UNIVERSIDADE FEDERAL DE VIÇOSA
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

JHONATAN DE SOUZA OLIVEIRA

REDES BAYESIANAS NA DETERMINAÇÃO DE FUNÇÕES TÁTICAS DE UM TIME DE FUTEBOL DE ROBÔS

VIÇOSA
2014

JHONATAN DE SOUZA OLIVEIRA

REDES BAYESIANAS NA DETERMINAÇÃO DE FUNÇÕES TÁTICAS DE UM TIME DE FUTEBOL DE ROBÔS

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 490 - Monografia e Seminário e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.
Orientador: Prof. Dr. Alexandre Santos Brandão

VIÇOSA
2014

JHONATAN DE SOUZA OLIVEIRA

REDES BAYESIANAS NA DETERMINAÇÃO DE FUNÇÕES TÁTICAS DE UM TIME DE FUTEBOL DE ROBÔS

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 490 - Monografia e Seminário e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovada em 28 de Novembro de 2014.

COMISSÃO EXAMINADORA

Prof. Dr. Alexandre Santos Brandão - Orientador
Universidade Federal de Viçosa

Prof. Dr. André Gomes Tôrres - Membro
Universidade Federal de Viçosa

Prof. B. Sc. Paulo Fábio Rocha - Membro
Universidade Federal de Viçosa

Pai, para compensar seu filho perna de pau, fiz um time de pernas de lata, não tão bom quanto o seu Cruzeiro, mas com a mesma vontade de aprender que você me ensinou.

Agradecimentos

Agradeço à minha família pelas orações e palavras certas, nos melhores momentos. Em especial, ao meu pai, Jair, minha mãe, Luzinete e minha irmã, Rafaela, pilares do meu saber. Aos amigos e professores da Eng. Elétrica e do BDP por toda ajuda, paciência e ensinamentos. Em especial, ao meu orientador Alexandre Brandão. Também, agradeço aos amigos Samuel e Fabiano, pelas ideias, conversas e contribuições, tão essenciais.

Este trabalho não seria possível sem vocês... e muito café.

"Tenhamos as cabeças abertas,
mas não tão abertas ao ponto de nossos cérebros se desprenderem delas."
Richard Dawkins

Resumo

A Small Size League (SSL) é uma das categorias de futebol de robôs integrantes dos campeonatos RoboCup. Dentre os diversos desafios na área de Inteligência Artificial aplicada ao futebol de robôs, destaca-se a possibilidade de troca de papéis entre os jogadores em campo. Por exemplo, um zagueiro pode vir a ser atacante, caso essa decisão seja mais favorável para a equipe. O problema na decisão de papel para os jogadores é o conflito de informações ou a ausência das mesmas quando as variáveis disponíveis são analisadas. Este trabalho propõe o uso da ferramenta Redes Bayesianas (RBs) para modelagem e inferência sobre esse sistema com incertezas.

Os objetivos deste trabalho incluem a implementação de dois times para SSL em um ambiente de simulação. Um time básico com papéis de jogadores fixos durante a partida, e outro time, chamado BayesBall, com troca de papéis. O gerenciamento da decisão sobre qual papel o jogador exercerá se dá pela absorção de evidências do sistema simulado, seguido de inferência sobre a RB.

Os resultados foram satisfatórios para os experimentos preliminares. Embora os poucos papéis disponíveis limitaram a dinâmica da partida, pode-se averiguar a precisão da RB julgando suas decisões em situações reais de jogo, e a consequente vitória do time BayesBall sobre o time básico.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O futebol é um esporte que apresenta desafios característicos das áreas de estudo de Inteligência Artificial (IA). Dentre eles, destaca-se o reconhecimento de padrões com poucos dados históricos e a tomada de decisão em ambientes com incerteza. O futebol de robôs é uma tentativa de fomentar pesquisas em robótica e IA através de uma plataforma comum para testes de diversas teorias, algoritmos e arquitetura de organização [1]. A *Robot World-Cup Soccer* (RoboCup) é uma competição anual internacional de futebol de robôs fundada em 1997 com o objetivo oficial de, em meados do século 21, um time de futebol de robôs humanóides vencer uma partida de futebol, pelas regras da FIFA, contra os vencedores mais recentes da Copa do Mundo. A liga de futebol de tamanho pequeno (do inglês, *Small Size League* - SSL) é uma das categorias mais promissora da RoboCup. Nela, a proposta é praticar uma partida de futebol, com regras adaptadas, entre dois times, cada um com seis jogadores, chamados *agentes*. Os agentes executam situações do futebol, como a condução e o passe de bola, a possibilidade de infração por colisão, o lançamento cruzado para a área adversária, dentre outros cenários que fazem parte da realidade do futebol humano. Durante a partida, um sistema central, chamado visão, é responsável por processar as imagens de duas câmeras colocadas em cima do campo, a fim de identificar os jogadores e a bola. A visão envia as informações processadas para a rede local, configurada para a partida. Um time é representado por um cliente, também chamado de módulo de inteligência, que se conecta à mesma rede local da visão, através do protocolo UDP, a fim de receber informações sobre os agentes. Por último, o módulo de inteligência envia comandos para os jogadores em campo, através de comunicação sem fio, finalizando um ciclo que configura um sistema em tempo real. A Fig. 1 ilustra esse processo que ocorre durante uma partida.

Redes Bayesianas [2] (RBs) é uma ferramenta gráfica para o gerenciamento de incertezas nos sistemas. Um sistema apresenta incerteza quando possui informações conflitantes ou ausentes. Através de ferramentas como a absorção de evidência, pode-se raciocinar sobre a modelagem do sistema e chegar a resultados esperados de um especialista. Essas

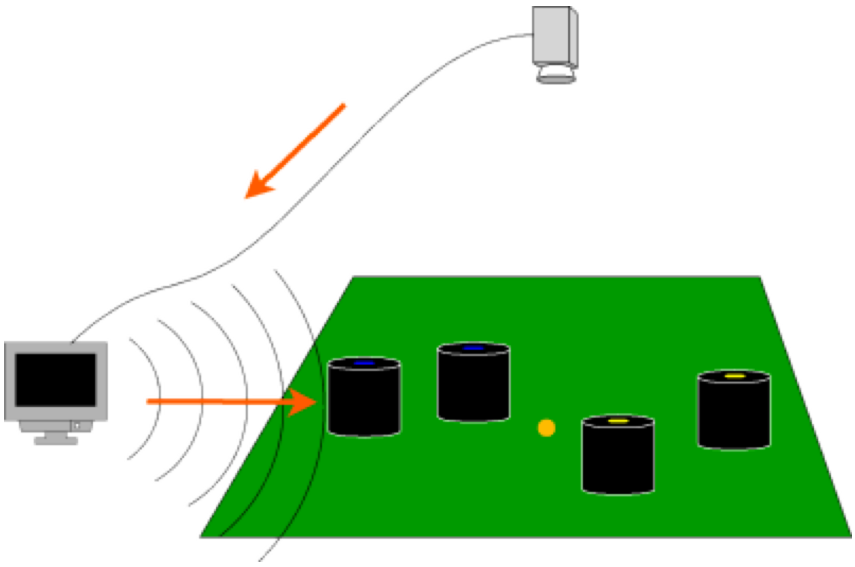

Figura 1: Visão geral do processo que ocorre durante uma partida: a câmera envia imagens para um computador central. As informações processadas são então utilizadas para as decisões enviadas, de forma sem fio, aos robôs em campo.

ferramentas são baseadas em teoria de probabilidade e apresentam robustez e semântica melhores do que em outras alternativas chamadas *ad hoc* como Lógica Fuzzy ou Sistemas Baseados em Regras [3]. Para se responder às perguntas (*queries*) sobre o modelo configurado em uma RB pode-se usar um algoritmo de inferência chamado *Variable Elimination* (VE) [4]. Caso a inferência seja mais longa que o ciclo do sistema em tempo real, pode-se optar por alternativas que usem pré-computação como o apresentado em [5] ou alternativas que reutilizam computação como exposto em [6].

A SSL possui diversas tarefas que envolvem informações conflitantes ou ausentes. Destaca-se a determinação do papel de cada jogador durante uma partida. Deixar um jogador com papel fixo pode causar situações indesejáveis como um zagueiro que não vai até a bola, mesmo próximo dela, só porque a mesma não se encontra em sua área de defesa [7]. Assim como em um jogo real, em algumas situações, o jogador pode atuar em uma posição diferente da qual foi originalmente designado. Por exemplo, durante uma ofensiva em contra ataque, o cabeça de área pode fazer a tarefa de um atacante, caso surja a oportunidade. Essa decisão de como atuar em uma determinada situação envolve, principalmente, o conflito de informações, típico de um ambiente dinâmico como o futebol.

A capacidade de um sistema inteligente de futebol de robôs tomar decisões baseadas em histórico e em situações novas durante a partida é discutido em [8]. Em especial, apresenta-se uma solução para a decisão em alto nível de qual estratégia adotar, dado as configurações mais atuais do jogo. A tomada de decisão é baseada em dois tipos de

aprendizagem: *offline* e *online*. O processo de aprendizagem pelo histórico de dados é denominado *offline*. Nessa etapa, os especialistas levantam um banco de dados com entradas que configuram uma das situações analisadas e ensinam o sistema inteligente qual a melhor estratégia para a situação exposta. O aprendizado *online* é o ajuste, durante a partida, do que se aprendeu *offline*. O trabalho propõe a combinação de redes neurais e RBs. Em linhas gerais, a rede neural do tipo self-organized map (SOM), uma rede clássica no treinamento não-supervisionados, é usada para mapear as entradas agrupadas nas saídas, denominados *clusters*. As entradas formam um vetor com o estado atual da partida, e os *clusters* são formados pelas manipulações das entradas, de forma a criar categorias das mesmas. As RBs atuam nas categorias descobertas pelas SOM, a fim de decidir a melhor estratégia para o atual estado da partida. O modelo SOM Bayesiano funciona como um sistema de tomada de decisão em um ambiente multi-agente e apresentou resultados significativos para uma solução que engloba escopos geralmente separados na literatura, o aprendizado *offline* e *online*.

