UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Dissertação de Mestrado

Benchmark Multiagente em Ambiente de Simulação de Futebol de Robôs

Telmo dos Santos Klipp

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande - FURG, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Computação

Orientador: Prof. Dr. Eder Mateus Nunes Gonçalves

Rio Grande, 2015

K65b Klipp, Telmo dos Santos.
Benchmark Multiagente em Ambiente de Simulação de Futebol de Robôs / Telmo dos Santos Klipp. – 2015.
150 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande - FURG, Centro de Ciências Computacionais, Programa de Pós-graduação em Engenharia de Computação, Rio Grande/RS, 2015.
Orientador: Eder Mateus Nunes Gonçalves.

1. Sistemas multiagente. 2. Coordenação multiagente. 3. Futebol de robôs. 4. Coalizões. 5. Benchmark. I. Gonçalves, Eder Mateus Nunes. II. Título.

CDU 004.891

Catalogação na Fonte: Bibliotecário Me. João Paulo Borges da Silveira CRB 10/2130

UNIVERSIDADE FEDERAL DO RIO GRANDE
Centro de Ciências Computacionais
Programa da Pós-Graduação em Computação
Curso de Mestrado em Engenharia de Computação

DISSERTAÇÃO DE MESTRADO

Benchmark em Ambiente de Simulação de Futebol de Robôs

Telmo dos Santos Klipp

Banca examinadora:

Prof. Dr. Augusto Loureiro da Costa (UFBA)

Prof. Dr. Diana Adamatti (FURG/C3)

Prof. Dr. Raquel de Miranda Barbosa (IF-RS)

Prof. Dr. Eder Mateus Nunes Gonçalves (FURG/C3)
Orientador(a)

Aos meus pais, Nauro José e Rosa Honorata,
e ao meu irmão Elton Davi.

AGRADECIMENTOS

Agradeço a Deus por me conceder a vontade livre e a vida da qual desfruto todos os dias. Agradeço aos meus pais que sempre me apoiaram concedendo-me educação para a vida.

Agradeço aos caros colegas que conheci durante o mestrado. Em especial, ao Lianderson, com o qual pude dividir as responsabilidades do mestrado, e que, frequentemente, tinha ensinamentos a me passar sobre sua própria vivência. Um agradecimento especial também ao caro colega Cristiano, com o qual igualmente dividi responsabilidades do mestrado e inclusive discussões sobre várias matérias da vida.

Aos colegas membros da sala Aquário que foram companhia presente no dia a dia de pesquisas e nos momentos de festa. Aos caros Alex, Amanda, Bruno, Felipe, Felipe, Joel, Leander, Luan, Matheus, Michael, Ricardo e Sidnei. Em especial também à colega Yuni, pelos incentivos a realização do trabalho e preocupações sinceras; e, ao colega Eduardo, pelo seu humor deliberado.

Agradeço ao meu professor orientador, Eder Mateus Gonçalves, por me acolher na empreitada do mestrado e por suas ponderações ao desenvolvimento do trabalho; ponderações estas que, vez por outra, deparava-me como sendo as mais corretas. Sobretudo, agradeço por sua paciência em me permitir, ao meu tempo, alcançar os objetivos necessários para que este trabalho se tornasse possível.

Aos professores participantes da minha banca de qualificação, Dr. Augusto Loureiro da Costa, Drª Diana Adamatti e Drª Raquel Barbosa. Embora meu trabalho estivesse, naquela ocasião, com cara apenas de prelúdio, a banca se esforçou em me indicar os acertos a serem feitos, as fontes a serem perseguidas, e alertar para o tamanho da empreitada.

À FURG por seu programa de mestrado (PPGComp) e aos seus professores que inspiram na busca do conhecimento. A CAPES e ao CNPq por possibilitarem que meus esforços fossem direcionados na realização do mestrado.

Mais uma vez, um agradecimento às pessoas que conheci e com as quais dividi instantes de vida. Considero que, nesta vida, inteirar-se das pessoas e aprender com tantas mentes diferentes tem sido a tarefa de maior proveito a realizar.

A mind that is stretched by a new experience
can never go back to its old dimensions.
— Oliver Wendell Holmes, Jr.

RESUMO

KLIPP, Telmo dos Santos. **Benchmark Multiagente em Ambiente de Simulação de Futebol de Robôs**. 2015. 150 f. Dissertação de Mestrado – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

O desenvolvimento de sistemas com complexidade necessária a uma abordagem multiagente carrega consigo também aspectos de complexidade relacionados à avaliação dos diferentes níveis e componentes desse sistema. Um sistema multiagente pode ser composto por uma série de agentes heterogêneos, que apresentam variabilidade quanto à sua arquitetura interna, modelos utilizados para o seu desenvolvimento, linguagem de programação, de especificação e validação. Agregam-se a isso, contextos específicos de cada solução para com o ambiente para o qual foi projetado. Deste modo, impõem-se mecanismos que permitam estabelecer métricas de avaliação para cada nível do desenvolvimento de um sistema multiagente, considerando dimensões como organização, comunicação entre agentes e os agentes em si. Esta dissertação apresenta como problemática, o estabelecimento de um *benchmark* para sistemas multiagente dentro do simulador de futebol de robôs *Soccer Server 2D*. Mais especificamente, este *benchmark* deve prover métricas e mecanismos de avaliação de esquemas organizacionais multiagente segundo os diferentes cenários que podem se estabelecer dentro da dinâmica de uma partida de futebol. Não obstante, deve-se permitir o estabelecimento de referências de avaliação da coletividade dos times implementados para o *Soccer Server 2D*, indiferente aos demais níveis de concepção do sistema.

Palavras-chave: Sistemas multiagente, coordenação multiagente, futebol de robôs, coalizões, benchmark.

ABSTRACT

KLIPP, Telmo dos Santos. **Multi-agent benchmark in a simulation environment for robot soccer**. 2015. 150 f. Dissertação de Mestrado – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

The development of systems with the required complexity for a multi-agent approach, also carries complexity aspects related to the evaluation of different levels and components of such a system. A multi-agent system can be composed of a series of heterogeneous agents, which have variability regarding its internal architecture, models used for its development, programming language, specification and validation. Added to this, are situated the particular contexts of each solution towards the environment for which it is designed. Therefore, it is needed mechanisms to establish evaluation metrics for each multi-agent system development level, taking into account dimensions such as organization, communication between agents and the agents themselves. This dissertation presents as a problem, establish a benchmark for multi-agent systems within the robot soccer simulator Soccerserver 2D. Specifically, this benchmark should provide metrics and evaluation mechanisms of multi-agent organization schemes according to different scenarios that can be established within the dynamics of a football match . Nevertheless, it should be allowed the establishment of assessment referrals for the Robocup teams collectivity, regardless to other levels of system design.

Keywords: Multi-agent systems, multi-agent coordination, robot soccer, coalitions, benchmark.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUT	Agent Under Test
CLang	Coach Lang
DSA	Distributed Stochastic Search
ETL	Electro Technical Laboratory
IA	Inteligência Artificial
IJS	Institute Jožef Stefan
MDPs	Markov Decision Processes
RoboCup	Robot World Cup Initiative
SBSP	Situation Based Strategic Positioning
SMA	Sistemas Multiagente
SPAR	Strategic Position by Attraction and Repulsion
SPSA	Simultaneous Perturbation Stochastic Approximation
SST	Soccer Scientia Tool
SSST	Soccer Server Statistical Extracting Tool
TA	Team Assistant

SUMÁRIO

1 INTRODUÇÃO

Problemas computacionais modernos apresentam requisitos que impõem soluções de alta escalabilidade, distribuição lógica e/ou geográfica de componentes, e a capacidade de incorporar soluções já existentes em contextos menores destes problemas. Sistemas multiagente (SMA) (FERBER, 1999) apresentam-se como um modelo capaz de gerar soluções que atendam a estes requisitos, seja como modelo de projeto, ou até mesmo por meio de ferramentas e/ou métodos para a devida implementação e validação destes sistemas.

Na sua forma mais simples de concepção, um SMA é um aglomerado de agentes que são organizados de modo a atender a uma série de objetivos dentro de um determinado ambiente. Estes objetivos podem ter uma dimensão individual, dos agentes que se beneficiam do grupo para realizar suas ações; ou uma dimensão social, um conjunto de objetivos globais que uma vez instanciados pelos agentes levam o sistema como um todo aos estados desejados.

A concepção de um SMA deve tratar de pelo menos três dimensões básicas: a organização do sistema, os aspectos de comunicação entre os agentes, e a formalização dos agentes em si (definição da arquitetura interna e métodos de implementação) (ROSA et al., 2013). Cada uma destas dimensões confere uma complexidade específica ao sistema, com seus próprios parâmetros e mecanismos de avaliação.

Além disso, a interação entre os agentes, e por consequência do SMA com o ambiente é única, na medida em que este possui seus próprios parâmetros e métricas, que são devidamente instanciados por meio de metas, desejos e intenções, independente do modelo de SMA e agentes utilizados. Nesse sentido, o ambiente é um componente relevante no que diz respeito à avaliação do desempenho do sistema, tendo em vista que define o que deve ser otimizado quanto aos mecanismos de atuação nele.

Um exemplo desta problemática dá-se quando da especificação, projeto, desenvolvimento e validação de um SMA para atuar no simulador *Soccer Server 2D* (CHEN et al., 2003). O *Soccer Server 2D* é um simulador de partidas de futebol entre robôs, que atuam em um ambiente de duas dimensões. Uma equipe é composta por onze robôs, que podem ser programas independentes sob diferentes arquiteturas. Para vencer uma partida,

os robôs devem ser organizados e coordenados sob uma estratégia coletiva de modo a potencializar suas ações individuais. Neste caso, conceber uma equipe significa definir:

- uma ou mais arquiteturas de agente, que incluem aspectos reativos e deliberativos de tomada de decisão;
- ferramentas e linguagens a serem utilizadas para a implementação dos agentes;
- um modelo de organização entre os agentes, de modo a definir estratégias coletivas, jogadas, metas, considerando os papéis e as funções de cada agente no time;
- técnicas/modelos para concepção de sistemas inteligentes.

Cada item destes, apresentados acima, impõe seus próprios parâmetros e métodos de avaliação. Saber exatamente qual componente está funcionando dentro de requisitos mínimos e qual não está é um problema a ser resolvido de modo a garantir a validação de um SMA como um todo.

O próprio simulador *Soccer Server 2D* surgiu com a premissa de ser uma plataforma de teste para a implementação e avaliação de técnicas da Integilência Artificial (IA) e tecnologias multiagente. A partir do simulador, as equipes de pesquisa testam seus esforços. Juntamente com o *Soccer Server 2D* foi criada a competição anual *Simulation League*, onde os times de futebol de robôs competem.

Embora o simulador e a competição indiquem os times, que se sagrando vencedores, supostamente têm implementadas as tecnologias mais efetivas e os SMA melhor equacionados na qualidade de times, não necessariamente signifique que os times com piores resultados possuem tecnologias dispensáveis. Torna-se necessária a produção de novos tipos de avaliações.

Seguindo esse contexto, notar-se-á que muitas das abordagens de avaliações desenvolvidas para o simulador *Soccer Server 2D* cobrem de forma mais abrangente as características individuais dos agentes e estatísticas de jogo. Com os já decorridos anos da competição *Simulation League*, muitas das características cognitivas e reativas dos agentes já alcançaram um nível de maturação dentro dos times de futebol de robôs. Assim, muitas das equipes de pesquisa se voltam para questões de estratégias de time, da coordenação dos agentes e demais aspectos sociais. Isso traz como prerrogativa a necessidade da construção de abordagens de avaliação mais voltadas a características de SMA, enquanto time de futebol, como um todo.

Neste trabalho será proposta uma abordagem de avaliação destinada a verificar a ocorrência de estruturas organizacionais denominadas coalizões. Uma coalizão é uma formação organizacional em que dois ou mais agentes cooperam momentaneamente para alcançar um objetivo em comum (HORLING; LESSER, 2004). No futebol de robôs

simulado, as coalizões podem emergir explicitamente quando da execução de jogadas ensaiadas, formações de ataque ou defesa e execução de planos coordenados.

Pretende-se verificar se as coalizões são benéficas aos times. Também existe a intenção de que, mesmo indiretamente, seja possível verificar quais tecnologias, implementadas pelas equipes de pesquisa em seus times, permitem a formação de mais coalizões.

1.1 Objetivos

Esta seção apresenta o objetivo geral e os objetivos específicos deste trabalho.

- **Geral**

  Definir um conjunto de indicadores que permitam avaliar a estrutura organizacional de um SMA enquanto time de futebol no ambiente *Soccer Server 2D*, mais precisamente na formação de coalizões.

- **Específicos**

  Para se atingir o objetivo geral proposto neste trabalho, faz-se necessário o cumprimento dos seguintes objetivos específicos:

  - delinear o papel do ambiente na análise de desempenho de um SMA;
  - definir meios para a distinção do contexto social do contexto individual quanto às avaliações do sistema;
  - definir uma abordagem de identificação e avaliação de coalizões em times implementados para o simulador *Soccer Server 2D*;
  - definir as métricas para a abordagem de avaliação das coalizões;
  - aplicar a abordagem de avaliação por meio de experimentos.

1.2 Motivação e justificativa

A *Robot World Cup Initiative (Robocup)* (ROBOCUP, 1997) já tem um histórico de contribuições que a habilita como um ambiente para a avaliação de diferentes técnicas voltadas ao desenvolvimento de robótica e sistemas inteligentes. Em quase vinte anos desde sua criação, inúmeras contribuições científicas e tecnológicas já foram geradas, no sentido de sua meta final: uma equipe de robôs humanoides apta a uma partida de futebol contra a seleção campeã mundial de futebol.

Contudo, a concepção de uma equipe, ou mesmo de um único robô, apta a jogar futebol na plenitude de suas demandas motoras e estratégicas, impõe restrições de diferentes níveis de complexidade. Assim, a tarefa de construção de um artefato computacional que permita chutar a bola deve estar integrada a soluções de deliberação e comunicação que

permitam inferir em que contexto uma ação deste tipo deve ser tomada. Não obstante, a integração de um conjunto de atuadores que respondam a todas as demandas motoras que se espera de um jogador de futebol eleva o número de componentes necessários para a construção de um sistema completo, agregado às possibilidades oriundas de estratégias sociais de resolução do problema.

Visto em um contexto mais genérico, sistemas de computação modernos devem ser modelados a partir da interação de uma série de componentes, onde cada componente agrega algum tipo de valor no desempenho final do sistema (SUCH et al., 2007). Conseguir avaliar o desempenho de cada componente, seja de forma isolada ou integrado a um subconjunto destes componentes, viabiliza mecanismos necessários à validação e verificação das soluções geradas frente aos requisitos do ambiente (EATON; COLLINS; SHEEHAN, 2001).

Nesse contexto justifica-se a definição de um conjunto de parâmetros e cenários em que os componentes de uma solução multiagente possam ser isolados e avaliados. Neste caso, pretende-se avaliar características organizacionais de um SMA enquanto time de futebol.

1.3 Metodologia

O projeto de um sistema computacional, seja desenvolvido pelo paradigma multiagente ou outro paradigma qualquer, dá-se no sentido de maximizar e/ou minimizar uma série de parâmetros do ambiente. Eventualmente, subconjuntos destes parâmetros podem operar em contraposição a outro subconjunto de modo que um equilíbrio nos mecanismos de otimização deve ser encontrado.

Para o *Soccer Server 2D*, deve-se mapear estes parâmetros, e qual a relação destes com os diferentes componentes do agente cognitivo, seja nos aspectos sociais ou individuais.

Uma vez identificados os componentes do agente cognitivo e suas relações com o desempenho do mesmo no *Soccer Server 2D*, uma análise restrita aos componentes do SMA será realizada. Para que seja possível uma análise robusta visando avaliar comportamentos individuais e sociais serão identificadas métricas e parâmetros que isoladamente se relacionam respectivamente com: o desempenho individual dos agentes e o desempenho da equipe.

Concomitante a isso, será realizada pesquisa bibliográfica das abordagens de avaliação instituídas no *Soccer Server 2D* ante aos aspectos individuais e sociais em conformidade com as características do simulador, com o problema de futebol, e com as nuances de SMA. Levar-se-á em consideração técnicas e ferramentas que foram produzidas para compor uma abordagem de avaliação voltada mais para o aspecto social.

Destaca-se que a abordagem que será instituída corresponde ao nível organizacional, por se tratar da identificação e avaliação da formação de coalizões entre agentes. Os

padrões individuais dos agentes relativos às trocas de passes serão expandidos para o reconhecimento de comportamentos sociais definidos na formação de coalizões, que também indicarão o nível de coletividade dos times.

Uma das principais questões é verificar que características das coalizões tornam uma equipe de futebol mais apta a conquistar vitórias. A metodologia de teste envolverá a análise de partidas entre as equipes que participaram da competição *Robot Soccer 2D League* do ano de 2014. As disputas ocorreram somente entre equipes desse ano, para facilitar os preparativos para a aplicação do teste de validação da abordagem de avaliação e considerar a equidade de condições entre os times.

1.4 Organização do trabalho

Este documento, além desta introdução, está organizado em mais sete capítulos que versam sobre os seguintes conteúdos:

No Capítulo 2 serão introduzidos os conceitos básicos relativos à área de sistemas multiagente, sendo estes: agente, comunicação, organização e ambiente. Serão apresentados também os conceitos relativos à coordenação de SMA. Ao final, serão revisadas as abordagens de teste até então empregadas para verificar, avaliar e validar SMA.

No Capítulo 3 discorrer-se-á sobre a organização RoboCup e seu principal objetivo: fomentar o desenvolvimento de uma equipe de robôs humanoides apta a uma partida de futebol contra o campeão mundial de futebol. Será apresentado o simulador de futebol de robôs *Soccer Server* conforme seu propósito de ser uma plataforma de teste e um ambiente para a atuação de agentes.

No Capítulo 4 serão abordados trabalhos relacionados diretamente às avaliações de agentes no *Soccer Server 2D*. Assim como, serão discutidos alguns trabalhos relacionados à análise e avaliação do futebol praticado por humanos, pois alguns de seus conceitos são igualmente aplicáveis ao futebol de robôs.

No Capítulo 5 serão apresentadas algumas ferramentas que instituíram abordagens de avaliação para o simulador *Soccer Server 2D*, sendo que todas as ferramentas relacionadas fazem uso dos arquivos de *log* fornecidos pelo seu servidor.

No Capítulo 6 serão feitas algumas considerações sobre aspectos coletivos e sociais de times, e sobre a execução de jogadas conjuntas entre os jogadores. Serão também apresentadas as técnicas idealizadas para viabilizar o reconhecimento e a avaliação da formação de coalizões entre os agentes dos times implementados para o simulador *Soccer Server 2D*.

No Capítulo 7 serão descritos os testes realizados para a aplicação da abordagem de avaliação, nos quais foram utilizados os times que participaram da competição *Simulation League*, na edição da *RoboCup* de 2014. Posteriormente, serão demonstrados e discutidos os resultados junto de análises empíricas fundamentadas na observação de algumas das

partidas utilizadas nos testes, e nas descrições das tecnologias utilizadas nos times por parte das equipes de pesquisa que os implementaram.

No Capítulo 8 serão realizadas as considerações finais e definidas as direções a serem tomadas em trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo serão introduzidos os conceitos básicos relativos à área de sistemas multiagente (SMA), sendo estes: agente, comunicação, organização e ambiente. Serão apresentados também os conceitos relativos à coordenação de SMA. Ao final, serão revisadas as abordagens de teste até então empregadas para verificar, avaliar e validar SMA.

Faz-se crucial a percepção de como as dificuldades na produção de testes, que são impostas pelas características do SMA, vêm sendo enfrentadas por trabalhos da área, tanto na sua forma teórica quanto aplicada. Salienta-se que os conceitos e características básicas que definem um agente, além das condições necessárias para a existência de uma sociedade de agentes, são fatores fundamentais na compreensão dos requisitos necessários à especificação, validação e testes voltados a SMA, bem como no entendimento de sua complexidade superior a outras abordagens existentes.

Em SMA, os agentes, suas tarefas e o ambiente em que estes atuam são interconectados muitas vezes de forma dependente e intrínseca. Ainda, devido à abordagem de desenvolvimento distribuída dos SMA, os vários módulos e componentes de um agente são especificados no sentido da sua imersão em um ambiente social, adequando suas potencialidades, expressas em seu conjunto de metas locais, às necessidades do ambiente, expressas pelas metas do sistema (GONÇALVES, 2006). Resultando daí a necessidade de não ignorar nenhum aspecto, quando da elaboração de métodos, parâmetros e propriedades para avaliações.

Destacam-se os níveis de organização relacionados à sociedade de agentes, que incluem métodos de formação de equipes, formação de coalizões, e considerações sobre fatores sociais como papéis dos agentes e estratégia dos times destinados ao *Soccer Server 2D*.

2.1 Sistemas Multiagente

Problemas computacionais modernos apresentam requisitos que impõem soluções de alta escalabilidade, distribuição lógica e/ou geográfica de componentes e a capacidade de incorporar soluções já existentes em contextos menores destes problemas. Sistemas

distribuídos são cada vez mais requisitados nos ambientes computacionais, fazendo-se cada vez mais presentes em nosso mundo físico e virtual, seja em dispositivos eletrônicos, equipamentos industriais, eletrodomésticos, automóveis, aeronaves, redes sociais, *sites*, ambientes simulados, entre outros.

O compartilhamento, seja de informações ou recursos, tem se apresentado como solução para diversos problemas de origem distribuída. Este assume um papel importante em diferentes tecnologias permitindo a economia de tempo, capacidade de processamento, vida útil de recursos, entre outros fatores. Determinados problemas de origem distribuída exigem um particionamento e distribuição entre entidades especializadas nas partes para uma possível solução. A distribuição permite uma série de benefícios no que tange a utilização de recursos, podendo-se aplicar vários conceitos computacionais que otimizam esse processo, tais como: paralelismo, multiprocessamento, divisão da carga de trabalho, entre outros (LEITÃO; INDEN; RÜCKEMANN, 2013).

A área de Sistemas multiagente (SMA), reconhecida inicialmente como subnível da área de Inteligencia Artificial (IA), ganhou consistência em meados da década de 80 e é tida desde então como uma área em seu direito próprio (D'INVERNO; LUCK, 2004). Seu surgimento deu-se a partir da necessidade de se modelar e conceber sistemas distribuídos e complexos. Para WOOLDRIDGE (2009), a área de SMA emergiu naturalmente da união de cinco importantes fatores de pesquisa na computação, os quais:

- **Ubiquidade** – Cada vez mais a capacidade de processamento computacional foi sendo inserida em diversos dispositivos, e sistemas embarcados fazem parte de contextos economicamente inviáveis tempos atrás (equipamentos industriais, meios de transporte, eletrodomésticos, vestimentas, entre outros). Grande parte disso se deve ao avanço na tecnologia de processadores e a diminuição de seu custo.

- **Interconexão** – A presença de internet e a interconexão entre sistemas computacionais são características comuns na atualidade. Sistemas distribuídos e concorrentes, geralmente situados em grandes redes, são condição essencial para a computação moderna.

- **Inteligência** – As tarefas automatizadas delegadas aos sistemas computacionais crescem em número e complexidade, em grande parte devido ao avanço da engenharia de sistemas e suas técnicas;

- **Delegação** – Refere-se à tendência do processo de delegar tarefas aos sistemas computacionais alcançar maiores níveis. Por exemplo, já é comum que um sistema computacional faça o controle de uma aeronave, o próximo passo seria permitir que um sistema realizasse o controle do tráfego aéreo.

- **Orientação humana** – Refere-se ao processo de tornar os sistemas computacionais mais próximos da percepção e entendimento humano. Esta mudança é perceptível

com o surgimento das interfaces gráficas e a evolução das linguagens de programação do baixo até o alto nível.

O avanço nesses fatores e a união das características necessárias aos sistemas atuais corroboraram para o surgimento da área de SMA (LUCK et al., 2005). Esses fatores ainda fornecem um entendimento sobre as vantagens idealizadas para o uso de SMA, dentre as quais, compreende-se a capacidade destes sistemas de se auto-adaptarem e evoluírem frente a erros e condições inesperadas, ou simplesmente para otimizar os objetivos para os quais foram propostos. Espera-se que esses sistemas tenham inteligência suficiente para manter-se operando nas devidas condições, sem a necessidade de intervenção externa (humana) ou técnicas de reengenharia. Há de se destacar que SMA permitem estender a capacidade individual de um agente conjuntamente com outros a problemas maiores ou mais elaborados.

Mediante suas potencialidades, SMA tornou-se uma área profícua para a pesquisa e desenvolvimento, sua raiz complexa e interdisciplinar direcionou-a na pesquisa e fundamentação teórica e técnica em diversas áreas como teoria dos jogos, economia e biologia. Também teve base em diversos ramos da computação, onde especificamente através da área de IA, é complementada por frentes de pesquisa como planejamento, métodos de raciocínio, métodos de busca e aprendizado de máquina (VIDAL, 2010).

As aplicações de SMA incluem pesquisa e modelagem do comportamento de enxame e coletivo apresentado por determinados grupos de animais (insetos, pássaros, peixes, entre outros), objetos inanimados (engarrafamentos, robôs), economia, sociedade e sistemas imunes (ODELL, 2007). Outras aplicações são mercado de ações, serviços *web* e modelagem de problemas relativos às ciências sociais e econômicas (comportamento individual e de grupos em sociedades). Também, tem-se procurado concretizar no agente conceitos abstratos de alto nível para uma representação cognitiva similar a humana como, crença, desejo e intenção, bem como, comportamentos de interação humana apresentados em sociedade (LUCK, 2005).

Essas aplicações, quando modeladas computacionalmente, primam por uma organização estrutural básica que reúne entidades, ambiente e um sistema organizado de forma que as entidades possam atuar e interagir. WOOLDRIDGE (2009) classifica os SMA como sistemas computacionais formados por uma coleção de entidades de software conhecidas como agentes, que uma vez imersos em um ambiente, interagem entre si para manter tanto os seus aspectos funcionais e objetivos operáveis quanto os do SMA ao qual pertencem. Para estruturarem o SMA, os agentes são capazes de executar ações autônomas e interagirem trocando dados ou mantendo atividades típicas de uma sociedade como cooperação, coordenação e negociação. A Figura 1 ilustra os principais componentes presentes em um SMA.

Assim, o universo de um SMA abrange o ambiente no qual este está inserido e suas

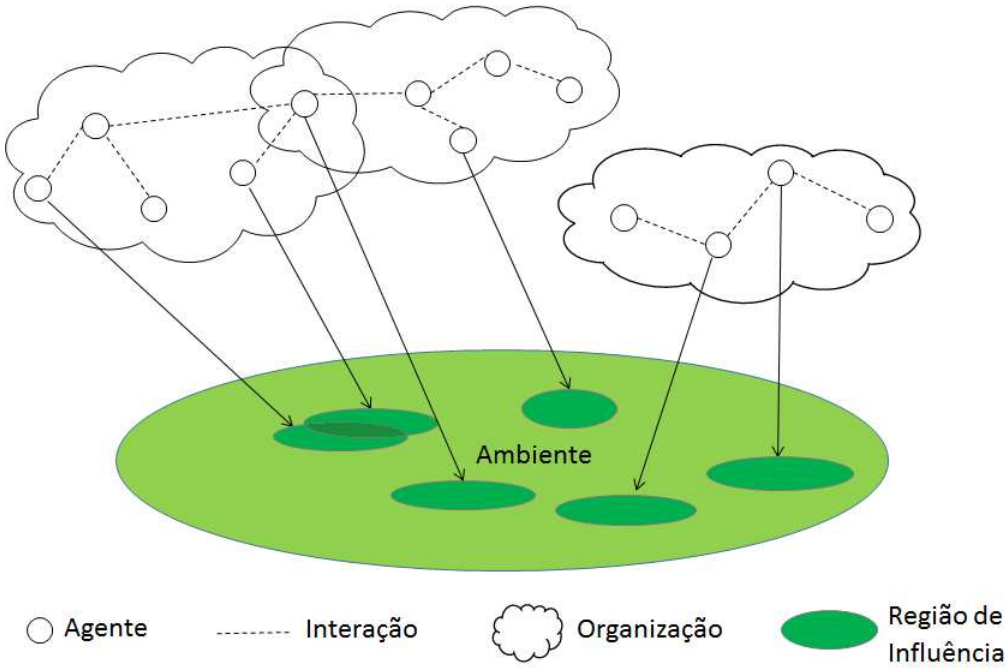


Figura 1: Principais componentes de um sistema multiagente. Extraído e adaptado de WOOLDRIDGE (2009).

dimensões básicas de concepção: a organização do sistema, os aspectos de comunicação entre os agentes e a formalização dos agentes em si (ROSA et al., 2013). Nas próximas subseções serão apresentados conceitos e questões referentes a estes quatro quesitos.

2.1.1 Agente

Não existe um consenso sobre a definição do termo "agente". Isto se deve à abundância e empregabilidade do termo em si e ao grande número de áreas que fazem uso dos conceitos relativos ao paradigma voltado a agentes. Desta forma, considerar-se-ão algumas definições que evidenciam importantes propriedades de um agente por estas serem largamente consolidadas (D'INVERNO; LUCK, 2004).

Para BOMAN; HOLM (2005), um agente é uma entidade que possui como principais características a autonomia, a proatividade e uma memória e estado interno que lhe fornecem base para a tomada de ações. Esse, pode ser caracterizado simplesmente como um *software* ou uma entidade física, a exemplo de um robô.

WOOLDRIDGE (2009) classifica um agente como um sistema computacional inteligente, que imerso em um determinado ambiente é capaz de agir de forma autônoma para alcançar seus objetivos. Ainda segundo WOOLDRIDGE (2009), a autonomia se caracteriza como sendo um espectro que abrange a capacidade da entidade de existir, pensar e agir por conta própria, podendo encontrar-se limitada no agente em qualquer um destes quesitos por razões de segurança. A inteligência, por sua vez, refere-se à capacidade de um agente apresentar reatividade (perceber e responder a estímulos do ambiente), proa-

tividade (tomar iniciativa nas ações) e habilidades sociais (interagir com outros agentes ou humanos) de forma a alcançar determinados objetivos.

Para ODELL (2007), há três propriedades básicas que um agente deve possuir:

- **Autonomia** – O agente age por conta própria e tem determinado grau de controle de seu estado interno e ações em virtude das suas experiências.
- **Adaptabilidade** – Refere-se à capacidade do agente de ajustar a si mesmo conforme sua interação com o ambiente e outros agentes.
- **Interatividade** – O agente é capaz de se comunicar com o ambiente e com outros agentes. As formas de comunicação podem ser as mais variadas.

Segundo ODELL (2007), estas três propriedades básicas variam em grau de abrangência nos agentes. A Figura 2 ilustra, de forma abstrata, o nível de abrangência de cada uma destas propriedades.

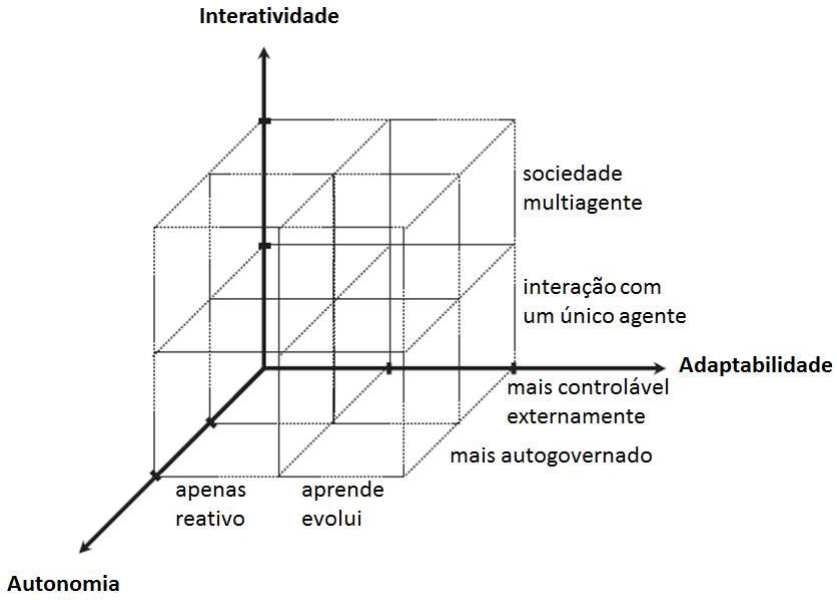


Figura 2: Níveis de abrangência das propriedades básicas de um agente. Extraído e adaptado de ODELL (2007)

Outras características idealizadas para os agentes comumente aceitas são (ODELL, 2007):

- **Continuidade temporal** – A execução do agente ocorre continuamente a partir do momento em que este passa a existir. Nesse caso, o agente pode estar ativo ou inativo (executando em *background*) durante sua existência.
- **Mobilidade** – Refere-se à capacidade do agente de migrar de um ambiente para outro.

- **Cooperação** – Um agente pode agir em conjunto com outros para alcançar um objetivo em comum.

- **Competição** – Um agente disputa com outros por recursos e procura realizar seus próprios interesses.

Uma das principais questões envolvendo um agente é a necessidade de adaptação. Um agente pode estar condicionado a uma tarefa simples em seu ambiente, no entanto, geralmente este é tido como dinâmico. Qualquer situação, ou condição, que leve o agente a ser impedido ou a ignorar os comportamentos necessários para a realização de sua tarefa, deve induzi-lo à adaptação, reconduzindo-o novamente aos comportamentos esperados e consequentemente ao estado desejado.

À vista disso, entende-se que um agente deva ser capaz de interagir em um ambiente por conta própria. Para que um agente apresente uma ou várias funcionalidades (que suportem a realização de tarefas), este deve ser projetado sobre a perspectiva de como solucionar um problema tendo em vista o ambiente no qual está inserido (WEYNS; OMICINI; ODELL, 2007), e portanto, deve interagir de forma a receber informações e agir nesse ambiente. Não obstante, a interação do agente com o ambiente ocorre de forma contínua, ou seja, o agente capta informações do ambiente por meio de sensores e executa as ações por meio de atuadores perpetuando ciclos de interação (RUSSELL; NORVIG, 2004). A Figura 3 ilustra um modelo teórico de agente.

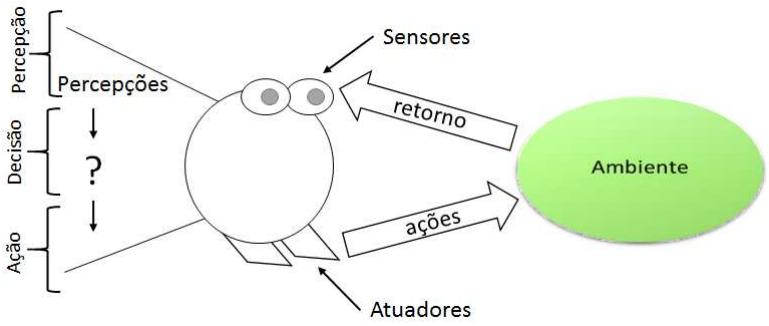


Figura 3: Agente cognitivo. Extraído e adaptado de WOOLDRIDGE (2009)

Um agente, ainda pode ser classificado quanto a seu modelo de concepção e estrutura interna em três tipos principais: (RUSSELL; NORVIG, 2004) (WOOLDRIDGE, 2009):

- **Agente reativo** – Atua no ambiente respondendo a estímulos que recebe deste. Sem dispor de um modelo interno do ambiente - por não manter memória interna - simplesmente faz um mapeamento das percepções captadas para as ações necessárias. Propostas para este agente visam arquiteturas de tempo real, para lidar de forma imediata com as mudanças que ocorrem no ambiente.

- **Agente cognitivo** – Este agente raciocina sobre as informações que possui, decidindo como agir no ambiente. Parte das informações é mantida em uma memória interna e pode indicar eventos, decisões e ações passadas ou quaisquer outras informações necessárias ao agente. Propostas para este agente produziram tipos variados de arquiteturas, nas quais, sempre estão presentes tecnologias que permitem a modelagem simbólica do mundo (*logical formulae*) e a capacidade de raciocínio (*logical deduction*, ou *theorem proving*) pelo agente.

- **Agente híbrido** – O agente híbrido é uma combinação das arquiteturas reativa e cognitiva. Visando equilibrar o agente em seus níveis de cognição, seja na capacidade de raciocínio ou na tomada rápida de decisões, procurou-se unir características das duas arquiteturas. Propostas para a arquitetura híbrida exigem no mínimo dois módulos: um atuando na tomada de decisões (reunindo características do modelo de mundo e capacidade de raciocínio) e outro atuando na resposta imediata a estímulos do ambiente (simples mapeamento entre estímulo e ação).

Independente da arquitetura do agente, as características do ambiente, assim como as funcionalidades e os objetivos esperados, exigem sempre, do agente, no mínimo as propriedades de autonomia, proatividade e adaptabilidade (WOOLDRIDGE, 2009).

2.1.2 Ambiente

O ambiente tem grande influência na complexidade de um SMA, tendo como principal motivo o fato deste ser parcialmente controlável, ou seja, é praticamente impossível considerar todos os estados e variáveis pertinentes à dinâmica de um ambiente e submetê-los às ações de controle. Assim, o nível e a amplitude de controle requerido impactam na complexidade computacional do sistema tanto pela capacidade de obter as informações necessárias quanto pela capacidade de agir no ambiente.

Segundo RUSSELL; NORVIG (2004), os sensores e atuadores de um agente, e a tarefa atribuída ao mesmo, assim como, as medidas de desempenho que qualificam a sua execução, estão intrinsecamente ligadas ao ambiente. Algumas propriedades do ambiente são úteis para compreensão do agente em si, tais como (RUSSELL; NORVIG, 2004):

- **Completamente observável *vs* parcialmente observável** – Se as informações do ambiente, captadas pelos sensores do agente em cada instante de tempo, são completas e precisas, este é considerado completamente observável. Se existem imprecisões de leitura, o ambiente é parcialmente observável.

- **Determinístico *vs* estocástico** – Se um estado futuro é determinável pelo estado e ação atual do agente, o ambiente é determinístico, do contrário, este passa a ser considerado estocástico, ou seja, sem qualquer previsibilidade diante da dinâmica deste ambiente.

- **Episódico *vs* sequencial** – No ambiente episódico, ações são atômicas e independem de ações anteriores. No ambiente sequencial, as ações atuais influenciam nas posteriores e consequentemente no resultado final.

- **Discreto *vs* contínuo** – Percepções e ações no ambiente discreto são conhecidas e ponderadas em intervalos de tempo. No ambiente contínuo as percepções e ações variam constantemente em espectros contínuos de valores e em intervalos distintos de tempo;

- **Único agente *vs* multiagente** – Há a presença de apenas um agente, ou há a presença de duas ou mais entidades que convivem no ambiente e agem de forma cooperativa ou competitiva.

Um projeto de sistema de controle deve levar em consideração o ambiente e suas propriedades. SMA, frequentemente, envolvem ambientes parcialmente observáveis, estocásticos e sequenciais com níveis exponenciais de complexidade.

WEYNS; OMICINI; ODELL (2007) propõem uma análise do papel que o ambiente desempenha na construção dos agentes sobre dois diferentes contextos:

- **Agente situado ou imerso** – Este contexto engloba os tipos de agentes cujo ambiente é visto como externo, sendo que o foco é voltado para a dinâmica do mesmo. A arquitetura interna destes agentes não possui modelo do ambiente, pois visa lidar de forma imediata com as mudanças que ocorrem no ambiente, sem maior necessidade de processamento. Desta forma, foram propostos inicialmente os agentes puramente reativos, posteriormente vieram os agentes baseados em comportamentos, aproveitando-se da execução paralela para ações mais eficientes no ambiente.

- **Agente cognitivo** – Agentes baseados neste contexto necessitam de um modelo do ambiente. A representação de conhecimento é vital para tarefas desempenhadas pelo agente (navegação, histórico de ações, entre outros), que raciocina sobre as informações para escolher como agir no ambiente. Agentes direcionados neste contexto possuem desde arquiteturas puramente cognitivas a abordagens híbridas.

Segundo WEYNS; OMICINI; ODELL (2007), cada tipo de agente requer técnicas específicas de percepção e comunicação para atuar no ambiente. Geralmente, agentes situados fazem uso da própria percepção, ou de características presentes no ambiente (abordagem de feromônios inspirada em animais, marcos no ambiente como padrões de cores, objetos, ou dispositivos eletrônicos) para comunicar-se com outros agentes e atuar em grupo. Já agentes cognitivos, geralmente, comunicam-se diretamente por canais de comunicação, mantendo informações do ambiente, bem como de si próprio e de outros agentes, compartilhando-as sempre que necessário.

2.1.3 Comunicação

Dado que um SMA é uma sociedade organizada em função de um objetivo maior, é necessário permitir-se que tanto a especificação e implementação dos agentes seja possível quanto a interação destes por meio de comportamentos sociais. Desta forma, a comunicação é definida como um dos requisitos fundamentais de SMA. A comunicação entre agentes pode ocorrer de três formas (CAO; FUKUNAGA; KAHNG, 1997) (PARKER, 2008):

- **Comunicação implícita** – Um agente observa os efeitos causados no ambiente devido às ações de outros agentes. Da observação inferem-se informações e denota-se o comportamento do grupo;
- **Reconhecimento de ação passiva** – Um agente obtém informações pela observação direta e reconhecimento das ações de outros agentes;
- **Comunicação explícita** – A troca de informações ocorre diretamente entre os agentes através de mensagens e de forma intencional.

Os primeiros dois tipos de comunicação são mais simples e fáceis de implementar, pois não necessitam de canais e protocolos de comunicação, bem como, não estão limitados a restrições de largura de banda e falhas de comunicação. No entanto, é de caráter fundamental a capacidade perceptiva e cognitiva dos agentes. O processamento das informações relativas às propriedades do ambiente e às ações dos agentes fica restringido às interpretações, assim como, a correta relação com a atuação do grupo e os objetivos do sistema.

A comunicação explícita é mais complexa e lida com diversas variáveis. Requisitos como arquitetura de comunicação (organização, protocolos, meios físicos, padrão de mensagens), bem como questões de falhas de comunicação (atraso e perdas de mensagens, postergação infinita, confiabilidade das informações) influenciam diretamente em tarefas de coordenação, ou na consistência do estado global, quando este é necessário. Tanto a coordenação quanto o estado global são dependentes da troca de informações locais e individuais dos agentes que não compartilham memória e, portanto, ficam sujeitos à sequência das mensagens já que o tempo não é compartilhado. Geralmente, a comunicação explícita resulta em qualidade no desempenho do sistema, mas pode ter grande impacto no custo computacional.

2.1.4 Organização

Quando um SMA é considerado em sua totalidade, o aspecto organizacional surge com a necessidade de técnicas que visem manter o SMA como uma unidade. A organização em um SMA visa resolver conflitos entre os propósitos globais do sistema e os

comportamentos autônomos dos agentes (WOOLDRIDGE; CIANCARINI, 2001). Esta pode vir orientada no sentido da cooperação ou da competição entre os agentes.

Questões de estrutura organizacional, normas e obrigações dos agentes, e de auto-organização das sociedades são fundamentais (LUCK et al., 2005). Há vários estudos e níveis de organização relacionados a sociedade de agentes, que incluem forma de estruturas organizacionais, métodos de formação de equipes, formação de coalizões, entre outros. Também são feitas considerações sobre fatores sociais como papéis, poderes institucionais, direitos e deveres dos agentes na sociedade.

Segundo HORLING; LESSER (2004) os principais paradigmas organizacionais utilizados em SMA são:

- **Hierarquias** – Neste tipo de organização os agentes são dispostos em uma estrutura de árvore (ver Figura 4(a)), onde cada agente está contido em um nó. Um agente que pertence a um nó superior, em relação a outros na árvore, tem maior acesso às informações, pois, os agentes de nós inferiores fornecem os dados que coletam. As decisões de alto nível partem de cima para baixo, seguindo a ordem de interações, que ocorrem apenas entre agentes conectados diretamente. Esta estrutura permite a quebra e atribuição de tarefas de forma hierárquica, propiciando uma abordagem eficiente de divisão e conquista. Também faculta ao agente o enfoque nas suas tarefas locais. Em contrapartida, torna possível a formação de gargalos e pontos únicos de falha, bem como, lentidão na comunicação devido às conexões diretas entre agentes serem limitadas conforme a estrutura da árvore.

- **Holarquias** – Este tipo de estrutura é designado pelo aninhamento hierárquico de organizações que possuem similaridades (ver Figura 4(b)), ou seja, grupos de agentes (também hierárquicos) que mantém semelhanças entre si. Os grupos de agentes, embora distintos entre si, derivam suas propriedades dos grupos imediatamente superiores na hierarquia, fator que implica nas similaridades. A estrutura final é vista como um todo, onde cada elemento, sejam os grupos, sejam os agentes que os compõem são insubstituíveis. Esta estrutura organizacional, além de permitir a quebra das tarefas em subconjuntos atribuíveis a grupos de agentes, suporta conexões entre grupos. A coordenação dos agentes pode ser feita de forma flexível garantindo certa resiliência diante da dinâmica do ambiente, no entanto, devido à proporção e distribuição dos agentes pode ser difícil calcular o desempenho do sistema.

- **Coalizões** – Nesta estrutura acontece a formação de subconjuntos de agentes (ver Figura 4(c)) para atender a objetivos específicos. Quando os objetivos mudam, o grupo é desfeito. A responsabilidade de coordenar as ações fica sob o encargo do grupo, que exteriormente pode ser visto como uma entidade. Grupos de agentes que

formam coalizões podem ser sobrepostos, ou mesmo, um grupo pode ser alocado dentro de outro. Neste caso, não ocorre coordenação entre agentes de grupos distintos e, caso haja dependência de tarefas, representantes de grupos intermedeiam um acordo. A principal ideia por traz da formação de coalizões é a otimização na execução de tarefas e atividades pela adição de entidades (que procuram a otimização individualmente). No entanto, isto só ocorre pela correta escolha de uma função de utilidade (que consiga avaliar a real necessidade de coalizão e o número imprescindível de agentes). O desafio em relação a estrutura de coalizão é a formação e disjunção de grupos, que pode incorrer em custo computacional elevado.

