Geral

Como era esperado todos os agentes tiveram uma melhora de performance quando utilizaram o PathFinding com LRA, com reparo local. Com o A* sendo calculado uma única vez, muitos agentes ficavam presos impossibilitados de coletar recursos.

Dentre os agentes com a alocação descentralizada, os que tinham comunicação tinham uma melhora de desempenho quando comparado ao que não podia se comunicar, uma vez que os agentes através da comunicação faziam melhor a escolha de que mina ele iria ser alocado, evitando de tal maneiras que os agentes ficassem ociosos esperando a mina ser desocupada. Além disso, sabendo quais minas já estavam ocupadas, os agentes procuraram coletar de mais minas, visitando então minas até que os outros agentes que não possuíam comunicação, nem chegaram a visitar.

Os agentes com alocação centralizada, o Auction Allocation, tiveram uma performance pior, pois foram alocados para pontos muito próximos, tumultuando o ambiente ao redor da mina, e o agente tinha que recalcular várias vezes a rota. O Elitist Allocation conseguiu um resultado melhor, devido ao fato dos agentes estarem mais espalhados pelo cenário, evitando criar regiões tumultuadas.

Comparando os dois tipos de coordenação, a centralizada teve melhores resultados, mesmo quando comparada com a individual com comunicação, pois na individual o agente quando aloca uma mina para si não tem essa mina desalocada por outro agente que esteja até mais próximo dela. Enquanto na centralizada todas as possibilidades são analisadas antes de ser feita qualquer escolha.

Os resultados mostram que o melhor resultado, entre as abordagens analisadas, foi obtido pelo agente cognitivo, com alocação centralizada baseada no Elitist Social Welfare e um pathfinding que permita recalcular a rota em momentos de colisão.

6 Conclusões e Trabalhos Futuros

Este capítulo tem como objetivo apresentar algumas considerações finais sobre os principais tópicos abordados nesta dissertação, incluindo as contribuições alcançadas e indicações para trabalhos futuros.

6.1 Considerações Finais

Apesar da clara aplicação de coleta de recursos multiagente em jogos, particularmente em jogos RTS, nenhum trabalho de análise foi encontrado na literatura. O objetivo deste trabalho foi atingido com sucesso, pois além de apresentar uma análise original de uma série de possíveis arquiteturas e abordagens para a coleta de recursos multiagente de jogos RTS, podemos citar, entre outras contribuições deste trabalho:

- O estudo realizado entre as diferentes técnicas e abordagens de coleta de recursos existente na literatura;
- Um estudo que possibilitou a definição exata do problema da coleta de recurso, conseguindo uma especificidade que permitirá a comparação dos agentes deste trabalho com outros estudos gerados posteriormente;
- Uma arquitetura no qual pode se analisar isoladamente cada sub-problema da coleta;
- Uma arquitetura que permita criar novas soluções para os sub-problemas da coleta, tendo um benchmark como base.

6.2 Trabalhos Futuros

Em trabalhos futuros pretende-se criar arquiteturas mais complexas expandindo o trabalho já existente, com as seguintes melhorias:

- Adicionar novos parâmetros para a formulação do agente como a criação de diferentes funções de utilidades além da distância utilizada neste trabalho.
- Também gostaria de ampliar as possibilidades dos parâmetros já existentes como diferentes abordagens de path-finding coletivo.
- Permitir a comunicação entre os agentes através de mensagens além do BlackBoard e flags, utilizados neste trabalho.
- Criar alocações individuais que permitam a negociação entre agentes, visando qual a melhor alocação para o time.

E por fim quando estivermos convencidos que cobrimos todas as possibilidades necessárias do estudo criar um agente para disputar a competição no AIIDE.

Referências

Adams, D. (2006). *The State of the RTS*. Available from: http://pc.ign.com/articles/700/700747p3.html [Accessado em 26 de Julho de 2009]

Machado A. Patrulha Multiagente: uma Análise Empírica e Sistemática. *Tese de Mestrado.* UFPE.

Berman, S.; Edan, Y.; Jamshidi, M., A foraging group of autonomous, mobile robots-implementation of hierarchical fuzzy behavior-based control. In: The 21st IEEE Convention of Electrical and Electronic Engineers in Israel, 2000..

BOURG, D. AND SEEMANN, G. (2004),. AI For Game Developers. Artificial Intelligence: A Modern Approach. 1st Edition. *O'Reilly*, 2004.

Chevaleyre, Y.; Dunne, P. E.; Endriss, U.; Lang, J.; Lemaˆıtre, M.; Maudet, N.; Padget, J.; Phelps, S.; Rodr´ıguez-Aguilar, J. A.; and Sousa, P. 2006. Issues in multiagent resource allocation. Informatica 30:3–31.

Strategies for Strategy Game AI. In Papers from the AAAI 1999 Spring Symposium on ArtificialIntelligence and Computer Games, Technical Report SS- 99-02, 24-27. AAAI Press.

HAYAT S, NIAZI M, Multi Agent Foraging- Taking a step further Q-Learning with Search. In: Proceedings of the 2005 IEEE, International Conference on Emerging Technologies. Islamabad 2005.

Moura, J. (2006). Uma Estratégia Eficiente de Coleta Multiagente para Jogos RTS. *Thesis, (Graduate Project)*. UFPE.

ORTS, (2003). *ORTS Homepage*. Available from: http://www.cs.ualberta.ca/~mburo/orts [Accessado em 19 Julho de 2009].

Ostegaard E., Sukhatme G., Mataric M., Emergent Bucker Brigading – A simple mechanism for improving performance in multi-robot constrained space foraging tasks.

Russell, S., Norvig, P. (2003). Artificial Intelligence: A Modern Approach. 2nd Edition. *Prentice Hall*, 2003, pp 41-44.

Wikipedia, The Free Encyclopedia. Disponível em: http://en.wikipedia.org/wiki/Real-time_strategy [Acessado em 20 de Julho de 2009]

SCHWAB, B. 2004. AI Game Engine Programming. Charles River Media, Setembro, 2004.

SHELL D, MATARIC M, On foraging strategies for large-scale multi-robot system. In:Proceedings of the 2006 IEEE/RSJ, Intl.Conference on Intelligent Robots and systems. Beijing, China October 2006

D. Silver. Cooperative pathfinding. In AIIDE, pages 117--122, 2005

Olivier Simonin, Construction of numerical potential fields with reactive agents. AAMAS 2005: 1351-1352

Smed J., Hakonen H. (2006),. Algorithms and network for computer games. 1st Edition. *Wiley*, 2006.

Sugawara, K. Sano, M. Yoshihara, I. Abe, K. Watanabe, T. Foraging behaviour of multi-robot system and emergence of swarm intelligence. In: IEEE International Conference on Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. Tokyo 1999

Sugawara, K. SanoK. Watanabe, T. Swarming robots: Foraging behavior of simple multi-robot system. In: 2002 IEEE/RSJ international conference on intelligent robots and systems, Lausanne 2002

Vieira, V. (2007). RTSCup: Ambiente de Simulação de Jogos RTS com Foco na Inteligência Artificial. *In: Proceedings of the 6th Brazilian Symposium on Computer Games and Digital Entertainment, 7-9 November 2007 São Leopoldo, Brazil*.

WANG S, TAN M, Relation between task-based diversity and efficiency in multi-robot foraging. In: Control, Automation, Robotics and Vision, 2002. ICARCV 2002.