A construção de um time de futebol de robôs simulado com RB e redes de Petri é proposto em [9]. A visão de cada agente, isto é o estado mais atual da partida, é modelado em um tipo de RB chamado RBs temporais. Nesse tipo, o tempo é considerado no modelo através de uma medida probabilística do quanto o estado anterior interfere no atual. Com essa ferramenta, pode-se considerar com maior precisão a dinâmica do sistema. As ações dos agentes são representadas em redes de Petri. Esses modelos são extensões da máquina de estado, porém adicionam a capacidade de executar eventos em paralelos, sem perder a hierarquia dos mesmos. No trabalho, as observações simuladas entram como evidências para atualizar o modelo representado pelas RBs. Após atualização, as redes de Petri, que estão acopladas à RB, podem alterar de estado, caso o limiar do atual estado seja alcançado. Dessa forma, as RBs atualizam o modelo, dado as informações mais atuais da partida, e, em seguida, as redes de Petri determinam as ações do agente. Embora estratégias mais complexas não tenham sido testadas na solução proposta, tarefas comuns durante uma partida foram alcançadas com sucesso.

Em [10], é investigado os diferentes objetivos da IA no contexto do futebol de robôs. O artigo aplica sua proposta em uma categoria para pequenos robôs de duas rodas. Porém, algumas soluções descritas envolvem o tratamento entre os agentes, de forma genérica, independente da estrutura física utilizada. Além do mais, assim como no SSL, a categoria também usa uma visão central, uma unidade de processamento por time, o cliente, e todo o processamento é feito em tempo pré-determinado, configurando um sistema em tempo real. Destaca-se o algoritmo para reconhecimento de estados, onde se utiliza técnicas

de controle clássico para determinar situações de jogo, como ofensivo ou defensivo. Na mesma categoria, o trabalho apresentado em [11] propõe um sistema usando lógica Fuzzy para determinar qual o papel de cada agente durante a partida. As regras de fuzificação envolvem a distância entre o agente e a bola, a orientação, ângulo para chute e possíveis obstáculos. Para defuzificação, o operador *or* é utilizado para somar o valor de afinidade entre o agente e o um papel em campo. O papel com maior afinidade para certo agente é então escolhido.

Um método para classificação de situações durante uma partida no futebol de robôs é esboçado em [12]. Para isso, utilizou-se árvores de decisão podadas pelos algoritmos Gini, Twoing e Deviance. A árvore em si foi construída com um algoritmo de aprendizagem chamado CART, utilizando um conjunto de dados históricos das partidas. Os resultados demonstram que árvores de decisão são capazes de identificar e prever situações de jogo de forma mais precisa que métodos heurísticos.

Levando em consideração o contexto até aqui apresentado, este trabalho tem como objetivo construir um time de futebol de robôs para a SSL no qual os agentes tem a capacidade de trocar de papel durante a partida. O time é implementado em um ambiente simulado, através do simulador oficial da categoria, o grSim. O gerenciamento de incertezas durante a troca de posição dos agentes será manipulado por uma RB. Para validação de resultados, o time com capacidade de troca de posição será confrontado em diversas partidas com um outro time, chamado básico, o qual usa papéis fixos por agente.

1.1 Small Size League

A SSL, também conhecida por F-180, dentre as outras categorias competitivas da RoboCup, tem grande foco acadêmico, dado seus diversos desafios de implementação. O principal foco dessa categoria é o problema na cooperação de multi agentes em um ambiente altamente dinâmico. Por exemplo, em uma competição normal do SSL, um ciclo completo de resposta, isto é, o processamento de todas informações, pode chegar a 180 ms. Os agentes em campo devem responder às decisões centrais da inteligência de cada equipe. O módulo de inteligência tem acesso à visão geral do jogo a cada ciclo e também às possíveis informações provindas da telemetria dos agentes. Dessa forma, a SSL pode ser classificada como sistema híbrido entre ambiente centralizado e distribuído.

Em uma partida do SSL, duas equipes se enfrentam com até seis robôs cada. Cada robô deve ser dimensionado fisicamente para ocupar um diâmetro máximo de 180 mm e

Figura 2: Uma possível configuração para uma partida no SSL.

altura máxima de 150 mm [13]. A bola utilizada nas partidas tem dimensões próximas à de uma bola de golfe e é tingida de laranja. O campo é feito por um carpete verde, com dimensões próximas à 4 m de largura e 6 m de comprimento. A Fig. 2 apresenta uma possível configuração para duas equipes se enfrentando no SSL.

A automação dos agentes em campo se dá através de um fluxo comum entre as equipes: um sistema de visão que rastreia os objetos em campo, um juiz humano que intervém na partida através de um software, o sistema de inteligência de cada equipe e a comunicação com os agentes em campo. A Robocup incentiva o compartilhamento de ferramentas e informações entre as equipes, como forma de fomentar e desenvolver a área. Assim, diversas ferramentas que se consolidaram entre as equipes com o passar dos anos, foram adotadas como oficiais da categoria. Esse é o caso do software de visão, o SSL Vision, do software simulador, o grSim, dentre outras.

Para o sistema de visão, duas câmeras de alto desempenho são colocadas a 4 metros de altura sobre campo. As imagens capturadas por ambas são transmitidas diretamente para um computador de alto desempenho. Esse computador executa o software SSL Vision, o qual processa as imagens de ambas câmeras, uma por vez, a fim de retirar as informações em campo. Essas informações são: posição dos robôs, da bola, identificação única dos robôs, identificação por time dos robôs e geometria do campo. Em seguida, as informações retiradas do processamento de imagem são repassadas para uma rede local, sob protocolo UDP, através de um padrão de encapsulamento chamado Protobuf.

O software de inteligência de cada equipe deve ser capaz de capturar os pacotes de informação disponibilizados na rede local do campeonato. As equipes, em seguida, devem processar as informações, de maneira autônoma, a fim de decidir o que cada agente deve fazer em campo. É esperado do módulo de inteligência das equipes a previsão de cenários, o reconhecimento de padrões, o desvio de obstáculos, planejamento de trajetória, dentre outros problemas característicos da IA.

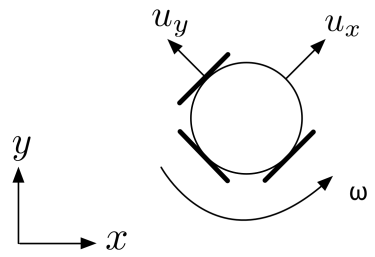

Figura 3: Modelagem de um robô de quatro rodas sobre dois sistemas referenciais.

Por fim, as equipes transmitem as informações para os agentes em campo através de módulos de rádio frequência. As regras do SSL delimitam as tecnologias permitidas para essa comunicação, mas, em geral, as equipes usam módulos de transmissão e recepção wireless wi-fi, xbee ou FM.

1.2 Modelo do Robô

Para o envio de comandos aos agentes no campo, é necessário a modelagem matemática dos mesmos. Com o modelo dos robôs, pode-se utilizar técnicas de controle a fim de fazê-lo mover para uma posição desejada. A teoria de controle utilizada neste trabalho pode ser encontrada em [14]. O robô considerado neste trabalho é um agente com quatro rodas, o que o coloca na categoria omnidirecional. Essa característica é um diferencial importante para as equipes, já que os agentes podem se locomover com maior liberdade, melhor precisão e em menor tempo.

Em termos gerais, na modelagem da cinemática envolvendo o robô móvel, deve-se considerar dois referenciais: o do robô e a do espaço no qual o robô descreve uma trajetória, chamado referencial espacial. Variáveis derivadas desses sistemas referenciais podem ser relacionadas através de modelos não lineares [15], como ilustrado na Fig. 3.

O modelo do robô de quatro rodas considerado neste trabalho é descrito a seguir. Na referência do próprio robô, a velocidade linear no eixo x, u_x, no eixo y, u_y e angular ω, podem determinar no referencial espacial, o qual o robô descreve a trajetória, a velocidade em x, $\dot{x}$, em y, $\dot{y}$ e angular $\dot{\psi}$. Esse modelo de transferência entre os referenciais pode ser descrito na forma matricial a seguir:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} cos\psi & -sen\psi & -a \cdot sen\psi \\ sen\psi & cos\psi & a \cdot cos\psi \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_x \\ u_y \\ \omega \end{bmatrix}$$

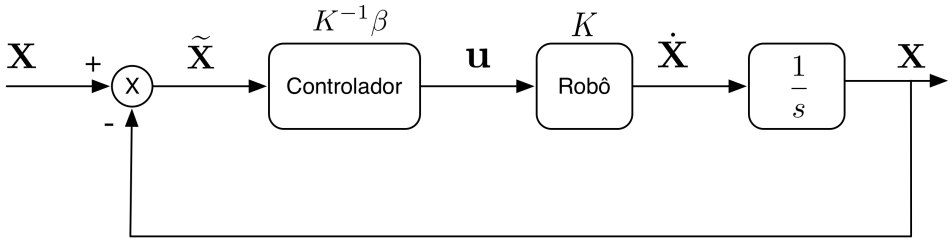

Figura 4: Malha de controle para simulação do posicionamento dos robôs.

De forma similar, a modelagem matricial pode também ser descrita na forma algébrica

$$\dot{\mathbf{X}} = K \cdot \mathbf{u},$$

representando, respectivamente, as matrizes de velocidade no referencial espacial, de constantes de modelagem e velocidade no referencial do robô.

Deseja-se, de forma geral, impor um ponto desejado para os agentes, para que os mesmos façam a trajetória de forma suave, corrigindo os possíveis desvios e erros. Porém, os robôs tem como entrada final a velocidade em cada roda, individualmente. Dessa forma, faz-se necessário um controlador para os agentes.

Um controlador linear simples, o qual atende aos requerimentos deste trabalho, é definido a seguir:

$$\mathbf{u} = K^{-1} \cdot \beta \cdot \widetilde{\mathbf{X}}, \tag{1.1}$$

onde $\mathbf{u}$ é a velocidade desejada em cada motor, β é uma constante de ajuste do controlador, K^{-1} é a inversa da matriz de parâmetros de modelagem do robô e $\widetilde{\mathbf{X}}$ o erro entre a postura atual do robô e a desejada. *Postura* é o conjunto espacial de posicionamento do agente, o qual inclui as coordenadas no plano cartesiano e o ângulo de rotação do mesmo.

Agora, pode-se definir uma malha de controle, utilizando o controlador, a fim de se obter uma previsão de resposta do sistema. A Fig. 4 ilustra o diagrama de blocos da possível malha aplicada aos agentes em campo quando se deseja posicionar os robôs.