- **Times** – Esta estrutura condiz com um grupo de agentes cooperativos (ver Figura 4(d)) que essencialmente precisam concordar com um objetivo a ser alcançado. Em virtude da cooperação, os agentes procuram otimizar o objetivo final do time, diferentemente das coalizões onde os agentes preconizam os objetivos individuais. Os agentes dentro de um time podem exercer diferentes papéis dependendo das subtarefas exigidas e da necessidade de replanejamento de suas execuções. Não obstante, os agentes podem ter crenças em comum, ou compartilhadas, que indicam planos e comportamentos de time. Embora esta estrutura permita a execução de problemas complexos, a comunicação entre os agentes pode ter um grande custo devido à necessidade de atender às restrições (sincronização entre agentes, troca de informações, negociações, entre outros) que visam o objetivo geral do grupo. A alocação de tarefas aos agentes, bem como a manutenção da consistência das atividades quando um plano é executado pelo grupo é um problema considerável.

- **Congregações** – Está estrutura permite a união estável de agentes em grupos pequenos de longa duração (ver Figura 4(e)). Congregações são formadas por agentes similares que possuem características que se complementam. Estes agentes podem não ter objetivos fixos e em comum, entretanto, o conjunto de habilidades ou requisitos necessários para a execução das tarefas individualmente motivam a união. Os agentes pensam em suas tarefas locais (calculando a utilidade), e caso não seja proveitoso fazer parte de um grupo, acabam por migrarem para outros grupos. Embora um agente possa fazer parte de múltiplas congregações, não há comunicação entre agentes de congregações diferentes. Um benefício promovido por esta forma de organização é a diminuição do custo de comunicação, no entanto, os agentes tendem a se comunicar apenas com aqueles inclinados a fornecer ajuda. Isto pode criar grupos de agentes isolados e imutáveis.

- **Sociedades** – Este tipo de estrutura é formado por grupos abertos e inspirados nas formas de sociedades presentes nos ecossistemas (ver Figura 4(f)). Os agentes costumam ter diferentes habilidades, objetivos e níveis de racionalidade, mas

a sociedade provê um ambiente em comum onde eles podem conviver (atuando e interagindo). Dentro da sociedade podem coexistir diferentes tipos de organizações. Para que a sociedade seja consistente são impostas regras sociais, normas e convenções que os agentes devem seguir, (ex: protocolos de comunicação e negociação, restrições a comportamentos, entre outros), inclusive podendo haver sanções e penalidades diante de transgressões. Sociedades podem ser estruturas complexas, uma vez que, deve se garantir que um agente respeite as restrições impostas, porém, que estas restrições permitam o alcance dos objetivos pelo agente. Também devem ser definidos os papéis, protocolos e regras que fundamentaram a formação da sociedade, bem como, os meios que garantam aos agentes externos reconhecerem a sociedade e estando aptos (haja compatibilidade) possam adentrar na mesma.

- **Federações** – Esta estrutura é baseada no modelo de governo federativo. O modelo de organização prevê um ou mais grupos de agentes. Cada grupo tem como característica principal um agente como representante do grupo, o qual recebe autonomia superior aos demais membros (ver Figura 4(g)). O representante do grupo passa a ser mediador entre o mundo externo e os agentes de seu grupo. O mediador trabalha recebendo tarefas externas (podendo decompô-las em subtarefas), anunciando as habilidades do grupo, centralizando a comunicação, entre outras atividades. Esta estrutura é interessante por prover uma organização complexa formada por grupos de agentes heterogêneos. Outra vantagem é abstrair a realização de atividades administrativas dos agentes do grupo. Estas são concentradas na figura do mediador que, no entanto, passa a ser um ponto único de falha na estrutura.

- **Mercado** – Um tipo de organização que facilita a modelagem de economias de mercado do mundo real. Nela estão representadas, por agentes, as figuras do comprador e do vendedor (pode ser designado como uma terceira parte, o leiloeiro) (ver Figura 4(h)). A mercadoria (recursos, serviços, tarefas ou bens) recebe ofertas dos agentes compradores, sendo que, os agentes vendedores podem renegociar os preços. Durante as negociações, as ofertas são analisadas e processadas e, então, o vendedor (ou leiloeiro) anuncia o ganhador. A princípio, todo problema que pode ser modelado pelo paradigma produtor-consumidor (mercado de ações, indústria de manufatura, varejo, entre outros) pode fazer uso deste tipo de estrutura organizacional. Embora com potencial para modelar diversos problemas, esta estrutura pode ser bastante complexa dependendo do nível de modelagem que se queira alcançar, destacando-se: a organização pode ser aberta ou fechada (com restrições para a entrada de agentes), a modelagem de mercado pode requerer que as formas de negociação dos preços mudem ao longo do tempo, bem como, requerer a inversão da figura que faz a oferta (comprador para o vendedor, e vice-versa). Além disso, comportamentos como especulações de mercado e a não confiabilidade dos

participantes pode ser desejável.

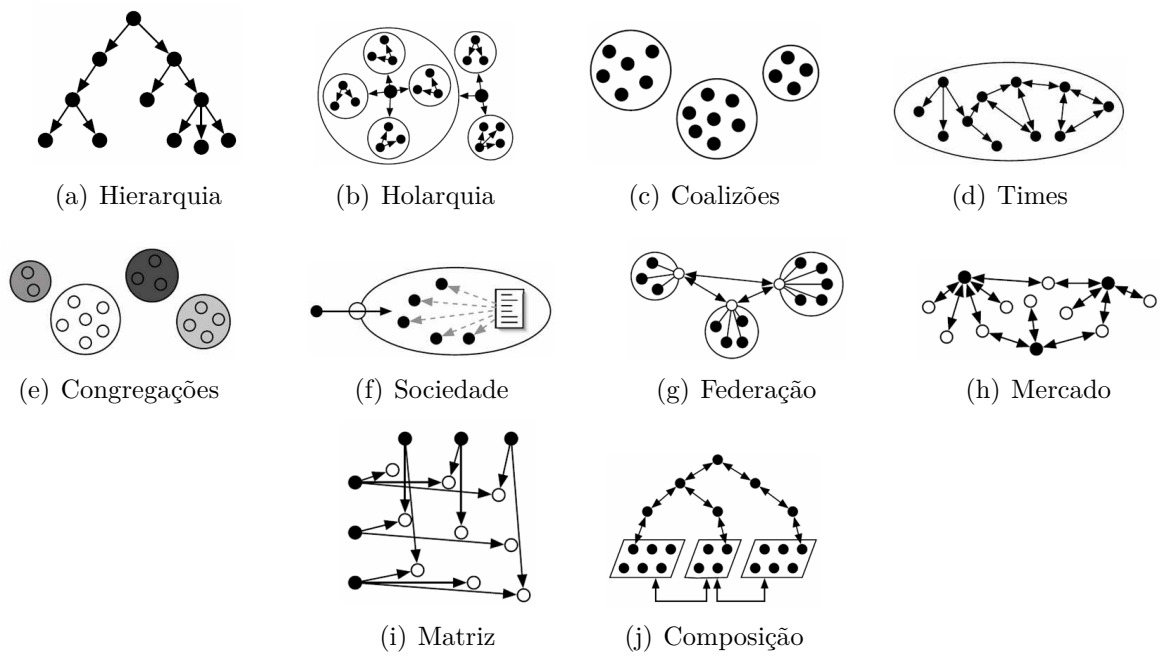
- **Matrizes** – Em uma estrutura de matriz os agentes possuem relações diretas com vários outros agentes. Cada agente possui seus próprios objetivos, características e relativa importância em relação aos outros da matriz. Agentes nomeados como administradores (ou pares) podem influenciar nas atividades de outros agentes. A matriz provê formalmente a distribuição dos agentes que podem ser influenciados por determinado agente administrador (ver Figura 4(i)). Vários administradores podem influenciar nas atividades de um único agente. Esta estrutura se aproxima de como os humanos vivem em sociedade, muitas vezes sendo influenciados ou pressionados por outras pessoas com quem interagem. Pode ser a organização ideal para atender aos objetivos globais de um sistema, já que todos os agentes podem ser direcionados para um objetivo em comum. No entanto, se houver discordância entre agentes administradores os objetivos globais podem ser comprometidos.

- **Composição** – Uma organização composta, parte do pré-suposto de que determinados problemas podem exigir mais de um paradigma organizacional para serem melhor equacionados. Diferentes partes de um sistema (controle, fluxo de dados, coordenação, entre outros) podem ser modelados por distintas formas organizacionais. Em suma, uma organização composta é um arranjo que pode associar os vários tipos organizacionais listados anteriormente (ver Figura 4(j)). A organização composta pode estar em constante estado de transição entre as diversas estruturas organizacionais (em diferentes áreas do sistema). Isto ocorre devido à adaptação aos diferentes objetivos presentes no sistema. Obviamente, esta estrutura pode ter grande complexidade, incluindo desde as já previstas nas demais estruturas, passando pelas condições que permitem as transições, até o fato de um agente poder assumir diferentes papéis simultaneamente. Questões de consistência se tornam mais importantes, uma vez que, diferentes tipos de estruturas organizacionais podem atuar em contradição entre si.

A Figura 4 ilustra os tipos de estruturas organizacionais mais comuns em SMA.

Aproximando-se do contexto deste trabalho, é possível identificar a estrutura organizacional "time" como inerente ao problema de futebol de robôs, mas não excludente das outras estruturas mencionadas. Há de se destacar que a dinâmica do jogo de futebol e o tempo reduzido para tomada de decisões primam por estruturas organizacionais menos complexas, limitando a escolha entre os tipos de estruturas.

Não obstante, outras formas de organização dentro da estrutura time podem ser projetadas ou mesmo surgir implicitamente. Em DROGOUL; COLLINOT (1998) são identificados quatro comportamentos individuais de agentes (passador, recebedor, bloqueador e defensor) e as dependências entre estes comportamentos que conduzem à formação de



(a) Hierarquia (b) Holarquia (c) Coalizões (d) Times

(e) Congregações (f) Sociedade (g) Federação (h) Mercado

(i) Matriz (j) Composição

Figura 4: Paradigmas de estruturas organizacionais em SMA. Extraído e adaptado de HORLING; LESSER (2004).

grupos similares a estrutura organizacional descrita como coalizões. A Figura 5 ilustra essas formações no campo.

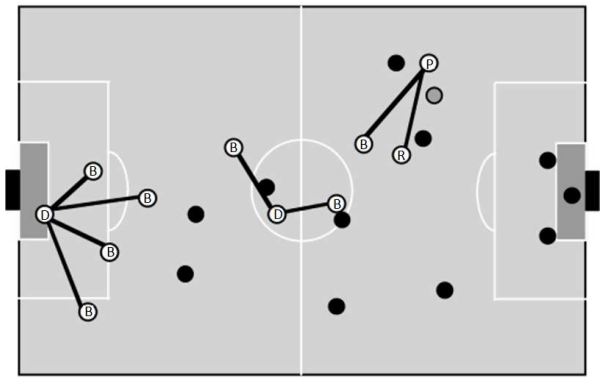


Figura 5: Dinâmica da formação de grupos. Extraído e adaptado de DROGOUL; COLLINOT (1998).

Durante uma partida os comportamentos definem os papéis dos jogadores na equipe e dinamicamente podem ser alternados conforme o grafo de influências ilustrado na Figura 6.

Os comportamentos básicos se engajam entre si em dependências tais como: o passador da bola necessita que o recebedor se posicione para haver o passe (ou tentativa), também pode depender que um bloqueador impeça o avanço de adversários e permita a tentativa de chute. Os comportamentos básicos podem ser vistos em três tipos de dependências relacionais (DROGOUL; COLLINOT, 1998):

- Dependências funcionais – Cujo desempenho de um comportamento depende da

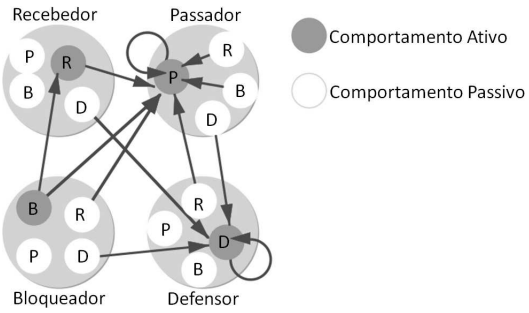


Figura 6: Grafo de influências de comportamentos. Extraído e adaptado de DROGOUL; COLLINOT (1998).

ativação prévia ou simultânea de outro comportamento.

- Dependências baseadas em recursos – Na qual um comportamento pode ser inibido devido ao conflito de recursos com outro comportamento.

- Dependências organizacionais – São atribuídas pelo projetista conforme o tipo de organização desejada. A definição das dependências de comportamentos pode estar relacionada às estratégias que não são alcançadas espontaneamente pelas dependências anteriores.

Ainda segundo DROGOUL; COLLINOT (1998), os jogadores estabelecerão os grupos conforme os papéis que estes assumem, as dependências instanciadas e o grafo de influências, bem como, um mecanismo de decisão como *contract net protocol*.

2.2 Coordenação em Sistemas Multiagente

A coordenação está relacionada à administração de interdependências entre atividades realizadas por entidades de um grupo, podendo ser agentes físicos (robôs) ou virtuais (*softwares*). Um mecanismo de coordenação é necessário quando as atividades que os agentes devem realizar interagem de algum modo, sendo passível de ocorrer da seguinte forma (YASUDA; TACHIBANA, 1994):

- Existe um recurso não compartilhável sendo requisitado por mais de um agente (problema da exclusão mútua);

- O início, algumas etapas, ou o término de uma atividade atribuída a um agente, depende do fim de outra atividade atribuída a outro agente;

- Em um terceiro caso, não há necessidade explícita de coordenação, porém um agente agindo de forma proativa favorece a outro fornecendo informações ou recursos, mesmo que potencialmente estes estejam disponíveis a ambos.

Os SMA tem um caráter distribuído, onde instâncias de execução ocorrem concorrentemente e a troca de informações entre as entidades do sistema ocorre de forma assíncrona. A comunicação e a sincronização tem um caráter fundamental na coordenação desses sistemas. O sincronismo deve ocorrer para que as entidades possam interagir da forma que seja necessária (exclusão mútua, dependência de atividades, ação proativa) e isso só acontece por meio da comunicação.

A coordenação, em virtude da comunicação e sincronismo, está relacionada à administração de conflitos e dependências para atingir objetivos parciais ou globais do sistema. Dessa forma, um fator crucial para a coordenação diz repeito ao tipo e quantidade de informação que deve ser trocada entre os agentes. O que é necessário que um agente saiba a respeito de outro (posição, estado, objetivo), e de que forma isso deve acontecer.

Devem ser considerados cuidadosamente os custos e benefícios das alternativas de comunicação (implícita, explícita, reconhecimento de ação passiva), para estimar aquela que de forma confiável alcance o nível desejado de desempenho do sistema. À medida que a complexidade de um sistema aumenta junto ao domínio da aplicação são necessárias técnicas de coordenação cada vez mais sofisticadas.

Observa-se que a complexidade está inserida em vários níveis dentro de um SMA. Geralmente, refere-se ao domínio do problema e à viabilidade computacional dos sistemas. Dessa forma, condições de complexidade estão ligadas à capacidade de um SMA de lidar com o ambiente de imersão, enquanto busca atingir os objetivos do sistema, através do planejamento de missões global ou local, da alocação de tarefas e a coordenação de interações entre os agentes. Notadamente, pode-se dividir o problema de complexidade em características do ambiente e características de otimização da coordenação (organização do sistema, planejamento de tarefas e capacidades dos agentes).

Pode-se frisar, no entanto, que no ambiente do simulador *Soccer Server 2D*, o número de habilidades básicas previstas para os agentes e as restrições do ambiente tornam necessária a consideração de um balanceamento entre as características cognitivas e de reação imediata, tanto para o agente quanto para as estratégias do time. Isso força as técnicas de coordenação a respeitarem limites de tempo quando da busca de soluções.

Como um dos principais aspectos que envolvem a coordenação é a interação entre os agentes, geralmente, para resolver problemas em conjunto, é necessário um entendimento sobre a atribuição de tarefas para esses agentes.

2.2.1 Descrição, Representação e Particionamento de Tarefas

Uma forma de descrever uma tarefa é definí-la como uma atividade com requisitos mínimos e um objetivo específico e que pode ser realizada por uma ou mais entidades. Os requisitos mínimos de uma tarefa podem ser descritos como habilidades necessárias à sua execução, tais como a detenção de uma ou mais ferramentas específicas, a capacidade de atender a um limite de tempo ou um estado específico da entidade executora.

Em SMA, uma tarefa pode ser definida como um objetivo parcial que pode ser alcançado independentemente do objetivo global de um sistema. Uma tarefa pode ser discreta (locomoção de um ponto a outro) ou contínua (monitoramento de um objeto) e variar em escala de tempo, complexidade e especificidade, entre outras formas.

Segundo DIAS et al. (2005), a forma mais comum entre as abordagens de atribuição de tarefas é delegar uma lista com tarefas em sua forma primitiva, ou seja, tarefas simples (atribuída a uma única entidade, geralmente requerendo uma única ação). Porém, tratando-se de um grupo, torna-se mais natural o uso do termo missão, que é uma descrição de alto nível para uma tarefa complexa cuja decomposição em subtarefas é possível. As subtarefas podem ser distribuídas para um grupo de agentes visando um melhor aproveitamento dos recursos disponíveis.

Quando existe um grupo de agentes que deve interagir para a realização de tarefas, também deve haver um tratamento em relação ao nível de descrição, modelagem e interdependência dessas tarefas. Também existe a necessidade de planejamento sobre o que deve ser feito. Em ZLOT (2006) é proposta uma terminologia para classificar os diferentes tipos de tarefas:

- **Decomposição e decomponibilidade** – uma tarefa t pode ser decomposta se puder ser representada por um conjunto de subtarefas σ_t, que satisfazem alguma combinação (relação) específica (ρ_t) de subtarefas. O conjunto de relações ρ pode definir restrições ou regras entre tarefas. O par (σ_t, ρ_t) é chamado decomposição de t;

- **Múltiplas decomponibilidades** – uma tarefa t que apresenta mais de uma forma de ser decomposta;

- **Tarefas Simples** – pode ser uma tarefa atômica ou uma tarefa simples decomponível;

    - **Tarefa atômica** – é uma tarefa que não pode ser decomposta, geralmente referida por uma única ação;
    - **Tarefa simples decomponível** – uma tarefa que pode ser dividida em passos e executada por uma única entidade;

- **Tarefa composta** – é uma tarefa que pode ser decomposta em um conjunto formado por tarefas simples, sendo que, deve existir ao menos uma tarefa simples decomponível;

- **Tarefas complexas** – uma tarefa que possui múltiplas decomponibilidades, onde existe ao menos um conjunto de tarefas alocadas entre várias entidades. Cada tarefa do conjunto pode ser simples, composta ou complexa.

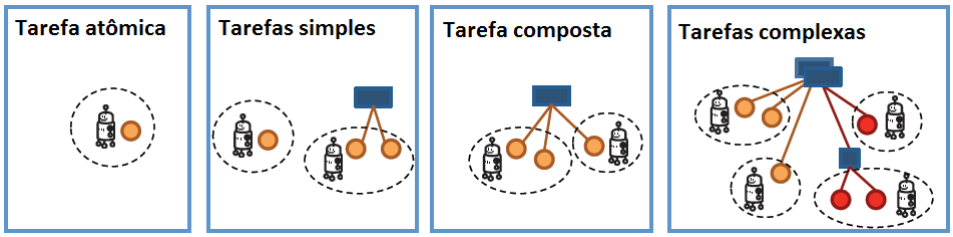


Figura 7: Representação de tarefas. Extraído e adaptado de KORSAH; STENTZ; DIAS (2013). Os círculos pontilhados representam possíveis alocações de tarefas para agentes, os círculos preenchidos são tarefas atômicas, os retângulos são tarefas decompostas que formam uma estrutura de árvore. No primeiro quadro é ilustrada uma tarefa atômica sendo atribuída a um agente; o segundo quadro ilustra uma tarefa atômica e uma tarefa simples decomponível cada uma sendo atribuída a um agente; o terceiro quadro ilustra uma tarefa composta formada por uma tarefa simples decomponível e sendo atribuída a dois agentes; o quarto quadro ilustra uma tarefa complexa alocada a múltiplos agentes e com várias possíveis decomposições (KORSAH; STENTZ; DIAS, 2013).

A Figura 7 ilustra a representação dos tipos de tarefas.

Segundo DIAS et al. (2005), a diferenciação entre os tipos de tarefas é necessária, pois no desenvolvimento de um algoritmo de alocação, tarefas complexas requerem uma modelagem de abstração em vários níveis. O algoritmo de alocação para tarefas complexas deve lidar com atribuições dinâmicas para produzir resultados de execução melhores. Nele, a decomposição (planejamento) de tarefas deve ser flexível, pois alterações podem ser necessárias (ZLOT, 2006). Isso ocorre devido à dinamicidade no ambiente dessas tarefas, a quantidade de caminhos para soluções e as capacidades dos agentes. Soluções melhores não são predeterminadas, mas ocorrem dinamicamente em resposta a demanda do sistema.

A diferenciação das tarefas também induz a correlação com a real aptidão das entidades (virtuais ou físicas) que, geralmente, são construídas com propósitos específicos, sendo praticamente impossível obter uma de propósito geral. Ainda que estas venham a incluir e superar níveis de habilidades que as predisponham a realização de mais tarefas, a especifidade e definição das tarefas preservarão questões de desempenho e coordenação entre as entidades. Como citado neste capítulo, algumas tarefas são mais complexas e requerem múltiplas habilidades e ações, sendo impossível sua execução por um único agente.

Entre os agentes pode haver uma grande variação de capacidades para o cumprimento de atividades específicas. A variedade pode advir dos componentes computacionais, das ferramentas (sensores e atuadores), do sistema de controle, ou dos propósitos e problemas que o agente deve resolver.

Segundo NEHMZOW (2003), há uma interdependência entre agente, tarefa e ambiente. Considerando-se, por exemplo, o problema de futebol de robôs e a tarefa de chutar uma bola situada à determinada distância, necessitar-se-á de um jogador as capacidades de locomoção, reconhecimento e o chute propriamente dito. A tarefa pode ser dividida na sequência: localizar a bola; locomover-se até a mesma e chuta-la. O jogador não opera

sozinho no ambiente, pois no percurso há movimentação de outros jogadores, então deve haver o desvio dos obstáculos. Notoriamente que, o jogador deve captar informações do ambiente para desempenhar suas funções e prever os movimentos dos jogadores e da bola requerendo capacidade cognitiva considerável. O estado do ambiente e as reais capacidades do agente estão intrinsecamente ligados ao cumprimento da tarefa. Assim, pode-se reconhecer uma tarefa básica como uma única ação (locomoção de um ponto a outro), mas que pode exigir mais de uma capacidade.

Em se tratando de futebol de robôs, e independente do ambiente ser real ou simulado, são necessárias aos agentes algumas habilidades básicas. Entre as quais se encontram a navegação e a localização, largamente estudadas em robótica móvel sobre diferentes abordagens e técnicas (SIEGWART; NOURBAKHSH, 2004). A navegação condiz com a capacidade do robô de se locomover em um ambiente seja ele fechado (*indoor* – salas, prédios, entre outros) ou aberto (*outdoor* - campos, oceano, cidade, entre outros). A localização é a descoberta pelo robô de sua posição no ambiente, seja local (parte do ambiente visível pelo agente) ou global (ambiente total em que o agente está imerso). Geralmente, a navegação e a localização atuam em conjunto.

Tarefas complexas podem ser decompostas em tarefas menores e independentes sempre que possível, desta forma, elas podem ser distribuídas para um grupo de agentes visando melhor aproveitamento dos recursos disponíveis. A descrição de uma tarefa no domínio de uma aplicação pode ser dada como um conjunto de variáveis (duração, início e término, habilidades necessárias), quando trata-se de tarefas simples a tendência é uma execução sequencial, em contrapartida tarefas complexas requerem mais flexibilidade na ordem de execução.

2.2.2 Coordenação em Times de Futebol de Robôs Simulado

A área de coordenação em SMA é complexa e extensa, envolvendo vários recursos e técnicas. Mesmo no problema de futebol de robôs essa complexidade é evidente pela quantidade de soluções. Serão listadas algumas das características e soluções consideradas no campo de futebol de robôs simulado a despeito da amplitude da área de coordenação em SMA, que não seria possível discorrer com profusão de detalhes.

O ambiente de futebol de robôs simulado *Soccer Server 2D* traz complicações para as abordagens de coordenação. As dificuldades presentes no simulador são (ALMEIDA; LAU; REIS, 2010):

- Existem muitas informações distribuídas em várias camadas e sensoreadas pelos agentes a todo instante, tornando o processamento difícil;
- O ambiente é imprevisível, tornando a previsão de estados futuros complicada;
- A comunicação é limitada e não confiável;

- Há incertezas na percepção do ambiente que podem levar à incorreta modelagem de estados e ao conflito entre os agentes.

Dessa forma, as principais questões de coordenação dizem respeito à (ALMEIDA; LAU; REIS, 2010):

- Percepção: como fazer o uso objetivo dos recursos de sensoriamento para favorecer a coordenação; como usar a visão na comunicação implícita; como estimar informações a partir de outros jogadores e determinar para os quais deve focar-se a atenção.

- Comunicação: quais jogadores devem se comunicar e quando, quais informações devem ser trocadas e de que forma.

- Ações: quais ações de um jogador favorecem o time em cada momento e como avaliá-las. Como executar uma tarefa simples (chute), ou composta (drible).

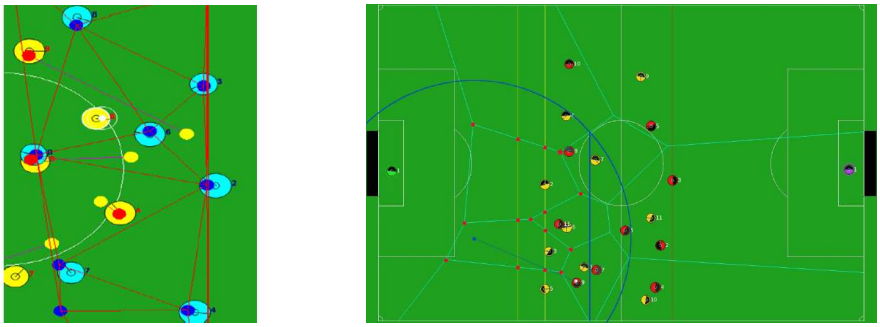
- Coordenação: como devem ser organizadas as dependências entre os jogadores e de suas ações junto à adaptação em tempo real. Como o treinador pode ser útil na coordenação.

Em virtude das questões listadas, as técnicas de coordenação podem ser agrupadas nos aspectos de comunicação e percepção inteligente, posicionamento (movimentação de defesa e movimentação de ataque), dos agentes individualmente e do time como um todo (ALMEIDA; LAU; REIS, 2010).

Apesar da limitação na troca de mensagens, entre as técnicas de coordenação por comunicação se encontra a modelagem do estado global. Em MALMIR et al. (2014), os jogadores utilizam a visão para mapear partes do campo e construir o estado global, enviando informações através do comando *say* (possibilita o envio de mensagens), alguns destes mantêm restrições de posicionamento no campo e do movimento do pescoço. Apenas o jogador que tiver o papel principal, em cada momento, tem acesso ao estado global. Em VELASHANI et al. (2014), passes longos, lançamentos ou passes em profundidade são determinados por meio da comunicação entre os agentes, também são coletadas informações dos jogadores aliados e oponentes, mas apenas um jogador sincroniza todas as informações coletadas. Na abordagem apresentada em MOHAMADI et al. (2014), apenas um jogador com uma posição visual favorecida fornece informações aos jogadores que estão engajados em tarefas ofensivas.

Alguns comandos específicos do simulador também ajudam na coordenação por comunicação, são estes: *point to*, que permite que um jogador indique uma posição (ex: para receber a bola), e *attention to*, que define um jogador para se prestar atenção (ex: enfoque do sensor que capta mensagens).

Entre as técnicas de posicionamento, cita-se a movimentação dos jogadores em relação a um ponto de referência (a bola ou os adversários) (PETER, 1998). Há também *Situation Based Strategic Positioning* (SBSP), na qual as formações táticas de um time são acionadas conforme as estratégias de jogo ativas, em cada formação os jogadores podem assumir diferentes papéis, regiões ou posições fixas (FERREIRA; REIS; LAU, 2004). Na técnica triangulação de *Delaunay*, o campo é dividido em triângulos - diferentes entre si - nos quais os jogadores devem encontrar as posições necessárias para atender a uma situação de jogo (ex: receber um passe) (MOHAMADI et al., 2014)(AKIYAMA; NODA, 2008). A técnica triangulação de *Delaunay* pode vir combinada à células de *Voronoi* ou esta última - que produz uma abordagem semelhante - pode ser utilizada isoladamente (PROKOPENKO; WANG; OBST, 2014). A Figura 8 ilustra as técnicas triangulação de *Delaunay* e células de *Voronoi*.



(a) Triangulação de Delaunay (b) Células de Voronoi

Figura 8: Técnicas de posicionamento no campo. A Figura 8(a) foi extraída de MOHAMADI et al. (2014). A Figura 8(b) foi extraída de PROKOPENKO; WANG; OBST (2014).

Outra técnica de coordenação por posicionamento é *Strategic Position by Attraction and Repulsion* (SPAR), na qual os jogadores se movimentam por atração em relação à pontos de interesse (jogador aliado com a posse da bola e o gol adversário) e repulsão em relação aos demais jogadores (adversários e membros de time sem a bola) (PETER, 1998).

Entre as técnicas de movimentação defensiva estão: marcação individual, marcação por região, marcação em linha, entre outras. Em ZHANG et al. (2014) é utilizada a técnica nomeada "rede defensiva". Nesta, cada jogador é responsável por uma região de marcação. Para isso um jogador deve escolher os oponentes a serem marcados. Estes têm as suas posições estimadas junto às regiões individuais a serem marcadas, que por sua vez, determinarão, após soma e normalização, a região final de marcação. A Figura 9 ilustra essa técnica.

Em MARIAN et al. (2014) são apresentadas técnicas de ação defensiva, como marcação em T, marcação em cone, marcação em antecipação, cobertura e *doubling up*. Todas envolvem a contenção do avanço do adversário ou a aproximação para o combate direto, salientado que, mesmo a marcação individual necessita da correta distribuição dos joga-

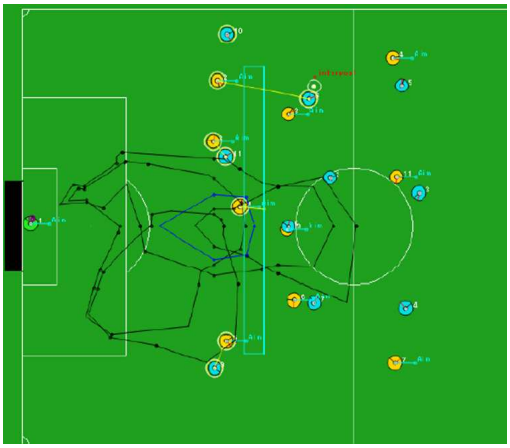
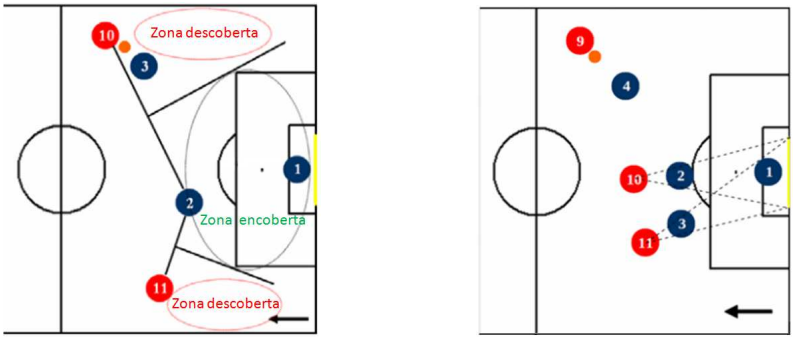


Figura 9: Técnica rede defensiva. Extraído de ZHANG et al. (2014). As linhas e círculos pretos indicam as regiões individuais de alguns oponentes, enquanto a linha e os círculos em azul indicam a região defensiva final do jogador que escolheu os oponentes.

dores no campo defensivo para ser efetiva. A Figura 10 apresenta duas dessas técnicas.



(a) Marcação em T (b) Marcação em cone

Figura 10: Técnicas de coordenação defensiva. Extraído e adaptado de MARIAN et al. (2014). Na Figura 14(a) a ação defensiva forma uma região encoberta, correspondente à toda área de gol e sua parte frontal externa, que se encontra na parte interna das duas marcações em T. Na Figura 10(b) os jogadores 2 e 3 se posicionam formando cones ou triângulos - com o intuito de proteger o gol - que têm por base as traves.

Já nas técnicas de movimentação ofensiva, encontra-se o deslocamento de jogadores na procura de espaços para a recepção de passes, deslocamento na direção do ataque para a recepção de lançamentos ou passes de infiltração e o posicionamento para evitar linhas de impedimento. Prover estas condições pode recorrer o uso de funções heurísticas ou outros métodos matemáticos como otimização por princípio de Pareto e *Simultaneous Perturbation Stochastic Approximation* (SPSA) (ALMEIDA; LAU; REIS, 2010). Pode também ser considerada a marcação sob pressão apresentada por MARIAN et al. (2014), onde o campo é dividido em três áreas, e cuja área próxima à saída de bola do oponente deve haver uma marcação agressiva, media marcação no meio campo e preenchimento de espaço no campo de defesa.

Entre as técnicas de coordenação individual de jogadores podem ser citadas o algoritmo *Q-learning*, lógica *Fuzzy* (AKIYAMA et al., 2014) e aprendizagem de máquina

(MALMIR et al., 2014). A coordenação individual está relacionada às tarefas locais dos agentes, à tomada de decisões e à gerência local de tarefas (ALMEIDA; LAU; REIS, 2010). *Q-learning*, por exemplo, é utilizado por OTTONI et al. (2014) para verificar a utilidade de estados do jogo diante de determinadas ações executadas pelos jogadores, uma vez que este algoritmo possibilita o aprendizado de funções de otimização sobre um espaço de estados-ações do ambiente.

As técnicas de coordenação de time incluem os *set-plays*, *Markov Decision Processes* (MDPs) e árvores de tarefas e técnicas de busca. Um *set-play*, por exemplo, é um plano de ações previamente definido para uma determinada situação de jogo que, geralmente, se repete (ex: cobranças de jogo) (PETER, 1998). Para a construção de *set-plays*, como descrito por FABRO et al. (2014), é necessária uma forma de modelagem de alto nível que considere os jogadores e suas ações, suportando o encadeamento de dependências e os critérios de transições das etapas no plano, bem como, do acionamento de um *set-play*. Para isso, FABRO et al. (2014), apresenta um *framework* (com linguagem de descrição de planos) e uma ferramenta para a criação de *set-plays* - *Splanner* - ilustrada na Figura 11.

Figura 11: Ferramenta para criação de *set-plays*, *Splanner* . Extraído de FABRO et al. (2014).

Outra forma de coordenar as ações de um time é realizar a modelagem do espaço de estados-ações de um jogo e construir uma árvore de busca que opere sobre o mesmo, determinado as sequências de ações de um plano, envolvendo um ou mais jogadores, conforme ilustrado na Figura 12 (AKIYAMA et al., 2014). Esse tipo de abordagem é adotado por vários grupos de pesquisa no campo de futebol de robôs simulado, sendo que, algumas das formas de modelagem são bem elaboradas e podem envolver a aplicação de táticas e restrições (PROKOPENKO; WANG; OBST, 2014), e de tarefas e políticas de recompensa ZHANG et al. (2014).

Em BAI; WU; CHEN (2012) é apresentado o *framework* MAXQ-OP. Este estabelece uma árvore de tarefas, a partir da qual são instanciadas cadeias de *markov*, cada uma

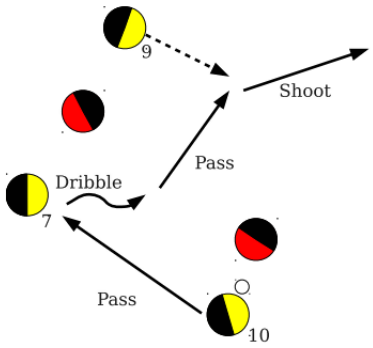


Figura 12: Exemplo de sequência de ações. Extraído de AKIYAMA et al. (2014).

representando uma subtarefa. A árvore de tarefas (ou comportamentos) comporta cadeias a serem performadas pelos jogadores que podem indicar sequências de ataques. Além disso, são utilizadas funções heurísticas e um método de busca de estados alcançáveis, para otimizar a procura das possíveis soluções na árvore de tarefas (BAI; WU; CHEN, 2012). A Figura 13 ilustra a árvore de tarefas do *framework* MAXQ-OP.

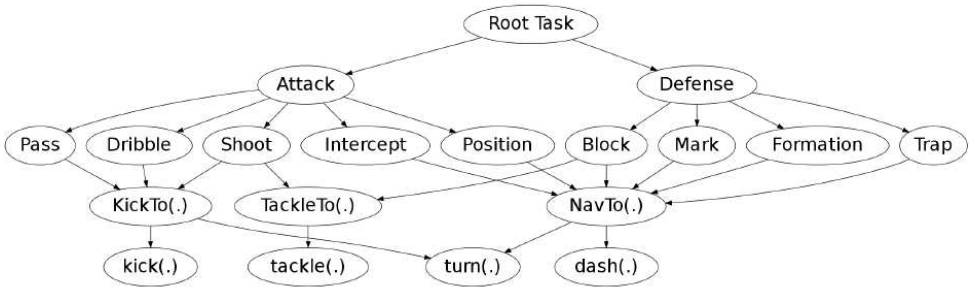
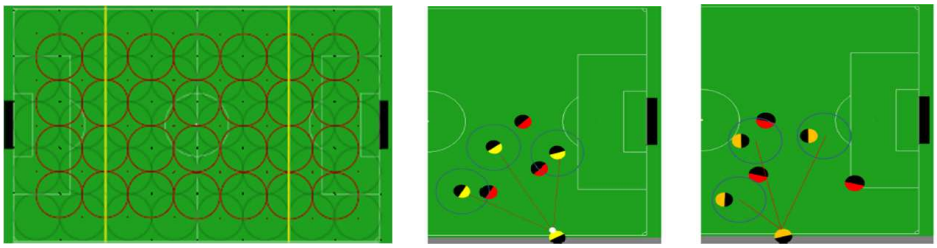


Figura 13: Árvore de tarefas do *framework* MAXQ-OP. Extraído de BAI; WU; CHEN (2012).

Em HIKICHI et al. (2014), a coordenação do time é realizada a partir do treinador e mensagens simples. Para isso, o campo é dividido em regiões circulares que determinam o raio de ação dos jogadores. Cada jogador tem uma área pessoal, no formato de círculo, cujo centro é o próprio jogador. O treinador conhece as posições dos jogadores e da bola e coordena as jogadas, ditando as posições e os momentos em que os passes devem ocorrer por meio de dois métodos: área pessoal e passe conduzido. Em ambos, o passe deve ocorrer para o jogador mais próximo, que esteja com a área circular livre de adversários, e cuja trajetória da bola esteja livre de obstáculos. A Figura 14 ilustra esse conceito.

Para a coordenação de um time, também pode ser considerado o uso do treinador e da linguagem *Coach Lang* (CLang). Esta é uma linguagem própria do treinador no *Soccer Server 2D*, utilizada na comunicação com os jogadores. O treinador tem acesso à mais informações do simulador e tem mais tempo para computar estratégias e ações a serem repassadas aos jogadores, o que envolve modelar o comportamento dos oponentes (IGLESIAS; LEDEZMA; SANCHIS, 2012).



(a) Divisão do campo (b) Área Pessoal (c) Passe conduzido

Figura 14: Coordenação de jogadores através do treinador. Na Figura 14(b) a direção do passe é o centro da área pessoal de cada jogador. Na Figura 14(c) a direção do passe é o centro de um círculo que resulta da adição de uma constante a área pessoal de cada jogador.

2.3 Abordagens de Teste em Sistemas Multiagente

Uma das condições necessárias a um sistema de computação - diante dos requisitos que sustentam seu projeto, desenvolvimento e execução em um ambiente - é a existência de técnicas e ferramentas com a capacidade de análise, verificação e validação do mesmo. O desenvolvimento de sistemas através do paradigma multiagente tem sido considerado o próximo passo a ser alcançado pela engenharia de software, desde o paradigma orientado a objetos (LUCK et al., 2005). Contudo, além do fato de SMA não serem a solução para todos os problemas no âmbito da computação, estes ainda não alcançaram maturidade suficiente para terem seu uso mais frequente nos problemas aos quais são opção (BOGG; BEYDOUN; LOW, 2008).

Uma das questões em aberto na concepção de SMA é a verificação e validação destes sistemas, que por sua vez, geralmente, são necessárias em todas as fases do processo de desenvolvimento. Atualmente, a área de SMA possui técnicas e ferramentas para projeto e implementação, mas sem um padrão de desenvolvimento e sem um padrão de teste definido, caracterizando-se muito mais como iniciativas de processos de desenvolvimento (HOUHAMDI, 2011). Segundo MILES et al. (2010), o principal motivo que dificulta a padronização de uma abordagem de teste e validação em SMA é a essência conceitual de um agente, bem como, de uma sociedade de agentes. As características dos agentes como autonomia, flexibilidade, imprevisibilidade, habilidades sociais, e comportamento dependente de contexto geralmente desconhecido, são fatores que implicam na difícil tarefa de produzir testes.

Considerando a concepção de um agente dotado de comportamentos que podem vir a ser assumidos para alcançar determinados objetivos, pode-se definir que para atingir a um objetivo, o agente utilizará um plano formado por: um estado final desejado para ambiente e tarefas ou subtarefas a serem realizadas (executadas através da combinação de comportamentos). Esta definição fornece a base para compreender as dificuldades encontradas ao se realizar testes em agentes, dentre as quais (MILES et al., 2010):

- **Difícil distinção do comportamento a ser testado** – Considerando que as

condições do ambiente, as características presentes no agente (pró-ativo ou reativo) e seu estado interno conduzem à várias possibilidades de planos e tarefas aleatórios a serem desempenhados, os comportamentos do agente vão estar condicionados ao contexto, que pode mudar e exigir adaptações. Inúmeros caminhos (planos) e arranjos de comportamentos dificultam qualquer abordagem de teste.

- **Aleatoriedade na execução de comportamentos** – Eventos e leituras do ambiente que incorrem na necessidade de reatividade por parte do agente podem gerar novas tarefas e ocasionar mudança de plano ou comportamento. As tarefas podem ou não ser interdependentes, sendo que, a arbitrariedade e a concorrência entre comportamentos dificultam abordagens de teste.

- **Rastreamento de ações** – É difícil traçar qual ação ou ações um agente executará para alcançar o estado final desejado, pode haver um grande número de possibilidades. Depurar o código, ou desenvolver um teste apropriado partindo-se do código é menos trivial do que em outras abordagens de programação.

- **Tratamento de falhas segundo o contexto** – Conforme o contexto de execução do agente e seu estado interno, as possíveis atividades concorrentes e a orientação assumida para alcançar um objetivo, há a elevação da dificuldade no problema de teste, quando da ocorrência de falhas e seu tratamento pelo agente. O número de casos de teste para as escolhas feitas pelo agente para o tratamento de falhas é grande devido à característica contextual.

Outros fatores que dificultam a realização de testes em agentes são (HOUHAMDI, 2011):

- A quantidade de dados pode ser enorme conforme o número de agentes e seus dados internos.

- É difícil determinar, com antecedência, as interações de um agente em um sistema durante sua execução.

- A comunicação entre agentes é comumente feita através de mensagens em vez de métodos. Isso torna as abordagens de teste da orientação a objetos praticamente não legáveis.

- A autonomia permite aos agentes agirem em discordância com as regras ou objetivos da sociedade a qual pertencem, mesmo que, agindo corretamente para si próprios. A mesma contradição pode ocorrer por parte da sociedade em relação ao agente.

Apesar das aparentes dificuldades, alguns trabalhos já têm se dedicado em criar abordagens de teste para agentes. Algumas destas abordagens podem ser vistas nos trabalhos

de revisão de testes em SMA de HOUHAMDI (2011) e (MILES et al., 2010). Nas próximas seções será apresentado o que se tem feito na área de testes em SMA. A princípio, pode-se salientar que as técnicas utilizadas para testes em agentes são focadas em características específicas da área. Dentre as características de teste estão: o agente isolado, a comunicação apenas, ou a organização do SMA apenas.

2.3.1 Foco no Agente

MILES et al. (2010) argumentam a favor da combinação de técnicas de verificação (fase de modelagem) e teste (agente já implementado), utilizando uma técnica intermediária que possa fazer a ligação entre as técnicas aplicadas nos modelos abstrato e concreto, respectivamente. A intenção é encontrar e validar o mesmo problema para estas duas fases distintas. Estes ainda apresentam o método evolucionário como suporte na fase de teste dos agentes.

O método evolucionário, detalhado por NGUYEN et al. (2009), faz uso de algoritmos evolucionários para testar agentes já inseridos no ambiente. Estes algoritmos (algoritmos meta-heurísticos) procuram guiar o aperfeiçoamento do agente segundo os nomeados "*soft goals*", que são características qualitativas inferidas e estabelecidas na análise de requisitos. Casos de testes mais difíceis são gerados para avaliar os agentes, que no caso de reprovação, devem ser melhorados para se adequarem aos testes. As variáveis qualitativas geralmente se referem ao desempenho ou segurança (ex: robustez, eficiência, diminuir custos).

Além disso, é feito uso de um algoritmo genético para mudar algumas características presentes no ambiente do agente em cada etapa (ex: mudar objetos físicos de lugar). Desta forma, o agente deve apresentar um desempenho adequado, em cada novo ambiente gerado. Os agentes que apresentam melhores resultados são selecionados para passar por um método de *crossover* (combinar por abordagem genética dois indivíduos) para gerar novos agentes (NGUYEN et al., 2009).

A abordagem evolucionária, no entanto, não avalia processos internos do agente, nem características sociais entre os agentes. Avalia apenas a adequação do agente segundo variáveis de otimização. A principal defesa a esse método é o baixo custo computacional e tempo despendido em avaliações, uma vez que outras abordagens implicam em avaliar o comportamento do agente por meio de cada método que este possa executar. Ainda é possível fazer considerações sobre um agente e seus processos internos através do comportamento apresentado nos testes, mesmo que partindo de uma abordagem analítica e não automática.

Em COELHO et al. (2006), o foco dos testes é nos comportamentos dos agentes ao realizarem papéis específicos. Para tanto é utilizado o conceito de "*Mock Agent*", que é um protótipo de agente com funções mínimas. Este agente contém apenas as interfaces dos métodos (sem implementação), para simular uma interação com o agente a ser tes-

tado. Esse conceito visa isolar o agente a ser testado, pois um agente tende a ter várias interações. O "*Mock Agent*" interage com o agente, enquanto outros componentes do sistema fornecerão dados da interação, notificarão falhas, e outras condições específicas ligadas ao agente testado.

