Emergência de Comportamentos em Agentes Cooperativos através de Composição de Técnicas de Inteligência Artificial

João Rogério Vieira Neto
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina
Florianópolis, Brasil
jrogerio.vn@gmail.com

Jerusa Marchi
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina
Florianópolis, Brasil
jerusa.marchi@ufsc.br

***Resumo*—Sistemas inteligentes são, em muitos casos, sistemas complexos, onde o cumprimento da tarefa pode demandar a ação integrada e coordenada de diversos agentes. Além disso, a complexidade das ações individuais pode demandar a utilização de diversas técnicas. Diversas arquiteturas foram propostas ao longo dos anos, dentre elas, uma em particular, proposta por [1], consiste na integração de diversas técnicas de IA. Nosso objetivo é integrar técnicas de inteligência artificial seguindo esta arquitetura e investigar os aspectos de coordenação entre os níveis, visando obter um agente capaz de reagir, perseguir um objetivo e alterar sua estratégia de ação, dependendo das circunstâncias do ambiente. Para isso, utilizaremos um jogo de captura à bandeira.**

***Index Terms*—Agentes Inteligentes, Arquiteturas de Agentes**

I. Introdução

Em muitas soluções desenvolvidas utilizando técnicas de Inteligência Artificial (IA) a autonomia do sistema é uma qualidade desejada, onde o uso de agentes passa a ser comum. Além disso, quando a complexidade do sistema é alta somente um agente pode ser insuficiente para implementar uma solução para o problema, necessitando da ação coordenada de um grupo de agentes [2], [3]. Dependendo da complexidade das ações esperadas de cada agente, também torna-se necessária, a composição de diversas técnicas visando cobrir todos os aspectos do comportamento desejado. Por outro lado, jogos digitais são, em sua maioria, sistemas modulares e complexos, onde cada módulo é responsável por realizar tarefas específicas [4], [5], sendo portanto um excelente domínio para testar arquiteturas de agentes e a integração de técnicas IA.

Neste sentido, este trabalho tem por objetivo a implementação de técnicas de IA seguindo uma arquitetura multinível, onde cada nível utiliza uma técnica distinta. A ação coordenada dos níveis provê o comportamento inteligente individual e, em nível global, a ação coordenada dos agentes provê o comportamento inteligente do sistema. Nosso objetivo é verificar a plausabilidade do uso integrado de técnicas para a obtenção do comportamento inteligente e principalmente, os aspectos de coordenação das ações entre os níveis.

O artigo está organizado da seguinte forma: na Seção II é apresentada a arquitetura multinível na qual a arquitetura implementada foi inspirada. Na seção III é descrito o cenário do jogo criado para instanciar os agentes. Na seção IV apresenta-se a composição das técnicas escolhidas para a constituição dos agentes e o detalhamento de cada nível da arquitetura, bem como seu funcionamento global. A seção V apresenta a validação da implementação realizada, com os testes individuais e a integração final da arquitetura e sistema. Por fim, são feitas algumas considerações finais e são apresentadas algumas limitações encontradas.

II. Arquitetura Multinível

A arquitetura implementada é inspirada na arquitetura multinível proposta por Bittencourt [1] e integra as principais técnicas de IA, como Redes Neurais, Sistemas Fuzzy, Algoritmos Genéticos e Raciocínio Simbólico, cujo intuito é o desenvolvimento de um ser artificial autônomo, capaz de aprender com sua interação com o mundo, correlacionar fatos, armazenar e extrapolar situações vividas, tomar decisões e reorganizar seu próprio conhecimento. O modelo da arquitetura ampara-se nas seguintes hipóteses [6]:

- Cognição é uma propriedade emergente de um processo cíclico e dinâmico baseado na interação de um conjunto de unidades funcionalmente independentes.
- Qualquer modelo da atividade cognitiva deve ser epistemologicamente compatível com a teoria da evolução.
- Aprendizado e atividade cognitiva estão fortemente relacionados, portanto, a modelagem cognitiva do agente deve depender do histórico do agente cognitivo.

Nesta proposta, Bittencourt estabelece a ação coordenada de três níveis de ciclos de pensamento/ações:

1) um nível inferior denominado *nível reativo* que é composto por padrões extraídos de informações sobre o

mundo externo, controles que produzem alguma ação neste mundo e uma população de cromossomos que unem percepção e ação. Tais cromossomos quando submetidos a um processo de seleção natural, no qual a função de fitness está associada com as emoções do agente, permite ao agente aprender e adaptar-se ao ambiente.

2) um nível intermediário, denominado *nível instintivo* que possui um mecanismo de memória de longo prazo, de forma que quando o processo evolucionário avança no nível reativo e as situações começam a se repetir no mundo, torna-se possível identificar populações responsáveis por ações relevantes em uma determinada situação, abstrair suas propriedades e obter uma descrição geral de ação para uma determinada situação. Estas descrições são armazenadas pelo agente na memória de longo prazo e são acessadas mediante a percepção de situações similares.
3) um nível alto, denominado *nível cognitivo* é encarregado da manipulação das descrições gerais obtidas no nível instintivo, aplicando funções cognitivas de dedução, abdução e indução. Este nível é constituído por duas atividades complementares: o aprendizado através de descrições de situações relevantes, e a geração de novas estratégias de ação.