A entrada da malha é a postura desejada, $\mathbf{X}$, a qual é subtraída da postura atual do robô, $\mathbf{X}$, a fim de se obter o erro, $\widetilde{\mathbf{X}}$. Em seguida, $\widetilde{\mathbf{X}}$ é aplicado ao controlador, $K^{-1}\beta$, o qual age com um comando de correção das velocidades em cada roda, $\mathbf{u}$, sobre o robô modelado com parâmetros, K. O fator integrador, $\frac{1}{s}$, ao final da malha garante a obtenção da postura em termos de posição e não velocidade.

As respostas obtidas com a simulação da malha da Fig. 4 foram satisfatórias, já que se esperavam respostas suaves, mais seguras para a aplicação dos robôs móveis. Por exemplo,

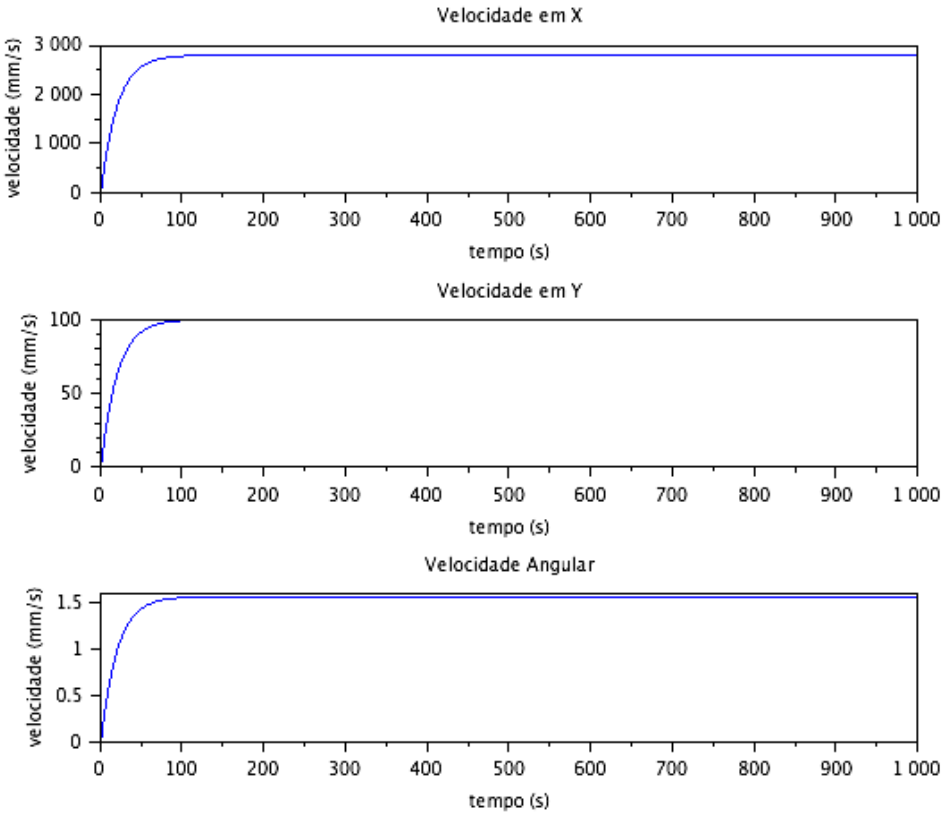

Figura 5: Exemplo de resposta da malha de controle da Fig. 4 para o posicionamento do robô em um ponto aleatório, onde a abscissa representa o tempo em segundos.

para um ponto aleatório no referencial espacial, a Fig. 5 apresenta a saída da malha, isto é, descreve como seria a trajetória descrita pelo robô ao se posicionar na postura desejada.

Assim, o controlador proposto na Eq. (1.1) exibe estabilidade e precisão suficientemente boa para a aplicação proposta neste trabalho.

1.3 Simulador

A implementação do método proposto neste trabalho será testada em um ambiente de simulação computacional. A comunidade do SSL, com a contribuição de diversas equipes, desenvolveu um simulador chamado grSim [16].

A principal função do grSim é substituir o sistema de visão. Para isso, o grSim deve lidar, em tempo real, com a simulação de modelos dinâmicos dos robôs, incluindo reações físicas, variáveis de estado e a renderização de um ambiente 3D para a visualização dos resultados. O simulador também envia as informações do ambiente pela rede local, através do protocolo UDP, assim como o sistema de visão, SSL-Vision. Dessa forma, as

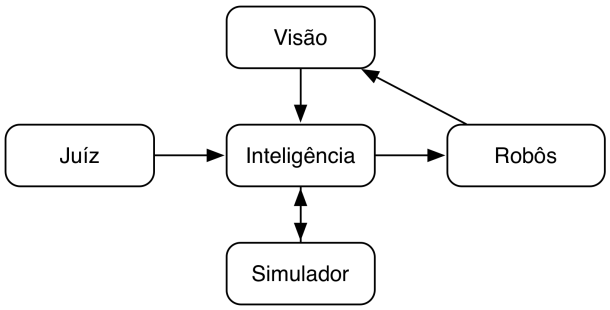

Figura 6: Visão geral do SSL, incluindo a possibilidade do simulador grSim.

equipes podem utilizar ambos sistemas sem muita ou nenhuma modificação no módulo de inteligência.

Agora, pode-se descrever a visão geral do SSL, incluindo a possibilidade do simulador, o sistema de visão formado pelas câmeras, o módulo de inteligência das equipes, o software do juiz controlado por intervenção humana e os agentes, representado pelos robôs, é ilustrada pelo diagrama da Fig. 6.

Com o fluxo de informação proposto na Fig. 6 as equipes têm disponível dois modos de operação funcionais e bem estruturados: o modo de partida e o modo de simulação, onde são desenvolvidos testes e pesquisas.

1.4 Objetivos

Este trabalho propõe a implementação do sistema *BayesBall*, um módulo de inteligência que se comunica com o simulador e gerencia os papéis dos agentes em campo através de uma RB. Na primeira, se desenvolve um time chamado básico, composto por três papéis: goleiro, zagueiro e atacante. Os seis jogadores do time básico recebem papéis fixos, dentre os três disponíveis, e exercem essas funções por toda a partida. Na segunda etapa, é desenvolvido o time BayesBall, também composto pelos três papéis descritos anteriormente. Porém, no BayesBall os seis jogadores recebem papéis decididos pela absorção de evidências e inferência em uma RB. Alguns objetivos específicos:

- o estudo e exposição da ferramenta RB e métodos relacionados,
- a implementação de um módulo de inteligência que seja escalável e seguro, capaz de suportar possíveis expansões de trabalhos futuros, e, por fim,
- a criação de um time mais competitivo para o SSL.

1.5 Distribuição

O restante desse trabalho segue como descrito a seguir. A introdução discute os problemas do SSL e apresenta a ferramenta RB. Também são apresentados a modelagem cinemática dos robôs e o simulador utilizado para teste do método. Em materiais e métodos, apresenta-se a implementação de um time básico, onde os agentes tem papéis fixos e regras simples. Em seguida, o time chamado BayesBall, utilizando o método proposto neste trabalho, é implementado. Para validação, os dois times se enfrentam em diversas partidas e os resultados serão discutidos a seguir. Por fim, as conclusões e propostas para trabalhos futuros serão traçadas.

2 Redes Bayesianas

Redes Bayesianas (RBs) é uma ferramenta para gerenciar e raciocinar sobre sistemas com incertezas. Define-se como incertezas sistemas com informações conflitantes ou ausentes. Por exemplo, considere um sistemas de busca na internet: se o usuário pesquisar por "Sistemas da Informações", dentre os resultados, duas páginas são plausíveis, uma com o título "Sistemas de Banco de Dados" e outra com "Informações de Recuperação". A falta de informação, nesse caso, de critérios, leva à incerteza sobre qual página exibir em primeiro lugar. Outro exemplo pertinente, considere um médico que examina um paciente com alguns, mas não todos, sintomas de uma doença específica. O conflito de informações, nesse caso, leva à incerteza sobre qual doença o paciente pode ter.

Em IA, o gerenciamento e raciocínio de incertezas pode ser tratado com diversas ferramentas já sugeridas na literatura, como Lógica Fuzzy ou Sistemas Baseado em Regras. Porém, RBs é a ferramenta que apresenta melhor clareza semântica, sem abrir mão da robustez provida pela consolidada teoria da probabilidade. Por exemplo, em [3], Lógica Fuzzy pode levar a conclusões nem sempre intuitivas. Considere o caso de lançar uma moeda não viciada. Pode-se definir duas sentenças para essa situação: uma para o caso "Cara" com valor de 0,5 e outra para o caso "Coroa" com valor de também 0,5, onde os valores representam a crença em que se tem nos resultados do lançamento da moeda. Em seguida, pode-se perguntar qual o valor esperado para lançar a moeda e se obter "Cara" ou "Coroa". Por definição, valores para sentenças compostas por "ou" em Lógica Fuzzy podem ser obtidos pelo valor máximo entre os valores das sentenças. Nesse caso, a resposta seria 0,5. Esse resultado não é intuitivo, pois se acredita que as chances de um lançamento de moeda ser "Cara" ou "Coroa" é 1.

Em outro exemplo [3], Sistemas Baseados em Regras podem ter carência de semântica e conjuntos de regras consideravelmente grandes. Considere uma simples situação onde se quer diagnosticar a causa da queda de conexão de um usuário de internet. As regras são introduzidas com estruturas "Se-Então" e um fator de confidência. Uma possível regra é a seguinte: "Se a página de internet nem chega a abrir, então o usuário pode ter perdido

a conexão; Fator de confidência: 9". O valor de confidência 9 não apresenta semântica clara, afinal, pode-se levantar questões como o por que desse valor, ou se é um valor baixo ou alto, dentre outras. Além do mais, para atender satisfatoriamente um sistema de diagnóstico, um Sistema Baseado em Regras pode requerer um conjunto de estruturas "Se-Então" muito grande ou inviável.

Por outro lado, há críticas à fase inicial de utilização das RBs. Para a modelagem do domínio de um problema em RB, é necessário selecionar as variáveis de interesse, explicitar as relações de dependência entre as mesmas e, por fim, a intensidade com que essas relações se dão. Geralmente, especialistas na área são chamados para essa modelagem. De certa forma, o modelo construído por especialistas tendem a ser de mais simples compreensão, manutenção e inspeção, em caso de futuros problemas ou expansões [17]. Porém, caso o modelo apresente alguma inconsistência semântica, essa será de difícil localização, além de comprometer a confiabilidade dos resultados.