Esta abordagem, nomeada "*Mock Agent*", é desempenhada por cinco componentes (COELHO et al., 2006):

- ***Test Suite*** – Um conjunto de casos de teste e operações que preparam o ambiente para teste;
- ***Test case*** – Define um cenário e série de condições que o agente sob teste enfrentará;
- ***Agent Under Test* (AUT)** – O agente sob teste;
- ***Mock Agent*** – Implementação falsa que irá interagir com o AUT;
- ***Test Monitor*** – Responsável por monitorar o ciclo de vida do agente para indicar seu estado ao *Test-Case*.

A partir dessa abordagem, passa a ser um quesito fundamental uma seleção de casos de teste que consigam corresponder com os erros mais comuns e graves e posteriormente expandir o quanto possível a detecção de erros. Os autores providenciam um processo de identificação de casos de testes, e orientam a produção de testes dirigidos a um único papel por vez. Para a implementação dos componentes relacionados ao teste, foi realizada a extensão do *framework* de teste *JUnit*. Além disso, o teste monitor é implementado através de extensão da linguagem orientada a aspectos do java, com o intuito de capturar a ocorrência de métodos no agente a ser testado (COELHO et al., 2006).

SIVAKUMAR (2012) apresenta uma abordagem de teste, também direcionada ao papel que o agente executa. É apresentado um modelo mais detalhado de definição do papel de um agente, que considera o conceito de responsabilidades. Cada papel pode incluir uma ou várias responsabilidades. Um teste sobre um agente deve considerar os objetivos, papéis desempenhados e responsabilidade de cada papel. Isto forma uma estrutura hierárquica que deve ser identificada no desenvolvimento do teste. Para tanto um roteiro de análise foi apresentado.

A abordagem de teste individual também considera a identificação da interação entre os agentes, eventos e pré-condições ligados às responsabilidades, e uma lista de critérios que satisfazem as responsabilidades. As responsabilidades definem se um papel é desempenhado da forma como se espera (responsabilidades são apenas um nome diferente para os comportamentos do agente) (SIVAKUMAR, 2012).

2.3.2 Foco na Comunicação

Considerando a troca de mensagens e a escalabilidade em plataformas multiagentes, mais especificamente o número de *hosts*, agentes, mensagens trocadas, e serviços prestados, SUCH et al. (2007), propõem a criação de quatro *benchmarks* para avaliação de SMA a partir de dimensões de desempenho na comunicação. Sistemas distribuídos e de larga escala sempre são influenciados pela adição ou restrição de componentes. Os autores pretendem dimensionar através de seus *benchmarks* este impacto.

São listados quatro *benchmarks*, que baseiam suas análises no termo *average round trip time* (RTT). Este é determinado por meio de cada agente em cada rodada de mensagens (envio e resposta), ou seja, é feita a mensuração do intervalo de tempo entre o envio e a resposta de uma mensagem. O valor final em cada *benchmark* corresponde ao valor médio entre todos os RTT dos remetentes (agentes) de mensagens que estão sendo avaliados. Outro parâmetro que pode ser modificado e avaliado é o tamanho da mensagem enviada. Os quatro *benchmarks* propostos são (SUCH et al., 2007):

- ***Benchmark* 1: número de *hosts*** – Testa o desempenho de uma plataforma multiagente segundo o número de *hosts* que ela suporta. Para isso cada *host* possui apenas dois agentes (remetente e destinatário), com a intenção de impactar minimamente no teste;

- ***Benchmark* 2: recepção massiva (1 receptor)** – Testa o desempenho de uma plataforma quando há massiva ocorrência de trocas de mensagens, por parte de um ou mais agentes entre diferentes *hosts*. Nesse caso, o teste procura analisar um serviço prestado por um agente com grande taxa e volume de requisições;

- ***Benchmark* 3: recepção massiva (N receptores)** – Testa o desempenho de uma plataforma quando há massiva ocorrência de trocas de mensagens, por parte de vários agentes presentes em diferentes *hosts*. A análise é feita sobre capacidade dos canais de transmissão de mensagens e o *buffer* dos agentes em si;

- ***Benchmark* 4: número de agentes por *host*** – Testa o desempenho de uma plataforma multiagente segundo o número de agentes suportados em um único *host*, e o aumento gradual das trocas de mensagens entre estes agentes e os de outros *hosts* . A análise incide sobre o comportamento dos *hosts* ao lidar com o acréscimo de agentes (sobrecarga, diminuição do RTT, entre outros).

2.3.3 Foco na Organização

KLEINER et al. (2013) desenvolveram o *framework RMASBench* para o teste de aplicações multiagentes de busca e resgate. O framework é baseado no ambiente simulado da competição de robótica *"Search and Rescue"* da RoboCup. Segundo os autores, dentro do

framework foram desenvolvidos alguns algoritmos que são o estado da arte em coordenação de sistemas multirrobôs, entre eles, *Distributed Stochastic Search* (DSA) e *Max-Sum*. O uso, implementação, ou mesmo teste de aplicações já desenvolvidas ocorre por meio de interfaces que possuem métodos que devem ser implementados. Não há nenhuma indicação de como a análise das aplicações é realizada dentro do *framework*, nem se detalhes específicos do comportamento dos agentes podem ser monitorados e analisados.

3 ROBOCUP E O SIMULADOR SOCCER SERVER 2D - DESCREVENDO O AMBIENTE

Neste capítulo discorrer-se-á sobre a organização *RoboCup* e seu principal objetivo: fomentar o desenvolvimento de uma equipe de robôs humanoides apta a uma partida de futebol contra a seleção campeã mundial. Também será apresentado o simulador de futebol de robôs *Soccer Server 2D* conforme seu propósito de ser uma plataforma de teste e um ambiente para a atuação de agentes. Para entender por que o *Soccer Server 2D* foi escolhido para viabilizar as ideias propostas neste trabalho (gerar abordagens de testes em SMA) será necessário a compreensão de como este ambiente interfere, muitas vezes restringindo e condicionando elementos dos times, assim como, de seu projeto e execução.

Será demonstrado que, embora a *RoboCup* tenha estabelecido uma problemática centrada em uma única questão - o futebol de robôs - esta é bastante abrangente tanto no quesito complexidade quanto na dimensão dos campos de pesquisa e, ainda, na quantidade de tecnologias exigidas para que se chegue a uma solução plausível do problema em um futuro próximo.

3.1 A RoboCup e seu Desafio

Ao longo da história da computação foram estabelecidos marcos que fomentaram sua evolução científica. Estes, geralmente, foram traçados através da descrição de determinados problemas para os quais se delimitavam soluções. Isso é particularmente notável na área de Inteligência Artificial (IA) em problemas como: desenvolver um sistema capaz de ganhar do campeão mundial de xadrez (solucionado através do computador *Deep Blue*) (BITTENCOURT, 2006) e implementar um sistema capaz de ser aprovado no teste de *Turing* (solucionado pelo software *Eugene Goostman*) (READING, 2014).

Mais recentemente, um problema capaz de atender aos anseios no desenvolvimento da ciência e tecnologia na área da robótica e IA foi estipulado por iniciativa da *RoboCup*. Este se traduz no desenvolvimento de uma equipe de robôs humanoides apta a disputar uma partida de futebol com a seleção campeã mundial, até o ano 2050. Por ocasião da disputa, tratar-se-á da seleção que for detentora do título da Copa do Mundo mais

recente (CHEN et al., 2003).

A *RoboCup* nasceu em 1997, como resposta às iniciativas anteriores que visavam converter o futebol de robôs em um problema padrão que promovesse ambas: ciência e tecnologia. Já em meados de 1993, foram conduzidos vários estudos de viabilidade, que indicaram sinal positivo para a criação da primeira competição de futebol de robôs em junho daquele ano. Capitaneada por um grupo de pesquisadores - ao qual se incluíam Minoru Asada, Yasuo Kuniyoshi, e Hiroaki Kitano - a iniciativa do projeto, ocorrida até então no Japão, foi nomeada *Robot J-League*. No entanto, em menos de um mês, a iniciativa recebeu atenção de pesquisadores de todo o mundo. Estes tinham interesse que a competição fosse estendida a um projeto internacional. Assim, propôs-se a mudança do nome da competição para *"Robot World Cup Initiative"*, que resultou no acrônimo *RoboCup* (ROBOCUP, 1997).

Contemporaneamente à iniciativa, já havia pesquisadores utilizando o domínio do futebol de robôs para fins de pesquisa. No Japão, Itsuki Noda, membro do centro governamental de pesquisa *ElectroTechnical Laboratory (ETL)* desenvolvia o que veio a ser a primeira versão do simulador oficial de futebol de robôs da *RoboCup*. Outros pesquisadores pioneiros, reconhecidamente importantes no domínio do futebol de robôs - além de Minoru Asada na *Osaka University* - foram Manuela Veloso e seu estudante Peter Stone na *Carnegie Mellon University* (ROBOCUP, 1997).

A primeira competição internacional de futebol de robôs aconteceu em 1997, já sob a responsabilidade da *RoboCup*. Desde àquele ano até então, muito se tem feito e produzido a partir das temáticas ligadas ao futebol de robôs. Igualmente, durante esse tempo, muitas instituições e laboratórios do mundo todo têm participado regularmente da competição internacional, assim como, de seus vários eventos paralelos (conferências de publicações, competições em diversas ligas, palestras, entre outros) (REIS, 2003).

A meta final estabelecida pela *RoboCup* garantiu relativa complexidade ao desafio, tal qual a possibilidade de um desenvolvimento gradual em diversas frentes de pesquisa, que, por sua vez, resultou em um equilíbrio entre os esforços empreendidos e os resultados conquistados. Grande parte disso ocorreu devido à forma de organização empreendida pela *RoboCup*, que com o intuito de alinhar, medir, integrar e ampliar os esforços direcionados na solução do problema proposto, criou diversas ligas de competições de futebol de robôs e ou correlacionadas ao problema, sendo estas (ROBOCUP, 2014):

- **RoboCup Soccer** – Liga para competições de futebol, com as subdivisões:
  - **Simulation League** – Voltada para a concepção de agentes de *software*, estratégias de times e técnicas de IA. Subdivide-se em: *2D Soccer Simulation* e *3D Soccer Simulation*;
  - **Small Size League** – Robôs com rodas e de pequeno porte;

  - **Medium Size League** – Robôs com rodas e de médio porte;
  - **Standard Plataform League** – Utiliza como padrão os robôs pípedes NAO da empresa Aldebaran Robotics;
  - **Humanoid League** – Robôs bípedes com maior proximidade das características humanas. Subdivide-se em: *KidSize*, *TeenSize*, *Adult-Size*.

- **RoboCup Rescue** – Liga para robôs de resgate em desastres ou situações de perigo, com as subdivisões:
  - **Robots League** – Adota robôs físicos;
  - **Simulation League** – Adota robôs virtuais.

- **RoboCup@Home** – Liga destinada a robôs que auxiliam em tarefas domésticas.

- **RoboCup@Work** – Liga destinada a robôs móveis que realizem trabalhos operacionais em aplicações industriais, cooperando com seres humanos, seja em tarefas de manufatura, automação, deslocamento de peças ou de logística.

- **RoboCupJunior** – Liga destinada a introduzir robôs aos alunos de séries primárias e secundárias. Subdivide-se em: *Soccer*, *Dance* e *Rescue*.

Inerente à essas ligas se encaixam estudos em Robótica e Inteligência Artificial (IA) que, segundo REIS (2003), incluem problemas como: coordenação, cooperação e comunicação multiagente, arquiteturas de agentes, aprendizagem, planejamento em tempo real, decisão estratégica e tática, comportamento reativo, visão computacional, processamento e análise de imagem, controle, sistemas de locomoção, sistemas sensoriais, fusão sensorial em tempo real, navegação e controle robótico.

Ainda segundo REIS (2003), os desafios proporcionados pelo futebol de robôs são muito superiores em complexidade frente a problemas de IA anteriores, entre estes, o já citado jogo de xadrez. Ambos, porém, xadrez e futebol de robôs permitem a comparação e avaliação de diversas arquiteturas, metodologias e algoritmos. No entanto, fica evidente a complexidade superior do futebol de robôs quando contrastado com o jogo de xadrez. Na Tabela 1 são dispostas as principais diferenças entre esses dois problemas.

Fornecido um novo problema padrão, muitas áreas desenvolveram estudos e, sendo de interesse para o presente trabalho, tem-se destacado sistemas multiagente (SMA). Em especial, a competição *Simulation League* abrange assuntos relativos ao desenvolvimento de SMA, e as extensões que pode adquirir. Como exemplo, tem-se o aprendizado e síntese de ações em ambientes multiagente, e aquisição de conhecimento em SMA, dentre outros temas (GONÇALVES, 2006).

Tabela 1: Diferenças dos domínios xadrez e futebol de robôs. Extraído e de REIS (2003).

Característica	Xadrez	Futebol de Robôs
Ambiente	Estático	Dinâmico
Mudança de Estado	*Por turnos*	*Tempo real*
Acessibilidade de Informações	Completa	Incompleta
Resultados das Ações	Determinístico	*Não Determinístico*
Sensores de Leitura	*Discreta (simbólica)*	*Continua (não-simbólica)*
Utilização de Atuadores	*Discreta (simbólica)*	*Continua (não-simbólica)*
Controle	Centralizado	Distribuído

3.2 Simulador Soccer Server

Parte da iniciativa de tratar o problema - proposto pela RoboCup - resultou na plataforma de teste *Soccer Server 2D*. Criado para atender à *Simulation League*, o *Soccer Server 2D* fornece um ambiente simulado de duas dimensões, onde duas equipes de futebol de robôs podem disputar uma partida. Cada equipe é controlada por um sistema de livre implementação, possibilitando o uso de várias técnicas, arquiteturas e estratégias diferentes. O ambiente simulado no *Soccer Server 2D* é considerado multiagente, de tempo real, semi-estruturado e, ainda, imprevisível, uma vez que o espaço de estados possíveis é amplo e quase sempre não determinístico (CHEN et al., 2003).

O *Soccer Server 2D* permite a implementação de agentes autônomos através de qualquer linguagem de programação (desde que suporte o protocolo UDP/IP). Para tanto, faz uso da abordagem cliente/servidor, sendo necessário que o cliente (no caso o agente) se conecte no programa servidor (trata-se do *soccerserver*, um componente do simulador de nome idêntico a este), o qual fornece um campo virtual e simula os comportamentos físicos da bola e dos agentes. Por meio de comunicação UDP/IP, o código de cada cliente pode enviar os comandos necessários ao servidor para executar ações no ambiente (CHEN et al., 2003). O campo virtual, bem como, a disposição dos robôs de duas equipes é ilustrado na Figura 15.

A constituição do simulador *Soccer Server 2D* é facultada principalmente a dois programas que executam em paralelo, o *soccerserver* e o *soccermonitor*, os quais (CHEN et al., 2003):

- **Soccerserver** – é um programa servidor que simula os movimentos da bola e dos jogadores, comunica-se com os clientes e controla o jogo de acordo com as regras.

- **Soccermonitor** – é o programa encarregado de gerar um campo virtual na tela do computador a partir do modelo numérico do *soccerserver*. Além de prover uma interface com o servidor, o *soccermonitor* apresenta, por padrão, as informações relativas ao placar do jogo, nome das equipes e as posições dos jogadores e da bola.

Figura 15: Interface do monitor do simulador *Soccer Server 2D*. Pode ser observado na figura que as equipes Yushan e Oxsy já possuíam todos os seus jogadores conectados ao servidor. Também estão presentes as informações do placar do jogo e o estado da partida, que não havia iniciado. O monitor encontrava-se em sua configuração padrão básica.

> Todavia podem ser selecionadas informações adicionais referentes aos jogadores e o campo para serem apresentadas.

A comunicação no *Soccer Server 2D* ocorre entre os clientes e o servidor através de conexão UDP/IP. Não há comunicação direta entre clientes, que devem se comunicar sempre por via do servidor (que redireciona as mensagens). Todo cliente é um processo único que se conecta em uma porta específica do servidor via *socket*, pela qual recebe informação visual e auditiva e envia comandos até o servidor (CHEN et al., 2003).

O protocolo UDP/IP, bem como a exigência de comunicação pelo servidor, deve ser entendida como um conjunto de restrições do ambiente, representando um meio não confiável e de banda limitada (GONÇALVES, 2006). Justamente, para tornar mais difícil o desafio, está prevista a possibilidade de o servidor enviar propositadamente ruídos na informação visual e auditiva destinada aos clientes, sendo necessário que estes estejam preparados para identificarem as imperfeições e se adaptarem as mesmas (CHEN et al., 2003).

Destaca-se que uma equipe de futebol deve ser composta por onze jogadores (dez na linha e o goleiro) e um treinador, somando um total de doze clientes conectados ao servidor por equipe. Cabe aos jogadores fazerem uso dos comandos disponíveis no *Soccer Server 2D* (chamadas de funções) para desempenharem capacidades básicas como chutar, passar, driblar, comunicar-se, entre outras ações. O treinador, especificamente, pode conhecer os ruídos presentes nas informações e se comunicar com os jogadores via servidor (CHEN et al., 2003). Além disso, um módulo interno do *Soccer Server 2D*, denominado Árbitro, provê a arbitragem da partida permitindo, todavia, a intervenção de um árbitro humano no caso de condições não previstas pelo algoritmo deste módulo

(GONÇALVES, 2006). A Figura 16 ilustra a arquitetura geral e os módulos que integram o *Soccer Server 2D*.

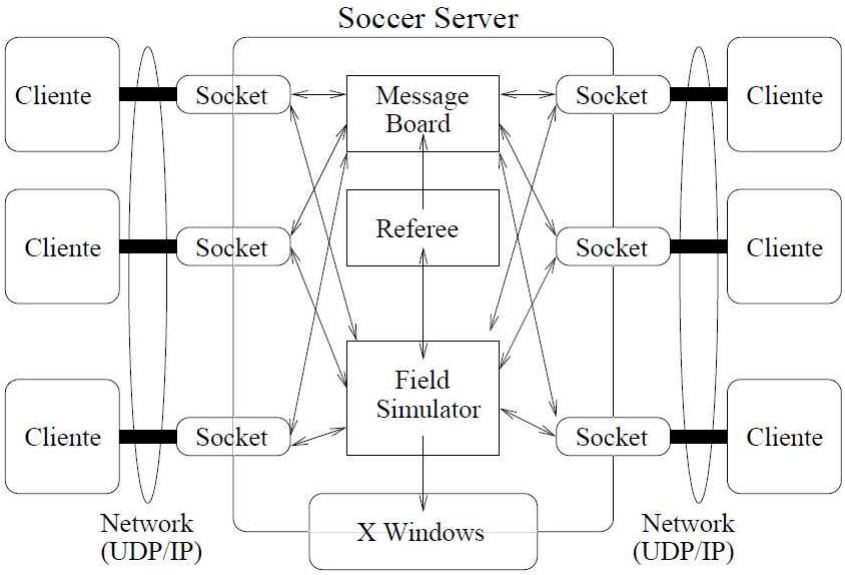


Figura 16: Arquitetura do *Soccer Server 2D*. Extraído de (GONÇALVES, 2006). O simulador *Soccer Server 2D* foi projetado em dois módulos principais, o *soccerserver* e o *soccermonitor*. Na arquitetura o *soccerserver* é representado pelos componentes *Message Board*, *Referee* e *Field Simulator*, que fornecem respectivamente a comunicação, a arbitragem e a simulação física dos elementos do jogo. Já o *soccermonitor* é representado pelo componente *X Windows*, uma vez que, este é o protocolo utilizado para o desenvolvimento da interface gráfica. Os clientes conectam-se ao simulador via *socket* e enviam mensagens auditivas e de comandos, assim como, recebem do simulador mensagens com as informações auditivas ou sensoriais

Cada agente no *Soccer Server 2D* possui três sensores, sendo estes (GONÇALVES, 2006):

- **Sensor auditivo** – Capta as mensagens difundidas pelos demais agentes, pelo árbitro e pelo treinador.
- **Sensor visual** – Fornece a informação relativa as distâncias e direções de objetos dentro do campo de visão do agente em intervalos regulares de 150ms.
- **Sensor corporal** – Informa o estado físico atual do agente dentro da partida como, por exemplo, energia, velocidade e ângulo do pescoço.

Estes sensores proporcionam a cada agente relativa similaridade com as condições de percepção que são enfrentadas por um jogador humano. Tudo com o intuito de proporcionar fiabilidade com o mundo real. Alguns aspectos são: condições de visão limitada a determinada distância, a audição também limitada à distância, percepção e ações sofrem interferência de ruídos, os jogadores tem energia limitada, entre outros. Na próxima subseção serão apresentadas características técnicas dessas limitações.

3.2.1 Restrições aos Sensores dos Agentes

O *Soccer Server 2D* prevê restrições aos sensores auditivo, visual e corporal dos jogadores. A intenção é condicioná-los às limitações similares às características humanas de cognição. As restrições são incorporadas por meio de parâmetros de configuração com valores padronizados. Alguns destes estão presentes no arquivo *"server.con"*, enquanto outros são compilados junto ao servidor. As informações dos sensores são encaminhadas pelo servidor via pacotes de dados, que por sua vez são estruturados para incorporarem os parâmetros.

Cada tipo de informação - seja auditiva, visual ou corporal - que chega a um agente tem estrutura e parâmetros próprios. Os termos que caracterizam a estrutura de dados que contém as informações do sensor auditivo de um jogador são (CHEN et al., 2003):

- ***Hear_decay*** – Define um valor de custo para cada mensagem ouvida pelo jogador. O *hear_decay* é decrescido da capacidade auditiva do jogador. Um agente deve ter, na capacidade auditiva, no mínimo o valor estabelecido em *hear_decay* para ouvir uma mensagem.

- ***Hear_inc*** – Define um valor a ser acrescido a cada ciclo na capacidade auditiva do agente.

- ***Hear_max*** – Define a capacidade máxima auditiva de um jogador.

- ***Audio_cut_dist*** – Define, em metros, a distância máxima entre dois jogadores em que uma mensagem pode ser ouvida.

- ***Say_msg_size*** – Define o tamanho máximo, em *bytes*, de uma mensagem.

Com o intuito de evitar a sobrecarga na capacidade de comunicação de ambos os times, os jogadores têm capacidades de audição diferentes. Para cada jogador, no entanto, estabelece-se, no mínimo, a capacidade de ouvir uma mensagem a cada ciclo. Um ciclo de sensoriamento auditivo é estabelecido em 1 segundo (CHEN et al., 2003). Os parâmetros do sensor auditivo e seus respectivos valores padrões são listados na Tabela 2.

Tabela 2: Parâmetros do servidor que influenciam o sensor auditivo. Extraído e adaptado de CHEN et al. (2003).

Parâmetros	Valor Padrão
audio_cut_dist	50
hear_max	2
hear_inc	1
hear_decay	2
say_msg_size	512

O sensor visual também incorpora em uma estrutura de dados os parâmetros do servidor que definem as informações visuais. Cada jogador recebe essas informações automaticamente do servidor em ciclos discretos. Os termos que caracterizam o campo de visão de um jogador - ilustrado na Figura 17 - são (CHEN et al., 2003):

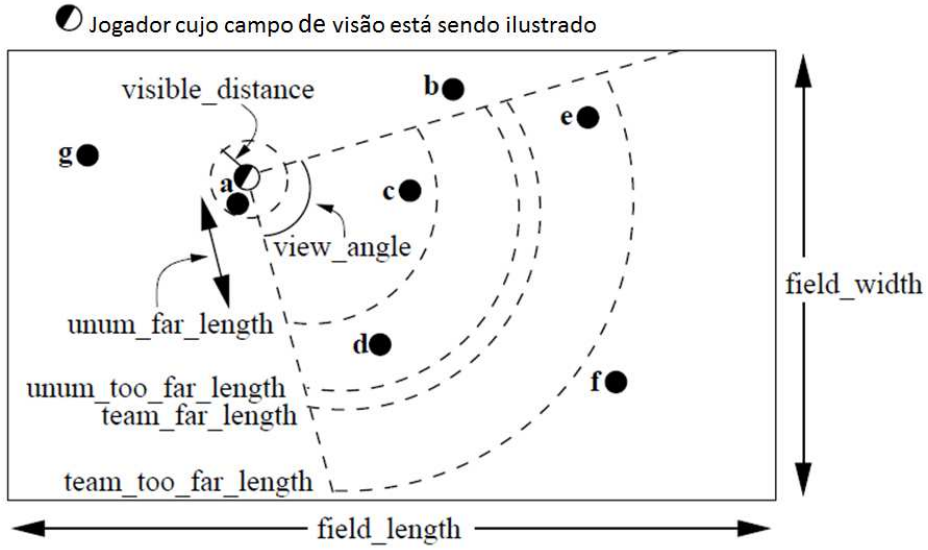


Figura 17: Campo visual de um jogador. Extraído de CHEN et al. (2003). Os pontos de *a* a *g* representam objetos presentes no campo de jogo e, apresentam condições diferentes de visualização para o jogador (CHEN et al., 2003).

- ***Sense_step*** – Determina cada intervalo de tempo (ciclo em milisegundos) em que a informação visual chega ao jogador.
- ***Visible_angle*** – Determina quantos graus possui o espectro de visão do jogador. Objetos fora da faixa fixada para o *visible angle* não podem ser vistos.
- ***Visible_distance*** – Determina a distância em metros que um objeto pode ser visto por um jogador.
- ***Unum_far_length, unum_too_far_length, team_far_length*** e ***team_too_far_length*** – Determinam os marcos (distâncias em metros) em que a capacidade visual do agente pode reconhecer características presentes em outros jogadores, que são respectivamente:
  - *unum_far_length* – Em distâncias abaixo deste marco, o número de uniforme e nome do time de outro jogador podem ser vistos.
  - *unum_too_far_length* – Em distâncias entre *unum_far_length* e *unum_too_far_length*, o nome do time é sempre visível, porém, a probabilidade de reconhecer o número de uniforme decresce de 1 à 0, conforme a distância aumenta.

  - *team_far_length* – Em distâncias abaixo deste marco, o nome de time pode ser visto.
  - *team_too_far_length* – Em distâncias entre *team_far_length* e *team_too_far_length*, a probabilidade de reconhecer o nome do time decresce de 1 à 0, conforme a distância aumenta.

- ***Quantize_step* e *quantize_step_l*** – Estes parâmetros são fatores que modificam os valores reais - obtidos por meio de cálculos - das distâncias dos objetos (bola e jogadores) e marcações (linhas e *flags*), em relação ao jogador observador.

Nas distâncias que ultrapassam *unum_too_far_length* e *team_too_far_length* não podem mais ser vistos, respectivamente, o número de uniforme e o nome de time CHEN et al. (2003). Os parâmetros relacionados com o sensor visual possuem os valores padrões listados na Tabela 4.

Tabela 3: Parâmetros do servidor que influenciam o sensor visual. Extraído e adaptado de CHEN et al. (2003).

Parâmetros	Valor Padrão
sense_step	150
visible_angle	90
visible_distance	3
unum_far_length	20
team_far_length	40
team_too_far_length	60
quantize_step	0.1
quantize_step_l	0.01

O sensor corporal, por sua vez, reporta as condições físicas correntes do jogador. Cada jogador recebe essas informações automaticamente do servidor em ciclos discretos. Os principais parâmetros relacionadas com o sensor corporal são (CHEN et al., 2003):

- ***Sense_body_step*** – Determina cada intervalo de tempo em que a informação corporal chega ao jogador. O valor padrão de cada ciclo de sensoriamento corporal é 100 milissegundos.

- ***View_mode*** – Possui as variáveis *ViewWidth* e *ViewQuality* que, caso modificadas, influenciam na frequência e qualidade de informação visual advinda do servidor. As opções possíveis são: *normal*, *narrow* e *wide* para *ViewWidth*; e *high* e *low* para *ViewQuality*.

- ***Stamina*** – Através das variáveis *Stamina* e *Effort* são indicados respectivamente: a quantidade de energia que o jogador possui e, o valor correspondente ao esforço

empregado em alguma ação - este último comporá o cálculo da quantidade de energia a ser decrescida da *Stamina*.

- ***Spead*** – É composta das variáveis *AmountOfSpeed* que indica a velocidade aproximada do jogador e *DirectionOfSpeed* que indica a direção aproximada em que o jogador se movimenta.

- ***Head_angle*** – Indica através da variável *HeadDirection* a posição relativa da cabeça do jogador.

- ***Variáveis de Contagem*** – São utilizadas para indicar o número de vezes em que determinados comandos foram executados pelo servidor. As variáveis são: *kick* (*kickCount*), *dash* (*DashCount*), *turn* (*TurnCount*), *say* (*SayCount*), *turn_neck* (*Turn_NeckCount*), *catch* (*CatchCount*), *move* (*MoveCount*), *change_view* (*Change_ViewCount*).

As variáveis de contagem representam a totalidade dos comandos que foram executados pelos jogadores em uma partida. Estes comandos coincidem com as ações disponíveis aos jogadores, entretanto, dependendo da função do jogador em campo, algumas ações não ficam disponíveis. Notadamente, os jogadores devem ser eficientes no uso dos comandos, pois, muitos deles representam gasto da energia disponível. Além das restrições impostas ao sensoriamento dos jogadores, outros parâmetros modelam e condicionam as ações no ambiente virtual.

3.2.2 Características da Simulação

O tempo, no *Soccer Server 2D*, é estipulado em intervalos discretos (ou ciclos) de 100 milissegundos. Uma simulação completa de uma partida ocorre em um total de 6000 ciclos (dez minutos), destinando-se 3000 ciclos (cinco minutos) para cada rodada da partida. As simulações ocorrem em tempo real. Como cada ciclo tem duração específica, as ações de agentes que precisam ser executadas em um dado ciclo devem chegar ao servidor durante o intervalo de tempo certo, sob pena de prejudicar a sincronização entre cliente e servidor (GONÇALVES, 2006).

Todos os comandos dos agentes precisam passar pelo servidor. Dessa forma, a comunicação, que envolve mensagens síncronas e assíncronas (ver Figura 18) é um ponto crucial a ser considerado (BITTENCOURT; DA COSTA, 1999). Algumas características específicas da troca de informações entre agente e servidor são (GONÇALVES, 2006):

- **Aceitação de comandos** – O servidor aceita um novo comando a cada 20 milissegundos.

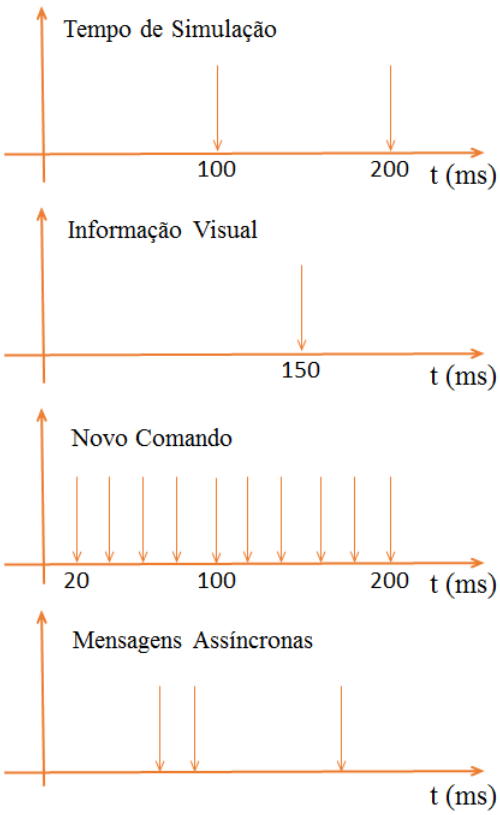


Figura 18: Temporização no *Soccer Server 2D*. Extraído de GONÇALVES (2006).

- **Informação visual** – Varia conforme os parâmetros *ViewWidth* e *ViewQuality*. Para o modo *normal* e qualidade da informação *high*, uma nova mensagem contendo a informação visual é recebida a cada 150 milissegundos.

- **Informações Auditivas** – São trocadas assincronamente pelo árbitro e pelos demais agentes.

Dentro do *Soccer Server 2D*, são viabilizadas para os agentes por meio de um conjunto específico de comandos básicos (chamadas de funções), ações de movimentação, aumento de velocidade, modificação do ângulo de visão, bem como chutar ou agarrar bola, entre outros. Cada comando possui restrições próprias, mas destaca-se a restrição do limite de tempo para a aceitação de comandos impostas pelo *Soccer Server 2D*. Embora, a regra geral seja de aceitação de um novo comando a cada ciclo de 20 milissegundos, comandos específicos como *turn*, *kick* ou *dash*, só podem ser executados em ciclos de simulação de 100 milissegundos (GONÇALVES, 2006).

3.2.3 O Registro das Partidas e o Programa Logplayer

Durante a execução de uma partida, o servidor (*soccerserver*) fica encarregado de gravar as informações geradas na simulação em dois arquivos de *log*. Em um dos arquivos (de extensão RCL) são registrados todos os comandos - de ações e de mensagens - solicitados ao servidor pelos jogadores e produzidos no ambiente virtual. No outro arquivo

(de extensão RCG), são registradas as mesmas informações que são enviadas ao monitor (*soccermonitor*) quando da reprodução virtual de uma partida. Fazendo-se o uso de programas específicos e dos dados gravados no *log* (RCG), é possível a repetição da partida como se fosse um vídeo.

Um dos programas que permite a reprise das partidas é o *Logplayer*. Este acompanha a instalação do simulador *Soccer Server 2D*. O *Logplayer* pode reprisar arquivos RGG ou apresentar uma partida em tempo real ao se conectar diretamente ao *soccerserver*. Além disso, pode enviar algumas de suas informações ao *soccermonitor*, embora possua sua própria interface gráfica. A Figura 19 ilustra o intercâmbio de informações entre os módulos do *Soccer Server 2D* e o *Logplayer*.

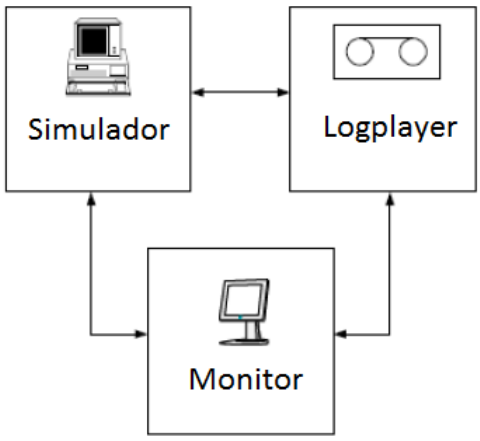


Figura 19: Interação dos módulos do *Soccer Server 2D* e o *Logplayer*. Extraído de REIS (2003). O *soccermonitor* pode tanto se conectar ao *soccerserver* e realizar a apresentação de uma partida em tempo real quanto se conectar ao *Logplayer* e apresentar uma partida gravada em um *log* RGG. Ao *Logplayer* é facultada a opção de se conectar ao *soccerserver* ou reprisar um arquivo de *log*

O *Logplayer* é considerado um programa auxiliar que permite a análise de alguns dos fatores relacionados a uma partida de futebol de robôs. Com a reprise do jogo e o uso das funções presentes nessa ferramenta é possível adotar abordagens de teste - geralmente visuais - nos comportamentos apresentados pelos agentes. A Figura 20 ilustra a interface do programa *Logplayer*, onde através dos botões do "menu", localizado na parte superior esquerda, é possível executar funções de vídeo tais como: iniciar, parar, avançar, diminuir ou aumentar a velocidade de reprise da partida.

O *Logplayer* tem funções de análise limitadas. Entretanto, existem outros programas que obtém avaliações a partir dos dados gravados nos arquivos de *log*. No Capítulo 5 serão apresentados e discutidos alguns destes programas, entre os quais o *Soccer Server Statistical Extracting Tool* (SSSET), que foi utilizado no âmbito deste trabalho. Destaca-se que os arquivos de *log* são a base para muitos dos processos de análise desenvolvidos para o *Soccer Server 2D*.

Existem diversas versões (ou protocolos) do arquivo de *log* do tipo RCG. Estas foram criadas ao longo das gerações do simulador *Soccer Server 2D*, na busca do aperfeiçoamento

Figura 20: Interface do programa *Logplayer* com algumas de suas funções ativas. Os círculos vazios ao redor do campo representam os *flags* de orientação visual. O valor impresso a direita de cada jogador se refere a sua *stamina* (energia). As linhas de impedimento (nas cores amarelo e ciano) são destacadas para ambas as equipes, assim como, a previsão do movimento de um dos jogadores, por meio de uma linha diagonal (na cor roxa).

da estrutura de registro do *log*. Para os fins do presente trabalho será utilizado a versão três do protocolo que se encontra documentada no manual do *Soccer Server 2D* para as versões do simulador 7.07 em diante. A versão três do arquivo de *log* contém a seguinte estrutura (CHEN et al., 2003):

- **Cabeçalho do arquivo** – Possui os caracteres "ULG", que indicam o início do arquivo.
- **Versão do arquivo** – Um único caráter que define a versão do arquivo de *log*.
- **Corpo do arquivo** – Contém registros com as informações das partidas, que seguem a seguinte estrutura:
  - *PM_MODE* – Um caráter que especifica em que modo a partida se situa (ex: pré-jogo, intervalo, intercorrendo, entre outros). Gravações no arquivo somente ocorrem se há mudança de modo.
  - *TEAM_MODE* – Uma estrutura que armazena dados sobre os times e o lado do campo em que estão atuando. Somente ocorrem gravações quando um novo time se conecta ao servidor ou o placar é modificado.
  - *SHOW_MODE* – Uma estrutura que especifica a posição dos jogadores e bola. Registros acontecem em instantes de tempo.
  - *MSG_MODE* – Registra o tipo, tamanho e conteúdo das mensagens enviadas.

- *PARAM_MODE* – Registrado no início do arquivo, lista os parâmetros e valores fixos que o servidor utiliza no controle da simulação (ex: aceleração da bola e velocidade máxima permitida, valores padrões definidos para os sensores e suas propriedades, entre outros).
- *PPARAM_MODE* – Registra os valores padrões dos parâmetros que indicam limitações físicas dos jogadores e dos comandos que estes executam (ex: força do chute, área de alcance do goleiro, carga máxima de energia, entre outros). É gravado do início do arquivo.
- *PT_MODE* – Registra os valores das propriedades dos jogadores antes do início de uma partida, sendo que, algumas podem ser alteradas ao longo de uma partida (ex: carga de energia, altura do jogador, força do chute, entre outros). É gravado do início do arquivo.

A estrutura do *log* permite o acesso à todas as informações geradas durante uma partida de futebol. Na seção 5.3 e no capítulo 6 será demonstrado como estas informações trarão os subsídios para as abordagens de avaliações pretendidas neste trabalho.

4 TRABALHOS RELACIONADOS

Neste capítulo serão abordados trabalhos relacionados diretamente às avaliações de agentes no *Soccer Server 2D*. Assim como, será brevemente discutido alguns trabalhos relacionados à análise e avaliação do futebol praticado por humanos, pois alguns de seus conceitos são igualmente aplicáveis ao futebol de robôs.

Durante os anos de história da Robocup, a *Simulation League* teve uma evolução gradual - porém expressiva - que pode ser observada através da plataforma de simulação *Soccer Server 2D* (AKIYAMA; DORER; LAU, 2014). Esta teve a capacidade de simulação incrementada por meio de mudanças nos atributos físicos dos jogadores, nas regras de jogo, nas características de comunicação, entre outros. Com isso, procurou-se fornecer características mais reais à simulação. Segundo AKIYAMA; DORER; LAU (2014), independente das conquistas já alcançadas, ainda há espaço para avanços, mesmo após não ter havido nenhuma mudança considerável no simulador (um dos fatores que induziam a evolução das equipes) desde 2010. Campos de pesquisa como a avaliação em tempo real das partidas junto à adaptação das equipes, e os aspectos táticos dos times têm tido relativa importância na busca de novos resultados.

4.1 Trabalhos de Análise e Avaliação Destinados ao Soccer Server

O *Soccer Server 2D* teve uma evolução gradual. Este possibilitou às equipes de pesquisa que trabalham com futebol de robôs terem sempre à frente um problema instigante com a possibilidade de adaptação às mudanças. No entanto, segundo GABEL; RIEDMILLER (2011), essa modificação gradual (ver Figura 21) também é um empecilho às abordagens de avaliação dos times e dos agentes que os compõem, destacando que, quando há mudanças, os times de novas edições da competição podem contar com vantagens que tornam a comparação com times de anos anteriores injusta.

Todavia, GABEL; RIEDMILLER (2011) conseguiram provar, a partir do período de competições entre 2003 e 2007 - no qual não houve mudanças drásticas no *Soccer Server 2D* - que os times continuaram a evoluir, competição após competição. Essa comparação

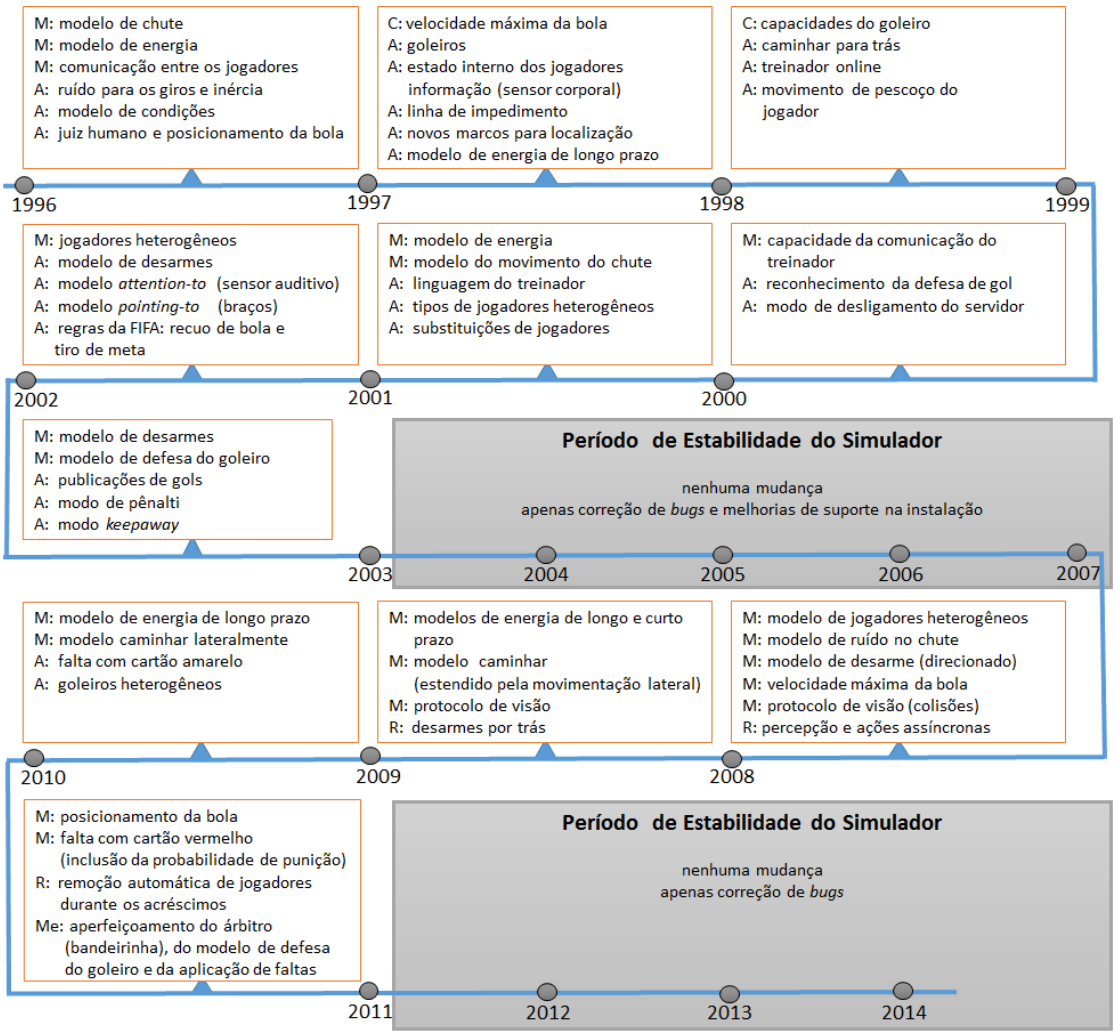


Figura 21: Histórico de mudanças no *Soccer Server 2D*. Extraído e adaptado de GABEL; RIEDMILLER (2011). A letra "M" indica os elementos do simulador que foram modificados, a letra "A" indica os elementos que foram adicionados ao simulador. A letra "R" indica as características que foram removidas e "Me" indica os elementos que foram melhorados.

foi feita com base no número de gols marcados e no número de vitórias, em partidas disputadas igualmente entre os times. Não obstante, GABEL; RIEDMILLER (2011) também provou, através da repetição das partidas referentes às finais, que não ocorreu convergência de desempenho entre os times, uma vez que os campeões mantinham um percentual de vitórias, geralmente constante, sobre seus vices.

Apesar da dificuldade na comparação, elencada por GABEL; RIEDMILLER (2011), muitas equipes de pesquisa no campo de futebol de robôs procuraram desenvolver técnicas e ferramentas de análise. Algumas iniciativas, como salientado por IGLESIAS; LEDEZMA; SANCHIS (2010), surgiram por meio dos desafios complementares da *Simulation League*. Destaca-se o desafio de modelagem dos comportamentos de agentes proporcionado pela competição de treinadores (*RoboCup simulation coach competition*), onde várias técnicas foram criadas (KAMINKA et al., 2003) (KUHLMANN; KNOX; STONE, 2006) (IGLESIAS; LEDEZMA; SANCHIS, 2012).

Geralmente, a modelagem dos comportamentos de agentes é voltada à compreensão das ações, uma vez que estas podem ser identificadas e delimitadas em eventos no jogo (chute, passe, drible, falta ou cobranças de jogo). Assim, surgem questões como: determinar por que um agente executou certa ação e se esta era a esperada para o contexto em que o agente se encontrava; denotar o comportamento de times e de grupos pertinentes a estes através das ações na partida; identificar estratégias e planos de ataque, entre outros. Por isso, BEZEK (2004) apresenta uma solução através de grafos, que permite a análise das sequências de ações, dos estados da partida e das estratégias de time adotadas. Essa técnica veio a ser implementada no programa *Logalyzer*, que ver-se-á no Capítulo 5.2.

Frisa-se que, essencialmente, um jogo de futebol é constituído de ações e interações, e dos eventos resultantes destes. Assim, muitas abordagens de análise são voltadas para a extração dessas características, ou seja, a extração de ações, interações e eventos produzidos pelos jogadores em uma partida.