A proposta desta arquitetura é bastante ampla e genérica, o que torna sua implementação completa um sonho audacioso. Desta forma, para este trabalho, optamos por nos inspirar na idéia da coordenação das ações dos três níveis, porém fazendo uso de técnicas específicas voltadas ao domínio de um jogo de captura à bandeira. Desta forma, antes de introduzir as técnicas implementadas e a coordenação entre níveis, é importante definir o cenário de aplicação dos agentes.

III. Cenário de Aplicação

O sistema consiste em um jogo no estilo captura à bandeira, que acontece em uma pequena arena. Existem dois times, denominados time A e time B, cujos objetivos são a captura da bandeira inimiga e a defesa de sua própria bandeira. Vence a equipe que conseguir capturar a bandeira adversária primeiro. O mapa da arena, mostrado na figura 1, possui cinco obstáculos estáticos e diversas áreas delimitadas, denominadas regiões, por onde os agentes podem se movimentar. As áreas são enumeradas em uma sequência específica para ser usada na identificação da posição dos agentes e elaboração do plano. A caracterização das regiões é a seguinte:

- Região superior: áreas 1 a 5;
- Região central: áreas 6 a 9;
- Região inferior: áreas 10 a 14;
- Região de defesa do time A: áreas 15 a 20;
- Região de defesa do time B: áreas 21 a 26.

Cada time é composto por 3 agentes. Um agente pode atacar a bandeira adversária ou defender uma área, conforme suas competências (descritas na seção IV-A). Além disso, estabeleceu-se a figura de um agente líder, que define a estratégia do time e estabelece tarefas aos subordinados. As próximas seções apresentam a arquitetura específica desenvolvida para este domínio, descrevendo em linhas gerais as técnicas escolhidas e implementadas em cada nível, bem como a coordenação geral do sistema de agentes.

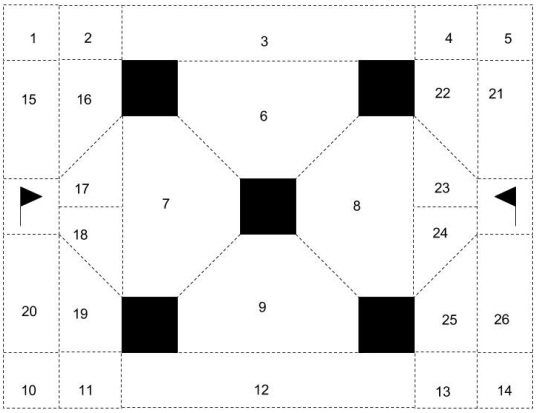


Figura 1. Representação das áreas e obstáculos da arena.

IV. Modelagem dos Agentes

A proposta do trabalho é a implementação de um sistema multiagente onde cada agente é definido por uma arquitetura em três níveis, conforme apresentado na seção II, e cujos comportamentos devem atender às demandas do domínio de aplicação descrito na seção III. Desta forma, é possível perceber que no nível cognitivo, o agente deve ser capaz de raciocinar sobre a estratégia do jogo, decidindo pelo ataque ou defesa; no nível instintivo, ele deve localizar-se na arena e coordenar seus movimentos conforme a decisão no nível cognitivo. E, de forma reativa, o agente deve buscar evitar colisões com os obstáculos estáticos, bem como deve *driblar* os oponentes ou deve colocar-se de forma a impedir que um agente adversário segue até a bandeira. Desta forma, tem-se para cada nível:

- Nível cognitivo: Para implementar o nível cognitivo foi escolhido o Modelo BDI - *Belief, Desire, Intention* [7] - pela facilidade em estabelecer planos e definir o comportamento do agente em alto nível, bem como pela simplicidade em estabelecer novos objetivos dada a dinâmica do jogo, através de seu ciclo de raciocínio bem estabelecido.
- Nível instintivo: Neste nível é utilizada uma busca em grafos com o algoritmo A*, que opera sobre grafos valorados realizando estimativa de custo para atingir um vértice destino. O algoritmo utiliza uma estratégia gulosa associada a uma função heurística para a estimativa de custo futuro [8], de forma a encontrar o melhor caminho para uma determinada área na arena.
- Nível reativo: O nível reativo foi implementado com o auxílio de um controlador *fuzzy* [9], [10], que recebe dados sobre a distância do agente a um determinado obstáculo e dependendo do comportamento (ataque ou defesa) impõe o próximo passo do agente.

A figura 2 ilustra os níveis da arquitetura e as técnicas implementadas em cada nível e as seções seguintes descrevem os detalhes de implementação de cada nível.