Antes de definir formalmente RBs, é necessário uma rápida revisão de conceitos de teoria da probabilidade, como distribuição de probabilidade conjunta, tabelas de probabilidade condicional, e grafos matemáticos, como os grafos acíclicos dirigidos.

A teoria da probabilidade tem ferramentas robustas, apropriadas e desejadas para lidar com incerteza [18]. Primeiramente, ela possui representação declarativa, com semântica clara. Também, possui poderosos padrões de raciocínio que possibilitam a criação de sistemas de tomada de decisão sob condicionamento. Por último, os métodos de aprendizado para sistemas probabilísticos são reconhecidos e consolidados pela comunidade acadêmica. Em especial, as RBs tem toda sua fundamentação teórica baseada na utilização de conceitos bem fundados pela comunidade científica, como os operadores regra da cadeia e o uso de independências condicionais.

Distribuição de Probabilidade Conjunta

Considere $U = \{v_1, v_2, v_3, \ldots, v_n\}$ um conjunto finito de variáveis discretas. Por exemplo, $U = \{familia{-}saiu, luzes{-}ligadas, cachorro{-}fora, problemas{-}intestinais, ouviu{-}latido\}$, onde todas variáveis são binárias. Cada variável v_i está associada a um domínio finito, denominado $dom(v_i)$. Por exemplo, $dom(famlia - saiu) = \{sim, não\}$. Seja V o produto cartesiano do domínio das variáveis, definido como $V = dom(v_1) \cdot dom(v_2) \cdot dom(v_3) \cdot \ldots \cdot dom(v_n)$. Uma *distribuição de probabilidade conjunta* (DPC) é uma função p em V tal que as seguintes condições sejam satisfeitas: (i) $0 \leq p(v) \leq 1$, para cada linha $v \in V$; e (ii) $\sum_{v \in V} p(v) = 1$. Caso não cause confusão, vai-se dizer nesse trabalho que p é uma DPC em U, ao invés de V.

Tabela 1: Exemplo de uma DPC para três variáveis binárias, onde p é uma abreviação para $p(cachorro - fora, problemas - intestinais, ouviu - latido)$.

cachorro-fora	problemas-intestinais	ouviu-latido	p
não	não	não	0,20
não	não	sim	0,15
não	sim	sim	0,06
sim	sim	sim	0,20
sim	sim	não	0,09
sim	não	não	0,15
não	sim	não	0,08
sim	não	sim	0,07

Exemplo 1. Seja $U = \{cachorro - fora, problemas - intestinais, ouviu - latido\}$, onde cada $v_i \in U$ tem domínio binário, isto é, $dom(v_i) = \{sim, não\}$. Seja p uma DPC em U, isto é, $p(cachorro - fora, problemas - intestinais, ouviu - latido)$. A DPC p pode ser representada por uma tabela, como mostrado na Tabela 1, onde as três variáveis são apresentadas nas três primeiras colunas e a quarta coluna é o valor associado à DPC, representado pela função p. Cada linha representa uma configuração dos possíveis valores das variáveis, e cada configuração possui um valor associado à DPC. Por definição, a somas dos valores de p totalizam 1.

A DPC é uma função importante na modelagem de sistemas. Através da DPC, pode-se responder perguntas sobre o mesmo e obter respostas matematicamente corretas, usando as consolidadas ferramentas da teoria da probabilidade. De forma similar, quando se deseja a saber a probabilidade de alguma variável condicionada à um valor específico de outra variável, obtém-se a *tabela de probabilidade condicional* (TPC). Por exemplo, pode-se perguntar para a DPC exposta na Tabela 1 qual é a probabilidade de $cachorro - fora$ ser sim, caso $problemas - intestinais$ e $ouviu - latido$ sejam $não$, isto é, deseja-se a construção da TPC $p(cachorro - fora | problemas - intestinais = não, ouviu - latido = não)$. Nesse exemplo, sem necessidade de manipulações matemáticas, encontra-se na sexta linha a configuração desejada, por isso, a resposta é 0,15. Outra pergunta plausível, qual é a probabilidade de $cachorro - fora$ dado que $ouviu - latido$ é sim. Nesse caso, a TPC não é facilmente obtida após inspeção da DPC na Tabela 1. Formalmente, deseja-se saber $p(cachorro - fora | ouviu - latido = sim)$. Para se obter a resposta, é necessário absorver a evidência inicial, isto é, o fato de que $ouviu - latido = sim$, e, em seguida, remover as variáveis não desejadas da tabela, processo chamado marginalização. A seguir, é apresentado a semântica da absorção de evidências e marginalização.

Tabela 2: DPC da Tabela 1 normalizada pela absorção da evidência $ouviu - latido = sim$.

cachorro-fora	problemas-intestinais	ouviu-latido	p
não	não	sim	0,3125
não	sim	sim	0,125
sim	sim	sim	0,417
sim	não	sim	0,146

Absorção de Evidências e Marginalização

Evidências são observações sobre o sistema nas quais se pode confiar. Quando uma observação é feita, a DPC do sistema pode ser atualizada a fim de refletir o estado mais atual do mesmo. Esse processo de atualização é chamado de absorção de evidências, mas também é conhecido por atualização de crenças (do inglês, *belief update*). Para uma DPC $p(U)$, representa-se uma pergunta com a absorção de evidências como uma TPC $p(X|Y = y)$, onde $X \cap Y = \emptyset$, $X \subset U$, $Y \subset U$ e $y \in dom(Y)$. Para atualizar a DPC, deve-se fazer o seguinte: (i) remover as linhas não condizentes com a evidência; e (ii) normaliza os valores de p.

Exemplo 2. Considere a evidência $ouviu - latido = sim$ sobre o sistema representado na DPC da Tabela 1. Para absorver a evidência, deve-se remover todas linhas onde $ouviu - latido = não$. Esse processo gera a DPC atualizada. Em seguida, os valores de p são normalizados, a fim de somarem 1 novamente, resultando na DPC $p(cachorro - fora, problemas - intestinais|ouviu - latido = sim)$ apresentada na Tabela 2.

A *marginalização* é o processo de soma de variáveis, a fim de reduzir o domínio da DPC. Considere a DPC $p(U)$. Caso se deseja $p(X)$, onde $X \subset U$, pode-se somar sobre todas variáveis $U - X$ para se obter $p(X)$, ou seja, $p(X) = \sum_{U-X} p(U)$. Note que a absorção de evidências seguida de marginalização podem ser usadas para gerar TPCs que são respostas para perguntas sobre o sistema modelado.

Exemplo 3. Considere que a pergunta $p(cachorro - fora|ouviu - latido = sim)$ foi feita para a DPC exposta na Tabela 1. Primeiramente, absorve-se a evidência $ouviu - latido = sim$, resultando na DPC $p(cachorro - fora, problemas - intestinais|ouviu - latido = sim)$ como demonstrado no Exemplo 2 e exposto na Tabela 2. Em seguida, deve-se marginalizar a Tabela 2 somando $problemas - intestinais$ a fim de se obter $p(cachorro - fora|ouviu - latido = sim)$. Assim, calcula-se a DPC marginalizada $p(cachorro - fora|ouviu - latido = sim) = \sum_{problemas-intestinais} p(cachorro - fora, problemas - intestinais|ouviu - latido = sim)$ exposta na Tabela 3.

Tabela 3: DPC da Tabela 2 atualizada pela marginalização de $problemas - intestinais$.

cachorro-fora	ouviu-latido	p
não	sim	0,4375
sim	sim	0,5630

Embora raciocinar sobre o sistema usando uma DPC é aparentemente útil e, de fato, apresenta ferramentas robustas como a absorção de evidências e marginalização, grande parte dos problemas envolvem mais variáveis, o que pode inviabilizar a utilização de uma DPC. Como exemplo, apresenta-se três problemas ao se utilizar DPC para modelar e raciocinar sobre um sistema. Primeiro, o problema da aquisição. Uma DPC com 3 variáveis binárias tem até $2^3 = 8$ configurações (linhas) diferentes. Da mesma forma, com 5 variáveis a DPC tem $2^5 = 32$ configurações. Caso os valores da DPC sejam atribuídos por um especialista consultor, o que ocorre na maioria dos casos práticos, estudar 8 casos de configurações para atribuir valores pode até ser plausível. Mas estudar 32 casos pode ser extenuante. Em muitos problemas práticos, o número de variáveis consideradas pode chegar a centenas, o que inviabiliza a análise de um especialista. Segundo, o problema da absorção de evidências. Considerando que o problema da aquisição foi vencido, isto é, conseguiu-se obter a DPC, para se absorver uma evidência sobre o sistema, um número exponencial de divisões pode ser feito, dependendo do número de variáveis na DPC. Terceiro, e último, o problema da marginalização. Mesmo após a absorção de evidência, para marginalizar a DPC a fim de se obter a resposta requerida pode ser necessário uma quantidade exponencial de somas, dependendo do número de variáveis. Esses problemas levaram a rejeição da teoria da probabilidade em aplicações de IA desde a década de 1970. Em 1988, Judea Pearl da University of California, Los Angeles, publicou uma série de trabalhos que viabilizaram o uso da teoria da probabilidade no gerenciamento de incertezas em IA. As novas ideias resumem o que hoje é conhecido como redes Bayesianas.

Independência Condicional

Seja $p(U)$ uma DPC, e X, Y, Z subconjuntos disjuntos de U. Diz-se que Y e Z são independentes com a condição de que seja dado X, denotado $I(Y, X, Z)$, se

$$p(Y, Z|X) = \frac{p(YX) \cdot p(ZX)}{p(X)}, \tag{2.1}$$

onde a *multiplicação* entre duas DPCs p_1 e p_2, denotado $p_1 \cdot p_2$. O teorema de Bayes é a base teórica para a Eq. (2.1) de independência condicional. Daí, o nome rede Bayesiana.

Exemplo 4. Considere a DPC $p(cachorro-fora, problemas-intestinais, ouviu-latido)$ apresentada na Tabela 1. Deseja-se saber se $problemas-intestinais$ é independente de $ouviu-latido$ dado $cachorro-fora$ é sim. Pode-se marginalizar a DPC em $\{cachorro-fora, problemas-intestinais\}$, em $\{problemas-intestinais, ouviu-latido\}$ e, por último, em $\{cachorro-fora\}$ obtendo-se, dessa forma, $p(cachorro-fora, problemas-intestinais)$, $p(problemas-intestinais, ouviu-latido)$ e $p(cachorro-fora)$, respectivamente. Por inspeção, pode-se mostrar que:

$$p(cachorro-fora, problemas-intestinais, ouviu-latido) = \\ = \frac{p(cachorro-fora, problemas-intestinais) \cdot p(problemas-intestinais, ouviu-latido)}{p(cachorro-fora)}.$$

Logo, pode-se afirmar que $I(problemas-intestinais, cachorro-fora, ouviu-latido)$.