Em RILEY; STONE; VELOSO (2001), por exemplo, foi criada uma abordagem para compreender as ações dos agentes por meio da inclusão na arquitetura destes da capacidade de gerar *logs* dos seus estados internos. Com isso, são fornecidas as condições necessárias aos desenvolvedores para que se identifiquem as razões que levaram os agentes a tomarem determinadas ações. A avaliação é feita em níveis de detalhamento, retroativamente ou no momento da ação. As informações do agente são organizadas em camadas, permitindo níveis de abstração. Essa abordagem provê: uma ferramenta de depuração para o desenvolvimento de agentes (ver Figura 22) e o controle interativo destes.

Em ALMEIDA et al. (2012), foi criada uma abordagem para extrair planos táticos de times na forma de *set-plays*. Um *set-play* é uma descrição de alto nível para um plano, no qual existe a representação dos passos a serem executados e suas condições de restrição, e dos agentes envolvidos e seus respectivos papéis.

Segundo ALMEIDA et al. (2012), para extrair um *Set-Play* é necessário um conjunto

Figura 22: Ferramenta de depuração criada através de uma extensão ao *Logplayer*. Extraído de RILEY; STONE; VELOSO (2001). A direita são apresentadas as informações de um jogador junto a opções de vídeo. O botão "P" permite selecionar o agente cujas informações serão dispostas. O botão "L" permite controlar a quantidade de informação apresentada (RILEY; STONE; VELOSO, 2001).

de *logs*, onde são identificados eventos relevantes como dribles, passes, gols ou faltas. Os eventos são ordenados conforme a sequência em que foram executados durante a posse de bola de um time. A sequência denota o comportamento cooperativo atingido pelo time por meio da execução de um plano.

A extração dos eventos é condicionada a determinados algoritmos que foram implementados no programa *SoccerScope2*. Os eventos são classificados segundo sua relevância. A sequência de ações é abstraída na representação de um *Set-Play*. Em cada *Set-Play* são identificadas os passos (eventos) que se sucederam e os jogadores envolvidos. No entanto, há uma limitação para a abordagem de análise criada, pois esta só considera as jogadas que se iniciaram de bola parada como cobranças de lateral e escanteios (ALMEIDA et al., 2012). Essa abordagem será melhor exemplificada mais a frente no Capítulo 5.3, pois resultou na criação de dois programas de análise (*Soccer Scientia Tool* e *Soccer Server Statistical Extracting Tool*), sendo que um destes foi diretamente utilizado no âmbito desta dissertação.

Em KARIMI; AHMAZADEH (2014) são utilizadas técnicas e algoritmos da mineração de dados para processar os *logs* do *Soccer Server 2D*. Para a obtenção de informações relevantes são utilizadas as posições e distâncias dos jogadores e bola. O algoritmo C4.5 é utilizado tanto para identificar quanto para prever o movimento dos jogadores.

Em CLIFF et al. (2014), é evidenciado a difícil tarefa de denotar comportamento cooperativo apenas pelas posições e movimentações dos jogadores. Em vez disso, uma nova abordagem de grafos direcionados - que representam pares de jogadores - é utilizada. Através desses grafos é disposta uma rede de transmissão e armazenamento de informação. A dinâmica da rede é analisada para o reconhecimento de comportamentos de coordenação apresentados pelo time, como o de enxame. A Figura 23 ilustra duas dessas redes no formato de diagramas.

Os diagramas demonstram as dependências entre os pares de jogadores e com isso

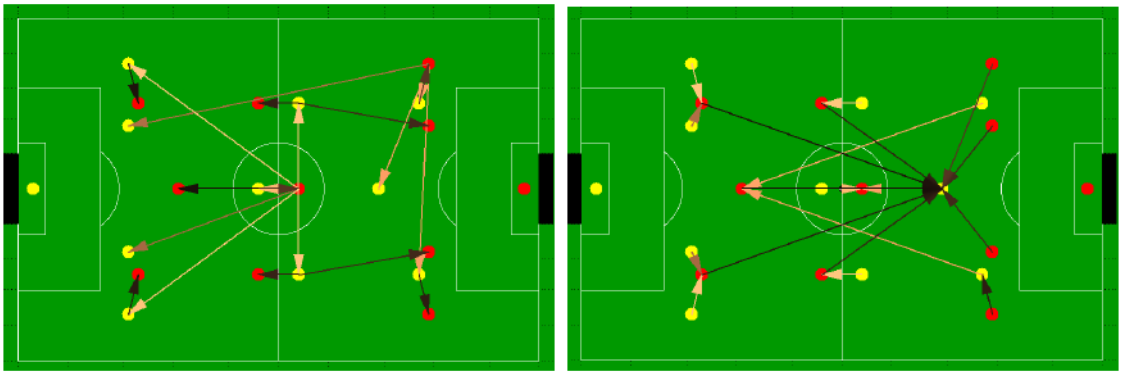


Figura 23: Diagramas de redes para dois times. Extraído de CLIFF et al. (2014). Os grafos direcionados representam o fluxo de informações, enquanto as cores nas setas indicam a intensidade da troca de informações.

é realizada uma analise tática. É notório que a dinâmica da rede e sua estrutura são fortemente ligadas à coordenação do time.

A extração de características no formato de sequências de jogadas aparece em grande parte das abordagens de análise, pois é uma condição natural pela dinâmica de uma partida de futebol. Essa natureza também é explorada em trabalhos de análise destinados ao futebol humano. Em TOVINKERE; QIAN (2001), são detectados os eventos e ações dos jogadores através das posições dos jogadores e da bola. Foi criado um esquema semântico para as definições dos eventos e ações, que são detectados algoritmicamente. Eventos básicos são utilizados para detectar os de mais alto nível.

Em PERIN; VUILLEMOT; FEKETE (2013) são analisadas fases no jogo, das quais também são extraídas as ações e eventos. Cada fase representa uma sequência de ações de um time até a perda da bola ou a concretização de um objetivo. Como se trata de análise por vídeo, os dados precisam ser preparados (por meio de anotação) e pré-processados, considerando o conhecimento de especialistas.

4.2 Considerações

Quando existe a pretensão de avaliação sobre um sistema já implementado, deve-se caracterizar até que nível da arquitetura funcional do sistema ou de seus componentes é possível exercer uma análise. Algumas plataformas e tecnologias de concepção de agentes permitem que se conheçam detalhes internos da arquitetura destes. No entanto, tem-se considerado que um agente deva ser um sistema fechado (visto como uma caixa preta), além de que, como visto na Seção 2.3, diversos aspectos dificultam uma análise interna do agente coerente e precisa.

As dificuldades inerentes à produção de testes em agentes estão igualmente presentes no ambiente *Soccer Server 2D*. Incluem-se nisso, os comportamentos dos agentes, sejam reativos ou de tomada de decisões e as interações de nível social. Embora o *Soccer Server*

2D seja um ambiente controlado, o espaço de estados-ações, em cada ciclo de execução, é considerável. Isso permite grande variabilidade na dinâmica do fluxo de execução. Monitorar internamente um agente, para verificar aspectos reativos ou cognitivos requer o acesso e o processamento de grande quantidade de dados. Seria necessário identificar como um agente organiza e integra os componentes responsáveis por ações reativas e ações deliberadas. Seria necessário considerar como um agente representa, organiza e executa planos de ações individuais e em grupo. Provavelmente, a abordagem que considerasse a arquitetura interna dos agentes somente seria aplicável à implementações que seguissem regras específicas.

Vide a proposição deste trabalho, pretende-se uma avaliação de aspectos de SMA dentro do ambiente *Soccer Server 2D* suficiente para produzir resultados e o mais externa possível para evitar detalhes específicos das tecnologias, arquiteturas, ou mesmo técnicas usadas na implementação, uma vez que estas podem ser as mais variadas possíveis. Nota-se que seria muito complexo criar métodos e parâmetros, que pudessem avaliar com integridade tamanha diversidade de soluções.

Todavia, há de se considerar que, quando se ignora a composição e execução interna dos agentes, criam-se dificuldades em determinar as interações destes, bem como, em determinar quando as interações resultam na realização de planos individuais ou de grupos. No *Soccer Server 2D*, os agentes podem interagir por comunicação direta ou indireta e, frequentemente, o fazem para seguir estratégias, formações táticas, jogadas, e quaisquer outras composições de jogo que envolvam dois ou mais jogadores e a coordenação destes.

Desconsiderando-se os aspectos internos de um agente, tem-se como alternativa focar nos seus comportamentos e, na medida do possível, nos aspectos de interação, tanto em relação ao ambiente quanto em relação à sociedade de agentes. Dentre as possibilidades de avaliação de um agente de forma isolada, encontra-se a de inserir componentes no ambiente para interagir com o agente e coletar dados da interação (COELHO et al., 2006) (ver Seção 2.3.1). No entanto, esta é logo descartada, pois não há espaço no *Soccer Server 2D* para adotar tal abordagem, enquanto se procura avaliar o SMA como uma equipe em sua totalidade (com doze clientes atuando em sua plenitude). Descartando-se qualquer intervenção externa, resta voltar-se ao registro de informações geradas no próprio ambiente.

O *Soccer Server 2D* fornece os arquivos de *log* como mecanismos para o registro dessas informações, seja das interações dos agentes ou dos comportamentos que estes apresentam. Inicialmente, os registros do *log* são informações básicas de todos os comandos que passam pelo *soccerserver*.

Considera-se que a abordagem de avaliação proposta se situará nos aspectos de organização, assim como, nos aspectos sociais das estruturas organizacionais “coalizões”, dentro dos times. Para tanto, pretende-se identificar quais jogadores interagem para formar essas estruturas, tentando apontar e avaliar os motivos incipientes. Antes de elucidar

a proposta, porém, no próximo capítulo serão apresentadas algumas ferramentas que fazem uso dos arquivos de *log* gerados pelo *Soccer Server 2D*, exemplificando o nível de avaliação social que estas alcançam.

5 FERRAMENTAS DE ANÁLISE

Neste capítulo serão apresentadas algumas ferramentas que instituíram abordagens de avaliação para o simulador *Soccer Server 2D*, sendo que todas as ferramentas relacionadas fazem uso dos arquivos de *log* fornecidos pelo seu servidor.

5.1 O Programa Team Assistant

O *Team Assistant* [1] (TA) é um programa multifuncional com funções gerais de projeto, desenvolvimento e análise de agentes para o *Soccer Server 2D*. O TA foi concebido na *Shaheed Beheshti University*. Lá foi utilizado como ferramenta auxiliar dos desenvolvedores incumbidos de projetar equipes de futebol de robôs (ZAREIAN et al., 2003). A Figura 24 ilustra a interface do TA com algumas das suas opções de análise acionadas.

A multifuncionalidade do TA advém das suas funções principais, que são (ZAREIAN et al., 2003):

- **Monitor/LogPlayer** – Com o TA é possível abrir arquivos de *log* rcg ou iniciar uma conexão direta com o servidor *soccerserver*, fornecendo opções de vídeo para os dois casos. Durante a simulação, há o acesso às propriedades dos agentes e a funções básicas de análise como rastrear o movimento dos jogadores e da bola, mostrar a área de visão dos jogadores e exibir os tipos e a quantidade de comandos enviados por estes.

- ***Treinador*** – Uma conexão direta com o servidor pode ser feita na função de treinador, o que permite o envio dos comandos próprios do treinador aos jogadores. O usuário também pode modificar o estado de um jogador (ex: posição, ângulo da cabeça e visão) e usar um programa embutido, o *plan-editor*, para definir modos de jogo e configurações táticas.

- **Depurador** – O arquivo de *log* do tipo rcl também é processado. Desta forma, é possível utilizar o TA como um depurador dos comandos enviados pelos jogadores.

[1] Disponível no endereço: http://sourceforge.net/projects/team-assistant/files/

Figura 24: Interface do *Team Assistant*. O programa possui opções de vídeo por meio dos botões da barra localizada na parte inferior da imagem. Os acontecimentos da partida são mostrados na janela com o campo à esquerda da imagem. Na parte superior, há caixas numeradas que permitem o fácil acesso às informações de cada jogador. A opção "*Draw Events*", entre as caixas numeradas, permite a disposição gráfica dos eventos no campo. A direita da imagem, é apresentado um gráfico com as estatísticas de passe para cada jogador através da janela de análise. Esta, que fica disponível a partir da barra de *menu* superior (opção "*Analyzer*") também possui outras opções de análise.

- **Detecção de eventos** – Além das regras de detecção de eventos presentes no programa, o usuário pode criar outras que desejar, através de máquinas de estado.

- **Relatórios** – Há o suporte de relatórios para os eventos criados pelo usuário, como o registro da frequência que estes ocorrem sucessivamente.

As últimas três funções estão mais ligadas à análise e avaliações de desempenho. Na função de depurador existe a representação gráfica dos estados e ações dos jogadores em uma simulação. Isto inclui as posições e a velocidade dos jogadores e da bola, e os comportamentos dos jogadores como passes, chutes, dribles ou prender a bola. É possível comparar os comandos enviados pelos jogadores e os que foram realmente aceitos pelo servidor (ZAREIAN et al., 2003).

Na função de detecção, os eventos originais do TA, bem como os criados pelo usuário, podem ser representados graficamente no campo (no momento em que ocorrem) ou exibidos em mensagens de texto na parte superior da interface do programa. Já a função de relatórios provê gráficos sobre a ocorrência de eventos de ambos os times em períodos de tempo determinados pelo usuário. Também provê a sequência ordenada dos eventos detectados de um time durante os intervalos de posse da bola (ZAREIAN et al., 2003).

5.2 O Programa Logalyzer

O *Logalyzer* [2] é um programa criado especificamente para a análise de partidas de futebol oriundas do simulador *Soccer Server 2D* por meio dos arquivos de *log* de extensão rcg. O programa é da autoria de Andraz Bezek e teve sua última versão (v 0.9 - desenvolvida para o sistema operacional *Windows XP*) disponibilizada pelo mesmo, em 2006, enquanto membro do *Department of Intelligent Systems* do *Institute Jožef Stefan* (IJS).

O *Logalyzer* possui desde funções básicas de vídeo e de análise tradicionais (como a verificação da trajetória dos jogadores e da bola) até funções de manipulação e criação de dados, e de reconhecimento dos papéis e ações dos agentes (BEZEK, 2005). A interface do *Logalyzer* é ilustrada na Figura 25.

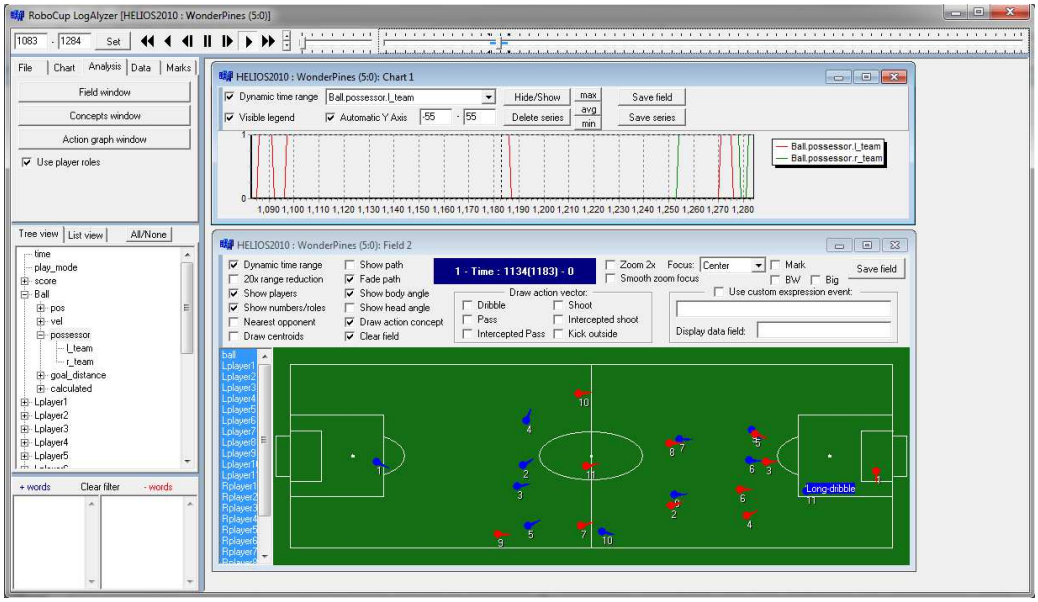


Figura 25: Interface do *Logalyzer*. O programa possui opções de vídeo por meio dos botões da barra localizada na parte superior da imagem. Além disso, são separadas diversas opções de análise em abas diferentes. Estas ficam localizadas no canto esquerdo, logo abaixo da barra de opções de vídeo. No quadro referente a cada aba de análise, encontram-se as opções referentes àquele tipo de análise. No quadro logo abaixo, encontram-se as abas *"Tree view"* e *"List view"*, que permitem selecionar, isoladamente, os dados de cada elemento do jogo. As janelas que apresentam os dados, a exemplo das duas que se encontram mais a direita ocupando a maior parte da imagem (janela com o campo virtual e janela com um gráfico da posse de bola), são incluídas através de cada aba de análise. Cada janela de apresentação de dados possui uma gama de opções. Atenta-se que os dados são apresentados conforme o instante de tempo da partida.

O *Logalyzer* distribui várias funções de análise em interfaces diferentes. No entanto, embora propicie análises individuais e, indiretamente, sociais dos agentes, o *Logalyzer* é de difícil uso por não possuir documentação. Além disso, é uma ferramenta de código fechado que só se encontra disponibilizada através de um arquivo executável binário. Desta forma, a sua modificação e reengenharia são praticamente impossíveis.

Sem que pese as dificuldades, algumas das virtudes do *Logalyzer* são a capacidade de

[2] Disponível no endereço: http://dis.ijs.si/andraz/logalyzer/

abrir mais de um *log* ao mesmo tempo e, com isso, mesclar dados de diferentes *logs*, bem como isolar os dados de um mesmo agente, independente do número de *logs* abertos.

Os dados gerados pelas análises podem ser manipulados e exportados para diferentes tipos de arquivos como c5, arff, txt, e raw. Da mesma forma, os dados presentes nestes arquivos podem advir de inúmeras tabelas que têm seus campos definidos pelo usuário (BEZEK, 2005). Nos arquivos de extensão c5 e arff, próprios para o uso em *Data Minning*, há suporte para a definição de classes. Também podem ser exportados gráficos do tipo gml, net, chaco, txt, e bmp (BEZEK, 2005).

5.3 Os Programas Soccer Scientia Tool e Soccer Server Statistical Extracting Tool

Os programas *Soccer Scientia Tool* (SST) e *Soccer Server Statistical Extracting Tool* (SSSET) fornecem estatísticas gerais (de times) e individuais (de jogadores) a partir de determinados eventos que ocorrem em uma partida. Ambos foram desenvolvidos com base em um terceiro programa, o *SoccerScope2* [3] . Este, por sua vez, é um software de análise e código aberto (escrito em java), desenvolvido por pesquisadores da *Electro-Communications Japan University* (CUNHA ABREU, 2011).

Segundo CUNHA ABREU (2011), houve condições para o uso do *SoccerScope2* para a criação do SST e do SSSET, estando a parte legada principalmente ligada à carga do arquivo de *log* do tipo rcg (abre somente as versões dois e três deste *log*). O SST e o SSSET resultaram da aplicação de algoritmos que visam a obtenção de estatísticas finais de jogo. Antes que se considere diretamente a extração dessas estatísticas, porém, é necessária a compreensão de como são modeladas a estrutura do campo e de uma partida para ambos os programas.

Como já foi dito anteriormente neste trabalho, uma simulação no *Soccer Server 2D* ocorre em 6000 ciclos de tempo. Baseado nessa condição, todas as informações que constituem uma simulação são organizadas, pelos programas SST e SSSET, linearmente, em um vetor de dados. Dessa forma, existe a representação do estado de todos os elementos do jogo para cada ciclo de simulação (posição de jogador e bola, energia e visão do jogador, comandos executados, entre outros) (CUNHA ABREU, 2011). A Figura 26 ilustra esse conceito de organização de uma simulação.

A divisão do campo em regiões é necessária para a extração de estatísticas. Segundo CUNHA ABREU (2011), essa divisão foi baseada na ontologia *SocOn ontology* que - tal qual a um sistema especialista - foi submetida a profissionais do mundo do futebol, determinando, ao final deste processo, as regiões a serem consideradas para o reconheci-

[3] O *SoccerScope2* não se encontra mais disponível na sua versão original (http://ne.cs.uec.ac.jp/*koji/SoccerScope2/index.htm), porém sua versão modificada se encontra disponível em: https://github.com/joaoportela/RoboCup-strategy-adviser-coach/tree/master/statistics_calculator.

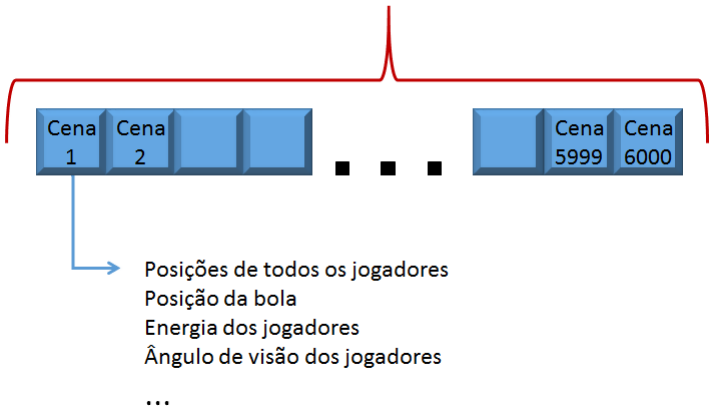


Figura 26: Representação de uma simulação no *Soccer Server 2D*. Extraído e adaptado de (CUNHA ABREU, 2011). Cada cena representa um ciclo da simulação e contém todos os estados dos elementos do jogo.

mento de eventos em uma partida (ex: inversão de bola, cruzamento, oportunidade de gol, chute a gol) (CUNHA ABREU, 2011). A figura 27 ilustra as regiões do campo que foram definidas.

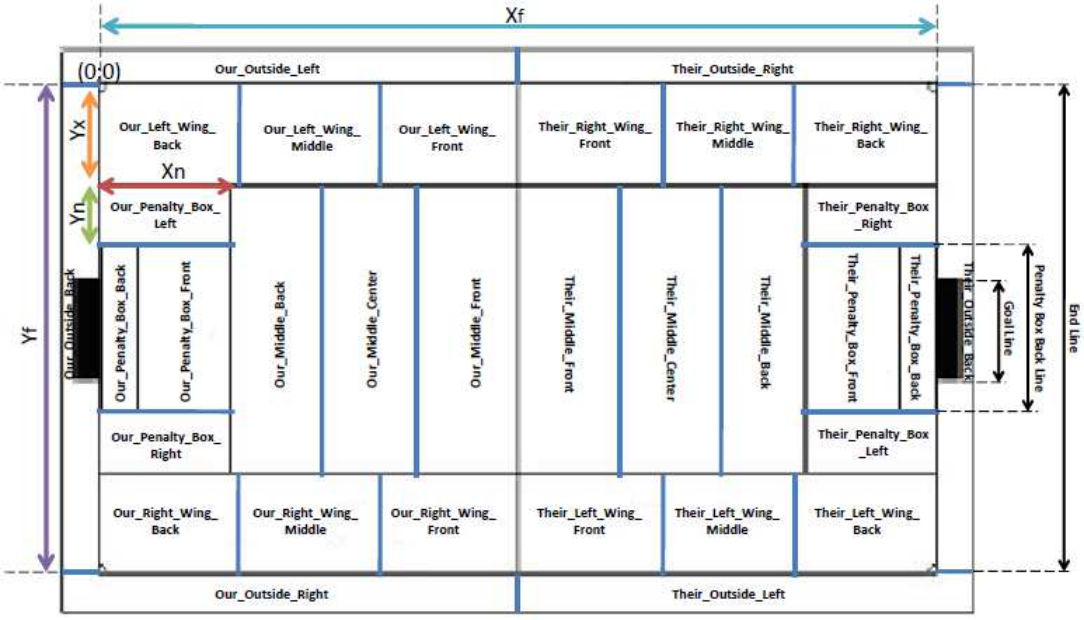


Figura 27: Regiões do campo. Extraído de CUNHA ABREU (2011). Cada lado do campo possui 16 regiões definidas em coordenadas relativas. Encontram-se definidas as regiões externas ao campo, necessárias para determinar a saída da bola. Também são definidas as variáveis (X_n, Xf , Y_x, Y_n, Y_f), que possibilitam um fácil redimensionamento do campo (presentes no código de ambas as ferramentas) (CUNHA ABREU, 2011).

Em relação à extração de estatísticas, CUNHA ABREU (2011) salienta que a maioria dos eventos de um jogo iniciam com um chute (a exemplo de passes, cruzamentos, dribles, chutes a gol). Um chute produz um aumento na velocidade da bola ou a mudança de sua direção. Esse conceito é provido pela Equação 1, dada por CUNHA ABREU (2011), na qual t_1 e t_0 são instantes de tempo e, V_{ball} e D_{ball} são respectivamente a velocidade e direção da bola.

$$Chute(t_0) \leftarrow (\|V_{\vec{ball}}(t_0)\| < \|V_{\vec{ball}}(t_1)\| \vee D_{ball}(t_0) \neq D_{ball}(t_1)) \wedge t_1 = t_0 + 1 \qquad (1)$$

Considerando-se que a simulação provê os estados dos elementos do campo, e que instantes específicos de tempo podem ser isolados, a extração de estatísticas é fundamentada na especificação de regras (que detectam eventos). Cada regra, por sua vez, é constituída de três condições (CUNHA ABREU, 2011):

- **Inicial** – Determina o momento em que um evento começa;
- **Restritiva** – Restringe o número de elementos do jogo que satisfazem uma regra no momento da busca;
- **Final** – Determina o momento final de um evento.

Detectar eventos, como pode ser observado no Algoritmo 1, significa efetuar laços de repetição no vetor de cenas (ciclos) de uma partida. Assim, em um primeiro momento todos os chutes são detectados e marcados como possíveis eventos no jogo. Logo após, os chutes são analisados diante das condições de restrição da regra em aplicação, limitando a busca. Por fim, o instante final do evento é determinado (CUNHA ABREU, 2011).

ALGORITMO 1: Algoritmo genérico de detecção de eventos. Extraído de CUNHA ABREU (2011).

```
for Cycle i = 0 to max_Cycles − 1 do
  if (kick.start_condition(scene[i], scene[i + 1])) then
    addKick(i)
  end if
end for
for all Event Class event do
  for all Cycle i in Kicks do
    for j = i + 1 to nextKick do
      if (! event.constrain(scene[j])) then
        break
      end if
    end for
    if (event.final_condition(scene[j])) then
      addEvent(event, i, j)
    end if
  end for
end for
```

Todas as regras existentes são aplicadas ao vetor de estados de um jogo, por meio de uma classe nomeada "*Statistics*" - presente nos programas SST e SSSET - considerando as condições para cada evento, que podem variar (CUNHA ABREU, 2011). Na próxima subseção serão descritos os eventos que podem ser detectados.

5.3.1 A Detecção de Eventos

Em conformidade com a definição de regras para a extração de estatísticas, alguns dos eventos que podem ser detectados são: *pass*, *wing chain*, *pass chain*, *ball possession*, *temporal sequences*, *shot*, *goal scoring*, *outside information* e *offside information*.

As informações de passes (certos e errados) são coletadas para ambos os times através do evento *pass*. Um passe certo, provém da identificação de dois chutes consecutivos entre jogadores de um mesmo time. A Equação 2, dada por CUNHA ABREU (2011), fornece este conceito, onde P_0 e P_1 são respectivamente o passador e o recebedor da bola, P é um jogador genérico; t_0, t_1 e t_2 são instantes de tempo e b representa a bola.

$$\begin{aligned} &SuccessfulPass(P_0, P_1, t_0, t_1) \leftarrow KicksBall(P_0, t_0) \wedge KicksBall(P_1, t_2) \wedge SameTeam(P_0, P_1) \wedge \\ &t_1 > t_0 \wedge t_2 > t_1 \wedge P_0 \neq P_1 \wedge (\neg\exists(P, t) : t > t_0 \wedge t < t_2 \wedge KicksBall(P, t)) \wedge \\ &(\neg\exists(P, t) : t > t_1 \wedge t < t_2 \wedge P \neq P_1 \wedge dist(b, P) < dist(b, P_1)) \end{aligned} \tag{2}$$

Assume-se que o jogador que tem o controle da bola pode executar chutes (não é considerado se a bola apenas rebater no jogador). Dessa forma, se entre dois chutes detectados, o segundo for (CUNHA ABREU, 2011):

- Do mesmo jogador, trata-se de um drible;
- De outro jogador do mesmo time, trata-se de um passe;
- De um jogador adversário, trata-se de um passe errado.

Wing chain é o evento que especifica um tipo especial de passe. Este se caracteriza como uma inversão de bola no campo e é determinado por meio da divisão do campo em três corredores: esquerdo, meio e direito. Um *wing chain* é registrado se houver um passe entre jogadores posicionados nos corredores esquerdo e direito, porém, esse também é registrado se dois passes consecutivos ocorrerem dos corredores esquerdo para o meio e meio para o direito (ou vice e versa) sequencialmente (CUNHA ABREU, 2011).

Pass chain é o evento em que se identifica passes certos consecutivos entre jogadores de um mesmo time (CUNHA ABREU, 2011).

Ball possession é o total de tempo em que a bola permanece com um time sem sair do campo ou sem ser interceptada por jogadores do time adversário. O evento *Ball possession* é computado por meio dos intervalos de tempo entre chutes (dribles, passes certos, inversão de bola) (CUNHA ABREU, 2011).

Já *temporal sequences* é o evento que calcula o tempo gasto por uma equipe até chegar na área adversária sem que a bola seja perdida. Esse tempo indica qualitativamente a ofensividade de uma equipe, considerando-se quatro níveis: devagar, médio, rápido e interrompido (CUNHA ABREU, 2011).

Shot é o evento no qual um jogador chuta a bola no seu campo de ataque em direção à linha de gol adversária com uma velocidade inicial suficiente para chegar até a mesma. A Equação 3, dada por CUNHA ABREU (2011), mostra essa concepção, onde $\vec{A}_{ball}$ representa a aceleração da bola, $Vel(b, t_0)$, $Pos(b, t_0)$ representam a posição e velocidade da bola no instante t_0, e P representa o jogador.

$$Shot(P, t_0) \leftarrow Belongs(P, team) \wedge KicksBall(P, t_0) \wedge InReg(Pos(P, t_0), AtckField(team)) \wedge \\ \left(\exists (t) : Pos(b, t_0)_X + Vel(b, t_0)_X t + \frac{\vec{A}_{ball_X} t^2}{2} > X_f \wedge Pos(b, t)_Y > 0 \wedge Pos(b, t)_Y < Y_f\right) \quad (3)$$

No caso de um chute a gol, são considerados três eventos (CUNHA ABREU, 2011):

- Chute no alvo, quando a bola é chutada na direção do gol com tolerância externa do gol de 0.5 metros em cada trave;
- Chute que não possui a direção certeira do gol, mas que deixa o campo pela linha de fundo nas dimensões da área de pênalti;
- Chute interceptado, ou seja, que possui as condições dos eventos anteriores, mas é defendido pelo goleiro ou interceptado por outro jogador.

Goal scoring ocorre quando a bola passa totalmente pela linha do gol deixando o campo. Ocorrendo o gol, nenhum outro evento é considerado. No entanto, também são avaliadas as oportunidades de gol quando há grande probabilidade. Para isso é constituído um triângulo virtual com o jogador de ataque e as traves como vértices. Se há menos do que dois jogadores oponentes na área do triângulo, é registrada a oportunidade de gol (CUNHA ABREU, 2011). A Figura 28 ilustra o conceito de triângulo virtual.

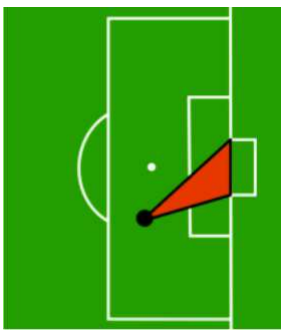


Figura 28: Triângulo virtual. Extraído de CUNHA ABREU (2011). As traves do gol e o jogador de ataque (indicado pelo ponto preto) formam o triângulo utilizado para computar as oportunidades de gol.

Em *outside information* são registrados os eventos de escanteios, cobranças de lateral, e de tiro livre indireto junto à posse da cobrança. Já em *offside information* é registrada a condição de impedimento, considerando os jogadores em posição irregular e o mais indicado como possível recebedor da bola (CUNHA ABREU, 2011).

Os algoritmos que especificam cada uma das regras e os eventos que as mesmas detectam estão presentes no anexo A deste trabalho. Nas próximas subseções serão demonstradas as especificidades dos programas SST e SSSET.

5.3.2 Soccer Scientia Tool

O *Soccer Scientia Tool* (SST) é uma ferramenta que calcula estatísticas finais de um jogo usando o sistema de coordenadas do *Soccer Server 2D* (CUNHA ABREU, 2011). É um programa de visualização por meio de interface gráfica e textual que recebe como entrada um arquivo de *log* e permite ao usuário a escolha das estatísticas a serem geradas, entre as quais encontram-se as opções: *pass* (passes certos e errados), *shot* (chutes interceptados, na direção do gol ou para fora), *goal*, *offside information* (impedimentos) e *outside information* (laterais, escanteios e tiros livres indiretos).

A interface do *Soccer Scientia Tool* é apresentada em três módulos principais: *animation module*, *chart module* e *table module* (CUNHA ABREU, 2011).

O *animation module* possui três janelas que provém ao usuário respectivamente: a visualização dos acontecimentos da partida e opções de vídeo, os eventos da partida com a opção de seleção para um evento específico e o histórico de passes bem sucedidos dos agentes na partida. O *chart module* provê a visualização gráfica das estatísticas referentes aos agentes como gols, passes bem sucedidos, chutes a gol, entre outros. Já o *table module* provê estatísticas coletivas de cada time em uma tabela de dados gerais (CUNHA ABREU, 2011).

5.3.3 Soccer Server Statistical Extracting Tool

Ao contrário do Soccer Scientia Tool (SST), o *Soccer Server Statistical Extracting Tool* (SSSET) não é um programa de interface gráfica. Sua estrutura principal é fornecida por um arquivo em código *python*. O código fonte, nomeado "*Statistics*", recebe como entrada arquivos xml, cujos conteúdos são os dados de eventos detectados nas partidas. Cada arquivo xml é produzido como saída da execução da versão alterada do *SoccerScope2* (com a implementação dos algoritmos de detecção). Para produzir o arquivo xml com os dados de uma partida, o programa aceita como entrada um arquivo de *log* do tipo rcg (somente arquivos na terceira versão).

Dessa forma, a extração de estatísticas finais é realizada em duas etapas, na primeira ocorre à detecção dos eventos e na segunda o processamento destes. No entanto, enquanto o programa em java pode processar os dados de uma partida por vez, o programa python "*Statistics*" pode processar dados de vários arquivos xml.

Indiferente a sua divisão em etapas, o SSSET possui, além das funções presentes no SST, outras seis: *wing chain*, *pass chain*, *goal opportunity*, *ball possession* e *temporal sequence*. Por meio dessas funções, estatísticas de mais alto nível podem ser processadas, sendo estas (CUNHA ABREU, 2011):

- ***Zone Dominace*** – São calculadas as médias de posse de bola em cada região do campo para ambos os times. Como resultado, tem-se as partes do campo em que as equipes tem mais perda ou mais controle da bola. É possível, então, fazer a distinção entre equipes ofensivas ou defensivas.

- ***Passes*** – É calculada uma taxa entre os valores de acertos e erros de passes. O acerto ou não de passes tem influência direta sobre as outras estatísticas.

- ***Wing Variation*** – É calculado o total de inversões de campo realizadas pelos agentes de ambas as equipes. Este tipo de jogada pode ser considerada uma característica do nível de coletividade de um time.

- ***Temporal Sequence*** – Sequências de ataques das equipes são classificadas segundo sua continuidade temporal, ou seja, quando ocorrem sequências de passes de um time em direção ao ataque. Se há a perda da bola, a sequência é marcada como interrompida, porém, se a equipe chega na zona à frente da área adversária sem perder a bola, a sequência leva uma anotação em seu nível de periculosidade ao gol adversário, conforme as condições de restrição do algoritmo que faz a avaliação. Nesse caso as classificações são devagar, média e rápida.

- ***Goal Opportunities*** – São consideradas e computadas as chances de gol para cada equipe. Chances de gol são um bom indicativo do nível de competitividade de uma equipe.

O SSSET, apesar de não ter interface gráfica, é mais completo do que o SST. A disposição dos resultados, após o processamento, fica por conta de dois arquivos csv, um com os dados dos times e outro com os dados de cada partida. Os arquivos csv, também podem ser utilizados para o processamento em mineração de dados.

5.4 Considerações

Um agente atua dentro do *Soccer Server 2D* por meio da execução dos comandos que são aceitos pelo servidor. O *Soccer Server 2D* mantém as informações sobre os jogadores e os times através dos registros desses comandos nos arquivos de *log*. Isso torna os arquivos de *log* um dos principais recursos em abordagens de avaliação, como já observado nos programas elencados. Estes, porém, exploraram muito mais a avaliação estatística tanto de agentes quanto de times.

O isolamento ou a quantificação dos comandos acionados pelos agentes em si, não fornece clareza nem para a avaliação individual do agente. Embora, possa ser de proveito, quando um desenvolvedor deseja comparar se o comando do agente foi aceito pelo servidor ou se este era o comando esperado. Por isso, muitos dos programas mencionados neste

capítulo implementam abordagens que se voltam à identificação dos comportamentos dos agentes, sendo que um comportamento pode envolver a execução de vários comandos.

Além disso, a avaliação estatística já tem sido explorada na descoberta de características, como área do campo que o jogador ou a equipe se movimenta, o tempo de permanência com a bola individual e coletivamente, o percurso percorrido pelo jogador e bola, número geral de passes, chutes, gols, entre outros. No entanto, características do nível social dos jogadores enquanto equipe são mais difíceis de serem identificadas e, portanto, de serem avaliadas. Incluem-se nisso, dados de movimentação no campo que caracterizam padrões de técnicas e táticas de time, como o estabelecimento de formação para a defesa com dois ou mais jogadores ou o estabelecimento de formação para o ataque, também com dois ou mais jogadores.

Nesse sentido, foi proposta a obtenção de informações relativas a estratégias de equipes. Mais precisamente, em relação ao aspecto organizacional de um sistema multiagente enquanto time de futebol, e na identificação de comportamentos de coalizão demonstrados pelos jogadores. Comportamentos de coalizão ocorrem quando dois ou mais agentes cooperam momentaneamente para alcançar um objetivo em comum. Desta forma, estes podem emergir explicitamente quando da execução de jogadas ensaiadas e formações de ataque ou defesa.

As possibilidades de formação de coalizões no *Soccer Server 2D* se situam no âmbito da troca de mensagens entre os agentes, mas também da identificação visual dos companheiros de equipe pelos agentes que a compõem (que possibilitam jogadas, alinhamentos, e outras formações). Nos próximos capítulos será descrita e demonstrada a abordagem para a identificação e avaliação de coalizões.

6 MODELO DE RECONHECIMENTO E AVALIAÇÃO DE COALIZÕES

Neste capítulo serão feitas algumas considerações sobre aspectos coletivos e sociais de times, e sobre a execução de jogadas conjuntas entre os jogadores. Serão também apresentadas as técnicas idealizadas para viabilizar o reconhecimento e a avaliação da formação de coalizões entre os agentes dos times implementados para o simulador *Soccer Server 2D*.

A abordagem desenvolvida conta com o reuso do código base do programa *SoccerScope2*, onde pequenas alterações foram feitas, e o reuso do código em python "*Statistics*", referido por CUNHA ABREU (2011) como parte do programa *Soccer Server Statistical Extracting Tool* (SSSET). Neste último, foram introduzidos novos algoritmos para identificar e avaliar o que se considera serem coalizões dentro dos times do futebol de robôs simulado.

6.1 Questões Sociais e Coletivas de Times e Coalizões

Resguardados os fundamentos individuais dos jogadores (drible, chute, interceptação, condução de bola, passe, entre outros), as questões coletivas se traduzem, primeiramente, pela formação organizacional "time" propriamente dito. Ademais, fundamentos sociais se traduzem em interações entre os jogadores por meio de passes (lançamentos, passe com infiltração, cruzamento, inversões, cobranças), distribuição tática do time, formações de ataque (deslocamento conjunto, contra ataque) e formações de defesa (cobertura, cerco, barreira, cobrir espaço).

Esses fundamentos são citados como sociais, por terem a premissa do ensaio, do treinamento e da disposição do que comumente se nomeia no mundo futebolístico de "espírito de time" (característica da organização e coletividade dos jogadores) (DROGOUL; COLLINOT, 1998).

No *Soccer Server 2D*, os padrões de jogadas executados por mais de um jogador são difíceis de ser verificados, se considerado o uso da comunicação implícita e do reconhecimento de ações passivas pelos agentes (ex: identificação visual de situações de jogo).

Interações visuais não podem ser determinadas, embora produzam padrões de ações que podem ser percebidas no campo. Por outro lado, a comunicação direta e explícita pode ser determinada, no entanto, esta não irá necessariamente produzir efeitos visíveis no campo, uma vez que, as informações transmitidas entre os jogadores podem não indicar qual o comportamento adotado por estes. Há de se destacar também que, as restrições de comunicação limitam os mecanismos de coordenação por este meio, devendo restringir-se à absoluta necessidade (FERREIRA; REIS; LAU, 2004).

Considerando-se, no entanto, eventos que ocorrem no jogo (passes entre os jogadores), podem ser identificadas formações organizacionais secundárias independentes da formação principal (time). A formação de grupos de jogadores, em um time, para a execução de jogadas, como considerado por DROGOUL; COLLINOT (1998), é neste trabalho tratada como a composição de coalizões entre os jogadores.

Assim, como descrito por DROGOUL; COLLINOT (1998), em que comportamentos básicos, dependentes entre si, conduzem ao comportamento coletivo de time, partir-se-á desses comportamentos e da construção de jogadas coletivas para a identificação da formação de coalizões.

6.2 Caracterização da Abordagem de Identificação e Avaliação de Coalizões

Infere-se que um sistema multiagente (SMA) enquanto time de futebol concentra seus esforços na obtenção de um placar favorável em uma partida. Para isto, os agentes devem ser organizados e coordenados sob uma ou mais estratégias coletivas, jogadas e metas, considerando os papéis e as funções de cada agente no time, e de modo a potencializar suas ações individuais.

A maior parte da dinâmica de um jogo de futebol está ligada à troca de passes. A partir desta pode ser indicado o nível de coletividade de um time. Desta forma, afirma-se que a troca de passes - geralmente ligada às questões táticas - pode denotar a estrutura organizacional de um time. Mais especificamente, procuram-se os aspectos resultantes da formação de coalizões dentro dos times.

A formação de coalizões, entre os agentes, pode ser denotada por meio da troca de passes que ocorre durante as jogadas definidas como tabelas e triangulações. Assim, a abordagem utilizada neste trabalho é fundamentada nos programas *SoccerScope2* e *Statistics*, pois estes fazem parte do processo de identificação e avaliação dessas jogadas.

Para que se identifiquem tabelas e triangulações é necessário, a princípio, o registro de todos os passes que ocorreram em uma partida, condição já fornecida pelo programa *SoccerScope2*. Este fornece como saída um arquivo xml contendo os dados de uma partida, definidos como eventos do jogo.

Foram feitas algumas alterações no *SoccerScope2*, principalmente para que fosse pos-

sível as inclusões da posição e da região em que os jogadores se encontram durante os registros de passes, passes errados e inversões de campo. Foram feitas alterações que permitem a adição dos jogadores envolvidos nas cobranças de jogo (escanteios, laterais e faltas), quando do registro das mesmas, já considerando as posições e regiões em que os jogadores se encontram no momento das cobranças. Também foi criado o código que registra, no arquivo xml, as faltas cometidas pelos jogadores, sendo que, estas são necessárias na etapa de validação das jogadas identificadas.

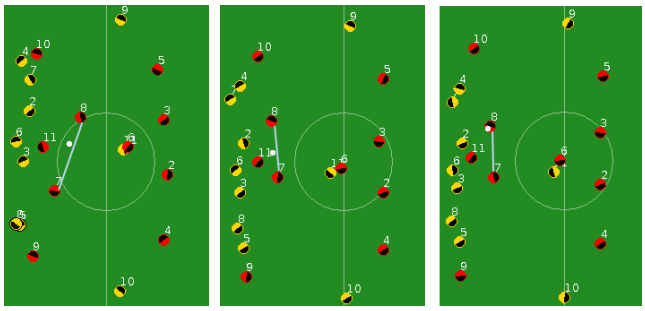
Conta-se também com o reuso do programa *Statistics*, no qual foram introduzidos novos algoritmos para identificar e avaliar as coalizões dentro dos times. Para isso, as sequências dos registros de passes dos arquivos xml são percorridas, buscando a identificação das tabelas e triangulações. A Figura 29 ilustra os componentes que integram a abordagem de avaliação.

Figura 29: Processo de identificação e avaliação de coalizões.

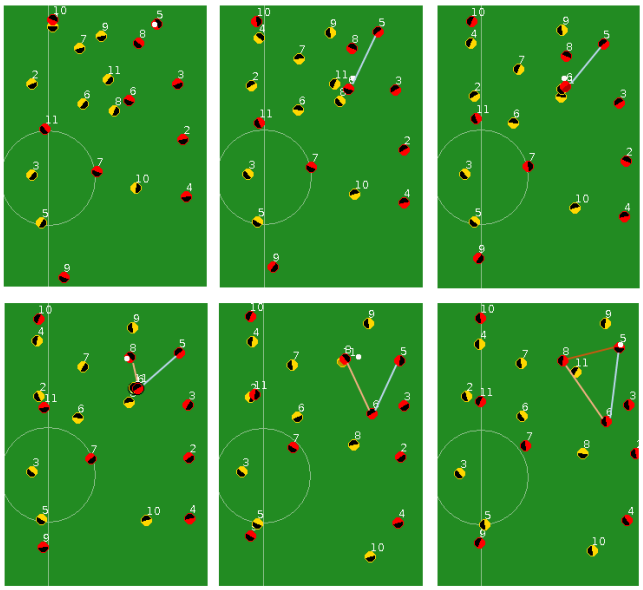
O fluxograma ilustrado na Figura 29 tem seu início no processamento dos arquivos de *log* rcg (na terceira versão) das partidas pelo programa *SoccerScope2*. Então, os eventos de cada partida são registrados em seus respectivos arquivos xml. Estes, por sua vez, são processados pelo programa *Statistics*, que faz a identificação e avaliação das tabelas e triangulações registrando-as em arquivos csv. Através do programa *Statistics*, também é possível a apresentação gráfica dos dados de interesse.