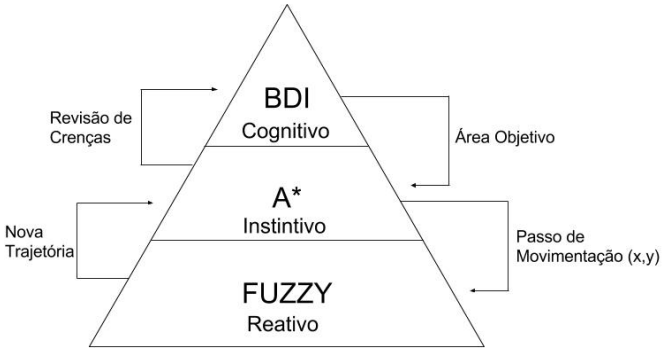


Figura 2. Estrutura da arquitetura proposta.

A. Nível Cognitivo

O nível cognitivo do agente é modelado utilizando *AgentSpeak* através do interpretador Jason, e suas ações são integradas com a aplicação Java utilizada para implementar a interface do jogo. Os agentes do sistema possuem crenças relativas às suas competências, à sua posição atual e às ordens recebidas do agente líder. As crenças relativas às competências de ataque e defesa dos agentes são fixas, ou seja, durante toda a execução os agentes sempre vão defender ou atacar as mesmas áreas. As crenças relativas à posição atual e às ordens vindas do agente lder mudam no processo de revisão de crenças dos agentes.

A comunicação dos agentes ocorre por meio de troca de mensagens individuais e *broadcast*. A troca de mensagens entre os agentes do mesmo time é sempre individual, o *broadcast* só é emitido quando um agente muda sua crença de posição atual, fazendo-se necessário informar a todos os agentes do ambiente sobre sua nova posição. As tabelas IV-A e IV-A mostram as diferenças nas configurações de cada time, o que distingue as competências de cada um.

	Time A		
	Líder	**Subord 1**	**Subord 2**
Crença de defesa:	áreas 15 a 20	áreas 16,18 e 20	não
Crença de ataque	não	não	sim
Elabora plano	sim	não	não

Tabela I
CONFIGURAÇÃO DO TIME A

O líder do time A fica observando a movimentação inimiga e identifica as áreas que precisam ser protegidas baseado na posição atual dos inimigos. Uma vez identificadas as áreas que devem ser protegidas, ele ordena que o subordinado 1 proteja a área que é capaz, enquanto o próprio líder protege outra. Enquanto o líder e o subordinado 1 estão sempre defendendo, o subordinado 2 está sempre focado em atacar.

No caso do time B, o líder fica observando a movimentação inimiga, ao identificar as áreas que precisam ser protegidas, ele tenta alocar os subordinados para defendê-las, caso um

	Time B		
	Líder	**Subord 1**	**Subord 2**
Crença de defesa:	áreas 21 a 26	áreas 22 e 23	24 e 25
Crença de ataque	não	sim	sim
Elabora plano	sim	não	não

Tabela II
CONFIGURAÇÃO DO TIME B

subordinado não seja capaz de defender nenhuma delas ele ordena que esse agente ataque. Caso os dois agentes sejam capazes de proteger a região, ele ordena que os dois protejam e seu time ficará sem atacar enquanto executa essa estratégia de defesa.

Ao elaborar uma estratégia, o agente líder emite uma ordem para seu subordinado, que por sua vez, gera uma ação Java a ser executada na aplicação. Quando um agente da aplicação Java recebe uma ação vinda de seu modelo cognitivo em Jason, é um indicativo que a partir de agora ele possui uma posição alvo definida. Então inicia o processo deliberativo de sua ação, ou seja, essa posição é enviada ao nível instintivo que deverá gerar uma trajetória até a área desejada.

B. Nível Instintivo

O objetivo do nível instintivo é gerar uma trajetória da posição atual do agente até uma posição de destino. Para efetuar essa tarefa é utilizado o algoritmo de busca A*, que opera percorrendo um grafo valorado gerado sobre as áreas da arena como exemplificado na figura 3.

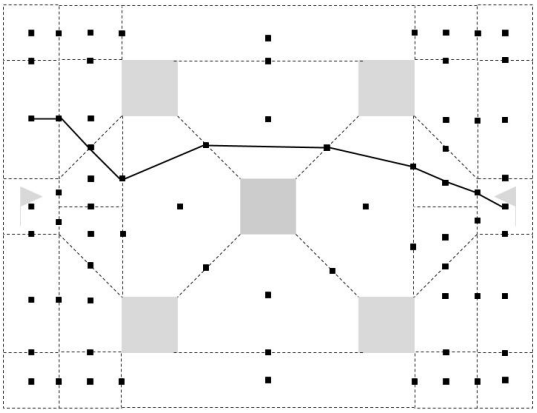


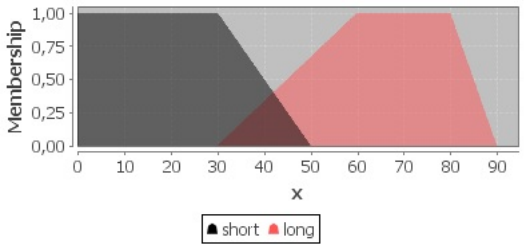
Figura 3. Exemplo de trajetória através dos pontos de controle .