Em termos práticos, o Exemplo 4 mostra que, uma vez que se sabe que o $cachorro-fora$ é sim, uma observação em $problemas-intestinais$ não mudará a crença nos valores de $ouviu-latido$, e vice-versa. O uso de independência condicional é de extrema relevância para o entendimento das premissas das RBs, já que a mesma trabalha com fatorizações da DPC original baseada em independências condicionais.

Regra da Cadeia

A *regra da cadeia* é um bem fundamentado operador da Teoria da Probabilidade. Ela pode fatorizar qualquer DPC usando a seguinte definição:

$$p(X|Y) = \frac{p(X,Y)}{p(Y)}. \tag{2.2}$$

Exemplo 5. Considere a DPC $p(cachorro-fora, problemas-intestinais, ouviu-latido)$ apresentada na Tabela 1. Pode-se aplicar a regra da cadeia sobre essa DPC utilizando a Eq. (2.2). Assim:

$$p(cachorro-fora, problemas-intestinais, ouviu-latido) = \\ = p(cachorro-fora) \cdot p(problemas-intestinais|cachorro-fora) \cdot \\ \cdot p(ouviu-latido|cachorro-fora, problemas-intestinais).$$

Aplicando a Regra da Cadeia e a Independência Condicional

A fundamentação teórica e o esboço formal para se provar a exatidão das RBs são baseados na aplicação da regra da cadeia e as independências condicionais. Ao se fatorizar

uma DPC usando a regra da cadeia, alguns desses fatores podem ser desconsiderados, caso haja independência condicional entre as variáveis do domínio. Assim, a fatorização da DPC será menor e, consequentemente, viável de aplicação.

Exemplo 6. Considere a DPC $p(cachorro-fora, problemas-intestinais, ouviu-latido)$ apresentada na Tabela 1. O Exemplo 4 demonstrou que $I(problemas-intestinais, cachorro-fora, ouviu-latido)$. Em seguida, a fatorização dessa DPC foi ilustrada no Exemplo 5. Por definição, dado a independência condicional, pode-se afirmar que

$$p(ouviu-latido|cachorro-fora, problemas-intestinais) = \\ = p(ouviu-latido|problemas-intestinais).$$

Dessa forma, a fatorização da DPC pode ser reescrita de forma mais compacta.

Note que o Exemplo 6 esboça como a DPC pode ser fatorizada em tabelas menores e plausíveis de construção em situações reais, aplicando-se a regra da cadeia a as independências condicionais. Para simplificar e elucidar melhor os conceitos expressos pela Teoria da Probabilidade e, mais especificamente, de independência condicional, pode-se também utilizar uma ferramenta gráfica, como apresentado a seguir.

2.1 Modelagem

Para apresentar a definição formal de RBs, resta apenas a introdução dos gráficos acíclico dirigido (GAD). Em um GAD, para qualquer *vértice* (ou nó) v, não há nenhum *caminho dirigido* (ou aresta dirigida) que comece e termine em v. A Fig. 7 ilustra um GAD com cinco vértices.

Os GADs podem representar o conhecimento expresso pelas DPCs e da interpretação causal para as mesmas. As variáveis de uma DPC podem ser expressas pelos vértices de um GAD, e as setas dirigidas podem representar relação causal entre as variáveis. Além do mais, existem ferramentas formais que comprovam independência condicional entre as variáveis de uma DPC através de análise gráfica em um GAD. D-separation [2] ou m-separation [19, 4] são duas ferramentas de manipulação gráfica para testar independência em um GAD.

Definição. Uma *Rede Bayesiana* (RB) em U é um par $(\mathcal{B}, C)$. $\mathcal{B}$ é um gráficos acíclico dirigido (GAD) com um conjunto de vértices U e C é um conjunto de tabelas de probabilidade condicional (TPC) $\{p(v_i|P(v_i))|v_i \in U\}$, onde $Pa(v_i)$ denota os pais

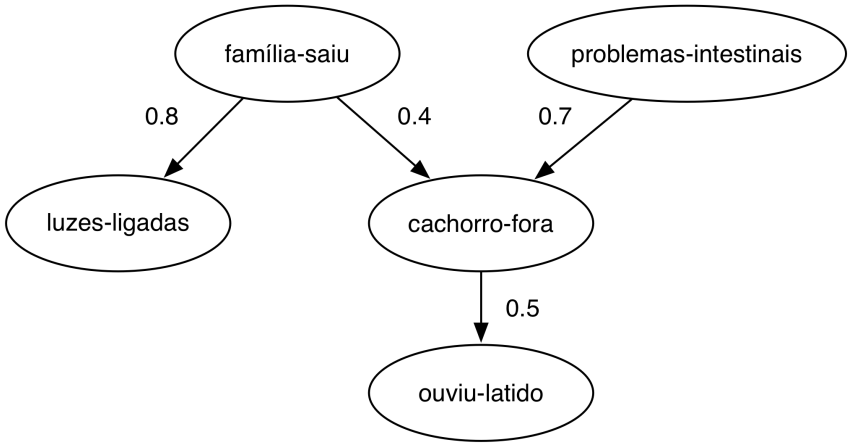

Figura 7: Um GAD representando uma RB com cinco variáveis e cinco TPCs.

(imediatos antecedentes) de $v_i \in \mathcal{B}$.

Se nenhuma confusão for gerada, $\mathcal{B}$ será usado para se referir à uma RB ou a um GAD. Também, XY será usado para denotar $X \cup Y$.

Exemplo 7. A Fig. 7 apresenta uma RB $\mathcal{B}$ com variáveis $U = \{famlia - saiu, luzes - ligadas, cachorro - fora, problemas - intestinais, ouviu - latido\}$. Para fins didáticos, as TPCs foram exibidas sobre os caminhos dirigidos. Note que, como as variáveis são binárias, precisa-se apenas determinar o valor da TPC de apenas um dos domínios. Sendo assim, em $\mathcal{B}$, pode-se interpretar que $familia - saiu$ ou $problemas - intestinais$ podem causar $cachorro - fora$ com 40% de chance e 70%, respectivamente. Também, $cachorro - fora$ pode causar $ouviu - latido$ em 50% das vezes.

O Exemplo 7 é uma expansão do Exemplo 1. Note que o domínio do problema pode ser facilmente expandido na modelagem por RBs. Dessa forma, RB apresenta características escaláveis e de fácil manutenção.

Uma das características principais de uma RB é a possibilidade de se obter partes da DPC de um sistema sem necessariamente precisar descrevê-la. As TPC são menores, mais plausíveis de construção, e, após manipulação, podem gerar novas TPCs que correspondem à partes da DPC original. Esse processo de manipulação sistemática é chamado inferência, como explicado a seguir.

2.2 Inferência

Inferência é o processo de fazer perguntas (*queries*) para a RB. isto é, dado a modelagem de um problema em tabelas de DPC, um GAD e, opcionalmente, evidências sobre o sistema, pode-se perguntar sobre qualquer variável da RB. Diz-se também que responder às perguntas de uma RB é o processo de cálculo de posteriores [20]. O processo de inferência envolve manipulação das TPCs originais para a criação de novas, as quais são resposta para a pergunta.

Um algoritmo de inferência é aquele capaz de manipulas as tabelas, a fim de se obter respostas às perguntas, de forma sistemática e correta. Nem toda manipulação das tabelas originais de uma RB pode gerar tabelas que realmente correspondem à partes da DPC do sistema. Dessa forma, os algoritmos de inferência garantem solidez nas respostas às perguntas, embora seus passos intermediários possam gerar tabelas que não são densidades de probabilidade, chamadas *potenciais*.

Existem diversos algoritmos de inferência em RB, com características próprias que os fazem melhores em determinadas situações. Dentre eles, destacam-se: *Arc-Reversal* (AR) [21, 22], o qual utiliza uma técnica de reversão dos arcos entre as variáveis de um GAD de uma RB a fim de se eliminar as variáveis não importantes para a pergunta em questão; *Lazy Propagation* (LP) [5], onde perguntas são respondidas por passagem de mensagens (potenciais) em uma segunda estrutura gráfica obtida da RB original, chamada *join tree*; e *Variable Elimination* (VE) [4], o qual elimina variáveis indesejadas uma por vez em um métodos iterativo.

Os algoritmos AR e LP possuem complexidade de implementação maior, já que o primeiro envolve complicada manipulação gráfica e o último utiliza uma segunda estrutura gráfica. Por ser mais simples, neste trabalho, o algoritmo de inferência utilizado é o VE. O VE responde à uma pergunta $p(X|Y)$ para uma RB $\mathcal{B}$ da seguinte forma:

- todas variáveis do tipo *barren* são removidas recursivamente, onde v é *barren* [4] se v não tem filhos (imediatos descendentes) e $v \notin XY$,
- todas variáveis independentes dado as evidências são removidas, restando $\mathcal{B}^s$, onde v é *independente dado as evidências* se $I(v, Y, X)$ é verdade em $\mathcal{B}$ usando d-separation,
- construir uma distribuição uniforme $1(b)$ para todas raízes de $\mathcal{B}^s$ que não for raiz de $\mathcal{B}$,
- absorver as evidências $Y = y$ em todas TPCs de $\mathcal{B}^s$,

- escolher uma ordem de eliminação das variáveis σ, através de um *gráfico moralizado* [23], o qual é definido pela troca das setas direcionadas pelas não direcionadas e ligação de todos pares de pais de um filho comum, gerando $\mathcal{B}_m^s$,
- seguindo σ, eliminar a variável v em $\mathcal{B}_m^s$ multiplicando todos potenciais envolvendo v, e então marginalizando v do produto,
- enfim, multiplicar todas tabelas que restarem e normalizar o produto, a fim de se obter $p(X|Y)$.

Embora mais simples que os outros algoritmos de inferência, o VE pode ser mais lento, dado sua recorrência a cada nova pergunta. Para mitigar possível problema de lentidão, sugere-se uma solução de pré-computação [24], a qual cria tabelas antecipadamente a fim de amenizar os cálculos durante a inferência; ou uma solução de reutilização [6], a qual sugere um método no qual cálculos anteriores feitos por VE podem ser utilizados para responder às próximas perguntas.