Para a compreensão do processo de identificação de tabelas e triangulações é necessário o entendimento sobre as condições que definem esses tipos de jogadas. Uma tabela se caracteriza por dois passes consecutivos entre dois jogadores, na qual o jogador que iniciou

a jogada - que geralmente faz um deslocamento e reposicionamento - recebe a bola em um novo passe. Já a triangulação é descrita por uma sequência semelhante, no entanto, esta envolve três jogadores e três passes. A Figura 30 exemplifica esses dois tipos de jogadas.



(a) Tabela



(b) Triangulação

Figura 30: Caracterização de uma tabela e de uma triangulação. Na Figura 30(a), a troca de passes entre os jogadores 7 e 8 resulta na tabela que está sendo ilustrada da esquerda para à direita. Na Figura 30(b), a troca de passes entre os jogadores 5, 6 e 8 resulta na triangulação que está sendo ilustrada da esquerda para à direita, e de cima para baixo. Em ambas as figuras, linhas entres os jogadores indicam o passe ou a troca de passes.

Como os registros dos passes no arquivo xml são feitos de maneira sequencial, conforme o tempo da partida, há registros para ambos os times, porém, cada registro traz informações suficientes para determinar os jogadores envolvidos e o time ao qual estes pertencem. Além disso, todos os eventos anotados no arquivo xml contêm o tempo exato em que ocorreram. Dessa forma, é possível determinar quando uma troca de passes foi interrompida e quais eventos de interesse ocorreram durante um intervalo de tempo determinado (gols, chutes, cobranças, faltas, cruzamentos). Essa característica é crucial para a validação das jogadas e para a etapa de avaliação das coalizões.

Assim, para ambos os tipos de jogadas (tabelas e triangulações) é necessário:

- Determinar os intervalos de tempo das sequências de passes de cada time.
- Determinar nesses intervalos as jogadas que ocorreram e os jogadores envolvidos.
- Verificar os requisitos: uma tabela envolve dois jogadores e dois passes consecutivos, e uma triangulação envolve três jogadores e três passes consecutivos.
- Verificar se durante o intervalo de uma suposta jogada ocorreram determinados eventos que podem invalidá-la, sendo estes: passe errado, perda e recuperação da bola por outro jogador do mesmo time, faltas e cobranças de jogo após o primeiro passe de cada jogada.
- Determinar os eventos que ocorreram após uma jogada, como: gols, gols perdidos, oportunidades de gol, chutes a gol e para fora, cruzamentos, passes errados, faltas e cobranças de jogo.

O Algoritmo 2 foi implementado para detectar, respectivamente, as tabelas e as triangulações. Os passos necessários ao registro das tabelas e triangulações são indicados por esse algoritmo.

Dada a definição da identificação das tabelas e triangulações, volta-se o interesse para características dessas jogadas nas partidas. Alguns pontos de interesse são a frequência em que as jogadas ocorrem, quais jogadores participaram, e outras características de avaliação que serão exemplificadas na próxima seção.

6.3 Métricas e Parâmetros de Avaliação das Coalizões

A avaliação das tabelas e triangulações deve levar em consideração os acontecimentos que resultaram direta ou indiretamente das mesmas, tanto pela ação dos jogadores que participaram efetivamente na execução da jogada quanto pela continuidade de ações do time (executadas por jogador da coalizão ou não) até a interrupção na posse de bola. Antes disso, podem-se levantar algumas das razões para que um time de futebol execute tabelas e triangulações, dentre as quais:

- Manter a posse de bola – a troca de passes em si já tem essa função. Tabelas e triangulações condicionam mais a posse de bola ao dificultarem o desarme do adversário.
- Abrir ou procurar espaços – a troca de passes por poucos jogadores em uma região de menor dimensão pode atrair a marcação adversária, abrindo espaços em outras regiões do campo. Também é possível a distribuição da bola para uma área livre de marcação ou de menor intensidade.

ALGORITMO 2: Algoritmo de detecção de jogadas. Durante o registro das tabelas e triangulações são verificados alguns eventos que ajudarão na etapa de avaliação.

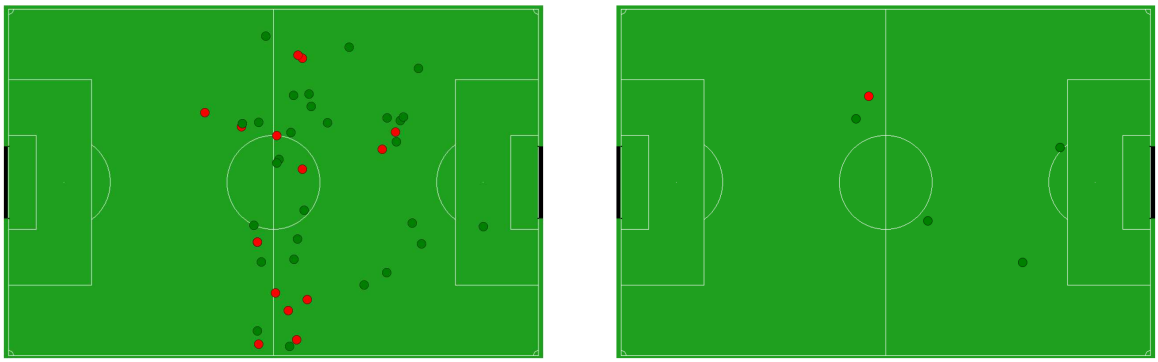
```
dominio ← analizar("arquivo.xml")
passes = [.]
t_eventos = {}
t_eventos['escanteios'] ← dominio.obterdados("corners").obterAtributo("time")
t_eventos['laterais'] ← obterdados("kickins").obterAtributo("time")
t_eventos['faltas'] ← obterdados("foulcharges").obterAtributo("time")
t_eventos['tiroslivre'] ← obterdados("freekicks").obterAtributo("time")
t_eventos['faltaslivre'] ← obterdados("freekickfaults").obterAtributo("time")
t_eventos['passeserrados'] ← obterdados("passmiss").obterAtributo("time")
dados ← dominio.obterdados("pass")
for passe i in dados do
   if passe.obterAtributo("team") = time_atual then
      passador(i) = passe.obterAtributo("kick")
      recebedor(i) = passe.obterAtributo("reception")
      passes(i) = addPass(passador(i), recebedor(i))
   else
      t_eventos['chuteoponente'] = passes.obterAtributo("kick").obterAtributo("time")
   end if
end for
for i = 0 to passador − 2 do
   if (recebedor(i + 1).num == passador(i).num) and recebedor(i).num == passador(i + 1).num
   and validaCadeiaDePasse(recebedor(i).time, passador(i + 1).time, t_eventos) then
      lista_tabelas = addTabela(time_atual, passes(i), passes(i + 1))
      calculaMovimentonoCampo(time_atual, passador(i), recebedor(i + 1)))
      padroesdegol(time_atual, dominio, passes(i + 1), passes)
      padroesdepasse(time_atual, passes(i + 1), passes, t_eventos)
   end if
   if (recebedor(i + 2).num == passador(i).num) and recebedor(i + 1).num == passador(i + 2).num
   and recebedor(i).num == passador(i + 1).num and recebedor(i).num! = passador(i + 2).num and
   validaCadeiaDePasse(recebedor(i).time, passador(i + 2).time, t_eventos)) then
      lista_triangulacoes = addTriangulacao(time_atual, passes(i), passes(i + 1), passes(i + 2))
      calculaMovimentonoCampo(time_atual, passador(i), recebedor(i + 2))
      padroesdegol(time_atual, dominio, passes(i + 2), passes)
      padroesdepasse(time_atual, passes(i + 2), passes, t_eventos)
   end if
end for
```

- Livrar-se da marcação – procura-se fazer chegar a bola a um jogador com mais liberdade. Para isso os jogadores que participam de uma jogada fazem movimentação e reposicionamento.

- Fornecer condições ao ataque – procura-se fazer chegar a bola a um jogador em condições de chutar a gol, cruzar a bola, fazer um passe de infiltração, entre outras possibilidades de ataque mais incisivas. Para isso, os jogadores que participam de uma jogada fazem movimentação e reposicionamento.

Esses cenários são difíceis de serem identificados, quando da conclusão de uma jogada. A abordagem deste trabalho não cumpre essa demanda para as três primeiras razões citadas. Porém, são consideradas outras características que tornam uma jogada passível de avaliação, entre as quais se encontra a movimentação no campo.

Sabe-se que o jogador que inicia uma jogada é também o último a receber o passe (em tabelas e triangulações). Considerando a posição do jogador no momento inicial e ao final da jogada, verifica-se, se esta proporcionou evolução na direção do campo de ataque. A Figura 31 ilustra as tabelas e triangulações do time *WriteEagle* conforme a posição do último jogador e indica se este avançou ou recuou no campo.

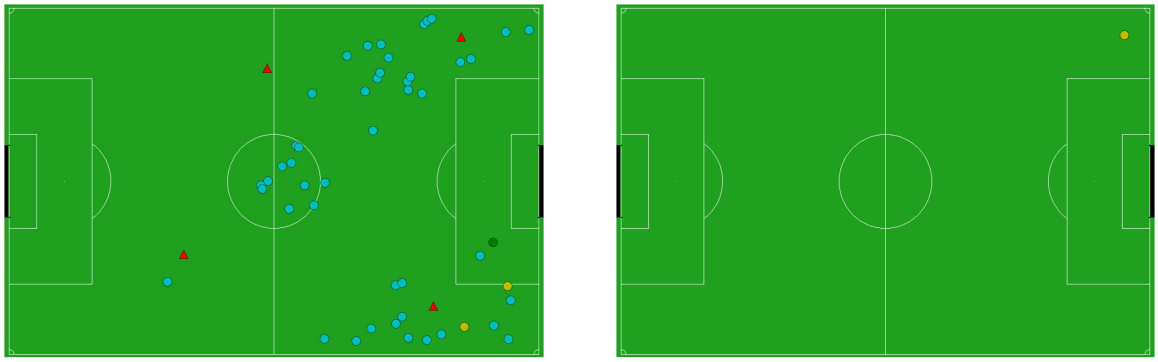


(a) Tabelas (b) Triangulações

Figura 31: Tabelas e triangulações do time *WriteEagle* em uma partida contra o time *Gliders*. Houve 41 tabelas e 5 triangulações. Os círculos indicam a posição do jogador que finaliza a tabela ou triangulação. Os círculos vermelhos indicam um recuo deste jogador em relação à posição em que se encontrava ao iniciar a jogada, já os círculos verdes indicam o avanço em direção ao ataque.

Também são considerados eventos típicos de jogo na avaliação da jogada, entre os quais: passes errados, chutes a gol, gols, entre outros. Há a diferenciação para o caso do evento ter resultado de ação direta do último jogador ou ocorrer em um momento posterior na mesma sequência de posse de bola, indicando participação indireta. A Figura 32 indica as tabelas e triangulações do time *YuShan2014* conforme a posição do último jogador e os eventos diretamente ligados às jogadas.

Considerando-se as peculiaridades listadas, adotou-se um sistema de pontos na abordagem de avaliação das jogadas. A quantização dos pontos em relação aos quesitos de avaliação das coalizões é descrita na forma:



(a) Tabelas (b) Triangulações

Figura 32: Tabelas e triangulações do time *YuShan2014* em uma partida contra o time *AUT-Parsian*. Houve 53 tabelas e 1 triangulação. Os círculos indicam a posição do jogador que finaliza a tabela ou triangulação. Os triângulos vermelhos indicam que o jogador errou o passe, os círculos amarelos indicam oportunidades de gol, o circulo verde indica que houve um gol enquanto os círculos em azul indicam que não houve nenhum evento diretamente ligado àquelas jogadas.

- Avanço em direção ao ataque – é concedido 0,5 (meio) ponto.
- Recuo em direção ao campo defensivo – é concedido 0 (zero) ponto.
- Evento resultante de ação direta à jogada, ou seja, uma ação desempenhada pelo último jogador de uma tabela ou triangulação:
  - Passe errado: é subtraído 1 (um) ponto;
  - Falta cometida: é subtraído 0,5 (meio) ponto;
  - Falta recebida: é concedido 0,7 (zero vírgula sete) pontos;
  - Oportunidade de gol: é concedido 1 (um) ponto;
  - Chute a gol: é concedido 1,5 (um vírgula cinco) pontos;
  - Gol marcado: é concedido 7 (sete) pontos.
- Evento resultante de ação indireta à jogada, ou seja, ocorrido após o jogador que finaliza uma tabela ou triangulação realizar um novo passe. O evento deve resultar da ação de qualquer jogador do mesmo time, durante o mesmo intervalo de posse de bola:
  - Passe errado: é subtraído 0,2 (zero vírgula dois) pontos;
  - Falta cometida ou impedimento: é subtraído 0,1 (zero vírgula um) ponto;
  - Falta recebida: é concedido 0,2 (zero vírgula um) pontos;
  - Cobrança de escanteio: é concedido 0,2 (zero vírgula dois) pontos;
  - Oportunidade de gol: é concedido 0,3 (zero vírgula três) pontos;
  - Chute a gol: é concedido 0,6 (zero vírgula seis) pontos;

  - Gol marcado: é concedido 2 (dois) pontos.

- Evento resultante de ação direta ou indireta à jogada:

  - Cruzamento: é concedido 0,2 (zero vírgula dois) pontos.

Há de se considerar que, durante um mesmo intervalo de posse de bola, as mesmas coalizões podem ocorrer mais de uma vez. As coalizões que voltaram a ocorrer podem estar ligadas à eventos positivos ou negativos. Neste caso, os eventos relacionados indiretamente às coalizões contabilizarão metade dos pontos ou serão desconsiderados. Assim, para que não haja supervalorização das jogadas, especificou-se a aplicação de pontos na forma:

- Passe errado: é subtraído 0,15 (zero vírgula quinze) pontos;
- Oportunidade de gol: é concedido 0,15 (zero vírgula quinze) pontos;
- Chute a gol: é concedido 0,3 (zero vírgula três) pontos;
- Gol marcado: é concedido 1 (um) ponto.
- Os demais eventos serão desconsiderados

Assim ficou estabelecida a avaliação das coalizões. Comparações individuais entre as coalizões são válidas, no entanto, o foco principal são comparações que envolvem a totalidade destas. Neste caso, podem ser feitas implicações - caso existirem - sobre as regiões nas quais as coalizões se concentram mais ou, em cujas ocorrências de recuos no campo e de passes errados são mais acentuadas. Nas comparações entre as equipes, o número de coalizões pode influenciar no resultado. Neste caso, é o que se procura, uma vez que se pretende avaliar a coletividade dos times. Atenta-se que será observado o número de repetições das coalizões com os mesmos jogadores em cada time, pois reforça o fato de se tratarem de coalizões.

No próximo capítulo serão descritos os testes utilizados para o emprego da abordagem de identificação e avaliação das coalizões.

7 VALIDAÇÃO DO MODELO

Neste capítulo serão descritos os testes realizados para a aplicação da abordagem de avaliação, nos quais foram utilizados os times que participaram da competição *Simulation League*, na edição da *RoboCup* de 2014.

Posteriormente, serão demonstrados e discutidos os resultados junto de análises empíricas fundamentadas na observação de algumas das partidas utilizadas nos testes, e nas descrições das tecnologias utilizadas nos times por parte das equipes de pesquisa que os implementaram.

7.1 Descrição dos Testes

Para a aplicação da abordagem especificada, foi idealizado um teste em que se considera uma quantidade x de partidas, realizadas entre uma quantidade y de times. Neste teste serão indicados aspectos relativos as coalizões como níveis de incidência, vantagem garantida ao time, características do comportamento coletivo apresentado, e os demais aspectos já mencionados.

Tendo em mente o teste com os times de futebol de robôs destinados ao ambiente simulado *Soccer Server 2D*, primeiramente, foram procuradas bases de dados e repositórios oficiais tanto da *RoboCup* quanto dos grupos de pesquisa que participaram das edições da *Simulation League*. Constatou-se que o repositório oficial para os arquivos das competições já realizadas (*logs* das partidas é arquivos binários dos times) se encontrava inativo.

Para os momentos de estudo e trabalho com o simulador, foi adquirido material disperso entre os grupos de pesquisa. No entanto, para fins de experimentação da abordagem de avaliação, optou-se pelo uso dos arquivos binários das equipes que participaram da *Simulation League*, na *RoboCup*[1] do ano de 2014, que ocorreu no Brasil. Entre os motivos que levaram a escolha dos times desse ano da competição para a realização do teste estão:

- Disponibilidade dos arquivos – Foi possível encontrar os arquivos no site[2] do evento

[1] A *RoboCup* 2014 aconteceu em João Pessoa na Paraíba. Seu site oficial é: http://www.robocup2014.org/

18th RoboCup International Symposium, que é realizado concomitantemente a competição.

- Base de comparação justa – Já houve pesquisa identificando que as diferenças das edições da competição devem ser levadas em consideração (ver seção 4.1). Optou-se por times pertencentes à uma mesma edição. Ainda que não tenha havido mudanças no simulador no período que se iniciou em 2011 até a atualidade.

- Facilidade de instalação – As equipes mais atuais foram desenvolvidas levando-se em consideração a versão mais atual do simulador e, ainda que, na evolução gradual desse fosse considerada a possibilidade de executar equipes de quaisquer edições, existem questões externas (versões do sistema operacional utilizado e de bibliotecas). Em suma, existe um aprimoramento na utilização de equipes mais atuais.

Foram, então, selecionados quatorze dos quinze times[2] que estão listados na Tabela 4 junto a sua posição de classificação na competição.

Tabela 4: Times da competição *Simulation League* de 2014 .

Time	Posição na Competição
WrightEagle	1
Gliders2014	2
Oxsy	3
HELIOS2014	4
CYRUS2014	5
YuShan2014	6
Infographics	7
UFSJ2D	8
tokA1[1]	9
FCP_GPR_2014	10
Ri-one2014	11
HfutEngine	12
AUT-Parsian	13
HERMES	14
Enigma	15

[1]Não foi possível resolver as dependências específicas de uma biblioteca para que o time *tokA1* executasse normalmente, então optou-se por deixá-lo de fora dos testes.

A aplicação da abordagem de avaliação dar-se-á em duas etapas. Na primeira etapa serão selecionados alguns dos times para realizar enfrentamentos em embates únicos.

[2]Os arquivos binários dos times se encontram disponibilizados em: http://fei.edu.br/rcs/2D/2014/

Os times serão escolhidos conforme suas posições de classificação na competição. Deste modo, pretende-se confrontar os times com melhores posições entre si, contra os times do meio da tabela de classificação e contra os times nas últimas posições. Também serão escolhidos times com classificação mediana para se enfrentarem entre si, e igualmente aqueles com as piores classificações para se enfrentarem entre si.

Na segunda etapa foram criadas baterias de teste, onde cada bateria é composta de treze partidas. Nas treze partidas um time se mantém sempre o mesmo, ou seja, em cada bateria o foco principal é um único time. São quatorze baterias em que cada time confronta os demais. Em cada partida serão identificadas as coalizões de ambos os times e serão registrados os eventos que ocorreram, mas somente o time imutável na bateria terá as suas coalizões avaliadas conforme a abordagem descrita na seção anterior. Como exemplo, a Tabela 5 apresenta a identificação das tabelas e triangulações que ocorreram em uma partida entre os times *YuShan2014* e *AUT-Parsian*. Estes dados são a saída do programa *Statistics*, presentes no arquivo csv.

Tabela 5: Dados das tabelas e triangulações na partida *YuShan2014 AUT-Parsian*.

Time	Total	Avanço no Campo	Recuo no campo	Passe errado	Gol	Chance de gol
YuShan2014						
Tabelas (jogadores)						
5 e 3	2	0	2	1	0	0
6 e 8	5	5	0	0	0	0
7 e 6	8	8	0	0	0	0
7 e 9	15	12	3	2	0	0
8 e 10	11	11	0	1	0	1
8 e 11	1	1	0	0	0	0
9 e 11	4	4	0	0	0	0
10 e 6	2	0	2	1	0	1
10 e 7	1	1	0	0	0	0
10 e 11	2	2	0	0	1	0
11 e 6	2	1	1	0	0	0
Triangulações (jogadores)						
9, 7 e 6	1	0	1	0	0	1
AUT-Parsian						
Tabelas (jogadores)						
7 e 9	1	0	1	0	0	0
10 e 6	1	1	0	1	0	0

Em uma partida já surgem dados interessantes, porém, ao se expandir o teste para uma bateria de disputas se pretende reforçar que as tabelas e triangulações são ocorrências de coalizões nos times, caso seja nítida a repetição destas nas equipes.

Ao longo de cada partida, ou bateria, pretende-se verificar a vantagem que as coalizões fornecem aos seus times, seja em manter a posse de bola, tornar o time mais ofensivo ou propiciar mais chances de ataque. Além disso, pretende-se verificar as nuances das

coalizões em cada time diante das mudanças de adversários.

7.2 Resultados

Os resultados serão apresentados, em sua maioria, através de gráficos dos dados e dos números gerais que ocorreram. Também serão feitas análises para cada caso. Na primeira etapa, cada caso se resume a uma única disputa entre dois times.

Dividiram-se as disputas individuais em três primeiras categorias:

- Disputas entre os times do topo da tabela de classificação;
- Disputas entre os times do meio da tabela de classificação e;
- Disputas entre os times na parte de baixo da tabela de classificação.

Nas próximas subseções serão listados os resultados dessas categorias.

7.2.1 Times do Topo da Tabela de Classificação

O teste iniciado na primeira etapa contou com disputas diretas entre os times que conquistaram as primeiras colocações. Iniciar-se-á pela descrição dos resultados obtidos na partida entre os times *WrightEagle* e *Gliders*.

Nessa primeira disputa, o time *WrightEagle* concretizou trinta e nove tabelas e sete triangulações enquanto o time *Gliders* concretizou vinte e três tabelas e cinco triangulações. Para que fosse possível ter uma ideia, mesmo que indireta, da ofensividade dos times, foram consideradas as duas metades do campo e destacado o que comumente se considera os campos ofensivo e defensivo de cada time - conforme o lado de campo em que estes atuam. A Figura 33 indica as proporções das tabelas e triangulações nos campos ofensivo e defensivo resultantes da primeira partida analisada.

Constata-se, a partir das proporções apresentadas na Figura 33, que o time *WrightEagle* é bem ofensivo, uma vez que, mantém quase a totalidade das tabelas e triangulações após a linha de meio campo. Por sua vez, o time *Gliders* teve uma distribuição maior dessas jogadas no campo defensivo, mas as proporções também demonstram relativa ofensividade deste time.

Quando considerado o percentual de posse de bola garantido pelas tabelas e triangulações em relação aos totais de posse de bola dos times, é constatado que essas jogadas têm relativa influência. Destacando-se que foi considerado apenas o tempo despedido entre os passes durante as jogadas, e o tempo decorrido até a ocorrência de eventuais chutes a gol e gols por meio do jogador que finaliza uma tabela ou triangulação (logo após a conclusão). Poderia também ter sido considerado o tempo decorrido em passes certos, atribuídos aos jogadores que finalizam as jogadas. Assim, os percentuais de posse de bola garantidos pelas tabelas e triangulações aumentariam consideravelmente. A Figura 34

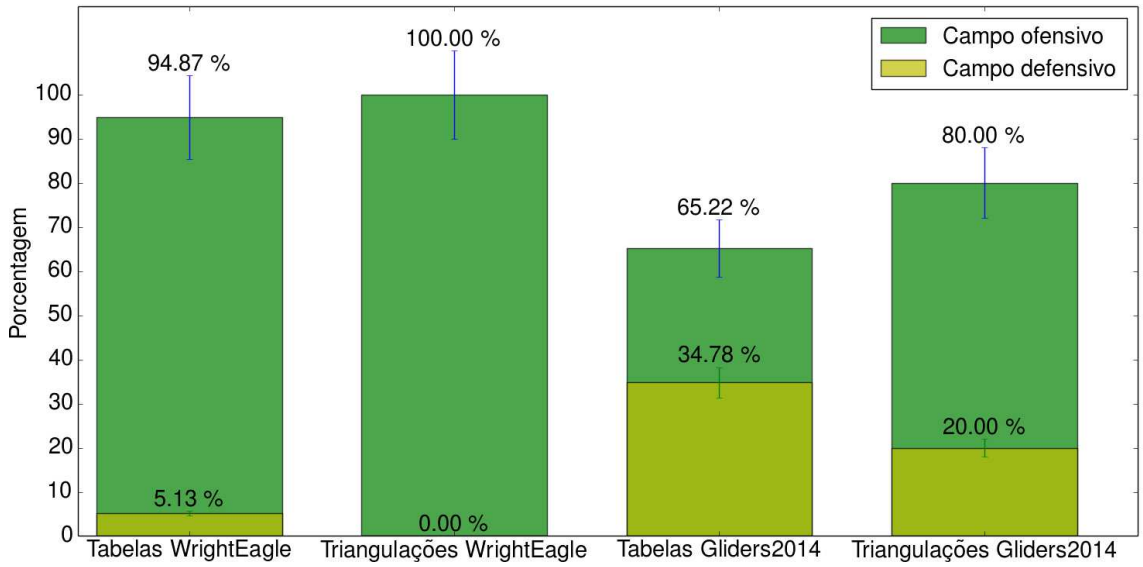


Figura 33: Distribuição das tabelas e triangulações no campo - partida *WrightEagle* x *Gliders*.

demonstra a posse de bola fornecida pelas tabelas e triangulações aos times *WrightEagle* e *Gliders*.

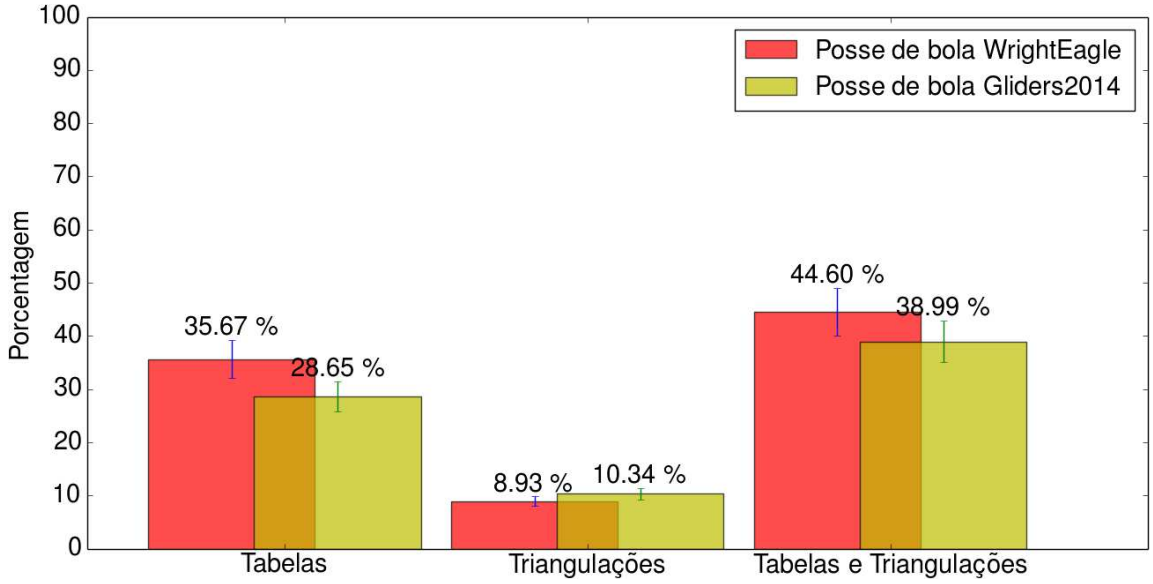


Figura 34: Percentual da posse de bola dos times *WrightEagle* e *Gliders* através das tabelas e triangulações.

No sistema de avaliação das tabelas e triangulações - por meio de pontos - o time *WrightEagle* obteve 36 pontos enquanto o time *Gliders* obteve 24,75 pontos. Lembrando que, o sistema leva em consideração uma série de possíveis eventos ligados direta ou indiretamente às jogadas. Neste caso, o time *WrightEagle* teve maior quantidade de tabelas e triangulações relacionadas a eventos na partida. Todavia, mesmo que o time *WrightEagle* tenha apresentado um número maior de tabelas, muitas destas pontuaram menos por estarem relacionadas a eventos negativos (passes errados). Na Figura 35, é possível a visualização da distribuição das tabelas do time *WrightEagle* no campo e das

demais informações obtidas.

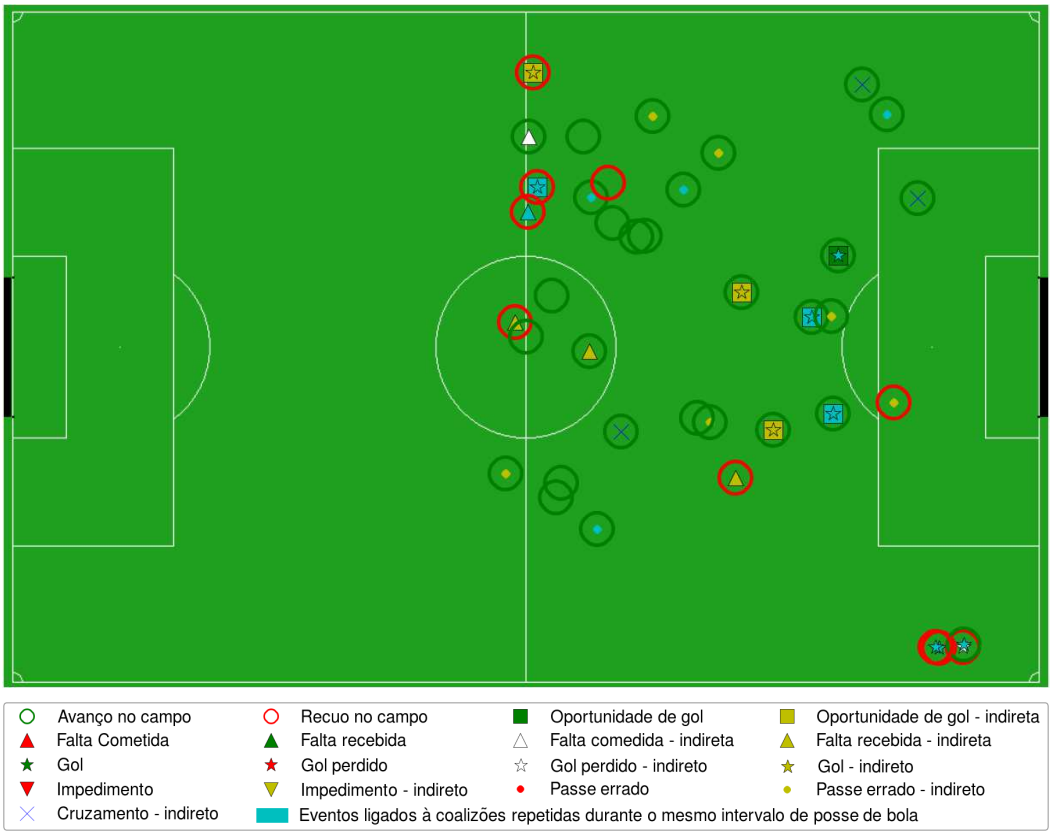


Figura 35: Tabelas do time *WrightEagle* ocorridas contra o adversário *Gliders*. Na mesma figura, segue a legenda com os possíveis eventos que influenciam na avaliação das tabelas.

Através da Figura 35 podem ser feitas algumas inferências a respeito do time *WrightEagle*. Primeiramente, pode ser observado que a maioria das oportunidades de gols e gols indiretos, relacionados às tabelas, aconteceram na região à frente da área do gol adversário. Todavia, através da forma gráfica utilizada não é possível distinguir se as tabelas estiveram ligadas a diferentes oportunidades de gol e gols, pois, durante um mesmo intervalo de posse de bola, várias tabelas ou triangulações podem ter participação nos eventos que ocorreram. No entanto, se uma tabela ou triangulação se repete durante o mesmo intervalo de posse de bola, os eventos ligados indiretamente ficam representados na cor ciano - conforme indicado na legenda. Essa distinção é considerada durante a atribuição de pontos como exemplificado na descrição da avaliação. Neste caso, a coalizão que se repete receberá pontos novamente, só que estes são reduzidos à metade da pontuação original ou são desconsiderados.

Ao checar o arquivo csv com os dados das tabelas e triangulações foi possível constatar que grande parte dos gols e oportunidades de gol foram atribuídos às tabelas entre os jogadores 7 e 11. Durante suas cinco ocorrências, foram registradas duas oportunidades de gol indiretas, uma oportunidade de gol direta e dois gols indiretos. Os jogadores 7 e 11, de fato, são os mais adiantados do time *WrightEagle* e provavelmente têm a função

de atacantes.

Ainda na Figura 35 é possível destacar a concentração de tabelas na parte superior ou lado esquerdo do campo (próximas à linha de meio campo). Nessa região houve passes errados e recuos, provavelmente devido à marcação acirrada. Também houve duas tabelas ligadas indiretamente a uma oportunidade de gol e gol. Isto se deve ao fato de os jogadores 3 e 10 desempenharem suas duas únicas tabelas na partida, de forma consecutiva, e possibilitarem um passe de infiltração para o jogador 7, o qual fez o gol.

Das sete triangulações registradas para o time *WrightEagle*, cinco foram atribuídas aos jogadores 4, 6, e 7. Outro dado pertinente foram as tabelas entre os jogadores 3 e 11, que registraram dez ocorrências. O time *WrightEagle* ainda demostrou uma distribuição de ocorrências de tabelas e triangulações mais uniforme do que seu adversário. O time *Gliders* concentrou grande parte de suas tabelas entre os jogadores 9 e 10 e, 10 e 11. Estes efetuaram respectivamente seis e oito ocorrências de tabelas (14 das 23 ocorridas). As demais jogadas efetuadas, não passaram de uma ou duas ocorrências. A Figura 36 ilustra as tabelas do time *Gliders*.

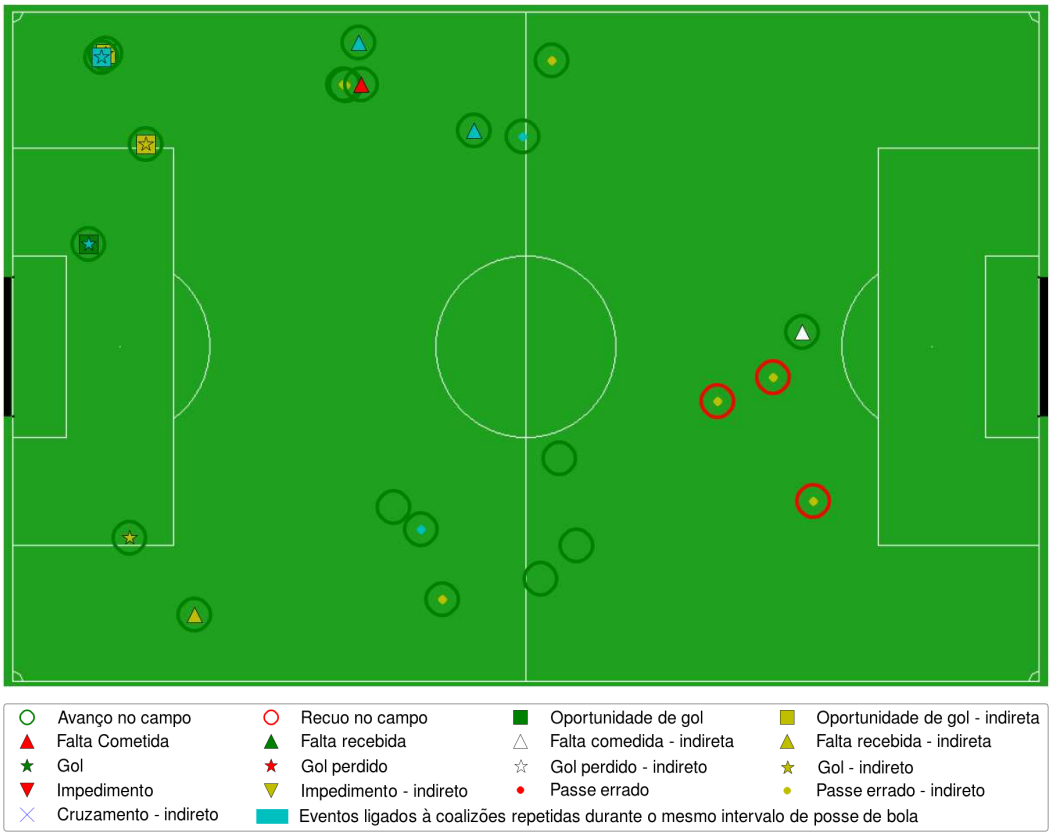


Figura 36: Tabelas do time *Gliders* ocorridas contra o adversário *WrightEagle*.

A partir da Figura 36, é possível observar que as tabelas do time *Gliders* ocorreram pelos lados do campo, provavelmente indicando jogadas com a participação dos laterais, embora não tenham sido registrados cruzamentos. Ao assistir à reprise da partida, foi possível constatar que os jogadores 10 e 9 são pontas avançados (laterais posicionados

mais à frente) que, geralmente, avançam em linha com o jogador 11, que é o atacante central. Os três gols do time *Gliders* surgiram de tabelas com, ou seguidas de, infiltrações entre os jogadores 9 e 10 e, 10 e 11. Essa partida terminou com placar de 6 a 3 a favor do time *WrightEagle*.

A segunda partida entre os times do topo da tabela de classificação contou com a participação dos times *Oxsy* e *Helios*. O time *Oxsy* obteve trinta e três tabelas e quatro triangulações enquanto o time *Helios* obteve treze tabelas e uma triangulação. No sistema de pontuação, o time *Oxsy* obteve 15,05 pontos e o time *Helios* obteve 3,85 pontos. A posse de bola mediante essas jogadas está ilustrada na Figura 37.

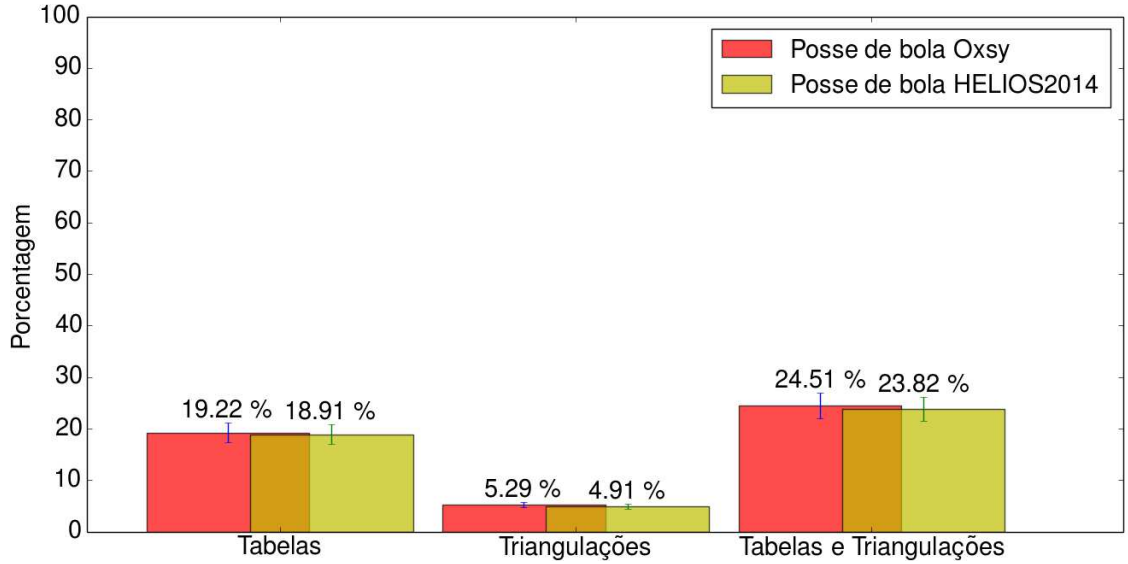


Figura 37: Percentual da posse de bola dos times *Oxsy* e *Helios* através das tabelas e triangulações.

O time *Oxsy* tinha, praticamente, o dobro de posse de bola em relação ao time *Helios*, dessa forma, as tabelas e triangulações, mesmo em maior número, não tiveram percentual de participação expressiva no total de posse de bola. A superioridade numérica também não se refletiu na pontuação sobre o time *Helios*. Isso se deve as doze tabelas efetuadas entre os jogadores 8 e 11, que geraram uma pontuação negativa devido a duas faltas cometidas, dois passes errados e oito recuos no campo (que não pontuam). Algumas posições em que esses jogadores se encontravam quando finalizaram as tabelas eram todas muito próximas, resultando em sobreposição quando graficamente representadas. Foi constatado que, em determinada situação, os jogadores 8 e 11 trocaram vários passes estando bem próximos após uma cobrança de lateral, e por fim perderam a bola. A Tabela 6 demonstra parte dos resultados das tabelas e triangulações dos times *Oxsy* e *Helios*.

As triangulações entre os jogadores 6, 7 e 10; 6, 10 e 11; e a tabela entre os jogadores 10 e 11 tiveram participação indireta em ao menos um gol a favor do time *Oxsy*. Ao rever a partida foi constatado que se trata de duas triangulações e uma tabela (consecutivas) que resultou em um dos dois gols a favor desse time. O placar da partida foi de 2 a 1.

Tabela 6: Dados das tabelas e triangulações na partida *Oxsy* x *Helios*.

Time	Total	Avanço no Campo	Recuo no campo	Passe errado (indireto)	Gol (indireto)	Chance de gol(indireta)
Oxsy						
Tabelas (jogadores)						
11 e 8	12	4	8	2	0	0
10 e 11	2	2	0	1	1	0
9 e 10	4	3	1	1	0	0
8 e 10	1	1	0	1	0	0
7 e 11	1	1	0	0	0	1
7 e 10	3	3	0	2	0	0
7 e 9	2	1	1	0	0	1
6 e 9	2	1	1	0	0	0
6 e 8	1	1	0	0	0	0
6 e 7	2	2	0	0	0	0
5 e 8	3	0	3	2	0	0
Triangulações (jogadores)						
7, 10 e 11	1	1	0	0	0	1
6, 10 e 11	1	1	0	0	1	1
6, 7 e 10	1	1	0	0	1	1
6, 7 e 9	1	1	0	0	0	0
Helios						
Tabelas (jogadores)						
9 e 11	8	8	0	3	0	0
8 e 11	1	1	0	0	0	0
7 e 9	2	2	0	1	0	0
6 e 11	2	1	0	1	0	0
4 e 9	1	1	0	1	0	0
Triangulações (jogadores)						
3, 5 e 8	1	0	1	0	0	0

O time *Helios* teve uma concentração de tabelas entre os jogadores 9 e 11, que produziu poucos pontos, por não participar de eventos positivos, e conduzir a três passes errados. Praticamente todas as tabelas do time *Helios* foram no campo defensivo, em uma região próxima à linha de meio campo. Ao assistir à reprise do jogo, foi constatado que o time *Helios* priorizava toques rápidos em diagonal para os jogadores mais adiantados. Geralmente, a bola parava com o jogador 9 (lateral) que conduzia a bola, mas tinha à frente vários jogadores e então recuava. Este fazia poucas tabelas com o jogador 11, sem avançar ao campo adversário. O único gol do time *Helios* foi devido a um erro de passe do time adversário.

7.2.2 Times do meio da Tabela de Classificação

Foram realizadas duas partidas nessa categoria. Na primeira partida, foram confrontados os times *Infographics* e *UFSJ2D*. O time *Infographics* obteve trinta tabelas e três triangulações chegando a 18,2 pontos. Já o time *UFSJ2D* obteve dezessete tabelas e uma triangulação, chegando a 9,6 pontos. O percentual de posse de bola obtido pelo time *Infographics* através das tabelas e triangulações foi de 47,74 por cento. Já o time *UFSJ2D* obteve 29,32 por cento. A distribuição das jogadas e os eventos ligados às mesmas são uns dos principais dados a serem considerados nessa partida. A Figura 38 ilustra a distribuição das tabelas do time *Infographics*.

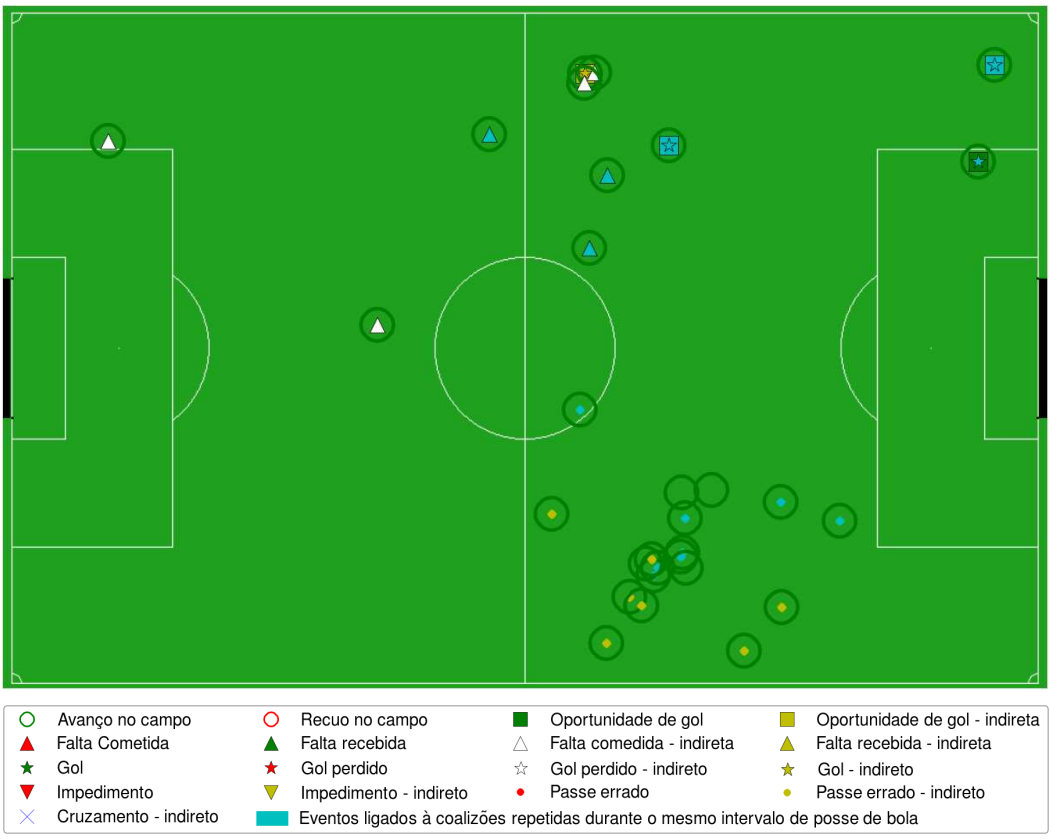


Figura 38: Tabelas do time *Infographics* ocorridas contra o adversário *UFSJ2D*.