Para cada área da arena sâo definidos alguns pontos de controle. Esses pontos são coordenadas bidimensionais que representam posições intermediárias que o agente usa para se deslocar, ou seja, representam os vértices do grafo utilizado como base para o planejamento da trajetória. Há também a necessidade de se calcular a distância entre dois pontos em dois momentos, e para isso usa-se a distância Euclidiana. O primeiro momento é na geração do grafo, onde o peso de cada aresta consiste na distância entre seus vértices. No segundo momento, cálculo da distância entre pontos é utilizado como função heurística para estimar a proximidade com a posição destino da trajetória.

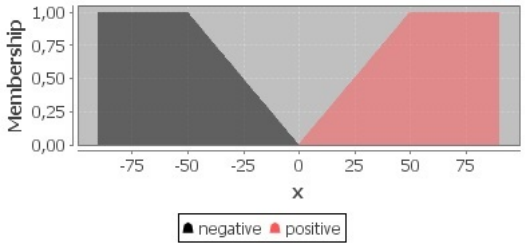
A cada atualização temporal no sistema o agente deve realizar um passo de movimentação, ou seja, um deslocamento parcial dentro da trajetória. Sempre que um passo de movimentação é gerado, ele é enviado ao nível reativo para que esse faça eventuais alterações, e também é feita uma verificação para identificar uma transição entre duas áreas. Ao identificar uma transição entre duas áreas, o nível instintivo deve informar ao nível cognitivo, para que este faça a revisão de crenças e elabore uma nova estratégia.

C. Nível Reativo

Para ser capaz de realizar as alterações no passo da trajetória, o nível reativo da arquitetura do agente utiliza um controlador fuzzy. A cada atualização da posição do agente, o nível recebe o passo de movimentação a ser aplicado e gera valores para as variáveis de entrada do controlador. O controlador é composto por cinco variáveis de entrada: distância, desvio no eixo x, desvio no eixo y, intenção e região, apresentadas nas figuras 4(a), 4(b), 5(a) e 5(b) respectivamente, onde é possível ver os universos de discursos definidos para cada variável bem como os conjuntos fuzzy definidos para cada universo.



(a) Distância.



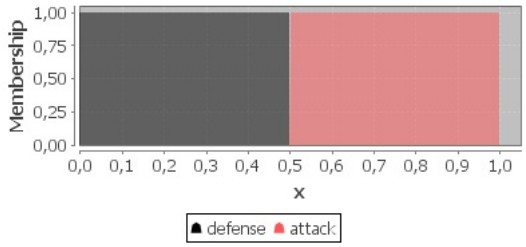
(b) Desvio nos eixos x e y.

Figura 4. Conjuntos fuzzy de entrada.

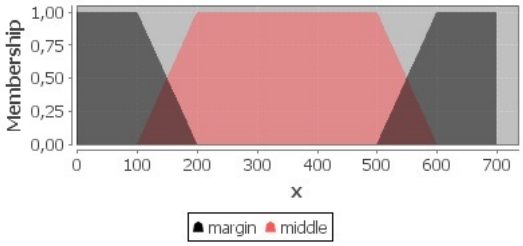
A variável distância trata da distância entre o agente e seu inimigo. A variável desvio é assinalada para os eixos x e y e indica a diferença na posição do agente com relação a trajetória estabelecida. A variável intenção define se o agente está atacando ou defendendo. A variável região define se o agente está no meio ou na borda de uma dada região.

Como variáveis de saída tem-se a movimentação no eixo x e a movimentação no eixo y que representam o valor do passo de movimentação do agente, sendo uma variável para cada eixo (x e y), apresentadas na figura 5.

Por fim, as regras fuzzy cobrem três casos: ataque pela região central, ataque pelas regiões de borda e defesa. No



(a) Intenção.



(b) Região.

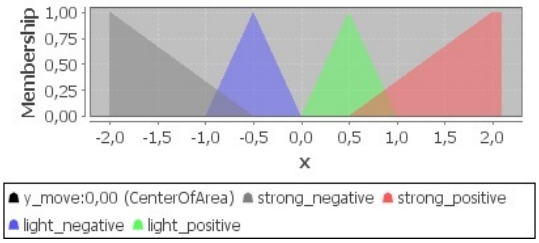


Figura 5. Conjunto fuzzy de saída: movimentação nos eixos x e y.

caso do ataque o agente mantém o valor do passo em um eixo, enquanto gera um valor de desvio para o outro eixo. O valor da movimentação no eixo selecionado será sempre no sentido contrário ao deslocamento do inimigo, com o objetivo de fazer o desvio se distanciando dele, e será um movimento brusco ou suave dependendo da distância entre ambos. Para a situação de ataque pela região central da arena, o eixo y é selecionado para desvio e o passo é mantido para o eixo x. No caso de um ataque pelas regiões de borda, tanto a verificação do desvio, como o valor do passo de movimentação levarão em consideração o eixo x. Quando a intenção do agente é de defesa, o comportamento é mais simples e não depende de tantas variáveis de entrada. As regras que levam em conta essa intenção basicamente vão gerar valores para a variáveis de saída que farão o agente defensor bloquear o caminho do atacante, e seguí-lo em caso de uma esquiva. O comportamento de defesa independe da região em que o agente se encontra.