3 Materiais e Métodos

A construção de um módulo de inteligência possibilitará os testes do método proposto em um ambiente de simulação. O módulo funcionará como um cliente para o simulador grSim. Dessa forma, informações do ambiente de simulação serão enviadas para rede local, para, então, serem processadas no módulo e enviadas de volta ao simulador, também pela rede. Os times básico e BayesBall tem suas implementações em paralelo dentro do módulo de inteligência, portanto, possuem variáveis de estado e decisão independentes.

O BayesBall é implementado na linguagem de programação Java [25], com a ajuda do ambiente de desenvolvimento integrado *NetBeans*. No desenvolvimento do software foram usados padrões de implementação como orientação a objetos e programação paralela, a fim de manter a escalabilidade da aplicação, a segurança durante o crescimento do código fonte e a modularidade, isto é, a capacidade de dividir o sistema em blocos concisos e fechados. Para o acompanhamento do progresso, cópias de segurança e rastreamento de versão, este projeto utilizou um sistema de controle de versão chamado Git [26]. Também, todo o código fonte do sistema BayesBall está disponível na plataforma Github no formato open-source, sob licença GPL [27], a fim de motivar o uso, modificação e distribuição do sistema na comunidade SSL.

A visão geral do funcionamento do BayesBall durante a simulação é apresentada no diagrama da Fig. 8. Cada time implementado no módulo tem suas variáveis locais, atualizações externas e saídas geradas em uma instância independente do BayesBall, chamada *mundo*. Em cada mundo, há três tarefas básicas para o andamento da simulação: atualizar, onde o mundo atualiza suas variáveis sobre o ambiente externo; pensar, no qual o sistema toma decisões baseadas nas novas informações externas; e executar, onde o BayesBall efetua cálculos de como alcançar as decisões propostas previamente. Para atualizar o mundo, o BayesBall primeiramente recebe o pacote de dados da rede local, através do protocolo UDP e encapsulamento Protobuf, tal como no sistema de visão. Em seguida, as variáveis locais representativas do mundo externo são atualizadas, como a posição da bola e posturas dos jogadores. Para pensar sobre seu mundo, antes de tudo, o BayesBall

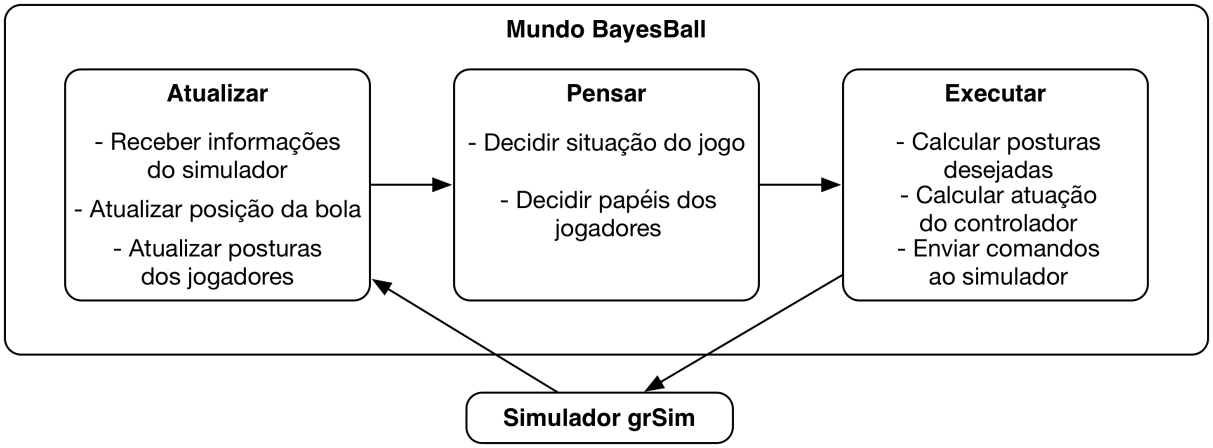

Figura 8: Visão geral da integração entre uma instância do BaysBall e o simulador grSim.

determina a situação corrente da simulação, a qual é baseada no atual comando enviado pelo software do juiz. Após, cada jogador recebe um papel em campo, dentre os disponíveis (goleiro, zagueiro e atacante). Para executar o mundo, o BayesBall primeiramente calcula as posturas desejadas, dado os papéis de cada jogador, decidido anteriormente. Em seguida, calcula a movimentação de cada agente através dos comandos do controlador. Por fim, envia os comandos para cada robô no simulador grSim, através da rede local, usando também o protocolo UDP e encapsulamento Protobuf.

Em sua instância do mundo, cada time pode re-implementar ou sobrescrever qualquer ou parte do fluxo proposto na Fig. 8. Note que a diferença entre a implementação do time básico e o time BayesBall é justamente na decisão de papéis, dentro da etapa pensar. No time básico, essa decisão é fixa para cada agente em campo.

No time BayesBall, a absorção de evidências para atualização da RB se dá por manipulações das variáveis internas do mundo, especificamente, as variáveis representativas do ambiente externo. A inferência se dá de modo a calcular o impacto das evidências no papel de cada jogador em campo. Cada agente fará sua análise individual em uma RB, dessa forma as evidências e inferências de cada um serão personalizadas. A implementação das RBs em Java foi feita com o pacote Hugin Expert, gentilmente cedido pela empresa, para este trabalho, a fins acadêmicos.

Após implementação dos dois times, a validação do método se dará através de partidas consecutivas, sem alteração no ambiente de simulação, entre o time básico e o BaysBall. Espera-se que a flexibilidade na assimilação de papéis aos jogadores, durante a partida, crie mais oportunidades de gol, além de encobrir possíveis fraquezas na defesa do time BayesBall.

3.1 Time Básico

A inteligência do time básico consiste apenas pelas tarefas previstas dos papéis dos jogadores disponíveis. Ou seja, cada agente recebe um papel no início da partida e exerce aquela função por todo o tempo. Os papéis dos jogadores são regras simples que descrevem uma tarefa fixa. Para ser dinâmico, essas regras consideram as variáveis mais atuais do ambiente, alterando assim as posturas desejadas, embora mantendo a mesma tarefa. A seguir, cada papel de jogador é descrito em mais detalhe.

O goleiro tem a intenção de proteger a área do gol se posicionando sempre como barreira à passagem da bola. A regra para o cálculo da postura do goleiro, dado as variáveis atuais do ambiente, é da seguinte forma: (i) calcular uma circunferência de raio r_g que compreende o centro do gol defensor como centro da mesma, (ii) traçar uma reta entre a bola e o centro do gol defensor, (iii) calcular a interseção entre o círculo e a reta, e (iv) posicionar o goleiro no ponto de interseção, com orientação na direção da bola.

O zagueiro exerce função similiar ao goleiro, com a intenção de defender não apenas a área mas também as laterais da região defensiva. Sendo assim, o zagueiro é posicionado seguindo os mesmos passos que o goleiro, com a diferença do raio da circunferência $r_z > r_g$. Observa-se também que r_z deve ser maior que a largura da área de defesa, já que o zagueiro não deve permanecer dentro da área defensiva por muito tempo.

O atacante possui comportamento ganancioso e segue a simples regra de levar (arrastar) a bola até o gol adversário. Dessa forma, propõe-se a seguinte sequência para o posicionamento do atacante: (i) traçar uma reta entre a bola e o centro do gol adversário, (ii) posicionar o atacante sobre a reta, a uma distância da bola, direcionado para o centro do gol adversário, e (iii) aumentar a velocidade do atacante, percorrendo a reta, em direção ao gol adversário. Assim, pode-se simular o comportamento do chute (ausente no simulador) com a aceleração do robô em direção ao gol adversário.

A movimentação tática do time básico é limitada às atividades de cada papel de jogador. Ou seja, a dinâmica dos agentes em campo seguem regras fixas e simples como descrita em cada papel. A distribuição de papéis para cada robô é feita de forma aleatória, no início da partida, seguindo um esquema de formação fixo: um goleiro, quatro zagueiros e um atacante. Os agentes, então, seguem as regras providas de seus papéis invariavelmente durante toda a partida.

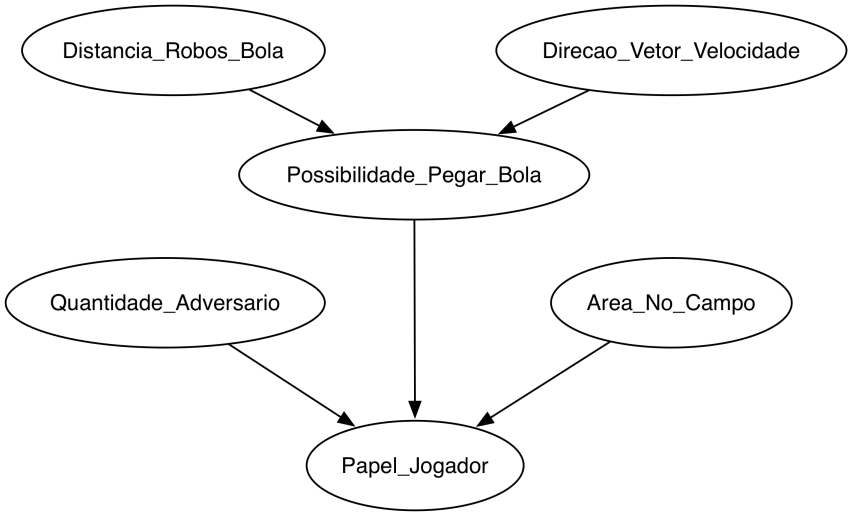

Figura 9: RB proposta para o sistema BayesBall.

3.2 Time BayesBall

O time BaysBall também disponibiliza os três papéis de jogadores descritos para o time básico, são eles o goleiro, zagueiro e atacante. Porém, esse time possui flexibilidade na assimilação de papéis por robôs. Sendo assim, um mesmo robô pode alterar seu papél e, consequentemente, suas regras de posicionamento, durante uma partida. A decisão pelo papel do jogador vem por inferência em uma RB.

A RB para decisão de papel é modelada de forma a considerar o ambiente dinâmico de uma partida do SSL. A ajuda de um especialista também é importante para validação e implementação da RB. O *especialista* [28] é aquele, ou aqueles, com conhecimento superior sobre o domínio do problema a ser modelado. Com a ajuda do especialista, pode-se validar as relações causais de uma RB, assim como decidir a intensidade dessas relações na construção das TPCs. Assim, modelou-se o ambiente de decisão de papéis de jogadores como a RB mostrada na Fig. 9. As TPCs estão expostas na seção Anexo A, para melhor disposição visual.