Conforme pode ser observado na Figura 38, o time *Infographics* teve passes errados e faltas cometidas relacionadas indiretamente às tabelas executadas. A execução de faltas era um artifício recorrente do time *Infographics*. A coalizão de maior ocorrência desse time - entre os jogadores 10 e 11 - resultou em dezenove tabelas, entre as quais, nove estiveram relacionadas a nove passes errados. A segunda coalizão de maior ocorrência, entre os jogadores 9 e 11, resultou em dez tabelas, entre as quais, cinco estiveram relacionadas à cinco faltas. Das outras quatro jogadas - três triangulações e uma tabela diferentes entre si - três estiveram relacionadas à faltas.

Embora todas as tabelas e triangulações do time *Infographics* tenham produzido o avanço em direção ao campo adversário, grande parte não produziu outros eventos positivos. Na maioria das ocasiões de jogo o time *Infographics* conduzia a bola até a região com concentração de tabelas ilustrado na Figura 38. Os jogadores 10 e 11 executavam tabelas nessa região e geralmente perdiam a bola em passes errados. A exceção foram quatro tabelas consecutivas entre os jogadores 9 e 11, que findaram em um passe para o jogador 10, o qual efetuou o único gol da partida.

Por sua vez, o time *UFSJ2D* teve muitas das tabelas executadas relacionadas à faltas recebidas. O critério de aplicação da pontuação considera que receber uma falta deva conceder mais pontos, uma vez que, geralmente, faltas sejam utilizadas para parar ataques com chances de êxito. Porém, o time *UFSJ2D* teve pontuação inferior devido à possuir quase somente a metade das jogadas em relação ao adversário, e devido ao lance de gol. A Figura 39 ilustra as tabelas do time *UFSJ2D*.

A coalizão do time *UFSJ2D* com maior número de ocorrências, sete no total, estabeleceu-se entre os jogadores 7 e 9. As demais não passaram de duas ocorrências. Os principais eventos relacionados às jogadas foram às faltas recebidas.

Na segunda partida foram confrontados os times *YuShan* e *FCP_GPR*. O time *YuShan* apresentou um número elevado de tabelas considerando as partidas anteriores, foram quarenta e três tabelas no total, mas em contrapartida apresentou apenas uma triangulação. O time *FCP_GPR* apresentou números muito inferiores. Foram somente quatro tabelas e duas triangulações. A avaliação por pontos concedeu 17,85 pontos ao time *YuShan* e 2,4 ao time *FCP_GPR*. Os percentuais de posse de bola garantidos pelas tabelas e triangulações encontram-se ilustrados na Figura 40.

O Time *YuShan* apresentou um percentual de posse de bola semelhante aos times que mantém um número elevado de tabelas. Notavelmente, manter a posse de bola é uma característica dessas jogadas. Considerando-se os tempos totais de posse de bola na partida, o time *YuShan* tem a vantagem de quase vinte por cento a mais do que o time *FCP_GPR*.

O time *YuShan* também apresentou a distribuição mais uniforme nas ocorrências de tabelas e triangulações em relação aos outros times estudados. No entanto, estas jogadas estiveram frequentemente envolvidas em recuos e passes errados. A segunda coalizão de

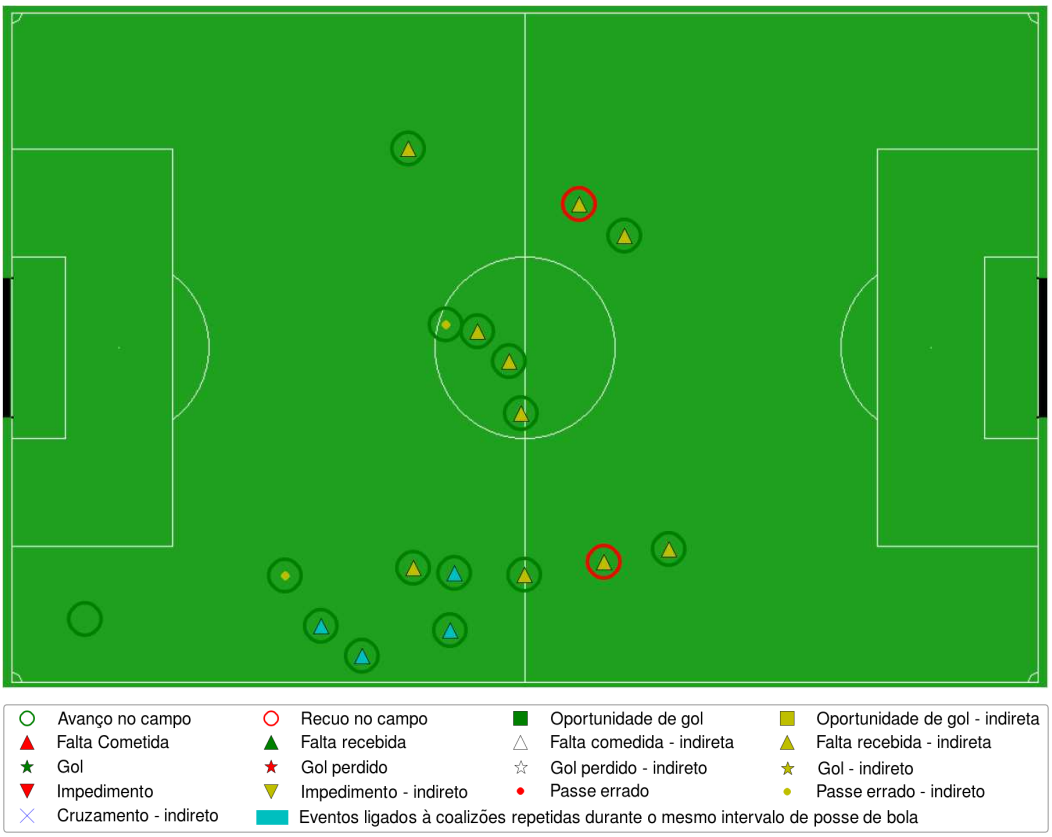


Figura 39: Tabelas do time *UFSJ2D* ocorridas contra o adversário *Infographics*.

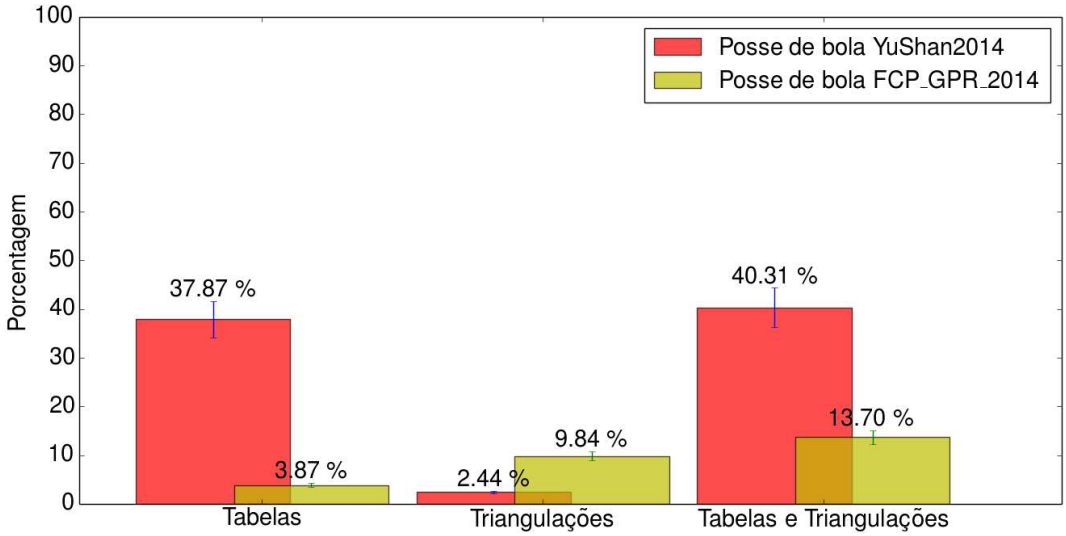


Figura 40: Percentual da posse de bola dos times *YuShan* e *FCP_GPR* através das tabelas e triangulações.

maior número, oito no total, esteve relacionada à quatro cruzamentos. Contando com outras jogadas, também envolvidas em cruzamentos, o time *YuShan* foi o time que mais fez uso desse fundamento do futebol até então. A Figura 41 ilustra a distribuição de tabelas do time *YuShan*.

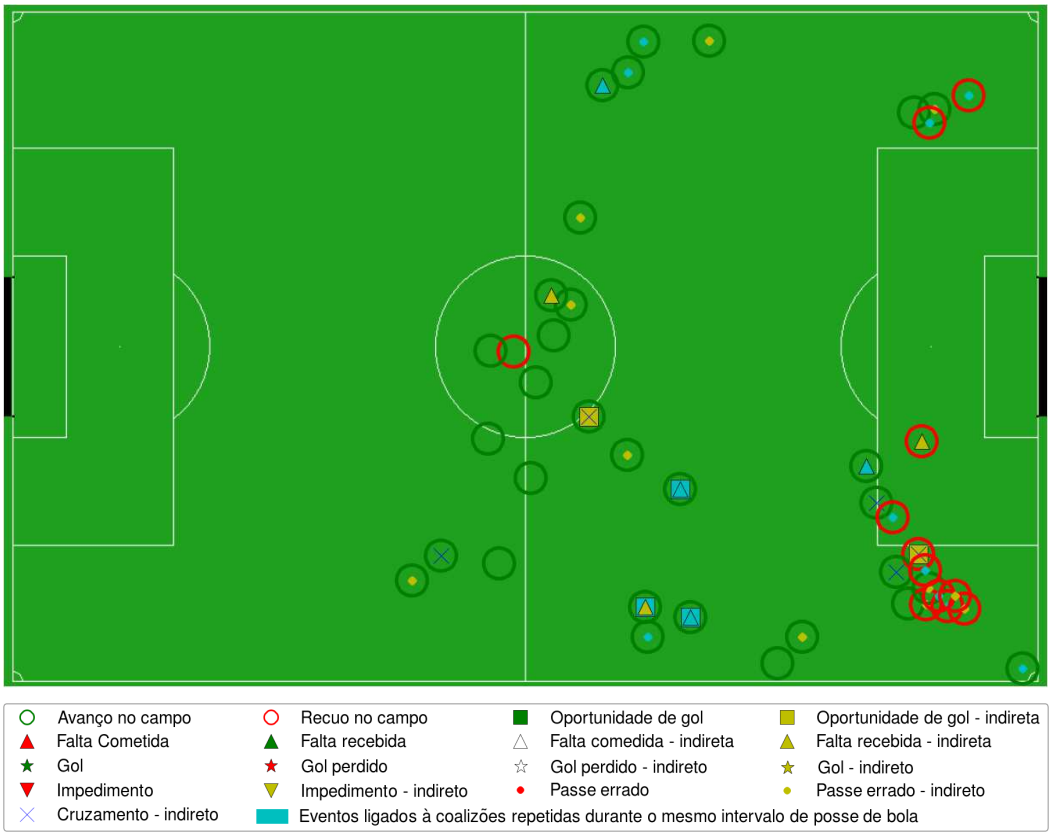


Figura 41: Tabelas do time *YuShan* ocorridas contra o adversário *FCP_GPR*.

Na Figura 41 pode ser observado que há uma concentração de tabelas entre o início da grande área e a zona de escanteios. Ao assistir à reprise da partida, foi possível constatar que uma jogada quase que recorrente do time *YuShan* era conduzir a bola até a linha de fundo e tentar passes ou chutes em direção à área. Geralmente, chegar até a linha de fundo envolvia a condução lenta da bola, e somente em alguns casos eram trocadas tabelas para atingir este objetivo. Ao chegar à linha de fundo, a área adversária já se encontrava lotada de defensores. Nesses momentos, tabelas curtas ocorriam na procura de espaços e alguns cruzamentos eram tentados, mas pouco efetivos.

Quanto ao time *FCP_GPR*, que desempenhou poucas jogadas, cabe salientar que eram priorizados os toques rápidos em diagonal e a condução de bola, mais lenta. Esse tipo de tática, pelas observações já realizadas, deixa os jogadores isolados e tendo pela frente vários oponentes. É possível, dependendo da situação em que se encontra a partida, haver áreas livres e espaços (propícias a toques longos em diagonal). No entanto, se o jogador que recebe a bola for conduzi-la, provavelmente perderá a vantagem de espaço livre conquistada.

7.2.3 Times na Parte de Baixo da Tabela de Classificação

Foi considerada apenas uma partida entre os times nas últimas posições da tabela. Esta partida contou com o confronto dos times *Hermes* e *Enigma*, respectivamente na décima quarta e na décima quinta colocações.

Apesar das últimas colocações, e tendo em consideração à proximidade de nível dos dois times, estes apresentaram um número considerável de jogadas. O time *Hermes* obteve vinte e oito tabelas e quatro triangulações enquanto o time *Enigma* obteve vinte e sete tabelas e nenhuma triangulação. Na avaliação de pontos o time *Hermes* obteve 7,55 pontos, já o time *Enigma* teve pontuação negativa de -0,35 pontos. A posse de bola gerada a partir das tabelas e triangulações é ilustrada na Figura 42.

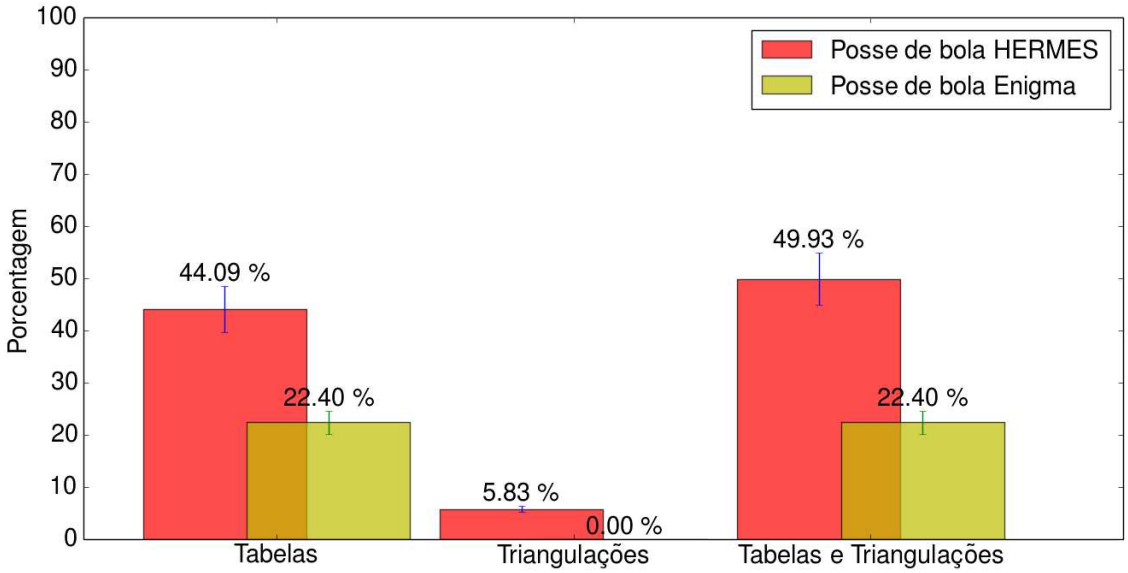


Figura 42: Percentual da posse de bola dos times *Hermes* e *Enigma* através das tabelas e triangulações.

O time *Enigma* possuía quase o dobro de posse de bola total em relação ao time *Hermes*. Essa diferença se reflete nos percentuais de posse de bola, embora os dois times possuam quase a mesma quantidade de tabelas.

Embora tenha havido bastantes jogadas, as pontuações foram baixas. Não há eventos positivos ligados às jogadas de ambos os jogadores. O time *Hermes*, teve principalmente passes errados indiretos relacionados às suas jogadas, mas também alguns recuos e faltas cometidas. As tabelas foram todas praticamente no campo defensivo, na região do meio campo. Todavia, grande parte das jogadas encerravam com avanço em direção ao campo adversário. A Figura 43 ilustra essas características.

O time *Enigma*, por sua vez, teve ainda mais passes errados e recuos relacionados às jogadas que executava do que seu adversário, conforme ilustrado na Figura 44. Por isso chegou a receber pontuação final negativa. No entanto, esse time esteve posicionado geralmente no campo ofensivo. Ao rever à reprise da partida foi possível constatar que o time *Enigma* teve muitas perdas de bola. No entanto, esse time tem uma troca de

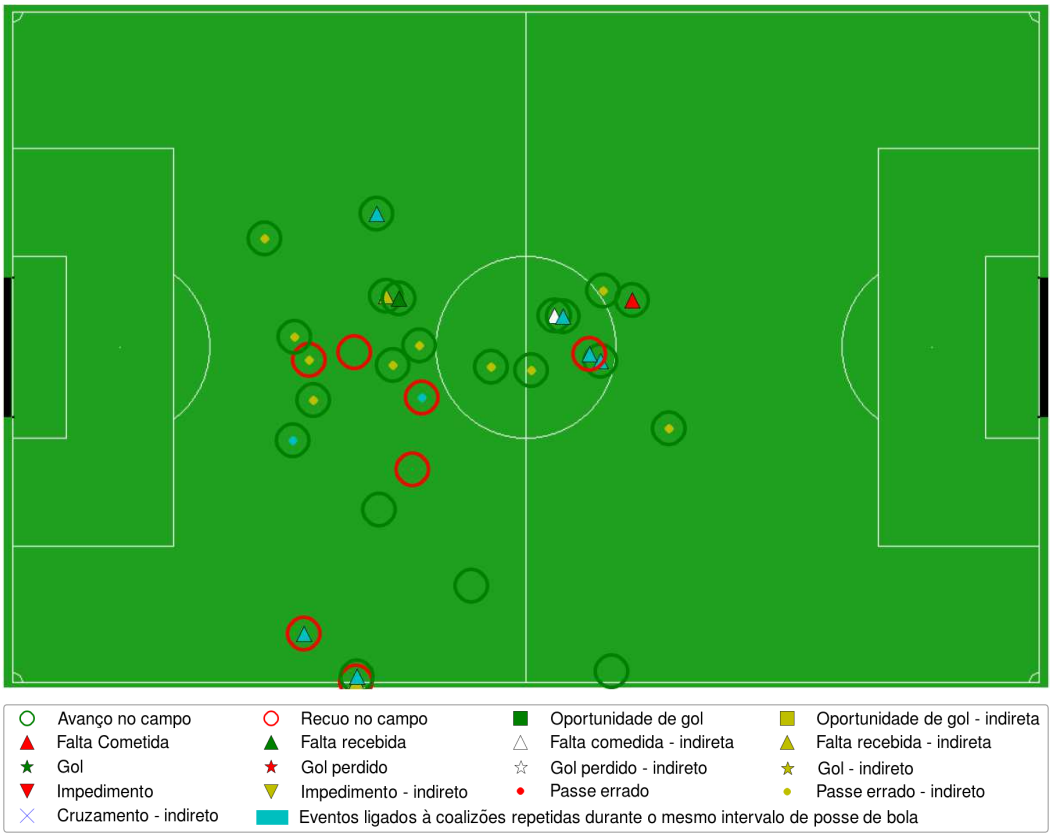


Figura 43: Tabelas do time *Hermes* ocorridas contra o adversário *Enigma*.

passes mais apurada que seu adversário e até do que outros times melhor classificados na competição. Também efetua tabelas e triangulações de forma consistente, geralmente procurando espaços ou na condição de manter a posse de bola. Todavia, aproximar-se da área adversária ainda é um desafio, por exigir um posicionamento mais encurtado e toques mais rápidos.

Executar tabelas e triangulações para manter a posse de bola é tarefa mais fácil do que utilizá-las para adentrar na área adversária. Geralmente no meio campo ou na defesa os espaços são mais abertos, os ângulos mais livres e os jogadores mais distantes. Isso possibilita mais liberdade para o posicionamento e troca de passes. Na área adversária há o afunilamento e diminuição de espaços. Ambos os times, *Hermes* e *Enigma* possuem boa troca de passes e tabelamento, no entanto é necessário a evolução destes times para considerar as situações encontradas próximo e dentro da área de gol.

Uma das condições que prejudicam a troca de passes nestes times é o domínio de bola e a condução da mesma. Comumente, estas deficiências resultam na quantidade de passes errados e necessidades de recuos.

Ainda pode ser observado na Figura 43, que o time *Hermes* avançava no seu próprio campo de defesa, e grande parte da pontuação que obteve se deve à isso. Por outro lado, o time *Enigma*, na Figura 44, muitas vezes recuava, mas já postado no campo adversário. Isto traz à tona a necessidade mudança na pontuação desse quesito, talvez considerando

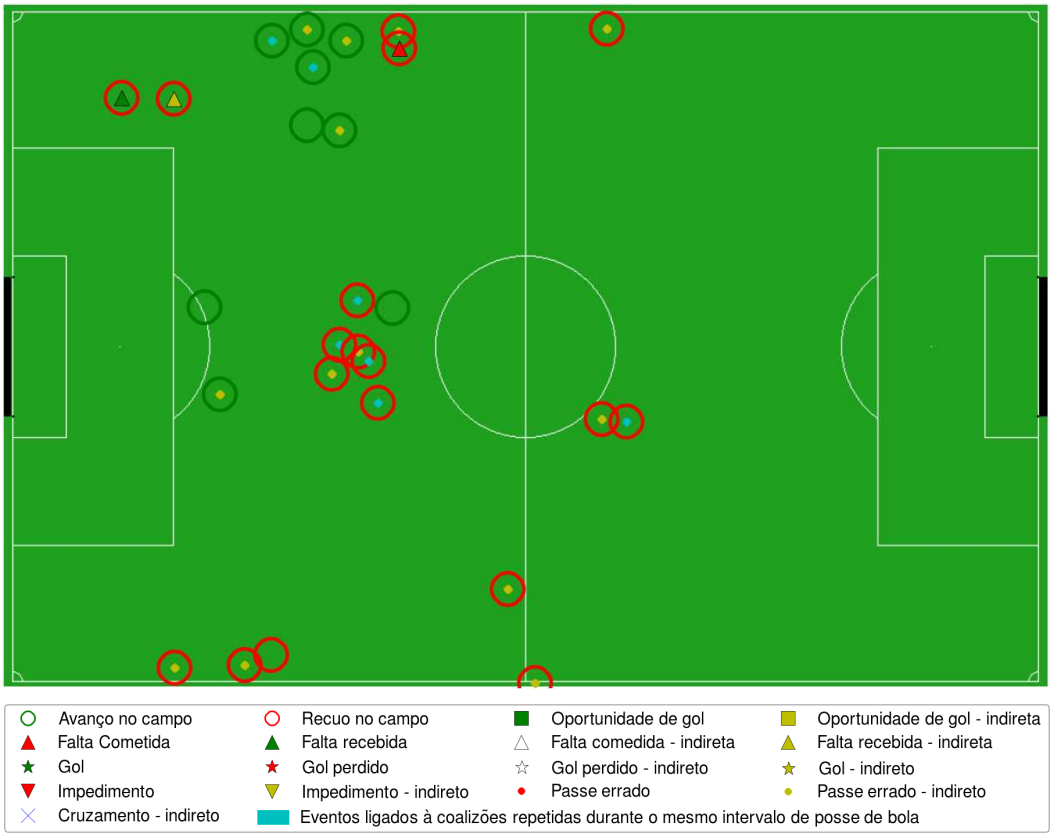


Figura 44: Tabelas do time *Enigma* ocorridas contra o adversário *Hermes*.

regiões no campo. Mesmo que não haja subtração de pontos devido à recuo no campo, posicionar-se no campo adversário frequentemente é mais produtivo do que permanecer na defesa.

7.2.4 Outras Disputas Consideradas

Resolveu-se considerar ainda outras duas disputas, para observar situações particulares. Primeiramente, confrontar times com certa diferença na tabela de classificação. Outra questão diz respeito a observações realizadas em testes.

Foi observado que o time *WrightEagle* frequentemente produz um número considerável de tabelas e triangulações em relação a outros times. Por outro lado, o time *Infographics* tem um tipo particular de sistema defensivo, no qual posiciona praticamente todos os jogadores em linha quando está sem a bola. Quando chega perto de sua própria área de gol, a formação em linha muda nas extremidades procurando o afunilamento na região da área, mas ainda mantendo uma linha de defesa sólida. Este time em questão, só desiste da formação em linha quando tem a posse de bola e ultrapassa a linha de meio campo.

A tática defensiva do time *Infographics* funciona bem quando considerados times com repertório de jogadas mais limitado. A linha que este produz é difícil de ser transposta, podendo para isso, ser considerado o uso de lançamentos, passes de infiltração e tocar a bola procurando espaços.

Portanto, na primeira partida foram confrontados os times *Infographics* e *WrightEagle*. O time *Infographics* executou sete tabelas e nenhuma triangulação e obteve 2,5 pontos. O time *WrightEagle* executou sessenta e uma tabelas e vinte e sete triangulações e obteve 52,25 pontos. Os percentuais de posse de bola garantidos pelas tabelas e triangulações estão ilustrados na Figura 45.

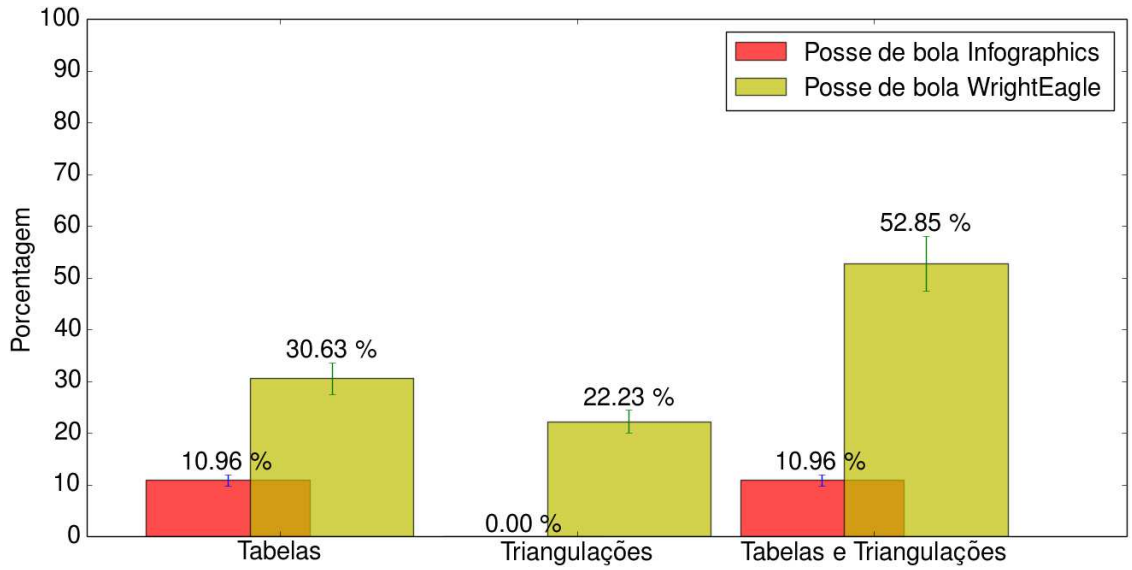


Figura 45: Percentual da posse de bola dos times *Infographics* e *WrightEagle* através das tabelas e triangulações.

O time *Infographics* executou todas as suas tabelas no campo defensivo. O tempo total de posse de bola desse time foi bem inferior em relação ao time *WrightEagle*. Embora as disputas contra o time *Infographics* - dependendo do nível de classificação dos times na competição - frequentemente tenham placares mínimos, a partida analisada terminou com o placar de 3 a 1, a favor do time *WrightEagle*. A Figura 46 ilustra a distribuição das tabelas do time *Infographics* durante a partida.

Para obter um placar favorável, o time *WrightEagle* fez uso das tabelas e triangulações para manter a posse de bola e procurar espaços. A distribuição das ocorrências de jogadas entre combinações de jogadores foi a mais uniforme até então. Frequentemente, essas jogadas estiveram relacionadas à execução de cruzamentos, onde praticamente todas as combinações de coalizões apresentaram esse fundamento. Houve também, com frequência, relações indiretas com passes errados, e de forma menos incidente recuos no campo. Cabe resaltar - mediante as partidas analisadas - a tendência do time *WrightEagle* em realizar as tabelas e triangulações sempre no campo adversário. A Figura 47 ilustra a distribuição de triangulações do time *WrightEagle*.

Ainda que o time *WrightEagle* tenha tido uma distribuição de ocorrências de tabelas e triangulações entre combinações de jogadores mais uniforme, algumas destas se destacaram. As tabelas entre os jogadores 3 e 10 e, 3 e 11 registraram respectivamente onze e doze ocorrências. Os jogadores 3, 10 e 11 ainda constituíram oito triangulações na

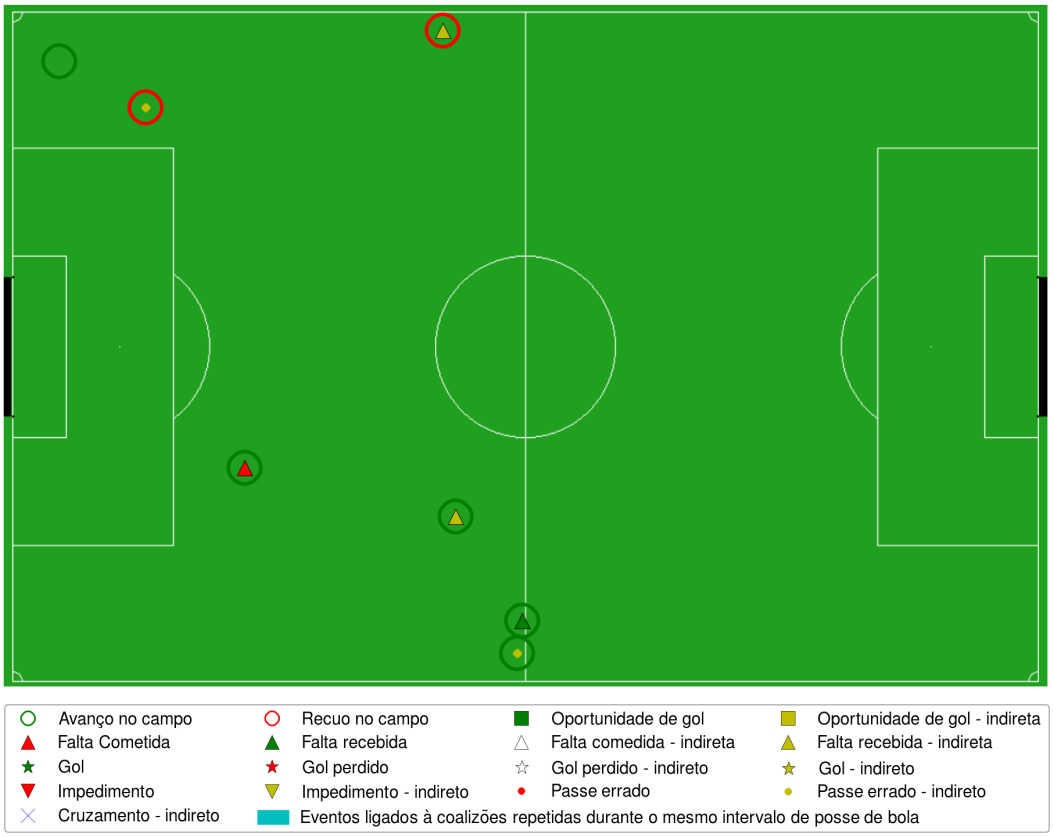


Figura 46: Tabelas do time *Infographics* ocorridas contra o adversário *WrightEagle*.

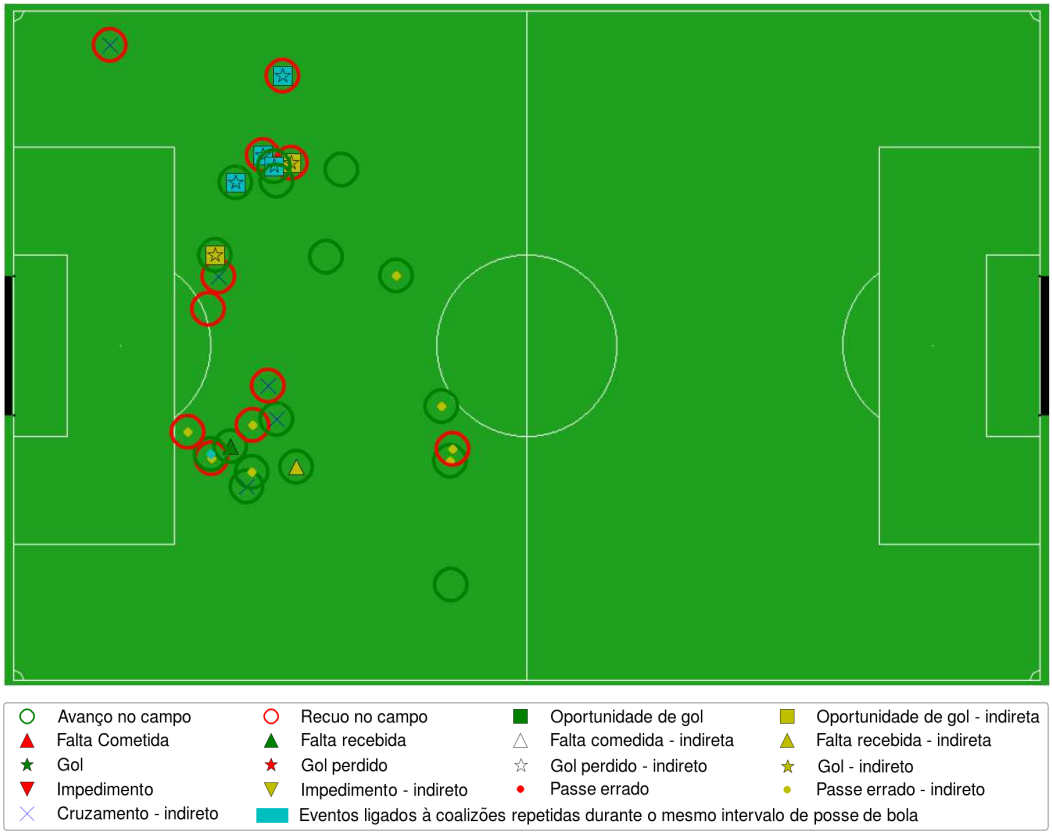


Figura 47: Triangulações do time *WrightEagle* ocorridas contra o adversário *Infographics*.

partida. Os jogadores 4, 6 e 7 constituíram nove triangulações na partida, enquanto os jogadores 6, 7, e 10 constituíram cinco triangulações. Ainda houve outras combinações de tabelas com ocorrências frequentes na partida. Estes números demonstram indiretamente a capacidade coletiva do time *WrightEagle*. Enquanto outros times mantêm ocorrências de combinações de tabelas e triangulações entre alguns poucos jogadores, o time *WrightEagle* é capaz de apresentar um repertório de ocorrências mais variado.

A segunda partida considerada foi realizada entre os times *AUT-Parsian* e *Cyrus*. Os motivos determinantes para a análise dessa partida, além da diferença nas posições de classificação, foram o fato do time *Cyrus* apresentar um número de tabelas e triangulações geralmente alto em relação à outros times. Por outro lado, o time *AUT-Parsian* apresenta um número de tabelas e triangulações sempre baixo.

Nesse novo confronto, o time *AUT-Parsian* apresentou três tabelas e duas triangulações, obtendo 1,15 pontos. O time *Cyrus* apresentou trinta e seis tabelas e nenhuma triangulação, obtendo 13,05 pontos. Os percentuais de posse de bola garantidos pelas tabelas e triangulações foram de 14,14 por cento para o time *AUT-Parsian* e de 28,14 por cento para o time *Cyrus*.

Ao assistir a reprise da partida, foi constatado que o time *AUT-Parsian* tem uma troca de passes e controle de bola menos eficiente. Os jogadores frequentemente demoram a tocar a bola, ou erram os passes direcionando a bola aos oponentes. Para a execução de tabelas e triangulações, além do posicionamento necessário à recepção da bola, é crucial o controle de bola ao receber o passe. Os jogadores do time *AUT-Parsian*, ficavam posicionados atrás dos oponentes em ângulos encobertos. Comumente também deixavam a bola escapar na recepção de passe ou faziam o domínio de bola com dificuldades, gastando tempo. Há exceção a ser feita sobre o time *AUT-Parsian* são a movimentação e o posicionamento defensivos, nas quais apresenta relativa eficiência. Os jogadores se movimentavam em sincronia, cobrindo bem os espaços no campo à cada movimento do adversário com a bola.

O time *Cyrus* geralmente executa tabelas e triangulações nas partidas que disputa. Contra o time *AUT-Parsian*, em específico, não houve triangulações. A preferência era por tabelas, dentre as quais, onze foram executadas entre os jogadores 9 e 10, com grande diferença em relação às demais combinações. O jogador 9 evitava passar a bola ao jogador 7, embora este estivesse posicionado mais próximo. As tabelas se concentraram em metade do campo adversário, onde principalmente os jogadores 9 e 10 procuravam avançar. O jogador 9 funcionava como um meia-atacante livre pela esquerda e todas as jogadas do time *Cyrus* passavam por este jogador.

7.3 Baterias de Teste

Nas baterias de teste, cada time foi avaliado contra os demais. Serão descritos os resultados gerais obtidos e serão feitas análises para casos específicos.

Cabe salientar que o teste feito em larga escala comprovou que as ocorrências de tabelas e triangulações tendem a se concentrar em alguns poucos jogadores. Essa característica é o principal fundamento na defesa destes tipos de jogadas como sendo coalizões dentro dos times. Não há certeza sobre as jogadas emergirem implicitamente ou explicitamente, porém, conclui-se que estas sejam especificadas e modeladas na implementação dos times. Durante os testes, era comum os times apresentarem padrões de comportamento, seja devido às regiões do campo que costumam atacar, nos tipos de jogadas que costumam executar e nos jogadores envolvidos.

Um dos fundamentos considerados para a avaliação das tabelas e triangulações é o avanço ou recuo em direção ao campo adversário. Os dados de posicionamento após o término das jogadas, quando plotados, produzem características passíveis de análise. No entanto, devido à quantidade de baterias realizadas, será priorizada a apresentação dos números gerais que ocorreram. A Figura 48 ilustra os dados de movimentação no campo gerados pelas coalizões (tabelas e triangulações).

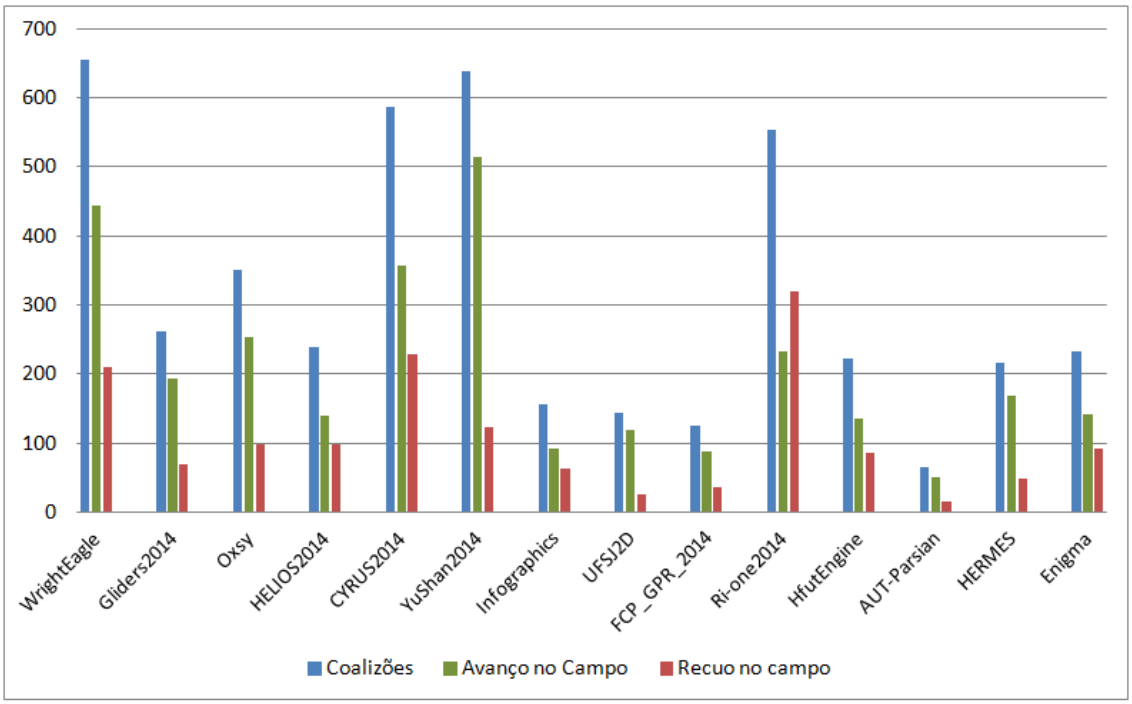


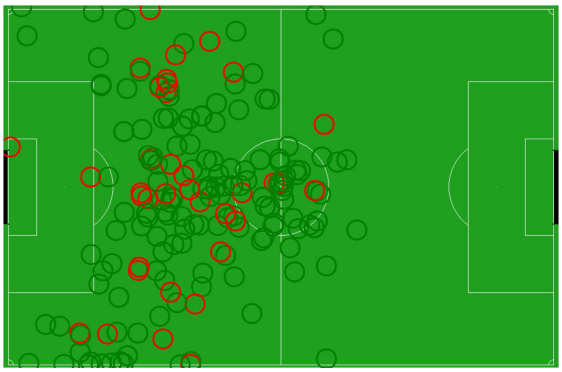
Figura 48: Quantidade de coalizões para cada time e avanços ou recuos no campo.

A abordagem de avaliação concede pontos aos avanços no campo, mas nenhum desconto de pontos é realizado caso haja recuos. A movimentação durante a troca de passes no futebol torna comum a necessidade de recuos, seja para encontrar novos espaços, atrair a marcação adversária ou livrar-se da marcação. Isso tudo foi previamente considerado.

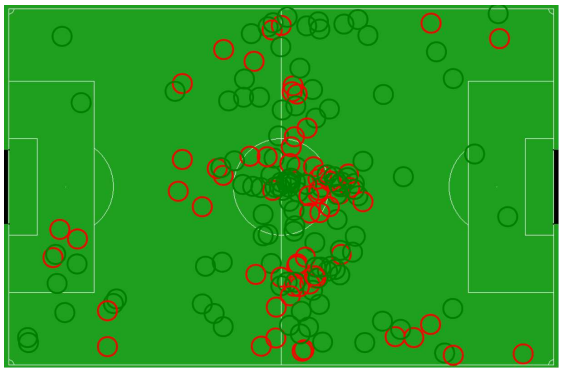
Conforme pode ser observado na Figura 48, as proporções entre avanços e recuos diferem bastante entre os times. O time *YuShan* possui a maior diferença entre as proporções de avanços e recuos. Diferenças menos expressivas não significam resultados negativos. Porém, há de se considerar que, se um time apresenta proporção próxima ou superior de recuos em relação aos avanços em campo, não conseguirá alcançar a área adversária com frequência.

Os números gerais de avanços ou recuos podem expressar a capacidade dos times de alcançarem o gol adversário, mas não são suficientes para isso. Considerando-se a movimentação própria do futebol, chegou-se à conclusão de que um time deve apresentar um balanceamento entre os avanços e recuos no campo.

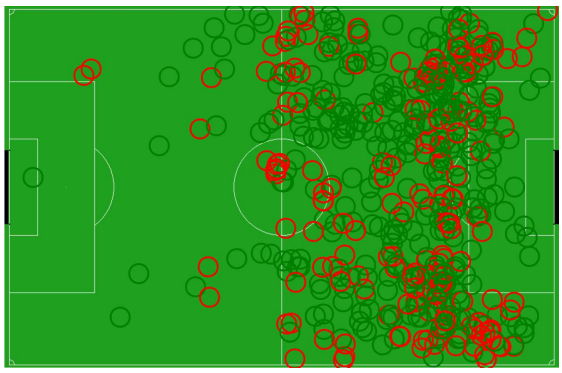
Concluiu-se também que a abordagem de avaliação - em relação à movimentação no campo - deve considerar as regiões que os jogadores efetuam as jogadas, uma vez que, um time pode apresentar um número grande de avanços, mas estar postado somente no seu campo de defesa. No caso contrário, um time realiza vários recuos, mas está postado no campo adversário. Para justificar essa consideração, na Figura 49 são ilustrados três padrões de movimentação no campo, demonstrados por três dos times analisados.



(a) Hermes



(b) Helios



(c) WrightEagle

Figura 49: Movimentação no campo, através das tabelas, apresentada por três times. As ocorrências de triangulações, nas baterias de teste, também seguiam o padrão indicado pelas Figuras 49(a), 49(b) e 49(c).

Ao se considerar a proporção entre recuos e avanços apresentados pelo time *Hermes*,

na Figura 48, e fazer uma comparação com os dados apresentados na Figura 49(a), chega-se a conclusão de que - embora apresente avanços no campo - o time *Hermes* está atuando, praticamente, só na defesa. Em contrapartida, o time *Helios* possui uma proporção de recuos e avanços bem próxima, no entanto, mantém a realização de tabelas em uma faixa do meio de campo, conforme a Figura 49(b). Não obstante, as jogadas do time *Helios* não continuam com muita frequência no campo adversário demonstrando que este tinha, igualmente, dificuldades de avançar. De posse de outros dados é possível indicar que as tabelas e triangulações do time *Helios* eram relacionadas constantemente à passes errados. Por outro lado, embora o time *WrightEagle* mantenha, aproximadamente, pouco mais do que o dobro de avanços em relação aos recuos, possui como tendência a realização de suas tabelas e triangulações quase que totalmente no campo adversário - conforme apresentado na Figura 49(c).

Ainda pode ser observado na Figura 49(c) que as regiões de maior concentração de recuos, geradas pelas tabelas do time *WrightEagle*, são as circunvizinhanças da área de gol do time adversário, onde as tabelas e triangulações eram realizadas para encontrar espaços.