O controlador tem um total de 12 regras, sendo 8 regras para o ataque e 4 para a defesa. No caso de ataque, o agente mantém o valor do passo em um eixo, enquanto gera um valor de desvio para o outro eixo. O valor da movimentação no eixo selecionado será sempre no sentido contrário ao deslocamento do inimigo, com o objetivo de fazer o desvio e se distanciar. O movimento pode ser forte ou leve dependendo da distância

entre ambos. Para a situação de ataque pela região central da arena, o eixo selecionado para desvio e mantido o passo para o eixo x, e as regras utilizadas são as seguintes:

```
R1:  IF region IS middle AND distance IS short
     AND
     yDeviation IS negative AND intention IS
     attack THEN
     yMove IS strongPositive;
R2:  IF region IS middle AND distance IS short
     AND
     yDeviation IS positive AND intention IS
     attack THEN
     yMove IS strongNegative;
R3:  IF region IS middle AND distance IS long AND
     yDeviation IS negative AND intention IS
     attack THEN
     yMove IS lightgPositive;
R4:  IF region IS middle AND distance IS long AND
     yDeviation IS positive AND intention IS
     attack THEN
     yMove IS lightNegative;
```

No caso de um ataque pelas regiões de borda, tanto a verificação do desvio, como o valor do passo de movimentação levarão em consideração o eixo x. Seguem as regras para a movimentação por essas regiões:

```
R5:  IF region IS margin AND distance IS short
     AND
     xDeviation IS negative AND intention IS
     attack THEN
     xMove IS strongPositive;
R6:  IF region IS margin AND distance IS short
     AND
     xDeviation IS positive AND intention IS
     attack THEN
     xMove IS strongNegative;
R7:  IF region IS margin AND distance IS long AND
     xDeviation IS negative AND intention IS
     attack THEN
     xMove IS lightgPositive;
R8:  IF region IS margin AND distance IS long AND
     xDeviation IS positive AND intention IS
     attack THEN
     xMove IS lightNegative;
```

Quando a intenção do agente é de defesa, o comportamento é mais simples e não depende de tantas variáveis de entrada. As regras que levam em conta essa intenção basicamente vão gerar valores para a variáveis de saída que farão o agente defensor bloquear o caminho do atacante, e segui-lo em caso de uma esquiva. O comportamento de defesa independe da região em que o agente se encontra, conforme mostram as regras abaixo:

```
R9:  IF xDeviation IS negative AND intention IS
     defense THEN xMove IS lightNegative;
R10: IF xDeviation IS positive AND intention IS
     defense THEN xMove IS lightPositive;
R11: IF yDeviation IS negative AND intention IS
     defense THEN yMove IS lightNegative;
R12: IF yDeviation IS positive AND intention IS
     defense THEN yMove IS lightPositive;
```

O nível reativo é ativado quando dois agentes se aproximam, atingindo uma certa distância. Ao identificar que o agente atacante conseguiu passar pelo defensor, o nível reativo comunica-se com o nível instintivo, caracterizando o processo de interação *bottom-up* na arquitetura.

D. Coordenação dos níveis funcionais da arquitetura

Dados os seus três níveis, as ações devem ser coordenadas de forma a assegurar o comportamento do agente. Desta forma há dois fluxos: o fluxo *top-down* e o fluxo *bottom-up*. O fluxo *top-down* inicia-se no nível cognitivo, que ao gerar uma intenção passa uma área alvo para o nível instintivo, que por sua vez, após gerar a trajetória envia o passo de movimentação para o nível reativo aplicar eventuais alterações. Eventualmente, o agente vai ter que refazer sua trajetória, além de fazer revisão de crença para alterar seu objetivo. Para isso é necessário definir também uma comunicação no sentido contrário (fluxo *bottom-up*) entre os níveis da arquitetura.

Quando o nível reativo é ativado, ele faz alterações no passo de movimentação a cada ciclo temporal da aplicação. Mas ele deve ser capaz de perceber quando não há mais necessidade de fazer alterações na trajetória. Então, quando o nível reativo identifica que a posição do agente ultrapassou a de seu inimigo no eixo definido, este indica ao nível instintivo que ele deve elaborar uma nova trajetória, partindo da posição atual do agente, e mantendo o mesmo destino. Por sua vez, o nível instintivo, após a aplicação de um passo de movimentação, deve sempre verificar se a mudança de posição resultou em uma transição para outra área da arena. Sempre que for identificada uma transição, o nível instintivo envia uma notificação para o nível cognitivo, para que este faça revisão de suas crenças, e adicione uma crença da sua nova posição atual.

Por fim, o nível cognitivo ao atualizar sua crença de posição atual, irá enviar uma mensagem *broadcast* aos demais agentes do ambiente, informando sua movimentação. Ação esta que fará com que o líder inimigo refaça o plano de ação de seu time, completando o processo inverso das ações da arquitetura e recomeçando o fluxo *top-down*.