As variáveis consideradas na RB da Fig. 9 refletem alguma característica do ambiente do jogo, e podem ser facilmente calculadas manipulando as informações do simulador. Todas variáveis, exceto *Possibilidade_Pegar_Bola* e *Papel_Jogador*, são binárias, como explanado a seguir. A variável *Distancia_Robos_Bola* representa quem está mais próximo da bola: o agente em análise ou qualquer outro agente em campo. Essa análise é feita pelo simples cálculo da distância entre cada robô em campo e a bola. Se o agente em análise estiver mais próximo ou não, uma evidência pode ser absorvida pela RB. A

variável $Direcao\_Vetor\_Velocidade$ representa a possibilidade do agente analisado estar em movimento na direção da bola ou contrário a ela. Se for possível deduzir essa informação, uma evidência pode ser absorvida nessa variável. A possibilidade de pegar a bola é ilustrada pela variável $Possibilidade\_Pegar\_Bola$, a qual tem cardinalidade três em suas possibilidades alta, média e baixa. A variável $Quantidade\_Adversario$ abrange a possibilidade de se gerar evidência na seguinte situação: dada uma região de disputa, uma circunferência ao redor da bola, pode-se apontar se há nenhum ou algum adversário no interior da mesma. A $Area\_No\_Campo$ representa de qual lado no campo o agente está: defensivo ou ofensivo. Essa observação pode, então, ser absorvida na RB por essa variável. Por fim, a variável $Papel\_Jogador$ tem cardinalidade três, compreendendo os papéis de jogador goleiro, zagueiro e atacante. Note que, dado as possíveis evidências sobre as outras variáveis, pode-se fazer inferência sobre $Papel\_Jogador$ para ajudar na escolha do papel do agente analisado.

Exemplo 8. Considere um agente do time BayesBall, durante uma partida, utilizando a RB da Fig. 9 para determinação do seu papel de jogador. Nesse momento, através das informações providas pelo simulador, pode-se observar que o agente está na direção da bola e na área defensiva do campo. Essas duas informações podem gerar duas evidências: $Direcao\_Vetor\_Velocidade = na\_direcao\_bola$ e $Area\_No\_Campo = defesa$. Se atualizarmos todas tabelas de DPC da RB e perguntarmos $p(Papel\_Jogador \mid Direcao\_Vetor\_Velocidade = na\_direcao\_bola, Area\_No\_Campo = defesa)$, o VE construirá uma tabela da qual se conclui que o papel goleiro tem valor 0,532, zagueiro 0,368 e atacante 0,1. Dessa forma, o agente analisado pode receber o papel goleiro.

A RB da Fig. 9 modela uma estrutura de decisão de papel de um agente em campo durante uma partida do SSL. Como ilustrado no Exemplo 8, a RB pode ser usada no raciocínio sólido, sistematizado e correto sobre informações aparentemente conflitantes e ausentes. As ferramentas de absorção de evidências e inferência são usadas a fim de auxiliar na decisão do papel de jogador, para o agente analisado.

4 Resultados e Discussões

O controlador para a movimentação dos agentes em campo foi simulado no ambiente de desenvolvimento *Scilab*, através de um sistema em malha fechada, como o apresentado na Fig. 4. Para ajuste fino dos parâmetros do controlador, variou-se a constante β da Eq. (1.1) em uma pequena faixa. A cada variação, efetuou-se a simulação do BayesBall a fim de averiguar a estabilidade (suavidade) do movimento. Para valores baixos de β, em uma faixa de 1 a 15, aproximadamente, os robôs descreveram trajetórias mais lentas, embora alcançavam a postura desejada de forma definitiva, isto é, sem erra-la e corrigi-la em seguida. Para valores altos de β, em uma faixa acima de 20, aproximadamente, os robôs movimentavam mais rápidos, embora erravam a postura desejada com frequência, o que acarretava deslocamento extra para o conserto da postura.

Durante a simulação, observou-se que os zagueiros e o atacante cometeram mais erros ao posicionar para a postura desejada. Por exemplo, a Fig. 10 ilustra o posicionamento de um agente, no papel de zagueiro, para uma postura desejada distante da postura atual do agente. Com as oscilações evidentes no posicionamento do agente, fica à mostra as sucessivas correções do controlador a fim de posicionar o mesmo. Essas oscilações não são tão intensas no posicionamento de um agente no papel de goleiro, na mesma situação de posicioná-lo em uma postura discrepante da atual. Dessa forma, evidencia-se nesse fato a influência da faixa de movimentação disponível para cada papel de jogador. O goleiro e o zagueiro possuem regras de posicionamento similares, mas o primeiro tem raio de abrangência menor que o último. Ou seja, com uma maior faixa de movimentação disponível, o zagueiro exibe maior inércia quando próximo da postura desejada, para um mesmo valor de β do goleiro.

A diferenciação entre os parâmetros do controlador para cada papel de jogador propiciou um ajuste fino necessário para evitar frequentes erros no posicionamento dos agentes. Os valores de β para cada papel foi determinado empiricamente, observando-se a movimentação de um agente no papel analisado se posicionar para uma postura discrepante da que estava incialmente. Assim, de forma empírica, para o goleiro $\beta = 10$, zagueiro $\beta = 1$

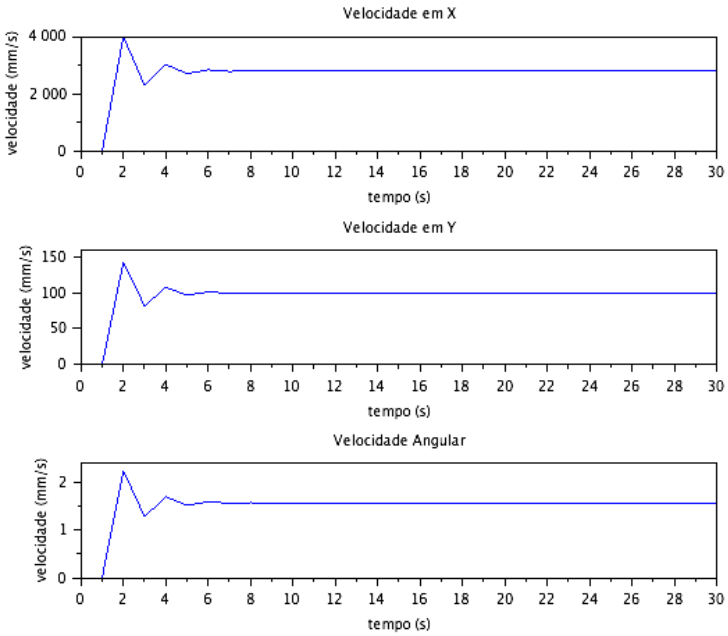

Figura 10: Posicionamento de um zagueiro para uma postura desejada distante do ponto atual do mesmo, utilizando $\beta = 25$.

e atacante $\beta = 5$. Esses valores, no simulador, proporcionaram movimentos mais suaves e precisos.

O time BayesBall utilizou uma RB, ilustrada na Fig. 9, implementada com a API Java do sistema Hugin Lite, provido pela empresa Hugin Expert. A modelagem da RB foi feita na interface gráfica do software Hugin Lite, onde foi possível construir, dar manutenção e simular as bases de dados da RB. A interface também oferece suporte à RB orientada à objetos e aprendizagem automática. Algumas estatísticas da RB gerada para este trabalho é apresentado na Tabela 4.

Após modelagem da RB, através do GAD e das TPC, efetuou-se alguns testes, com observações aleatórias, a fim de se conferir os resultados inferidos para papéis de jogadores.

Tabela 4: Estatísticas da RB implementada para este trabalho, ilustrada na Fig. 9.

Quantidades	Valor
Total de nós	6
Nós de probabilidade	6
Arestas	5
Total de tabelas para os nós discretos	56
Nós raízes	5
Nós folhas	1

Tabela 5: Matriz confusão para casos de validação da RB.

	goleiro	zagueiro	atacante
goleiro	5	1	0
zagueiro	0	6	0
atacante	0	0	6

A princípio, as respostas após absorção de evidências e inferência não condiziam com o esperado, evidenciando a necessidade de ajuste fino das probabilidades expressas nas tabelas. Para aprendizado, utilizou-se a técnica conhecida como *expectation–maximization algoritm* (EM) [29] para ajuste das probabilidades condicionais em caso de dados ausentes. O especialista foi consultado novamente, a fim de se gerar um conjunto de situações ideais de jogo, chamados casos.

O homologação da RB se deu pela avaliação, por um especialista, da resposta da RB dado situações reais de jogo. Inferiu-se da RB alguns casos que ilustram possíveis situações no simulador, isto é, evidências que poderiam ser observadas em uma situação real de jogo, e a saída mais provável de uma pergunta sobre o papel do jogador. De forma geral, após 18 casos analisados, a RB apresentou um desempenho de 94,45% de acertos, na opinião do especialista. Esse resultado é ilustrado pela matriz confusão da Tabela 5.

Para corroboração dos acertos feito pela RB, o placar da partida entre o time Bayes-Ball e o time básico foi de 1 x 0. A movimentação em campo era limitada pelas regras de posicionamento dos papéis, o que enfraqueceu a dinâmica da partida, exibindo muitas situações de bola travada entre os jogadores, laterais em excesso e colisões que atrapalhavam o andamento. Porém, ainda assim, pode-se evidenciar a influência da troca de papéis no dinamismo do time BayesBall. Em diversas situações, a troca de papéis possibilitou que novas situações fossem exploradas, finalizando, em uma delas, em um gol a favor do time BayesBall.

5 Conclusões

Esse trabalho propôs a implementação de um time simulado para a categoria SSL da competição de futebol de robôs RoboCup. Um dos problemas enfrentados no futebol de robôs é a possível flexibilização na troca de papéis entre os jogadores, durante a partida. A fim de raciocinar sobre as incertezas envolvidas na decisão de qual papel cada jogador deve exercer durante a partida, esse trabalho propõe o uso de RBs.

As variáveis envolvidas no processo de decisão dos papéis dos jogadores podem conflitar ou se ausentar em diversas situações de jogo. Dessa forma, a manipulação coerente das informações disponíveis no ambiente (evidências sobre o sistema) fizeram as RBs acertarem 94,45% dos casos analisados por um especialista. Corroborando com o resultado positivo, o time BayesBall venceu o time básico em uma partida livre no ambiente de simulação.