Após a execução de todas as baterias de teste, foram compilados alguns dados gerais, entre os quais encontram-se: os números de tabelas e triangulações apresentados pelos times, os dados já anteriormente expostos e a pontuação final adquirida na abordagem de avaliação por cada time. Esses dados serão apresentados na Tabela 7 para o início da próxima discussão.

Fora algumas discrepâncias, a abordagem de avaliação funciona conforme o idealizado. Se as jogadas dos times apresentam avanços no campo são pontuadas. A tendência inicial é de que quanto mais jogadas, maior será a pontuação. Embora não seja considerada a quantidade de campo avançada, a principal discrepância surge pela desconsideração de descontos em caso de recuo no campo. Este quesito deve ser modificado para considerar as regiões que os jogadores executam os recuos.

Ainda assim, os recuos refletem na pontuação final dos times. Isto ocorre não só pelo fato da ausência de pontuação, mas também devido ao distanciamento dos jogadores de outros eventos que influenciam na avaliação. Se um time fizer muitos recuos ficará longe da meta de gol, e por consequência de eventos como oportunidades de gol, gols, chutes a gol e cruzamentos.

Para confirmar a ponderação feita no parágrafo anterior, cabe observar os dados apresentados pelo time *Ri-one*. Este possui um número elevado de tabelas e triangulações que frequentemente estão relacionadas à recuos - conforme a Figura 50. O padrão de movimentação das jogadas do time *Ri-one* indica dificuldades na saída de bola, pois este, realiza uma concentração grande de recuos na frente de sua área de gol e continua a realizar recuos no seu campo de defesa e parte do campo adversário. Curiosamente, na metade final do campo adversário, o time *Ri-one* passa apresentar avanços na quase totalidade

Tabela 7: Resultados gerais dos times após as baterias de teste.

	Coalizões	Tabelas	Triangulações	Avanço no Campo	Recuo no campo	Pontos
WrightEagle	656	541	114	445	211	368,45
Gliders2014	262	218	44	193	69	139,20
Oxsy	351	284	67	253	98	243,25
HELIOS2014	244	179	64	140	99	37,65
CYRUS2014	615	502	113	371	244	201,00
YuShan2014	638	565	73	514	124	376,05
Infographics	156	137	19	93	63	72,20
UFSJ2D	145	129	16	119	26	74,05
FCP_GPR_2014	126	115	11	89	37	59,60
Ri-one2014	554	469	85	234	320	178,40
HfutEngine	222	202	20	135	87	95,35
AUT-Parsian	66	55	11	50	16	18,30
HERMES	217	185	32	168	49	54,65
Enigma	234	213	21	142	92	73,10

das jogadas executadas.

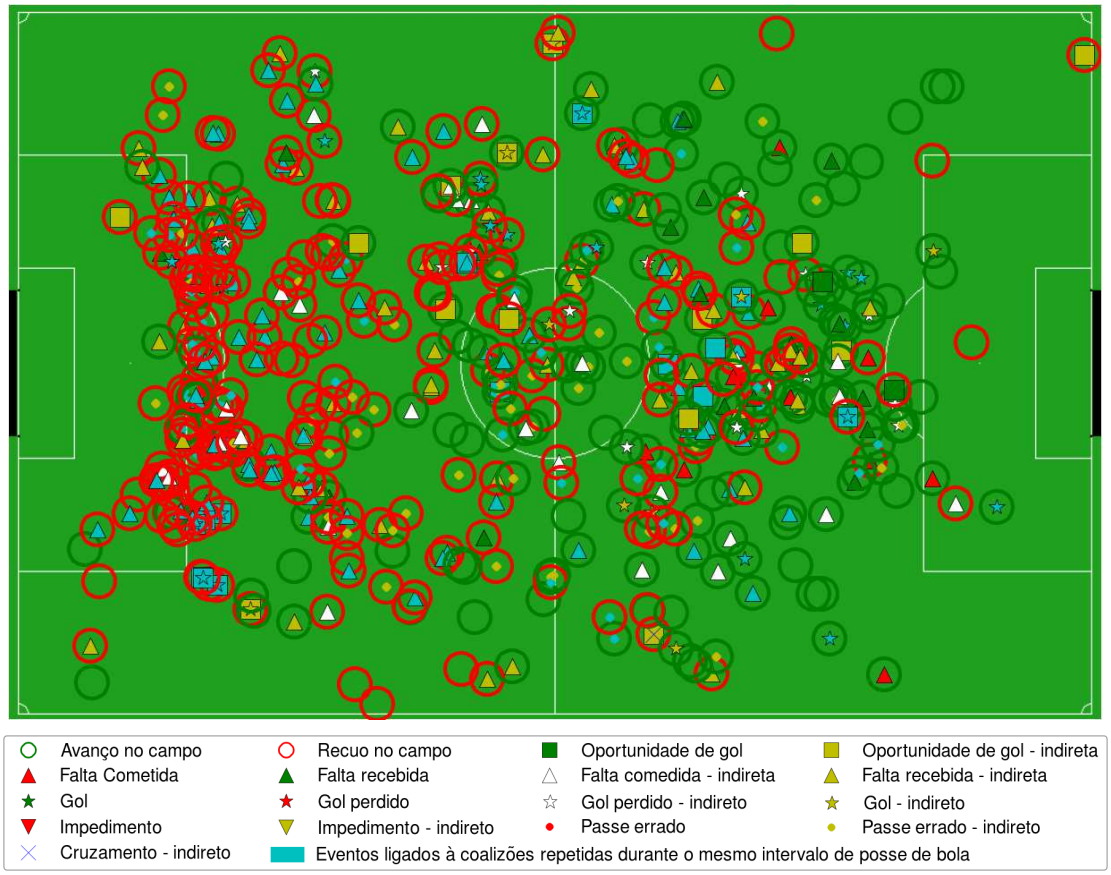


Figura 50: Tabelas do time *Ri-one* ocorridas durante sua bateria de teste.

O principal fator que influencia na pontuação é a quantidade de eventos realizados nas proximidades da área de gol. Ao observar os dados referentes ao time *Helios*, na Tabela 7, é possível constatar o impacto que os recuos têm na pontuação final. Como visto anteriormente, o time *Helios* frequentemente mantinha suas jogadas numa faixa do meio de campo. Este não conseguia se desprender dessa região e errava muitos passes. No entanto, apesar da quantidade de passes errados, o que mais impacta a pontuação do time *Helios* é o baixo número de eventos na região da área de gol. As partidas que o time *Helios* disputou, geralmente, terminavam com o placar zerado. A Figura 51 apresenta as ocorrências dos eventos da região da área de gol para o time *Helios* e os demais.

Os cruzamentos e escanteios também são considerados por representarem situações de ataque aos times. No entanto, estes fornecem poucos pontos por serem oportunidades de ataque menos contundentes. Além disso, são considerados os passes errados, faltas cometidas e faltas recebidas. O mais comum na ocorrência desses eventos é a forma indireta, ou seja, o evento não é realizado pelo jogador que finalizou a tabela ou triangulação, mas por outro jogador diante da continuidade de posse de bola. Neste caso, esses eventos subtraem ou somam poucos pontos. A Figura 52 ilustra as ocorrências dos demais eventos que impactam na avaliação das coalizões.

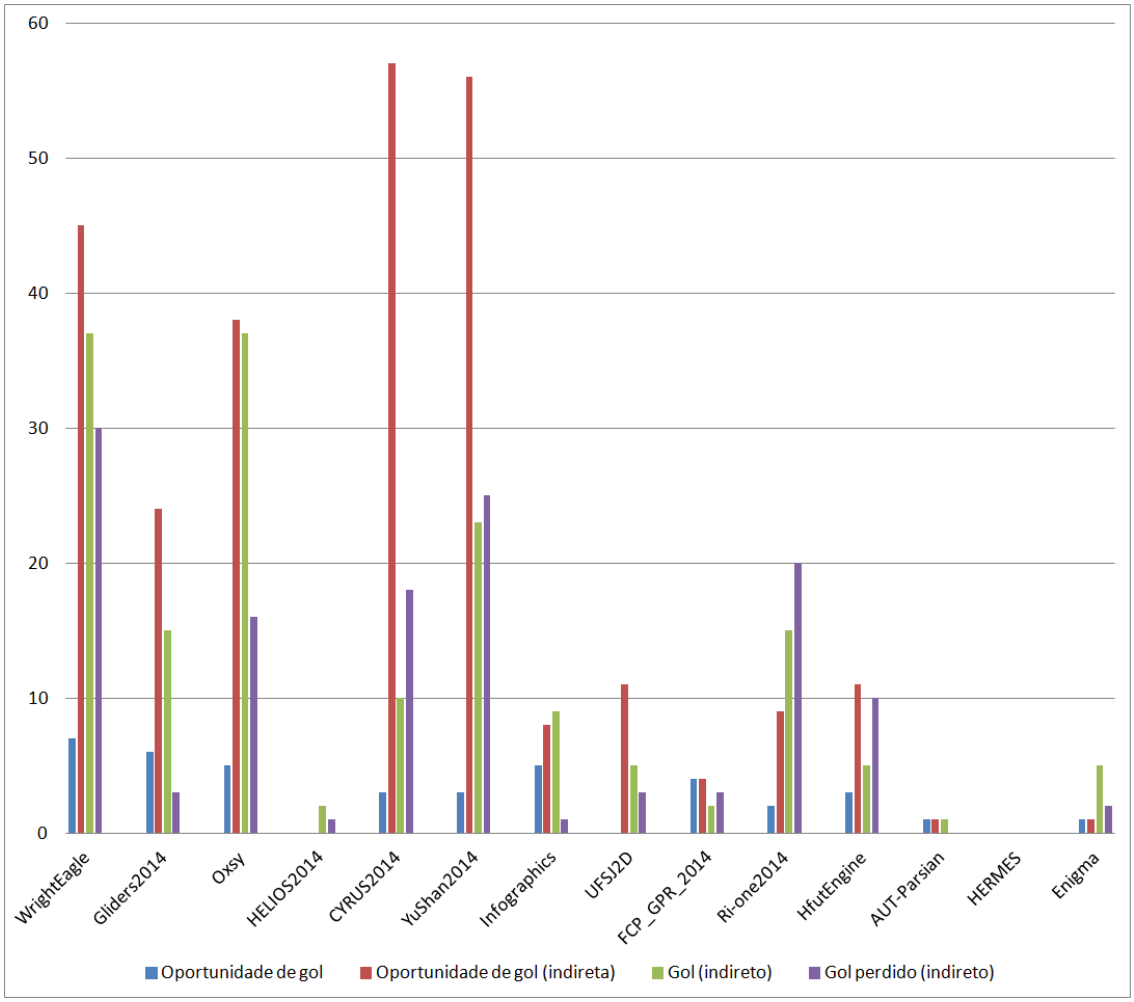


Figura 51: Quantidade de ocorrências de eventos para cada time.

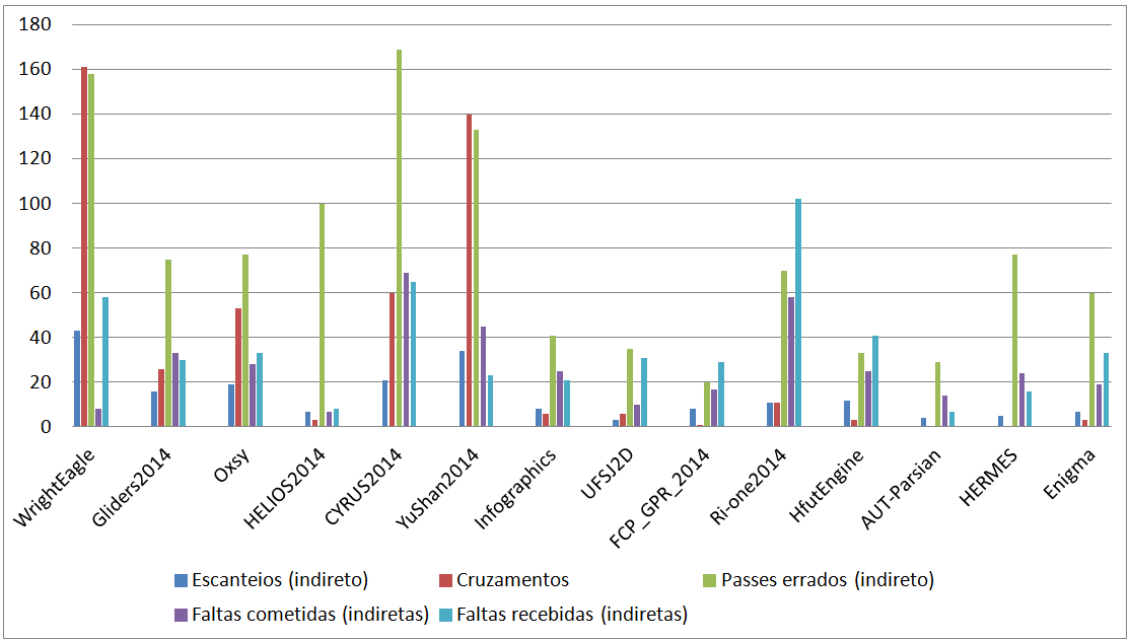


Figura 52: Quantidade de ocorrências de eventos para cada time.

É possível constatar a partir da Figura 52 que o evento mais prejudicial a continuidade de ações dos times, após a realização das tabelas e triangulações (culminando na perda da posse de bola), são os passes errados.

A quantidade de cruzamentos e escanteios também demonstra a presença de alguns times nas proximidades da área de gol adversária. Isto, por sua vez, culmina em mais chances de fazer gols. É possível considerar, de forma indireta, que a menor quantidade de faltas recebidas ou cometidas indica a capacidade do time de fazer uso das tabelas e triangulações para livrar-se da marcação adversária, priorizando o toque de bola.

Outra questão levantada neste trabalho diz respeito à incidência de ocorrências das coalizões nos times. Cada coalizão produz uma jogada que resulta de uma combinação única de jogadores. Se uma combinação se repete por inúmeras vezes, torna-se possível reforçar o argumento de que se trata de uma coalizão dentro de um time.

Durante as baterias de testes, os times apresentaram quantidades diversas de combinações únicas. Poder-se-ia pensar que a quantidade de combinações de coalizões diferentes tendesse a acentuar com o número de coalizões produzidas por um time. Isso, porém, praticamente, não ocorre. Times com uma quantidade menor de coalizões, por vezes, apresentaram maior número de combinações. Esse é o caso, por exemplo, dos times *Yushan* e *Rio-One*. O time *Yushan* teve um total de 638 ocorrências de coalizões a partir de 32 combinações diferentes, enquanto o time *Rio-One* teve um total de 554 ocorrências de coalizões a partir de 62 combinações diferentes.

O total de combinações diferentes de coalizões é provavelmente influenciado pelas técnicas implementadas nos times junto às concepções estruturais e às táticas de jogo. As quantidades de ocorrências para cada combinação de coalizão também podem indicar a flexibilidade de um time na criação de jogadas. Quanto mais a concentração das ocorrências de jogadas for distribuída em mais combinações, maior será a tendência de flexibilidade em uma partida. No entanto, a dispersão das ocorrências entre muitas combinações promove a não objetividade em um time. As Figuras 53 e 54 apresentam as quantidades de ocorrências das coalizões dos times *Hermes* e *Yushan* respectivamente.

A partir da Figura 53 é possível constatar que o time *Hermes* possui pouca flexibilidade na execução de suas jogadas, uma vez que, concentrou grande parte destas nas tabelas da combinação entre jogadores 10 e 11. Por outro lado, na Figura 54, o time *Yushan* concentrou a execução das jogadas em uma maior quantidade de combinações de jogadores.

Conclui-se, então, a descrição de alguns dos resultados das baterias de teste. Na próxima seção serão discutidos os resultados gerados até então.

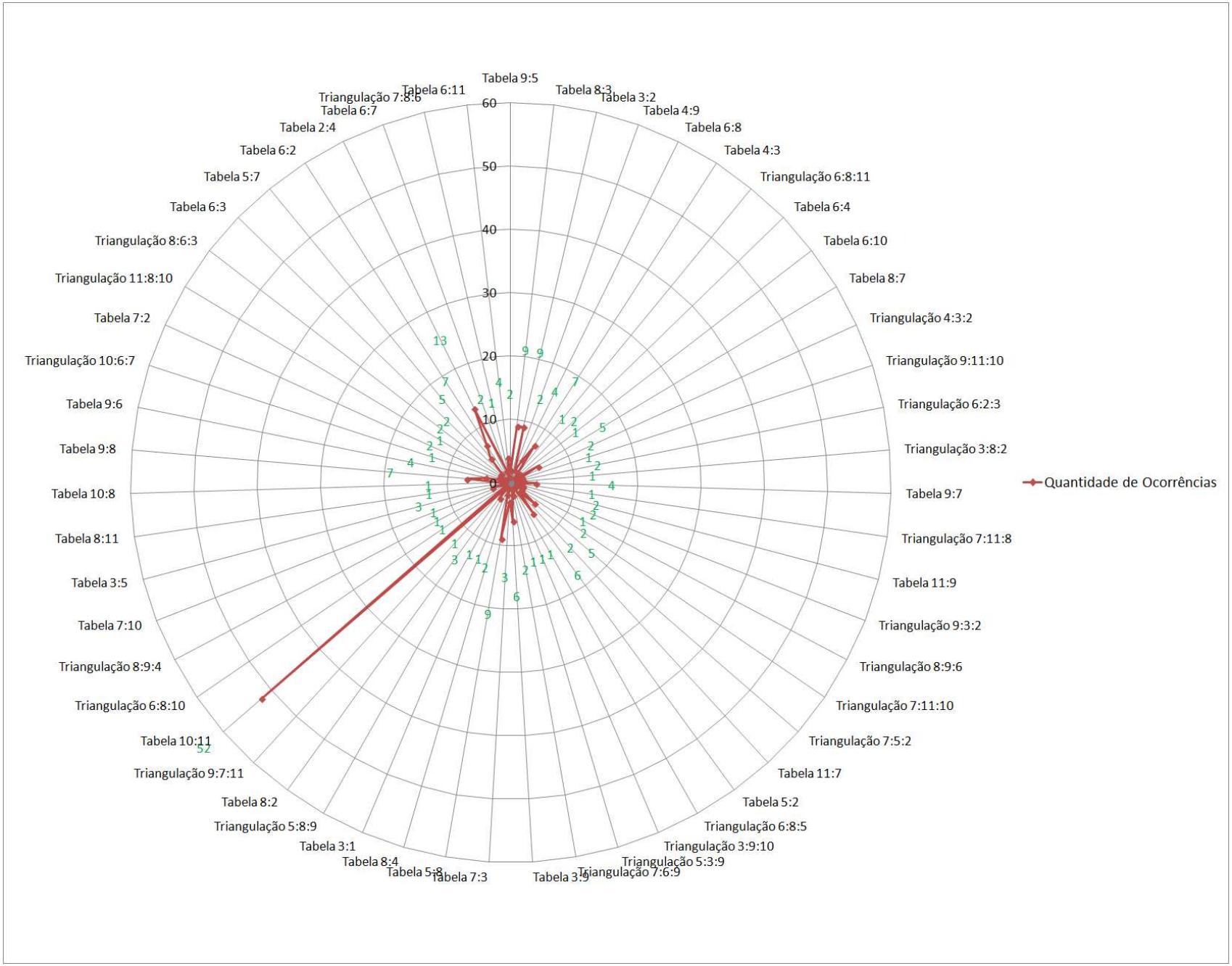


Figura 53: Quantidade de ocorrências de coalizões do time *Hermes* após a bateria de teste.

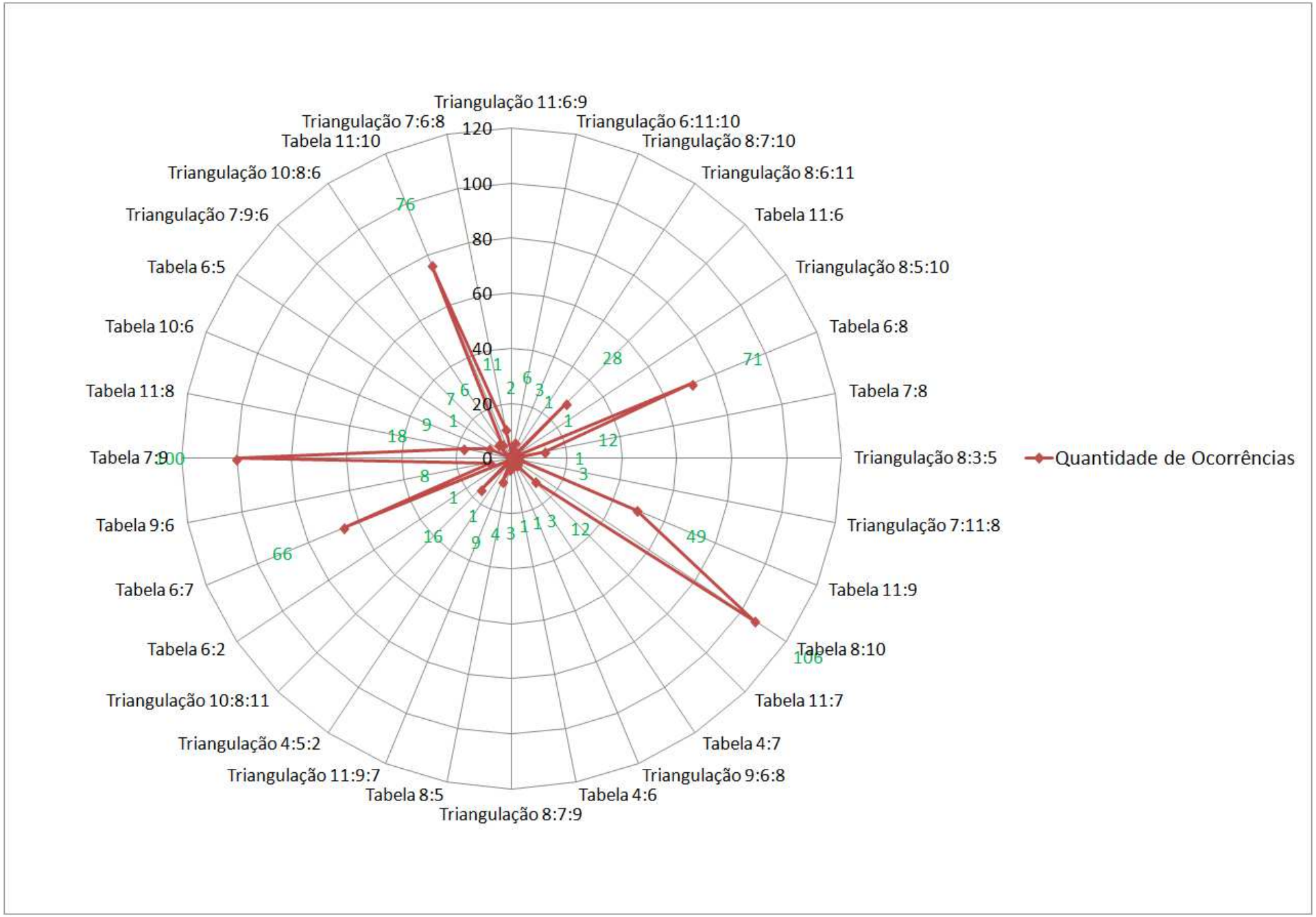


Figura 54: Quantidade de ocorrências de coalizões do time *Yushan* após a bateria de teste.

7.4 Discussão dos Resultados

Depois de decorrida a validação do modelo, por meio da execução dos testes - seja nas partidas disputadas uma única vez ou nas baterias que envolviam todos os times - será procurado discorrer sobre algumas inferências que foram alcançadas.

Dentre os argumentos realizados na Seção 6.3 para o uso das tabelas e triangulações está a retenção da posse de bola. Provou-se que manter a posse de bola é um dos ganhos obtidos na execução dessas jogadas. Os times que têm um percentual maior do total de posse relacionado às tabelas e triangulações, salvo algumas exceções, apresentam mais posse de bola frente aos adversários. Manter a posse de bola significa poder trabalhar para que as condições de ataque aconteçam.

As tabelas e triangulações também estão relacionadas à procura de espaços e à fuga da marcação adversária. Tem-se, dessa forma, que: se os times executam com maior efetividade as jogadas podem avançar no campo com mais frequência. Essa efetividade não está relacionada somente à diminuição dos recuos, mas principalmente à esquiva de eventos como faltas (cometidas ou recebidas) e passes errados.

Outra constatação importante é a região do campo na qual as jogadas são realizadas. Alguns times como o *WrightEagle* mantém a execução das jogadas quase totalmente no campo adversário. No caso do *WrightEagle* isto é alcançado independente do adversário combatido. Isso indica a ofensividade dos times. As regiões e os eventos relacionados às jogadas também podem indicar quais as dificuldades que os times apresentam ao tentar avançar no campo, seja devido à faltas ou passes errados.

Fornecer condições de ataque é também uma determinante na execução de tabelas e triangulações. Não é surpresa que eventos de oportunidades de gol e gols estejam ligados, mesmo que indiretamente, às tabelas ou triangulações. Não obstante, as ocorrências de tabelas e triangulações repetiam as mesmas combinações de jogadores nos seus times, sendo que, em alguns deles essas jogadas aconteciam praticamente por via de jogadores específicos. Estes geralmente possuíam a função de ataque, evidenciando o grau de importância e responsabilidade adquirida pelas coalizões dentro dos times.

A importância mencionada, muitas vezes, torna-se um problema quando os times não apresentam um repertório de variações nas ocorrências de coalizões. Em alguns times foi observado que a reincidência de jogadas em determinadas regiões do campo ou entre determinados jogadores prejudicaria o desempenho na partida, uma vez que essas jogadas eram, quase sempre, ineficazes na busca de eventos positivos.

Por fim, constata-se que os melhores resultados a partir das tabelas e triangulações surgem pela junção de: concentrações de ocorrências em combinações variadas, efetividade na execução das jogadas relacionando-as com eventos positivos e dar preferência a realização das jogadas no campo adversário. A discussão realizada nesta seção será continuada no próximo capítulo, com as conclusões deste trabalho.

8 CONCLUSÃO

Neste capítulo serão realizadas as considerações finais e definidas as direções a serem tomadas em trabalhos futuros.

Considera-se que a abordagem de avaliação desenvolvida e descrita neste trabalho alcançou resultados satisfatórios. A principal conquista refere-se à forma de avaliação não considerar apenas estatísticas individuais dos jogadores, mas integrar conceitos dantes produzidos para avaliar determinados tipos de jogadas coletivas (as tabelas e triangulações) que resultam da formação de grupos de jogadores. Estes grupos, por sua vez, são considerados coalizões dentro dos times.

Ao longo da formulação da abordagem de avaliação, das inferências realizadas e, posteriormente, de posse dos resultados via a aplicação de testes, chegou-se à conclusão de que a avaliação das coalizões deve considerar não somente a pontuação alcançada, mas a análise dos outros dados. Essa análise consiste na verificação das regiões onde as coalizões se concentram; na verificação dos eventos relacionados às coalizões; e na verificação das quantidades de ocorrências para cada combinação de coalizão.

Esse tipo de cuidado permite compreender as dificuldades que os times encontram no uso das tabelas e triangulações e, no caso contrário, as potencialidades. Também permite a verificação da flexibilidade que um time possui para a execução das jogadas, que, provavelmente, pode se estender na validação ou não das técnicas utilizadas na implementação dos times.

É muito provável que as tabelas e triangulações decorram da modelagem, organização e implementação dos times, seja implicitamente ou explicitamente. Técnicas que flexibilizem as ocorrências de jogadas em combinações que estejam relacionadas aos eventos positivos devem ser priorizadas.

No tocante ao papel do ambiente na análise de desempenho, foram feitas descrições sobre as restrições impostas pelo *Soccer Server 2D* aos agentes. As restrições dizem respeito à comunicação, às limitações dos comandos e, aos parâmetros dos jogadores. Destaca-se que, embora o *Soccer Server 2D* seja um ambiente controlado, o espaço de estados-ações na dinâmica de uma partida é um desafio considerável aos times, principalmente pela demanda de coordenação dos jogadores. Foram listadas algumas técnicas

utilizadas na coordenação. Estas tornam possíveis as tabelas e triangulações.

Quanto à separação do contexto individual do social, foram listadas as ações dos jogadores que frequentemente são definidas como individuais. Em contrapartida, os jogadores atuam em uma estrutura organizacional de time e, portanto, se engajam em jogadas coletivas para alcançarem os objetivos, dentre as quais, a mais simples se traduz no passe. Para empreenderem o comportamento coletivo no time e, frequentemente, em estruturas menores que envolvem grupos de alguns jogadores, é necessário técnicas que possibilitem aos jogadores resolverem dependências entre si. Estas podem variar (construção de planos, árvores de tarefas, descrição de papéis e posicionamento no campo, entre outros), mas sempre requerem o uso da comunicação (implícita ou explícita).

Durante a fase de testes verificou-se a importância das coalizões aos times. Estas podem garantir desde a posse de bola até terem a atribuição de construir as jogadas de ataque. Muitas vezes, a troca comum de passes não conduz a gols, chances de gol, aproximação da área adversária, ou mesmo o avanço no campo. Assim, da mesma forma que no futebol praticado por humanos, as tabelas e triangulações têm função vital no desempenho de um time e expressam o nível de entrosamento e coletividade dos jogadores.

Tem-se, então, que os indicadores de avaliação das coalizões são variados devido aos objetivos primordiais das tabelas e triangulações, os quais: manter a posse de bola; livrar jogadores da marcação adversária; encontrar espaços; fornecer condições de ataque, e em último caso, flexibilizar as jogadas do time. O que define o sucesso ou não destes objetivos são os eventos relacionados às jogadas e as regiões em que estas são executadas.

A abordagem de avaliação desenvolvida neste trabalho pode ser utilizada, principalmente, para a avaliação de aspectos coletivos dos times. Primeiramente, na questão das coalizões que executam tabelas e triangulações e foram confirmadas como jogadas importantes para a obtenção de resultados melhores no futebol de robôs simulado. É possível identificar os padrões de ações dessas coalizões e do comportamento de um time durante uma partida. É possível descobrir quais combinações de jogadas entre os jogadores estão fornecendo resultados positivos. É possível ter a compreensão de como times com bons resultados fazem uso dessas jogadas durante as partidas que disputam.

Não obstante, os dados podem ajudar as equipes de pesquisa a determinarem as tecnologias que efetivamente trazem ou trarão benefícios aos times que implementam.

Os dados gravados nos arquivos csv ainda podem ser reprocessados através de mineração de dados na busca de potenciais informações. Há muitos dados que podem ser sujeitados a outras análises, desde os eventos associados indiretamente as coalizões até as variações de ocorrências destas nos times ou entre os times. Além disso, a abordagem de avaliação pode ser entendida para incluir proposições de outros trabalhos dentre aqueles citados nesta dissertação.

Tendo isso em vista, podem ser incluídos como trabalhos futuros desde a avaliação dos dados gerados por meio de técnicas da mineração de dados até a integração de ou-

tras características produzidas durante um intervalo de posse de bola na abordagem de avaliação. Pretende-se incluir a identificação dos vários tipos de jogadas desempenhadas por jogadores, incluindo as tabelas e triangulações descritas neste trabalho para uma avaliação mais uniforme, em que se possa unir ou contrastar os elementos do jogo. Neste caso, citam-se padrões de cobranças, padrões de sequências de tabelas e triangulações e de fundamentos mais básicos como dribles, condução de bola, entre outros.

É também de interesse que se avalie outras características sociais não cobertas por este trabalho. Entre as quais estão os comportamentos da movimentação coletiva dos jogadores sem a bola no campo, tanto da parte ofensiva quanto defensiva. No entanto, sabe-se que características de movimentação sem a bola - apesar da facilidade com que são identificadas visualmente - são mais difíceis de serem verificadas computacionalmente. A razão é que, diferentemente da abordagem utilizada neste trabalho, não existem eventos que delimitem esses acontecimentos no jogo.

REFERÊNCIAS

AKIYAMA, H.; DORER, K.; LAU, N. On the Progress of Soccer Simulation Leagues. In: THE 18TH ANNUAL ROBOCUP INTERNATIONAL SYMPOSIUM: SPECIAL TRACK ON THE ADVANCEMENT OF THE ROBOCUP LEAGUES, 2014, João Pessoa, PB, Brasil. **Anais...** [S.l.: s.n.], 2014. n.13.

AKIYAMA, H.; NAKASHIMA, T.; YAMASHITA, K.; MIFUNE, S. HELIOS2014 Team Description Paper. In: ROBOCUP INTERNATIONAL SYMPOSIUM, 18., 2014. **Anais...** [S.l.: s.n.], 2014. Disponível em: <http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/>, Acesso: em 12/01/15.

AKIYAMA, H.; NODA, I. Multi-agent Positioning Mechanism in the Dynamic Environment. In: VISSER, U.; RIBEIRO, F.; OHASHI, T.; DELLAERT, F. (Ed.). **RoboCup 2007**: robot soccer world cup xi. [S.l.]: Springer Berlin Heidelberg, 2008. p.377–384. (Lecture Notes in Computer Science, v.5001).

ALMEIDA, F.; ABREU, P. H.; LAU, N.; REIS, L. An automatic approach to extract goal plans from soccer simulated matches. **Soft Computing - A Fusion of Foundations, Methodologies and Applications**, n/a, v.17, n.5, p.835–848, 2012.

ALMEIDA, F.; LAU, N.; REIS, L. P. A Survey on Coordination Methodologies for Simulated Robotic Soccer Teams. In: MALLOW, 2010. **Anais...** CEUR-WS.org, 2010. (CEUR Workshop Proceedings, v.627).

BAI, A.; WU, F.; CHEN, X. Online planning for large MDPs with MAXQ decomposition. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS-VOLUME 3, 11., 2012. **Proceedings...** [S.l.: s.n.], 2012. p.1215–1216.

BEZEK, A. Modeling multiagent games using action graphs. In: MODELING OTHER AGENTS FROM OBSERVATIONS (MOO 2004), 2004. **Proceedings...** [S.l.: s.n.], 2004.

BEZEK, A. **Logalyzer**: a powerful tool for visualization and analysis of robocup log files. <http://dis.ijs.si/andraz/logalyzer/>, 2005. Home Page.

BITTENCOURT, G. **Inteligência artificial**: ferramentas e teorias. [S.l.]: Editora da UFSC, 2006. (Série didática).

BITTENCOURT, G.; DA COSTA, A. Soccer server: um simulador para o futebol de robôs da robocup federation tutorial-3. In: ENCONTRO NACIONAL DE INTELIGêNCIA ARTIFICIAL, 1999. **Anais...** [S.l.: s.n.], 1999.

BOGG, P.; BEYDOUN, G.; LOW, G. When to Use a Multi-Agent System? In: BUI, T.; HO, T.; HA, Q. (Ed.). **Intelligent Agents and Multi-Agent Systems**. [S.l.]: Springer Berlin Heidelberg, 2008. p.98–108. (Lecture Notes in Computer Science, v.5357).

BOMAN, M.; HOLM, E. Multi-Agent Systems, Time Geography, and Microsimulations. In: OLSSON, M.-O.; SJOSTEDT, G. (Ed.). **Systems Approaches and Their Application**. [S.l.]: Springer Netherlands, 2005. p.95–118.

CAO, Y. U.; FUKUNAGA, A. S.; KAHNG, A. Cooperative Mobile Robotics: antecedents and directions. **Autonomous Robots**, Hingham, MA, USA, v.4, n.1, p.7–27, mar 1997. Disponível em: <http://dx.doi.org/10.1023/A:1008855018923>, Acesso: em 15/03/14.

CHEN, M.; DORER, K.; FOROUGHI, E.; HEINTZ, F.; HUANG, Z.; KAPETANAKIS, S.; KOSTIADIS, K.; KUMMENEJE, J.; MURRAY, J.; NODA, I.; OBST, O.; RILEY, P.; STEFFENS, T.; WANG, Y.; YIN, X. **Users Manual**: robocup soccer server — for soccer server version 7.07 and later. [S.l.]: The RoboCup Federation, 2003.

CLIFF, O.; LIZIER, J.; WANG, X.; WANG, P.; OBST, O.; PROKOPENKO, M. Towards Quantifying Interaction Networks in a Football Match. In: BEHNKE, S.; VELOSO, M.; VISSER, A.; XIONG, R. (Ed.). **RoboCup 2013**: robot world cup xvii. [S.l.]: Springer Berlin Heidelberg, 2014. p.1–12. (Lecture Notes in Computer Science, v.8371).

COELHO, R.; KULESZA, U.; STAA, A. von; LUCENA, C. Unit Testing in Multi-agent Systems Using Mock Agents and Aspects. In: INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING FOR LARGE-SCALE MULTI-AGENT SYSTEMS, 2006., 2006, New York, NY, USA. **Proceedings...** ACM, 2006. p.83–90. (SELMAS '06).

CUNHA ABREU, P. M. H. da. **Artificial Intelligence Methodologies Applied to the Analysis and Optimization of Soccer Teams' Performance**. 2011. Tese (Doutorado em Ciência da Computação) — FEUP, Estrada da Circunvalação (EN12).

DIAS, M. B.; ZLOT, R. M.; KALRA, N.; STENTZ, A. T. **Market-Based Multirobot Coordination**: a survey and analysis. Pittsburgh, PA: Robotics Institute, 2005. (CMU-RI-TR-05-13).

D'INVERNO, M.; LUCK, M. **Understanding agent systems**. [S.l.]: Springer, 2004.

DROGOUL, A.; COLLINOT, A. Applying an Agent-Oriented Methodology to the Design of Artificial Organizations: a case study in robotic soccer. **Autonomous Agents and Multi-Agent Systems**, [S.l.], v.1, n.1, p.113–129, 1998.

EATON, M.; COLLINS, J.; SHEEHAN, L. Toward a benchmarking framework for research into bio-inspired hardware-software artefacts. **Artificial Life and Robotics**, [S.l.], v.5, n.1, p.40–45, 2001.

FABRO, J. A.; OENNING, B. d. O.; BRENNER, V.; REIS, L. P.; LAU, N. FCP_GPR_2014 Team: joining setplays from fcportugal with reinforcement learning from gpr2d to improve decision-making in multi-option setplays. In: ROBOCUP INTERNATIONAL SYMPOSIUM, 18., 2014. **Anais...** [S.l.: s.n.], 2014. Disponível em: <http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/>, Acesso: em 13/01/15.

FERBER, J. **Multi-Agent Systems**: an introduction to distributed artificial intelligence. 1st.ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

FERREIRA, R.; REIS, L. P.; LAU, N. Situation Based Communication for Coordination of Agents. In: SCIENTIFIC MEETING OF THE PORTUGUESE ROBOTICS OPEN, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.39–44.

GABEL, T.; RIEDMILLER, M. On Progress in RoboCup: the simulation league showcase. In: SOLAR, J. Ruiz-del; CHOWN, E.; PLÖGER, P. (Ed.). **RoboCup 2010**: robot soccer world cup xiv. [S.l.]: Springer Berlin Heidelberg, 2011. p.36–47. (Lecture Notes in Computer Science, v.6556).

GONÇALVES, E. M. N. **Uma abordagem para especificação de conhecimento para sistemas multiagentes cognitivos**. 2006. Tese (Doutorado em Engenharia Elétrica). Universidade Federal de Santa Catarina, Florianópolis, SC.

HIKICHI, M.; FURUTA, Y.; WADA, R.; TADA, Y.; BANNDOU, S.; YAMAGUCHI, K. 2D Soccer Simulation League Team Description Ri-one. In: ROBOCUP INTERNATIONAL SYMPOSIUM, 18., 2014. **Anais...** [S.l.: s.n.], 2014. Disponível em: <http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/>, Acesso: em 10/01/15.

HORLING, B.; LESSER, V. R. A survey of multi-agent organizational paradigms. **Knowledge Eng. Review**, [S.l.], v.19, n.4, p.281–316, 2004.

HOUHAMDI, Z. Multi-Agent System Testing: a survey. **International Journal of Advanced Computer Science and Applications(IJACSA)**, [S.l.], v.2, n.6, 2011.

IGLESIAS, J. A.; LEDEZMA, A.; SANCHIS, A. The RoboCup Agent Behavior Modeling Challenge. In: XI WORKSHOP OF PHYSICAL AGENTS 2010 (WAF-2010), 2010. **Proceedings...** [S.l.: s.n.], 2010. p.179–186.

IGLESIAS, J. A.; LEDEZMA, A.; SANCHIS, A. Caos coach 2006 simulation team: an opponent modelling approach. **Computing and Informatics**, [S.l.], v.28, n.1, p.57–80, 2012.

KAMINKA, G.; FIDANBOYLU, M.; CHANG, A.; VELOSO, M. Learning the Sequential Coordinated Behavior of Teams from Observations. In: KAMINKA, G.; LIMA, P.; ROJAS, R. (Ed.). **RoboCup 2002**: robot soccer world cup vi. [S.l.]: Springer Berlin Heidelberg, 2003. p.111–125. (Lecture Notes in Computer Science, v.2752).

KARIMI, M.; AHMAZADEH, M. Mining RoboCup Log Files to Predict Own and Opponent Action. **International Journal of Advanced Research in Computer Science**, [S.l.], v.5, n.6, 2014.

KLEINER, A.; FARINELLI, A.; RAMCHURN, S.; SHI, B.; MAFFIOLETTI, F.; REFFATO, R. RMASBench: benchmarking dynamic multi-agent coordination in urban search and rescue (extended abstract). In: AUTONOMOUS AGENTS AND MULTI-AGENT SYSTEMS, 2013., 2013, Richland, SC. **Proceedings...** International Foundation for Autonomous Agents and Multiagent Systems, 2013. p.1195–1196. (AAMAS '13).

KORSAH, G. A.; STENTZ, A.; DIAS, M. B. A comprehensive taxonomy for multi-robot task allocation. **I. J. Robotic Res.**, [S.l.], v.32, n.12, p.1495–1512, 2013. Disponível em: <http://dx.doi.org/10.1177/0278364913496484>, Acesso: em 02/10/13.

KUHLMANN, G.; KNOX, W. B.; STONE, P. Know Thine Enemy: a champion robocup coach agent. In: TWENTY-FIRST NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2006. **Proceedings...** [S.l.: s.n.], 2006. p.1463–68.

LEITÃO, P.; INDEN, U.; RÜCKEMANN, C.-P. Parallelising multi-agent systems for high performance computing. In: THE THIRD INTERNATIONAL CONFERENCE ON ADVANCED COMMUNICATIONS AND COMPUTATION (INFOCOMP'13), 2013. **Anais...** IARIA, 2013. Disponível em: <http://hdl.handle.net/10198/9655>, Acesso: em 20/04/14.

LUCK, M. **50 facts about Agent-Based Computing**. University of Southampton on behalf of AgentLink III. AgentLink, 2005. Disponível em: <www.agentlink.org/roadmap>, Acesso: em 05/04/14.

LUCK, M.; MCBURNEY, P.; SHEHORY, O.; WILLMOTT, S. **Agent Technology**: computing as interaction (a roadmap for agent based computing). [S.l.]: AgentLink, 2005.

MALMIR, M.; BOLUKI, S.; KAMALI, M.; BABABEIG, M.; YEGANEJOU, M. AUT Parsian Team Description Paper 2014. In: ROBOCUP INTERNATIONAL SYMPOSIUM, 18., 2014. **Anais...** [S.l.: s.n.], 2014. Disponível em: <http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/>, Acesso: em 10/01/15.

MARIAN, S.; LUCA, D.; SARAC, B.; COTARLEA, O. OXSY 2014 Team Description. In: ROBOCUP INTERNATIONAL SYMPOSIUM, 18., 2014. **Anais...** [S.l.: s.n.], 2014. Disponível em: <http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/>, Acesso: em 10/01/15.

MILES, S.; WINIKOFF, M.; CRANEFIELD, S.; NGUYEN, C. D.; PERINI, A.; TONELLA, P.; HARMAN, M.; LUCK, M. Why testing autonomous agents is hard and what can be done about it. In: AOSE TECHNICAL FORUM, 2010. **Anais...** [S.l.: s.n.], 2010. Disponível em: <http://www.pa.icar.cnr.it/cossentino/AOSETF10/docs/miles.pdf>, Acesso: em 02/04/14.

MOHAMADI, S. E.; PASHAEE, A. A.; KAVIANI, P.; FOROOTANNEZHAD, A. M. HERMES Soccer 2D Simulation Team Description Paper. In: ROBOCUP INTERNATIONAL SYMPOSIUM, 18., 2014. **Anais...** [S.l.: s.n.], 2014. Disponível em: <http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/>, Acesso: em 10/01/15.

NEHMZOW, U. **Mobile robotics - a practical introduction (2. ed.).** [S.l.]: Springer, 2003. I-XVI, 1-280p.

NGUYEN, C. D.; PERINI, A.; TONELLA, P.; MILES, S.; HARMAN, M.; LUCK, M. Evolutionary testing of autonomous software agents. In: AAMAS (1), 2009. **Anais...** IFAAMAS, 2009. p.521–528. Disponível em: <http://dblp.uni-trier.de/db/conf/atal/aamas2009-1.html#NguyenPTMHL09>, Acesso: em 15/04/14.

ODELL, J. **Agent Technology**: what is it and why do we care? Enterprise Architecture, executive report. Cutter Consortium, Arlington, MA, v.10, n.3, p.1-25, 2007. Disponível em: <http://www.jamesodell.com/publications.html#presentations>, Acesso: em 02/03/14.

OTTONI, A. L. C.; JUNIOR, H. M. R.; MACHADO, I. G.; CORDEIRO, L. T.; NEPOMUCENO, E. G.; PEREIRA, E. B.; SILVA, R. I. d.; OLIVEIRA, M. S. d.; VASCONCELOS, L. O. R.; LAMOUNIER, A. M.; LOBO, F.; NOMIYA, F. M.; NETO, F. A. R.; ROCHA, J. G. UFSJ2D (UaiSoccer2D + RoboCap): team description paper robocup 2014. In: ROBOCUP INTERNATIONAL SYMPOSIUM, 18., 2014. **Anais...** [S.l.: s.n.], 2014. Disponível em: <http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/>, Acesso: em 13/01/15.

PARKER, L. E. Multiple Mobile Robot Systems. In: SICILIANO, B.; KHATIB, O. (Ed.). **Springer Handbook of Robotics**. [S.l.]: Springer, 2008. p.921–941. Disponível em: <http://dx.doi.org/10.1007/978-3-540-30301-5_41>, Acesso: em 25/9/13.

PERIN, C.; VUILLEMOT, R.; FEKETE, J.-D. SoccerStories: a kick-off for visual soccer analysis. **Visualization and Computer Graphics, IEEE Transactions on**, [S.l.], v.19, n.12, p.2506–2515, Dec 2013.