V. Validação

Para realizar a validação da estrutura proposta, foram executados testes isolados nos níveis cognitivo e instintivo, depois um teste de integração entre os níveis cognitivo e instintivo e por fim a integração completa do sistema. A seguir, apresentamos um cenário de teste do nível cognitivo e um cenário de teste do nível instintivo. Em seguida apresenta-se a integração entre o cognitivo e o instintivo - onde é possível ver o agente criando a trajetória baseado no objetivo gerado no nível cognitivo e também a alteração da estratégia dos times; e um cenário de teste da arquitetura como um todo, onde é possível ver todas as suas propriedades, além de realizar todas as ações do cenário anterior, o agente nessa execução realiza alterações na trajetória de acordo com a necessidade, e posteriormente retoma sua trajetória.

Para as execuções a seguir são instanciados dois times, o time A está representado pela cor azul é instanciado no lado esquerdo da arena, enquanto o time B é representado pela cor vermelha e é instanciado ao lado direito da arena.

A. Cenário 1: Teste do Nível Cognitivo

O primeiro cenário a ser apresentado é do nível cognitivo funcionando isoladamente. Primeiramente os agentes são iniciados e os líderes de cada time elaboram um plano inicial, como na figura 6 são mostradas as ações acontecendo na saída da execução. Enquanto o líder A e subordinado A_1 defendem as áreas 17 e 18 respectivamente, o subordinado A_2 ataca

a bandeira inimiga na área 28. Já o plano inicial elaborado pelo líder do time B consiste em ordenar ao subordinado B_1 defender a área 23, ao subordinado B_2 defender a área 24, enquanto ele próprio defende a área 21.

Figura 6. Nível cognitivo rodando isoladamente em Jason.

Após os agentes executarem as ações definidas no plano inicial, o subordinado B_1 muda sua posição para área 3, emitindo uma mensagem *broadcast* para os demais agentes. Ao receber a mensagem o líder A elabora um plano de ação de seu time para reagir a ação do agente inimigo. Finalizando com sucesso um ciclo de ações dentro do nível cognitivo.

B. Cenário 2: Nível Instintivo

O segundo cenário é uma execução bastante simples, onde um agente é instanciado em uma posição arbitrária da arena. O nível instintivo age gerando uma trajetória até a área da bandeira no extremo oposto da arena. A cada atualização temporal o nível instintivo impõe um passo de movimentação ao agente seguindo a trajetória criada. Ao final da execução, como mostra a figura 7, o agente obtém sucesso ao atingir a posição destino.

C. Cenário 3: Teste de integração - Cognitivo/Instintivo

Após os agentes serem instanciados, os líderes de cada time elaboram sua estratégia inicial. Como mostra a figura 8 o líder do time A e um de seus subordinados defendem

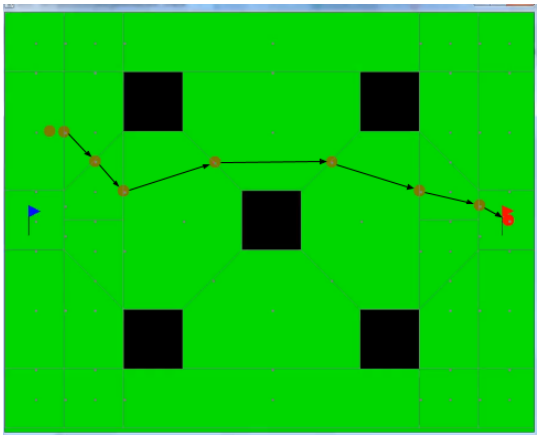


Figura 7. Agente realizando planejamento de trajetória no nível instintivo.

as áreas centrais de acesso à sua bandeira, enquanto o outro subordinado se move em direção à região em que se encontram os inimigos. A estratégia do time B é totalmente defensiva, os agentes subordinados defendem as áreas centrais de acesso à sua bandeira, enquanto o líder defende uma das áreas de acesso superiores.

As áreas objetivo das trajetórias vêm do nível cognitivo. Para este cenário, o subordinado do time A que se desloca para a região inimiga, tem como objetivo a área do canto superior direito da arena. Sempre que este agente atravessa de uma área para outra é emitido um *broadcast* avisando os demais agentes da aplicação, então, no momento que este agente passa para a região superior da arena, o líder do time B identifica a necessidade de alterar o plano de ação. A alteração do plano de ação do time B, ilustrada na figura 9 consiste em ordenar ao subordinado B_1 que proteja a área de acesso superior, enquanto o subordinado B_2 deve atacar a bandeira inimiga.

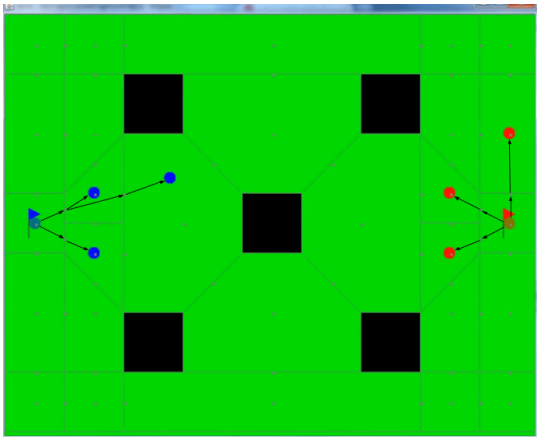


Figura 8. Times executando a estratégia inicial na integração entre os níveis cognitivo e instintivo.

Portanto, o cenário mostra que os dois níveis da aplicação integrados geram o comportamento que se esperava dos agentes, mas ainda falta o terceiro nível da arquitetura para que a

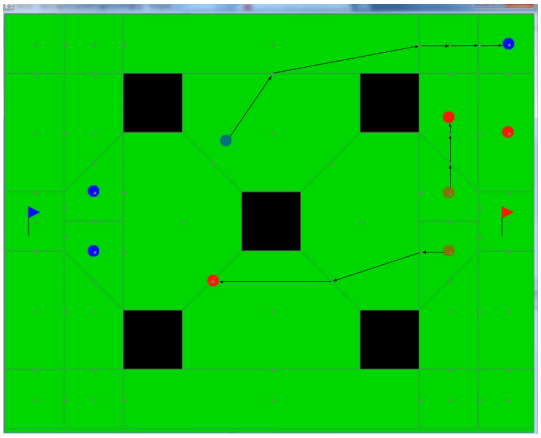


Figura 9. Time B alterando sua estratégia na integração entre os níveis cognitivo e instintivo.

integração seja validada por completo.

D. Cenário 4: Integração completa dos três níveis

No último cenário, a estratégia inicial do time B consiste em defender as áreas superiores, enquanto um de seus subordinados ataca pela região central. O time A inicialmente defende as áreas de acesso centrais enquanto seu subordinado ataca pela região central como mostra a figura 10. Ao receber a informação via *broadcast* de que o subordinado do time A entrou na região central, o líder do time B reformula seu plano de ação, passando a ter uma postura totalmente defensiva, como mostra a figura 11, quando então ordena que seus dois subordinados defendam as áreas centrais de acesso à sua bandeira.

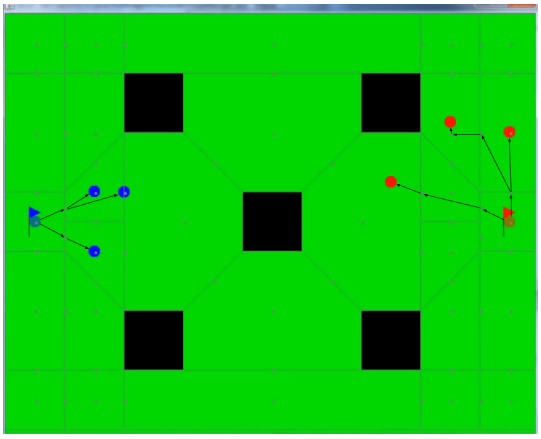


Figura 10. Times executando a estratégia inicial na integração entre os três níveis.

O subordinado A persiste na sua movimentação de ataque pela região central, ativando por fim, o nível reativo, como mostra a figura 12. Neste momento, a cada ciclo temporal, o controlador *fuzzy* é executado e passo de movimentação é alterado conforme as regras nebulosas. Sendo assim, o subordinado do time A começa a se deslocar em direção

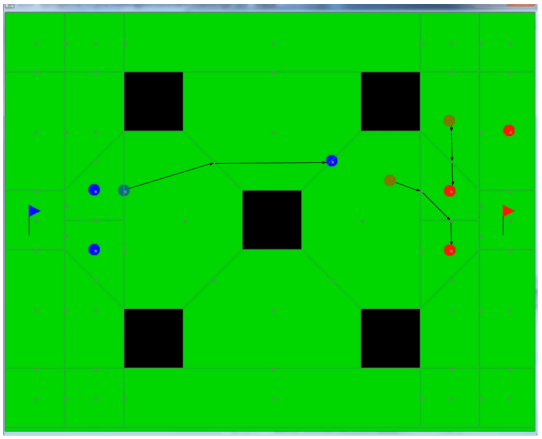


Figura 11. Time B alterando sua estratégia na integração entre os três níveis.

contrária ao posicionamento de seu inimigo, enquanto o agente do time B começa a seguí-lo. Ao verificar que conseguiu passar por seu inimigo, o agente do time A reformula sua trajetória visando a bandeira inimiga. Nesse momento o líder do time B identifica a proximidade do inimigo e passa a seguí-lo também, mas sem sucesso em alcançá-lo. Dessa forma o agente do time A consegue alcançar a bandeira inimiga, finalizando a execução.