PETER, S. **Layered learning in multi-agent systems**. 1998. Tese (Doutorado em Ciência da Computação) — School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

PROKOPENKO, M.; WANG, P.; OBST, O. Gliders2014: dynamic tactics with voronoi diagrams. In: ROBOCUP INTERNATIONAL SYMPOSIUM, 18., 2014. **Anais...** [S.l.: s.n.], 2014. Disponível em: <http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/>, Acesso: em 10/01/15.

READING, U. of. **Press Realeases**. <http://www.reading.ac.uk/news-and-events/releases/PR583836.aspx>, 2014. Home Page.

REIS, L. P. **Coordenação em Sistemas Multi-Agente**: aplicações na gestão universitária e futebol robótico. 2003. Tese (Doutorado em Ciência da Computação) — FEUP, Estrada da Circunvalação (EN12).

RILEY, P.; STONE, P.; VELOSO, M. Layered Disclosure: revealing agents' internals. In: CASTELFRANCHI, C.; LESPÉRANCE, Y. (Ed.). **Intelligent Agents VII. Agent Theories, Architectures, and Languages — 7th. International Workshop, ATAL-2000, Boston, MA, USA, July 7–9, 2000, Proceedings**. Berlin: Springer, 2001. n.1986, p.61–72. (Lecture Notes in Artificial Intelligence).

ROBOCUP. **The RoboCup Federation**. <http://www.robocup.org/>, 1998-2014. Home Page.

ROBOCUP. **RoboCup Federation Wiki**. <http://wiki.robocup.org/wiki/Main_Page>, 2014. Home Page.

ROSA, A. M. da; GULARTE, A.; JUNG, M.; GONÇALVES, E. M. Two different perspectives about how to specify and implement multiagent systems. In: VII WORKSHOP-ESCOLA DE SISTEMAS DE AGENTES, SEUS AMBIENTES E APLICAçõES, 2013. **Anais...** [S.l.: s.n.], 2013. p.141–144.

RUSSELL, S.; NORVIG, P. Inteligência Artificial. **Editora Campus**, [S.l.], 2004.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robots**. Scituate, MA, USA: Bradford Company, 2004.

SIVAKUMAR, V. K. Agent Oriented Software Testing – Role Oriented approach. **International Journal of Advanced Computer Science and Applications(IJACSA)**, [S.l.], v.3, n.12, 2012.

SUCH, J. M.; ALBEROLA, J. M.; MULET, L.; ESPINOSA, A.; GARCIA-FORNES, A.; BOTTI, V. Large-scale multiagent platform benchmarks. In: LANGUAGES, METHODOLOGIES AND DEVELOPMENT TOOLS FOR MULTI-AGENT SYSTEMS (LADS 2007). PROCEEDINGS OF THE MULTI-AGENT LOGICS, LANGUAGES, AND ORGANISATIONS-FEDERATED WORKSHOPS, 2007. **Anais...** [S.l.: s.n.], 2007. p.192–204.

TOVINKERE, V.; QIAN, R. Detecting semantic events in soccer games: towards a complete solution. In: MULTIMEDIA AND EXPO, 2001. ICME 2001. IEEE INTERNATIONAL CONFERENCE ON, 2001. **Anais...** [S.l.: s.n.], 2001. p.833–836.

VELASHANI, H. A.; YAZDANMEHR, N.; MAKOU, A. B.; RAJAEI, H. Enigma 2014 Team Description Paper. In: ROBOCUP INTERNATIONAL SYMPOSIUM, 18., 2014. **Anais...** [S.l.: s.n.], 2014. Disponível em: <http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/>, Acesso: em 10/01/15.

VIDAL, J. M. **Fundamentals of Multiagent Systems with NetLogo Examples**. [S.l.]: Unpublished, 2010. Disponível em: <http://multiagent.com/p/fundamentals-of-multiagent-systems.html> , Acesso: em 23/05/14.

WEYNS, D.; OMICINI, A.; ODELL, J. Environment As a First Class Abstraction in Multiagent Systems. **Autonomous Agents and Multi-Agent Systems**, Hingham, MA, USA, v.14, n.1, p.5–30, Feb. 2007.

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems**. 2nd.ed. [S.l.]: Wiley Publishing, 2009.

WOOLDRIDGE, M.; CIANCARINI, P. Agent-oriented Software Engineering: the state of the art. In: FIRST INTERNATIONAL WORKSHOP, AOSE 2000 ON AGENT-

ORIENTED SOFTWARE ENGINEERING, 2001, Secaucus, NJ, USA. **Anais...** Springer-Verlag New York: Inc., 2001. p.1–28.

YASUDA, G.; TACHIBANA, K. A Parallel Distributed Control Architecture for Multiple Robot Systems Using a Network of Microcomputers. **Computers and Industrial Engineering**, Tarrytown, NY, USA, v.27, n.1-4, p.63–66, Sept. 1994.

ZAREIAN, A.; TABARI, E.; SAMIMI, R.; HEDAYATI, A.; SHIVA, F. A.; KAZEMI, V. **Team Assistant 2003.** SBCE's game presentation and analysis, 2003. Disponível em: <http://sourceforge.net/projects/team-assistant/files/TeamAssistant%202003%20-%202D%20Soccer/Manual/>, Acesso: em 11/09/13.

ZHANG, H.; LU, G.; CHEN, R.; LI, X.; CHEN, X. WrightEagle 2D Soccer Simulation Team Description 2014. In: ROBOCUP INTERNATIONAL SYMPOSIUM, 18., 2014. **Anais...** [S.l.: s.n.], 2014. Disponível em: <http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/>, Acesso: em 12/01/15.

ZLOT, R. M. **An Auction-Based Approach to Complex Task Allocation for Multirobot Teams**. 2006. Tese (Doutorado em Ciência da Computação) — Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave. (CMU-RI-TR-06-52).

APÊNDICE A ALGORITMOS IMPLEMENTADOS

Este apêndice contém o código em *python* responsável pela identificação de padrões de tabelas e triangulações desempenhadas pelos agentes projetados para o *Soccer Server 2D*.

Lista A.1: Funções de identificação das coalizões no código Statistics.py.

```
import re
import os
import abc
import xml.dom.minidom as minidom

teamnamespattern = re.compile(r'^\d+-(.*)(?:_\d+)-vs-(.*)(?:_\d+)(?:_convert(?:ido)?)?\.
    rcg\.gz\.xml$')
yearpattern = re.compile(r'\d{4}')
onetwo_List = []
triangulation_List = []
indirect_events_list = []

def closedpasschains(side, dom):
"""
        Function that computes patterns related with passes, such as one-two chains,
            triangulations chains or circular pass chains. For each pattern it's
            verified whether it perceives certains events like, goal, goals miss, goal
            opportunity, pass miss and others.
"""
    teamid = side_identifier(side)
    chainprefix1 = "closed_passchain_ONE_TWO"
    chainprefix2 = "closed_passchain_TRIANGULATION"
    res = {"closed_passchain_ONE_TWO": 0, "closed_passchain_TRIANGULATION": 0}
    i = 0
    cadeia = 0

# Vectors to store the necessary data to identify the pass chains
    kicker = []
    receiver = []
    passe = []
    t_opponentkick = []

# Getting the necessary data and keeping it in the vectors
    dom_passchains = dom.getElementsByTagName("passes")[0]
    for passes in dom_passchains.getElementsByTagName("pass"):
        if passes.getAttribute("team") == teamid:
```

```
            kicker.append(Player(passes.getElementsByTagName("kick")[0].getAttribute("
                player"),
                                 passes.getElementsByTagName("kick")[0].getAttribute("
                                     time"),
                                 passes.getElementsByTagName("kick")[0].getAttribute("
                                     positionX"),
                                 passes.getElementsByTagName("kick")[0].getAttribute("
                                     positionY"),
                                 passes.getElementsByTagName("kick")[0].getAttribute("
                                     zone")))
            receiver.append(Player(passes.getElementsByTagName("reception")[0].
                getAttribute("player"),
                                 passes.getElementsByTagName("reception")[0].
                                     getAttribute("time"),
                                 passes.getElementsByTagName("reception")[0].
                                     getAttribute("positionX"),
                                 passes.getElementsByTagName("reception")[0].
                                     getAttribute("positionY"),
                                 passes.getElementsByTagName("reception")[0].
                                     getAttribute("zone")))
            passe.append(Pass(kicker[i], receiver[i]))
            i += 1
        else:
            t_opponentkick.append(passes.getElementsByTagName("kick")[0].getAttribute("
                time"))

# Starting the pass chains and others patterns recognition
    for i in range(0, len(kicker) - 2):

        if receiver[i + 1].unum == kicker[i].unum and receiver[i].unum == kicker[i + 1].
            unum and not validatepasschain(teamid, dom, receiver[i].time, kicker[i + 1].
            time, t_opponentkick):
            res[chainprefix1] += 1
            pass_chain_patterns(teamid, dom, passe[i + 1], passe, t_opponentkick, (i +
                1),
            str(':' + receiver[i].unum + ':' + receiver[i+1].unum + ':'), 'onetwo')
            onetwo_List.append(Onetwo(teamid, passe[i], passe[i + 1],
                calculate_field_move(teamid, kicker[i], receiver[i + 1])))

        if receiver[i + 2].unum == kicker[i].unum and kicker[i + 2].unum == receiver[i +
             1].unum
        and kicker[i + 1].unum == receiver[i].unum and kicker[i + 2].unum != receiver[i
            ].unum and not validatepasschain(teamid, dom, receiver[i].time, kicker[i +
            2].time, t_opponentkick):
            res[chainprefix2] += 1
            pass_chain_patterns(teamid, dom, passe[i + 2], passe, t_opponentkick, (i +
                2),
            str(':' + receiver[i].unum + ':' + receiver[i+1].unum + ':' + receiver[i+2].
                unum + ':'), 'triangulation')
            triangulation_List.append(Triangulation(teamid, passe[i], passe[i + 1],
                passe[i + 2], calculate_field_move(teamid, kicker[i], receiver[i + 2])))

    print "\nTotal One-two ", res[chainprefix1]
    print "Total Triangulation", res[chainprefix2]
    return res.items()

def validatepasschain(teamid, dom, lowerlimit, upperlimit, t_opponentkick):
"""
```

```
        Function that verifies whether on a supose pass chain there was: corners kicks,
            Kickin, pass misses, faults or opose team pass.
"""
    chain_stopped = False
    dom_corners = dom.getElementsByTagName("corners")[0]
    dom_kicksins = dom.getElementsByTagName("kicksin")[0]
    dom_freekicks = dom.getElementsByTagName("freekicks")[0]
    dom_foulcharges = dom.getElementsByTagName("foulcharges")[0]
    dom_freekickfaults = dom.getElementsByTagName("freekickfaults")[0]
    dom_passmisses = dom.getElementsByTagName("passmisses")[0]

    for i in range(0, len(t_opponentkick)):
        if int(lowerlimit) < int(t_opponentkick[i]) < int(upperlimit):
            chain_stopped = True
            break
    for foulcharges in dom_foulcharges.getElementsByTagName("foulcharge"):
        if int(lowerlimit) < int(foulcharges.getAttribute("time")) < int(upperlimit):
            chain_stopped = True
            break
    for corners in dom_corners.getElementsByTagName("corner"):
        if int(lowerlimit) < int(corners.getAttribute("time")) < int(upperlimit):
            chain_stopped = True
            break
    for kicksins in dom_kicksins.getElementsByTagName("kickin"):
        if int(lowerlimit) < int(kicksins.getAttribute("time")) < int(upperlimit):
            chain_stopped = True
            break
    for freekicks in dom_freekicks.getElementsByTagName("frekick"):
        if int(lowerlimit) < int(freekicks.getAttribute("time")) < int(upperlimit):
            chain_stopped = True
            break
    for freekickfaults in dom_freekickfaults.getElementsByTagName("freekickfault"):
        if int(lowerlimit) < int(freekickfaults.getAttribute("time")) < int(upperlimit):
            chain_stopped = True
            break
    for passmisses in dom_passmisses.getElementsByTagName("passmiss"):
        if int(lowerlimit) < int(passmisses.getElementsByTagName("kick")[0].getAttribute
            ("time")) < int(upperlimit) and passmisses.getAttribute("team") == teamid:
            chain_stopped = True
            break
    return chain_stopped

def calculate_field_move(teamid, kicker, receiver):
"""
        Simple function to analize if there is a gain of movement against opose team
            field, in others words, the principal goal oriented objective of soccer.
"""
    if teamid == "LEFT_SIDE":
        if float(kicker.getposx()) > float(receiver.getposx()):
            return False
        else:
            return True
    else:
        if float(kicker.getposx()) > float(receiver.getposx()):
            return True
        else:
            return False

def reincidentcoalition(passes, i, time, pattern):
```

```
"""
        Function that verifies if the events of a phase (ball possession interval) was
            already related with patterns (onetwo or triangulation) that repeated in the
             same phase.
"""
    if pattern == 'onetwo':
        for item in indirect_events_list:
            if item.find(':00:'):
                if (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':01:'):
                if (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':02:'):
                if (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':03:'):
                if (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':04:'):
                if (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':05:'):
                if (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':06:'):
                if (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
    if pattern == 'triangulation':
        for item in indirect_events_list:
            if item.find(':00:'):
                if (item.find(':' + str(passes[i-2].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1)
                            \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':01:'):
                if (item.find(':' + str(passes[i-2].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1)
                            \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':02:'):
```

```
                if (item.find(':' + str(passes[i-2].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1)
                            \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':03:'):
                if (item.find(':' + str(passes[i-2].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1)
                            \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':04:'):
                if (item.find(':' + str(passes[i-2].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1)
                            \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':05:'):
                if (item.find(':' + str(passes[i-2].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1)
                            \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
            if item.find(':06:'):
                if (item.find(':' + str(passes[i-2].sender.unum) + ':') != -1) \
                        and (item.find(':' + str(passes[i-1].sender.unum) + ':') != -1)
                            \
                        and (item.find(':' + str(passes[i].sender.unum) + ':') != -1) \
                        and item.find(':' + time + ':') != -1:
                    return True
    return False

def verifycrossing(teamid, j, passe, passes):
        if teamid == "LEFT_SIDE":
            if (passes[j].sender.zone == "TopRightright" and passes[j].receiver.zone ==
                "MiddleRightright")
            or (passes[j].sender.zone == "BottomRightright" and passes[j].receiver.zone
                == "MiddleRightright"):
                passe.receiver.setwingchange("indirect")
        if teamid == "RIGHT_SIDE":
            if (passes[j].sender.zone == "TopLeftleft" and passes[j].receiver.zone == "
                MiddleLeftleft")
            or (passes[j].sender.zone == "BottomLeftleft" and passes[j].receiver.zone ==
                 "MiddleLeftleft"):
                passe.receiver.setwingchange("indirect")

def verifypassmiss(teamid, passe, passes, dom_passmisses, i, j, item, pattern,
    opponentkicktime):

        for passmisses in dom_passmisses.getElementsByTagName("passmiss"):
            if int(passmisses.getElementsByTagName("kick")[0].getAttribute("time")) >
                opponentkicktime and opponentkicktime != 0:
                return 0
            if int(passe.receiver.time) < int(passmisses.getElementsByTagName("kick")
                [0].getAttribute("time")) < int(passes[j].sender.time) and passmisses.
```

```
                getAttribute("team") == teamid:
                if int(passe.receiver.unum) == int(passmisses.getElementsByTagName("kick
                    ")[0].getAttribute("player") and passe.receiver == passes[j - 1].
                    receiver):
                    passe.receiver.setpassmiss("True")
                    return int(passmisses.getElementsByTagName("kick")[0].getAttribute("
                        time"))
                if reincidentcoalition(passes, i, passmisses.getElementsByTagName("kick"
                    )[0].getAttribute("time"), pattern):
                    passe.receiver.setpassmiss("samephase")
                    return int(passmisses.getElementsByTagName("kick")[0].getAttribute("
                        time"))
                passe.receiver.setpassmiss("indirect")
                indirect_events_list.append(item + '04:' + passmisses.
                    getElementsByTagName("kick")[0].getAttribute("time") + ':')
                return int(passmisses.getElementsByTagName("kick")[0].getAttribute("time
                    "))
        return 0


def verifymatchecharges(passe, passes, dom_corners, dom_kicksins, i, j, item, pattern):

        for corners in dom_corners.getElementsByTagName("corner"):
            if int(passe.receiver.time) < int(corners.getAttribute("time")) < int(passes
                [j].sender.time):
                if reincidentcoalition(passes, i, corners.getAttribute("time"), pattern)
                    :
                    passe.receiver.setcorner("samephase")
                    return True
                passe.receiver.setcorner("True")
                indirect_events_list.append(item + '03:' + corners.getAttribute("time")
                    + ':')
                return True

        for kicksins in dom_kicksins.getElementsByTagName("kickin"):
            if int(passe.receiver.time) < int(kicksins.getAttribute("time")) < int(
                passes[j].sender.time):
                return True
        return False


def verifyfault(teamid, passe, passes, dom_foulcharges, dom_offsides, i, j, item,
    pattern, passmisstime):

        for foulcharges in dom_foulcharges.getElementsByTagName("foulcharge"):
            if int(foulcharges.getAttribute("time")) > passmisstime and passmisstime
                !=0:
                return 0
            if int(passe.receiver.time) < int(foulcharges.getAttribute("time")) < int(
                passes[j].sender.time) and passe.receiver == passes[j - 1].receiver:
                if foulcharges.getAttribute("team") == teamid:
                    passe.receiver.setfoulcharge("done")
                    return int(foulcharges.getAttribute("time"))
                else:
                    passe.receiver.setfoulcharge("taked")
                    return int(foulcharges.getAttribute("time"))
            if int(passe.receiver.time) < int(foulcharges.getAttribute("time")) < int(
                passes[j].sender.time):
                if foulcharges.getAttribute("team") == teamid:
```

```
                    if reincidentcoalition(passes, i, foulcharges.getAttribute("time"),
                        pattern):
                        passe.receiver.setfoulcharge("indirect_done_samephase")
                        return foulcharges.getAttribute("time")
                    passe.receiver.setfoulcharge("indirect_done")
                    indirect_events_list.append(item + '06:' + foulcharges.getAttribute(
                        "time") + ':')
                    return foulcharges.getAttribute("time")
                else:
                    if reincidentcoalition(passes, i, foulcharges.getAttribute("time"),
                        pattern):
                        passe.receiver.setfoulcharge("indirect_taked_samephase")
                        return int(foulcharges.getAttribute("time"))
                    passe.receiver.setfoulcharge("indirect_taked")
                    indirect_events_list.append(item + '06:' + foulcharges.getAttribute(
                        "time") + ':')
                    return int(foulcharges.getAttribute("time"))

        for offsides in dom_offsides.getElementsByTagName("corner"):
            if int(passe.receiver.time) < int(offsides.getAttribute("time")) < int(
                passes[j].sender.time) and passe.receiver == passes[j - 1].receiver:
                if offsides.getAttribute("team") == teamid:
                    passe.receiver.setoffside("true")
                    return int(offsides.getAttribute("time"))
            if int(passe.receiver.time) < int(offsides.getAttribute("time")) < int(
                passes[j].sender.time):
                if offsides.getAttribute("team") == teamid:
                    if reincidentcoalition(passes, i, offsides.getAttribute("time"),
                        pattern):
                        passe.receiver.setoffside("samephase")
                        return int(offsides.getAttribute("time"))
                    passe.receiver.setoffside("indirect")
                    indirect_events_list.append(item + '05:' + offsides.getAttribute("
                        time") + ':')
                    return int(offsides.getAttribute("time"))
        return 0

def verifypossession(passe, passes, t_opponentkick, j):

    for z in range(0, len(t_opponentkick)):
        if int(passe.receiver.time) < int(t_opponentkick[z]) < int(passes[j].sender.time
            ):
            return int(t_opponentkick[z])
    return 0

def pass_chain_patterns(teamid, dom, passe, passes, t_opponentkick, i, item, pattern):
"""
        Function that computes events related with one-two chains and triangulations
            chains. After the chain was completed it is verified whether - direct or
            indirect - events take place. The events are: goal, goal miss, goal
            opportunity, pass miss, fault, offside, corner, kickin and crossing.
"""
    chain_stopped = False
    dom_goals = dom.getElementsByTagName("goals")[0]
    dom_goalmisses = dom.getElementsByTagName("goalmisses")[0]
    dom_goalopportunities = dom.getElementsByTagName("goalopportunities")[0]
    dom_corners = dom.getElementsByTagName("corners")[0]
    dom_kicksins = dom.getElementsByTagName("kickin")[0]
    dom_offsides = dom.getElementsByTagName("offsides")[0]
```

```
dom_foulcharges = dom.getElementsByTagName("foulcharges")[0]
dom_passmisses = dom.getElementsByTagName("passmisses")[0]

for j in range(i+1, len(passes)):

    if chain_stopped:
        break
    opponentkicktime = verifypossession(passe, passes, t_opponentkick, j)
    if opponentkicktime > 0:
        chain_stopped = True
    passmisstime = verifypassmiss(teamid, passe, passes, dom_passmisses, i, j, item,
         pattern, opponentkicktime)
    if passmisstime > 0:
        chain_stopped = True
    faulttime = verifyfault(teamid, passe, passes, dom_foulcharges, dom_offsides, i,
         j, item, pattern, passmisstime)
    if faulttime > 0:
        chain_stopped = True
    if verifymatchecharges(passe, passes, dom_corners, dom_kicksins, i, j, item,
        pattern):
        break
    if not chain_stopped:
        verifycrossing(teamid, j, passe, passes)

    for goalopportunities in dom_goalopportunities.getElementsByTagName("opportunity
        "):
        if int(passe.receiver.time) <= int(goalopportunities.getAttribute("start"))
            < int(passes[j].sender.time) and goalopportunities.getAttribute("team")
            == teamid:
            if int(passe.receiver.time) < faulttime < int(goalopportunities.
                getAttribute("start")) or int(passe.receiver.time) <
                opponentkicktime < int(goalopportunities.getAttribute("start")) or
                int(passe.receiver.time) < passmisstime < int(goalopportunities.
                getAttribute("start")):
                break
            if int(passe.receiver.unum) == int(goalopportunities.getAttribute("
                player")) and passe.receiver == passes[j - 1].receiver:
                passe.receiver.setgoalopportunity("True")
                break
            if reincidentcoalition(passes, i, goalopportunities.getAttribute("start"
                ), pattern):
                passe.receiver.setgoalopportunity("samephase")
                break
            passe.receiver.setgoalopportunity("indirect")
            indirect_events_list.append(item + '02:' + goalopportunities.
                getAttribute("start") + ':')
            chain_stopped = True
            break
    for goalmiss in dom_goalmisses.getElementsByTagName("goalmiss"):
        if int(passe.receiver.time) <= int(goalmiss.getAttribute("time")) < int(
            passes[j].sender.time) and goalmiss.getAttribute("team") == teamid:
            if int(passe.receiver.time) < faulttime < int(goalmiss.getAttribute("
                time")) or int(passe.receiver.time) < opponentkicktime < int(
                goalmiss.getAttribute("time")) or int(passe.receiver.time) <
                passmisstime < int(goalmiss.getAttribute("time")):
                break
            if int(passe.receiver.unum) == int(goalmiss.getElementsByTagName("kick")
                [0].getAttribute("player")
```

```
                                                and passe.receiver == passes[j - 1].
                                                    receiver):
                        passe.receiver.setgoalMiss("True")
                        chain_stopped = True
                        break
                    if reincidentcoalition(passes, i, goalmiss.getAttribute("time"), pattern
                        ):
                        passe.receiver.setgoalmiss("samephase")
                        break
                    passe.receiver.setgoalmiss("indirect")
                    indirect_events_list.append(item + '01:' + goalmiss.getAttribute("time")
                         + ':')
                    chain_stopped = True
                    break
            for goal in dom_goals.getElementsByTagName("goal"):
                if int(passe.receiver.time) <= int(goal.getAttribute("time")) < int(passes[j
                    ].sender.time) and goal.getAttribute("team") == teamid:
                    if int(passe.receiver.time) < faulttime < int(goal.getAttribute("time"))
                         or int(passe.receiver.time) < opponentkicktime < int(goal.
                        getAttribute("time")) or int(passe.receiver.time) < passmisstime <
                        int(goal.getAttribute("time")):
                        break
                    if int(passe.receiver.unum) == int(goal.getElementsByTagName("kick")[0].
                        getAttribute("player")
                                                and passe.receiver == passes[j - 1].
                                                    receiver):
                        passe.receiver.setgoal("True")
                        chain_stopped = True
                        break
                    if reincidentcoalition(passes, i, goal.getAttribute("time"), pattern):
                        passe.receiver.setgoal("samephase")
                        break
                    passe.receiver.setgoal("indirect")
                    indirect_events_list.append(item + '00:' + goal.getAttribute("time") + '
                        :')
                    chain_stopped = True
                    break

def sum_ocurrence_events(event, lista):
""" Function to compute the points of each coalition (onetwo or triangulation). """
    event['total'] += 1

    if lista.getinfront():
        event['avanco'] += 1
        event['pontos'] += 0.5
    else:
        event['recuo'] += 1
        event['pontos'] += 0
    if lista.end.getreceiver().getgoal() == "True":
        event['goal'] += 1
        event['pontos'] += 7
    if lista.end.getreceiver().getgoal() == "indirect":
        event['goal_ind'] += 1
        event['pontos'] += 2
    if lista.end.getreceiver().getgoal() == "samephase":
        event['pontos'] += 1
    if lista.end.getreceiver().getgoalopportunity() == "True":
        event['goalopport'] += 1
        event['pontos'] += 1
```

```
    if lista.end.getreceiver().getgoalopportunity() == "indirect":
        event['goalopport_ind'] += 1
        event['pontos'] += 0.3
    if lista.end.getreceiver().getgoalopportunity() == "samephase":
        event['pontos'] += 0.15
    if lista.end.getreceiver().getgoalmiss() == "True":
        event['goalmiss'] += 1
        event['pontos'] += 1.5
    if lista.end.getreceiver().getgoalmiss() == "indirect":
        event['goalmiss_ind'] += 1
        event['pontos'] += 0.6
    if lista.end.getreceiver().getgoalmiss() == "samephase":
        event['pontos'] += 0.3
    if lista.end.getreceiver().getpassmiss() == "True":
        event['passmiss'] += 1
        event['pontos'] += -1
    if lista.end.getreceiver().getpassmiss() == "indirect":
        event['passmiss_ind'] += 1
        event['pontos'] += -0.3
    if lista.end.getreceiver().getpassmiss() == "samephase":
        event['pontos'] += -0.15
    if lista.end.getreceiver().getcorner() == "True":
        event['corner'] += 1
        event['pontos'] += 0.2
    if lista.end.getreceiver().getwingchange() == "indirect":
        event['wingchange_ind'] += 1
        event['pontos'] += 0.5
    if lista.end.getreceiver().getoffside() == "True":
        event['offside'] += 1
        event['pontos'] += -0.3
    if lista.end.getreceiver().getoffside() == "indirect":
        event['offside'] += 1
        event['pontos'] += -0.1
    if lista.end.getreceiver().getfoulcharge() == "done":
        event['foulcharge'] += 1
        event['pontos'] += -0.5
    if lista.end.getreceiver().getfoulcharge() == "taked":
        event['foulcharge'] += 1
        event['pontos'] += 0.7
    if lista.end.getreceiver().getfoulcharge() == "indirect_done":
        event['foulcharge'] += 1
        event['pontos'] += -0.1
    if lista.end.getreceiver().getfoulcharge() == "indirect_taked":
        event['foulcharge'] += 1
        event['pontos'] += 0.2

def verify_current_ocurrences(lista, pattern, coalitions):
"""
        This function verifies whether a pattern (one-two or triangulation) it already
            exists.
    :rtype : returns the string 'false' in case of negative (pattern doesn't exist) or
        the string identifying the pattern in case of positive.
"""
    if pattern == "one-two":
        for item in coalitions:
            if (item.find(':' + str(lista.start.getsender().getunum()) + ':') != -1)\
                    and (item.find(':' + str(lista.end.getsender().getunum()) + ':') !=
                        -1):
                return item
```

```
    if pattern == "triangulation":
        for item in coalitions:
            if (item.find(':' + str(lista.start.getsender().getunum()) + ':') != -1)\
                    and (item.find(':' + str(lista.middle.getsender().getunum()) + ':')
                        != -1)\
                    and (item.find(':' + str(lista.end.getsender().getunum()) + ':') !=
                        -1):
                return item
    return "false"

def set_ocurrence_events(lista):
    new_event = {"total": 0, "goal": 0, "goalopport": 0, "goalmiss": 0, "passmiss": 0, "
        avanco": 0,
                 "recuo": 0, "corner": 0, "goal_ind": 0, "goalopport_ind": 0, "
                     goalmiss_ind": 0,
                 "passmiss_ind": 0, "wingchange_ind": 0, "offside": 0, "foulcharge": 0,
                     "pontos": 0}
    sum_ocurrence_events(new_event, lista)
    return new_event

def count_patterns(side, pattern, todolista):
""" Function to verify and count the ocurrences of patterns between players: one-two and
     triangulation. """
    teamid = side_identifier(side)
    res = {}
    coalitions = []
    prefix1 = "Tabela:"
    prefix2 = "Triangulacao:"

#    print "\n-Side", side + "-"
    if pattern == "one-two":
        for lista in todolista:
            old_ocurrence = verify_current_ocurrences(lista, pattern, coalitions)
            if lista.side != teamid:
                continue
            if old_ocurrence != "false":
                sum_ocurrence_events(res[old_ocurrence], lista)
#                print "Summing up old one"
            else:
                res[prefix1 + str(lista.start.getsender().getunum()) + ":" + str(lista.
                    end.getsender().getunum()) + ":"] = set_ocurrence_events(lista)
                coalitions.append(prefix1 + str(lista.start.getsender().getunum()) + ":"
                     + str(lista.end.getsender().getunum()) + ":")
#                print "Had a new item"

    if pattern == "triangulation":
        for lista in todolista:
            old_ocurrence = verify_current_ocurrences(lista, pattern, coalitions)
            if lista.side != teamid:
                continue
            if old_ocurrence != "false":
                sum_ocurrence_events(res[old_ocurrence], lista)
#                print "Summing up old one"
            else:
                res[prefix2 + str(lista.start.getsender().getunum()) + ":" + str(lista.
                    middle.getsender().getunum()) + ":" + str(lista.end.getsender().
                    getunum()) + ":"] = set_ocurrence_events(lista)
                coalitions.append(prefix2 + str(lista.start.getsender().getunum()) + ":"
                     + str(lista.middle.getsender().getunum()) + ":" + str(lista.end.
```

```
                    getsender().getunum()) + ":")
#                 print "Had a new item"
    return res.items()

def matches_patterns_events():
    patterns = {}
    with open('filenames.txt') as fns:
        for fname in fns:
            print "processing {0}".format(fname.strip())
            fname = fname.strip()
            # allow comments...
            if fname.startswith("#"):
                continue

            dom = minidom.parse(fname)
            dirname = os.path.dirname(fname)
            bname = os.path.basename(fname)
            (year, tournamentatom) = tournamentinfo(dirname)
            (t1,t2) = teamnames(bname)
            print "\tyear:{0} atom:{1} t1:{2} t2:{3}".format(year, tournamentatom, t1,
                t2)

            patternsleft = []
            patternsright = []
            gdataleft = []
            gdataright = []
            patternsleft.extend(closedpasschains("left", dom))
            patternsright.extend(closedpasschains("right", dom))
            gdataleft.extend(count_patterns("left", "triangulation", triangulation_List)
                )
            gdataright.extend(count_patterns("right", "triangulation",
                triangulation_List))
            gdataleft.extend(count_patterns("left", "one-two", onetwo_List))
            gdataright.extend(count_patterns("right", "one-two", onetwo_List))

            gdataleft = mergepatterns(gdataleft)
            gdataright = mergepatterns(gdataright)
            updatepatterns(patterns, (t1, gdataleft), (t2, gdataright))
            print "processed {0}\n".format(fname)
    return patterns

def print_matches_patternscsv(fname, matchespatterns):

    eventnames = matchespatterns.values()[0].values()[0].keys()
    with open(fname,'w') as f:
        # table head
        f.write(";".join([""] + eventnames) + "\n")
        # table rows
        for team, patterns in matchespatterns.items():
            f.write(team + "\n")
            for pattern,events in patterns.items():
                f.write(pattern)
                for e in events:
                    f.write(";" + str(events[e]))
                f.write("\n")

if __name__ == '__main__':
    import sys
    if len(sys.argv) > 1:
```

```
        if sys.argv[1] == "teamnames":
            printteamnames()
            sys.exit(0)

# Compute patterns
    (patterns) = matches_patterns_events()

# print patterns and events
    print_matches_patternscsv("matchespatterns.csv", patterns)
```

APÊNDICE B RESULTADOS DE ALGUMAS BATERIAS DE TESTE

Neste apêndice, serão apresentados os dados gerais de algumas das baterias de teste.

Na Figura B.1 seguem as coalizões do time *Cyrus*. Estas formam um padrão no campo, no qual existem mais tabelas e triangulações pelo lado esquerdo. Neste lado, igualmente, se concentram os avanços no campo e os eventos positivos ligados às coalizões, variando entre oportunidades de gol, gols, gols perdidos e cruzamentos. É possível observar que, embora o time *Cyrus* possua um número elevado de oportunidade de gols, há uma quantidade pequena de conversões dessas oportunidades em gols de fato. Nota-se também, que houve bastantes gols perdidos.

Na Tabela B.1 (com dados do time *Osxy*), as ocorrências das jogadas comprovam que a maioria das coalizões se concentram em poucas combinações de jogadores. É possível observar que, embora houve cinquenta e duas combinações diferentes entre tabelas e triangulações, apenas onze combinações monopolizaram as ocorrências.

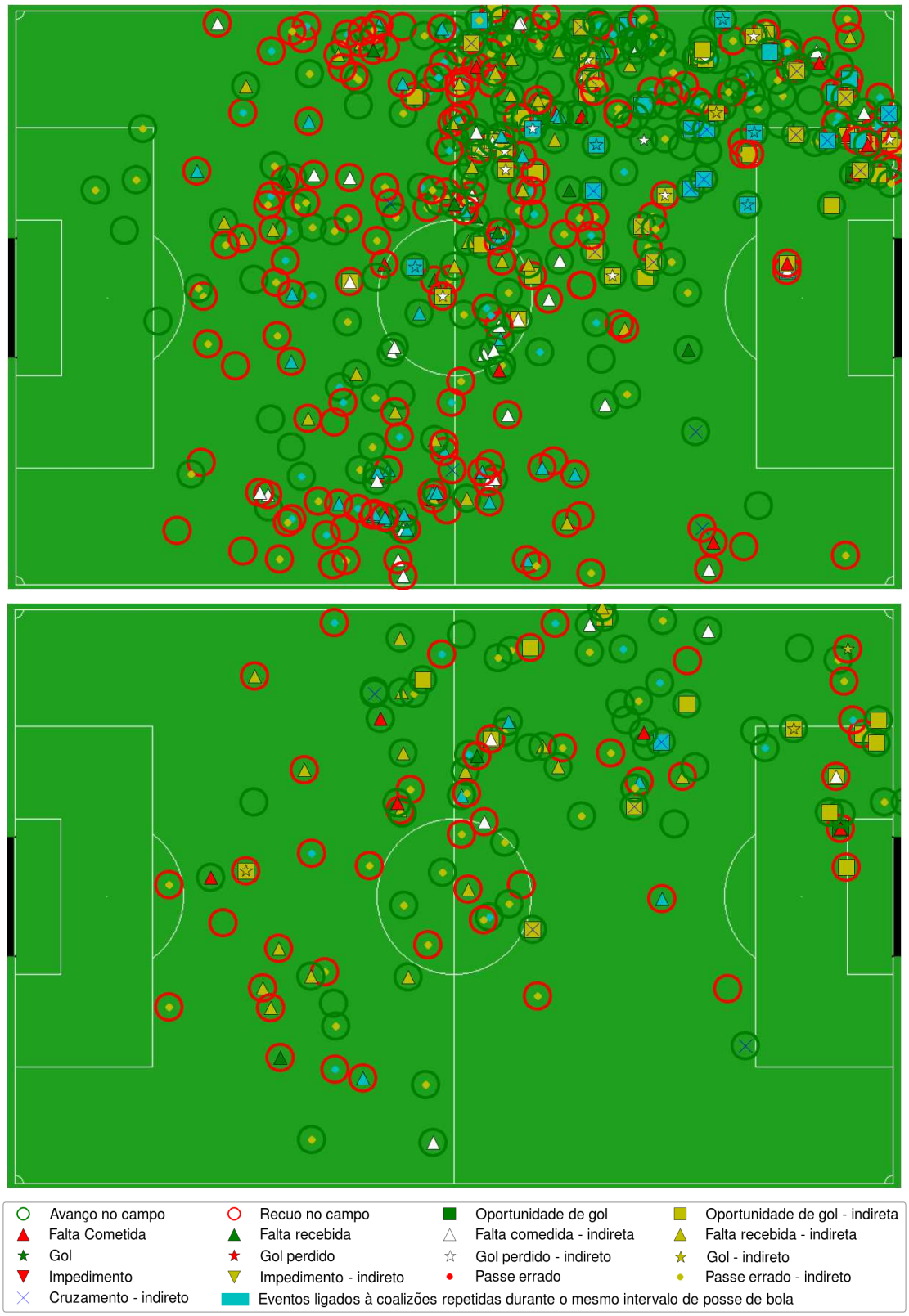


Figura B.1: Tabelas e triangulações do time *Cyrus* ocorridas durante sua bateria de teste.

Tabela B.1: Resultados gerais da bateria de teste do time *Osxy*.

Coalizões (jogadores)	Total	Avanços	Recuos	Pontos	Gols (ind.)	Gols Perdidos (ind.)	Oportunidades de Gol	Oportunidades de Gol (ind.)	Faltas Cometidas	Faltas Recebidas	Passes Errados (ind.)	Escanteios (ind.)	Cruzamentos (ind.)
10:6:11	1	1	0	2,8	1	0	0	1	0	0	0	0	0
4:5	1	0	1	-0,3	0	0	0	0	0	0	1	0	0
2:6:3	2	1	1	0,2	0	0	0	0	0	0	1	0	0
3:6:7	1	0	1	-0,3	0	0	0	0	0	0	1	0	0
7:10:8	5	4	1	3	0	2	0	0	0	1	2	0	1
8:10:6	3	3	0	3,4	1	0	0	1	0	0	2	0	1
7:6	28	21	7	23,75	4	2	0	4	3	1	7	2	2
10:6:9	1	1	0	0,5	0	0	0	0	0	0	0	0	0
7:11	8	5	3	5,95	1	1	0	1	1	1	1	0	3
2:9	4	3	1	1,2	0	0	0	1	0	0	2	0	0
7:8:11	5	4	1	1,75	0	0	0	0	0	0	1	0	1
6:9:8	1	0	1	3	1	0	1	0	0	0	0	0	0
9:6:7	3	2	1	1,5	0	1	0	1	1	0	1	0	0
6:8	2	1	1	1,05	0	0	0	0	0	2	0	0	0
7:8	10	6	4	13,5	3	0	0	2	0	2	1	0	1
9:7	9	8	1	9,8	1	1	0	1	3	0	0	1	4
9:7:8	1	1	0	0,5	0	0	0	1	0	0	1	0	0
2:7:9	1	1	0	0,2	0	0	0	0	0	0	1	0	0
10:11:5	2	2	0	5,3	1	0	1	1	0	0	0	0	0
8:10:11	2	1	1	0,4	0	0	0	0	0	0	1	1	0
10:9	38	33	5	29,85	3	2	2	3	4	3	7	2	12
5:11:8	1	0	1	-0,3	0	0	0	0	0	0	1	0	0
2:6:7	2	1	1	0,5	0	0	0	0	0	0	0	0	0
2:7	4	3	1	1,4	0	0	0	0	1	0	0	0	0
6:8:7	5	2	3	0,45	0	0	0	0	0	0	1	1	0

Continua na próxima página

Tabela B.1 – Continuação da página anterior.

Coalizões (jogadores)	Total	Avanços	Recuos	Pontos	Gols (ind.)	Gols Perdidos (ind.)	Oportunidades de Gol	Oportunidades de Gol (ind.)	Faltas Cometidas	Faltas Recebidas	Passes Errados (ind.)	Escanteios (ind.)	Cruzamentos (ind.)
7:3	2	2	0	0,55	0	0	0	0	0	0	1	0	0
2:1	2	1	1	0,1	0	0	0	0	2	0	0	1	0
8:4	1	0	1	0	0	0	0	0	0	0	0	0	0
6:7:10	9	7	2	8,25	2	1	0	1	1	1	2	2	1
5:1	2	2	0	5,7	2	0	0	0	0	1	0	0	1
8:2	1	0	1	2,3	1	0	0	1	0	0	0	0	0
6:1:2	1	0	1	0	0	0	0	0	0	0	0	0	0
7:11:6	1	1	0	1,1	0	1	0	0	0	0	0	0	0
11:7:10	3	3	0	3,45	0	2	0	1	1	0	0	0	2
5:7	1	1	0	0,2	0	0	0	0	0	0	1	0	0
7:10	43	35	8	21	1	1	0	2	3	2	11	2	4
10:8	7	6	1	2,8	0	0	0	0	0	2	5	2	0
7:5:11	1	1	0	0,2	0	0	0	0	0	0	1	0	0
9:6	26	15	11	22,2	6	0	1	5	1	4	5	0	5
6:7:4	1	1	0	0,5	0	0	0	0	0	0	0	0	0
11:8	32	16	16	9,2	1	0	0	1	3	4	5	1	4
6:3	1	1	0	0,5	0	0	0	0	0	0	0	0	0
9:10:11	2	2	0	1,55	0	0	0	0	0	1	0	0	1
10:6	15	14	1	6,35	0	0	0	1	1	1	2	0	0
8:5:7	6	2	4	1,1	0	0	0	0	0	1	1	1	0
4:2	1	1	0	0,5	0	0	0	0	0	0	0	0	0
8:4:5	1	0	1	-0,1	0	0	0	0	1	0	0	0	0
5:8:1	2	2	0	4,7	1	0	0	0	0	0	0	0	2
2:6	12	6	6	7,35	1	0	0	2	0	1	3	0	3
11:10	16	16	0	16,2	2	1	0	3	0	2	4	1	3

Continua na próxima página

Tabela B.1 – Continuação da página anterior.

Coalizões (jogadores)	Total	Avanços	Recuos	Pontos	Gols (ind.)	Gols Perdidos (ind.)	Oportunidades de Gol	Oportunidades de Gol (ind.)	Faltas Cometidas	Faltas Recebidas	Passes Errados (ind.)	Escanteios (ind.)	Cruzamentos (ind.)
5:11	1	1	0	0,5	0	0	0	0	0	0	0	0	0
5:8	17	9	8	12,75	3	0	0	2	2	3	4	2	2
10:9:7	3	3	0	2,4	0	1	0	1	0	0	0	0	0
Total	351	253	98	243,25	37	16	5	38	28	33	77	19	53

ANEXO A ALGORITMOS DE DETECÇÃO

Os programas *Soccer Scientia Tool* (SST) e *Soccer Server Statistical Extracting Tool* (SSSET) fornecem estatísticas gerais de jogo a partir de determinados eventos que ocorrem em uma partida. Esses eventos são detectados por meio de algoritmos que foram implementados no programa *SoccerScope2*. Neste anexo estão relacionados os principais algoritmos ligados à detecção de eventos.

ALGORITMO 1: Algoritmo de detecção de passes. Extraído de CUNHA ABREU (2011).

```
for all Cycle i in Kicks do
  cycle ← false
  originalKicker ← getPlayerKicking(i)
  secondKicker ← getPlayerKicking(nextKick(i))
  if originalKicker ≠ secondKicker ∧ sameTeam(originalKicker, secondKicker) then
    for j = nextKick − 1 to i + 1 do
      for all Player p in Players do
        if p ≠ secondKicker ∧ distance(p, Ball, j) < distance(secondKicker, Ball, j) then
          receiveCycle = j + 1
          cycle ← true
          break
        end if
      end for
      if cycle then
        break
      end if
    end for
    AddSuccessfulPass(originalKicker, secondKicker, i, receiveCycle)
  end if
end for
```

ALGORITMO 2: Algoritmo para prever o possível recebedor da bola. Extraído de CUNHA ABREU (2011).

```
IniCycle ← Cycle of Initial Kick
for all Cycle c in BallPath do
    for all Player p in Teammates do
        dist ← distance(p, Ball, c, IniCycle)
        addToProbabilityVector(p, Probability(dist, c − IniCycle))
    end for
end for
return HighestProbability(ProbabilityVector).Player
```

ALGORITMO 3: Algoritmo de detecção de gols e chutes a gol. Extraído de CUNHA ABREU (2011).

```
for all Cycle i in Kicks do
    kicker ← getPlayerKicking(i)
    for j = i + 1 to nextKick(i) do
        BallInitialPosition = BallPosition(i);
        BallFinalPosition = BallPosition(j);
        if InRegion(BallPosition(j + 1), OutsideBack) then
            if Intercepts(BallInitialPosition, BallFinalPosition, GoalLine) then
                AddGoal(kicker, i)
            else if Intercepts(BallInitialPosition, BallFinalPosition, GoalLine + 0.5) then
                AddShotonTarget(kicker, i)
            else if Intercepts(BallInitialPosition, BallFinalPosition, PenaltyBoxBackLine) then
                AddShot(kicker, i)
            end if
        end if
    end for
end for
```

ALGORITMO 4: Algoritmo de detecção de laterais, escanteios e tiro livre indireto. Extraído de CUNHA ABREU (2011).

```
for all Cycle i in Kicks do
    kicker ← getPlayerKicking(i)
    for j = i + 1 to nextKick do
        if InRegion(BallPosition(j), Outside) then
            outsideType = calculateOutsideType(BallPosition(j))
        end if
    end for
end for
```

ALGORITMO 5: Algoritmo de detecção de impedimentos. Extraído de CUNHA ABREU (2011).

```
for all Cycle i in Kicks do
    originalKicker ← getPlayerKicking(i)
    secondKicker ← getPlayerKicking(nextKick(i))
    if !sameTeam(originalKicker, secondKicker) then
        receiver = DetermineReceiver()
        if InvalidPosition(receiver, i) then
            AddInterceptedOffside(receiver, i)
        end if
    end if
    if originalKicker ≠ secondKicker ∧ sameTeam(originalKicker, secondKicker) then
        if InvalidPosition(secondKicker, i) then
            AddOffside(secondKicker, i)
        end if
    end if
end for
```