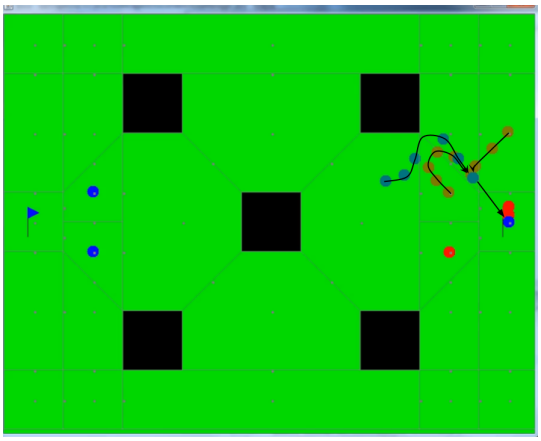


Figura 12. Agentes reagindo ao movimento do inimigo na integração entre os três níveis.

Nesse cenário é possível perceber todas as propriedades da arquitetura. No sentido *top-down*, os times formulam suas estratégias e os agentes geram áreas de destino. Com um destino definido o nível instintivo gera uma trajetória, e o nível reativo altera o passo de movimentação quando identifica a proximidade de um inimigo. E no sentido *bottom-up*, quando o agente identifica que conseguiu desviar do inimigo, imediatamente indica ao nível instintivo que deve ser gerada uma nova trajetória. Por fim, ao transitar de uma área para outra, o nível instintivo comunica ao cognitivo a alteração da posição atual, que acontece quando o time B muda sua estratégia para

uma postura totalmente defensiva.

VI. Conclusão

O objetivo deste trabalho foi a implementação coordenada de técnicas distintas de IA em uma arquitetura multinível, onde cada nível torna-se responsável por parte do comportamento do agente e suas ações integradas permitem a elaboração de um comportamento inteligente mais elaborado. Como visto na validação, a proposta alcançou o comportamento esperado dos agentes. Foram fatores limitantes do trabalho alguns problemas decorrentes da integração Jason/Java. Dentre os problemas encontrados, estão:

(i) interferência na comunicação dos agentes - o *loop* principal do jogo, implementado em Java, provocou problemas na troca de mensagens entre os agentes modelados em Jason. Ao elaborar um plano, os líderes emitiam mensagens, mas seus subordinados não as recebiam. Este problema foi solucionado através da criação de uma ação Java que os líderes emitem para interromper a execução do *loop*, retomando-a após realizada a comunicação.

(ii) referência nula completa ou incompleta na inicialização dos agentes - na ocorrência deste erro, os agentes do sistema não são, total ou parcialmente, inicializados, resultando em times incompletos ou na arena vazia. Este problema acontece de forma aleatória e não nos foi possível identificar a sua causa. No entanto, tais erros estão relacionados com a integração Java/Jason, e não com o funcionamento da arquitetura, que demonstrou-se estável em todas as execuções corretamente inicializadas.

De modo geral, foi possível constatar que o uso conjunto de técnicas coordenadas permite que se obtenha comportamentos complexos, desonerando o nível cognitivo e permitindo o surgimento de comportamentos reativos e/ou instintivos que podem lidar com situações onde o objetivo previamente estabelecido para o agente ainda é válido.

A. Trabalhos Futuros

Como trabalhos futuros, sugere-se a investigação mais profunda da integração Java/Jason visando sanar os problemas citados. Neste trabalho foi visto que é possível utilizar a arquitetura em um ambiente específico, mas não foi atribuído nenhum requisito de desempenho nas ações dentro das regras do jogo, o que poderia ser melhor explorado. Ainda é possível estudar melhor a composição das técnicas, com o objetivo de fazer implementações mais complexas e por fim, seria interessante inserir alguma forma de aprendizado com o objetivo de fazer com que a arquitetura apresente maior competitividade.

Referências

[1] G. Bittencourt, (1997). In the quest of the missing link. *Proceedings of IJCAI, Nagoya, Japan, August 23-29, pages 310-315. Morgan Kaufmman(ISBN 1-55860-480-4)*.

[2] M. Wooldridge, (2002b). *An Introduction to Multiagent Systems*. John Wiley and Sons.

[3] N. R. Jennings, (2000). On agent-based software engineering. *Artificial Intelligence*, 117.

[4] M. McShaffry and D. Graham, (2013). *Game Coding Complete*. Course Technology, a part of Cengage Learning, 4th edition.

[5] B. Schwab, (2009). *AI Game Engine Programming*. Course Technology, a part of Cengage Learning, 2nd edition.

[6] G. Bittencourt and J. Marchi, (2006). *Artificial Cognition Systems*, chapter An Embodied Logical Model for Cognition, pages 27–64. IDEA Group Inc.

[7] M. Wooldridge, (2002a). Intelligent agents. *Multi-Agent Systems and Applications II*.

[8] S. J. Russel and P. Norvig, (2003). *Artificial Intelligence*. Prentice Hall, Upper Saddle River, New Jersey, 2nd edition.

[9] R. Tanscheit, (2003). Sistemas fuzzy. *Anais de Minicursos do VI SBAI - Simpósio Brasileiro de Automação Inteligente*.

[10] G. Chen and T. T. Pham, (2001). *Introduction to Fuzzy Sets, Fuzy Logic and Fuzzy Control Systems*. CRC Press.