Os padrões de desenvolvimento utilizados durante a implementação do módulo de inteligência, provaram-se de grande valia quando a base de código aumentou de forma considerável. A execução de processos paralelos fizeram o sistema BayesBall mais rápido, quando o mesmo já apresentava sinais de lentidão. O uso de orientação a objetos manteve o código estável nos seus diversos estágios e escalável para futuras implementações.

De forma geral, os objetivos propostos foram alcançados de forma satisfatórias. Pequenos problemas durante a implementação, como os ajustes finos no controlador e nas TPC da RB, foram contornados sem prejudicar o andamento do projeto. Para trabalhos futuros, propõe-se o desenvolvimento de mais papéis disponíveis, além do aperfeiçoamento dos atuais, a fim de dar mais dinamismo às partidas e proporcionar melhor rotatividade entre os jogadores. Espera-se que a maior disponibilidade de papéis e a melhoria das regras dos mesmos criem situações de jogo ainda mais favoráveis e defesas mais aprimoradas. Também, um possível nó para a proposta RB seria a situação de jogo (defensivo, ofensivo, ou variações dos mesmos), baseado não apenas nos comandos do juiz mas também na disposição espacial dos agentes em campo.

Referências

[1] KITANO, H. et al. Robocup - a challenge problem for ai. *AI Magazine*, v. 18, n. 1, p. 73–85, 1997.

[2] PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*: Morgan Kaufmann, 1988.

[3] OLIVEIRA, J. de S. *Introduction to Bayesian Networks with Jhonatan de Souza Oliveira*. 04 2014. Disponível em: <http://machinelearningmastery.com/introduction-to-bayesian-networks-with-jhonatan-de-souza-oliveira/>.

[4] ZHANG, N. L.; POOLE, D. A simple approach to bayesian network computations. In: *Proceedings of the Tenth Canadian Artificial Intelligence Conference*, 1994. p. 171–178.

[5] MADSEN, A. L.; JENSEN, F. Lazy propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, v. 113, n. 1-2, p. 203–245, 1999.

[6] BUTZ, C. J.; OLIVEIRA, J. de S. O.; MADSEN, A. Bayesian network inference using marginal trees. In: *Seventh European Workshop on Probabilistic Graphical Models (PGM)*, 2014. p. 81–96.

[7] GUPTA, G. S. et al. The special cubs : Multi- agent co-operative systems for playing robot soccer. In: *FIRA Robot World Cup France '98 Proceedings*, 1998. p. 33–35.

[8] CHEN, B.; ZHANG, A.; CAO, L. Autonomous intelligent decision-making system based on bayesian {SOM} neural network for robot soccer. *Neurocomputing*, v. 128, n. 0, p. 447 – 458, 2014. ISSN 0925-2312.

[9] BONILLA, A. *Simultaneous Planning and Acting in Robotic Soccer*. Dissertação (Mestrado) — University of Puerto Rico, 2014.

[10] MESSOM, C. Robot soccer – sensing, planning, strategy and control, a distributed real time intelligent system approach. In: *International Symposium on Artificial Life and Robotics, AROB*, 1998. p. 422–426.

[11] SNG, H.; GUPTA, G.; MESSOM, C. Strategy for collaboration in robot soccer. In: *The First IEEE International Workshop on Electronic Design, Test and Applications*, 2002. p. 347–351.

[12] CRUZ, R. R. *Uso de Aprendizado de Máquina para Classificação de Situações de Jogo em Competições de Futebol de Robôs*. Dissertação (Mestrado) — Universidade Federal de Viçosa, 2012.

[13] COMMUNITY. *Laws of the RoboCup Small Size League 2014*, 2014.

[14] KATSUHIKO, O. *Modern Control Engineering*. 4th. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. (0130609072).

[15] MARTINS, F. *Introdução ao Controle de Robôs Móveis*. Disponível em: <http://www.slideshare.net/felipenm/introduo-ao-controle-de-robs-mveis>.

[16] MONAJJEMI, V.; KOOCHAKZADEH, A.; GHIDARY, S. grsim – robocup small size robot soccer simulator. In: RÖFER, T. et al. (Ed.). *RoboCup 2011: Robot Soccer World Cup XV*: Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7416). p. 450–460. ISBN 978-3-642-32059-0.

[17] MCCLOSKEY, S. *Probabilistic Reasoning and Bayesian Networks*. 01 2000. Disponível em: <http://www.cim.mcgill.ca/ scott/RIT/researchPaper.html>.

[18] KOLLER, D.; FRIEDMAN, N. *Probabilistic Graphical Models: Principles and Techniques*: MIT Press, 2009.

[19] LAURITZEN, S. L. et al. Independence properties of directed markov fields. *Networks*, v. 20, p. 491–505, 1990.

[20] SPIEGELHALTER, D. D.; RICE, K. *Bayesian statistics*. Disponível em: <www.scholarpedia.org/article/Bayesianstatistics>.

[21] OLMSTED, S. *On Representing and Solving Decision Problems*. Tese (Doutorado) — Stanford University, 1983.

[22] SHACHTER, R. D. Evaluating influence diagrams. *Operations Research*, v. 34, n. 6, p. 871–882, 1986.

[23] LAURITZEN, S. L.; SPIEGELHALTER, D. J. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, v. 50, p. 157–244, 1988.

[24] COZMAN, F. G. Generalizing variable elimination in Bayesian networks. In: *Workshop on Probabilistic Reasoning in Artificial Intelligence*Atibaia, Brazil: , 2000.

[25] CAELUM. *Java e Orientação a Objetos*, Novembro 2014.

[26] GIT: Version Controlling. Disponível em: <http://git-scm.com>.

[27] COMMUNITY. *General Public License*. 2014. Disponível em: <http://www.gnu.org/copyleft/gpl.html>.

[28] CASTILLO, E.; GUTIÉRREZ, J.; HADI, A. *Expert Systems and Probabilistic Network Models*: Springer, 1997.

[29] LAURITZEN, S. The em algorithm for graphical association mod- els with missing data. In: *Computational Statistics and Data Analysis*, 1995. v. 19, n. 2, p. 191–201.

ANEXO A – Tabelas de Distribuição de Probabilidade Condicional

As seguintes TPCs são as originalmente projetadas pelo especialista. As TPCs, porém, foram alteradas iterativamente pelo algoritmo de aprendizado, a fim de adequar as TPCs originais à algumas novas situações de jogo. As TPCs modificadas não são exibidas neste trabalho.

Tabela 6: Distância entre o robô e a bola.

$Distancia\_Robos\_Bola$	p
$maior\_chance\_pegar\_bola$	0,40
$menor\_chance\_pegar\_bola$	0,60

Tabela 7: Direção do vetor velocidade.

$Direcao\_Vetor\_Velocidade$	p
$na\_direcao\_bola$	0,40
$oposto\_bola$	0,60

Tabela 8: Quantidade de adversários.

$Quantidade\_Adversario$	p
$zero$	0,30
$maior\_que\_zero$	0,70

Tabela 9: Área no campo.

$Area\_No\_Campo$	p
$defesa$	0,60
$ataque$	0,40

Tabela 10: Possibilidade de pegar a bola.

$Possibilidade\_Pegar\_Bola$	$Distancia\_Robos\_Bola$	$Direcao\_Vetor\_Velocidade$	p
$baixa$	$menor\_chance\_pegar\_bola$	$na\_direcao\_bola$	0,3
$media$	$menor\_chance\_pegar\_bola$	$na\_direcao\_bola$	0,5
$alta$	$menor\_chance\_pegar\_bola$	$na\_direcao\_bola$	0,2
$baixa$	$menor\_chance\_pegar\_bola$	$oposto\_bola$	0,6
$media$	$menor\_chance\_pegar\_bola$	$oposto\_bola$	0,3
$alta$	$menor\_chance\_pegar\_bola$	$oposto\_bola$	0,1
$baixa$	$maior\_chance\_pegar\_bola$	$na\_direcao\_bola$	0,1
$media$	$maior\_chance\_pegar\_bola$	$na\_direcao\_bola$	0,2
$alta$	$maior\_chance\_pegar\_bola$	$na\_direcao\_bola$	0,7
$baixa$	$maior\_chance\_pegar\_bola$	$oposto\_bola$	0,3
$media$	$maior\_chance\_pegar\_bola$	$oposto\_bola$	0,5
$alta$	$maior\_chance\_pegar\_bola$	$oposto\_bola$	0,2

Tabela 11: Papel do jogador.

Papel_Jogador	*Quantidade_Adversario*	*Possibilidade _Pegar_Bola*	*Area_No_Campo*	*p*
goleiro	*zero*	*baixa*	*defesa*	0,3
zagueiro	*zero*	*baixa*	*defesa*	0,6
atacante	*zero*	*baixa*	*defesa*	0,1
goleiro	*zero*	*media*	*defesa*	0,2
zagueiro	*zero*	*media*	*defesa*	0,7
atacante	*zero*	*media*	*defesa*	0,1
goleiro	*zero*	*alta*	*defesa*	0,1
zagueiro	*zero*	*alta*	*defesa*	0,8
atacante	*zero*	*alta*	*defesa*	0,1
goleiro	*zero*	*baixa*	*ataque*	0,1
zagueiro	*zero*	*baixa*	*ataque*	0,2
atacante	*zero*	*baixa*	*ataque*	0,7
goleiro	*zero*	*media*	*ataque*	0,1
zagueiro	*zero*	*media*	*ataque*	0,1
atacante	*zero*	*media*	*ataque*	0,8
goleiro	*zero*	*alta*	*ataque*	0,0
zagueiro	*zero*	*alta*	*ataque*	0,1
atacante	*zero*	*alta*	*ataque*	0,9
goleiro	*maior_que_zero*	*baixa*	*defesa*	0,8
zagueiro	*maior_que_zero*	*baixa*	*defesa*	0,1
atacante	*maior_que_zero*	*baixa*	*defesa*	0,1
goleiro	*maior_que_zero*	*media*	*defesa*	0,8
zagueiro	*maior_que_zero*	*media*	*defesa*	0,1
atacante	*maior_que_zero*	*media*	*defesa*	0,1
goleiro	*maior_que_zero*	*alta*	*defesa*	0,6
zagueiro	*maior_que_zero*	*alta*	*defesa*	0,3
atacante	*maior_que_zero*	*alta*	*defesa*	0,1
goleiro	*maior_que_zero*	*baixa*	*ataque*	0,2
zagueiro	*maior_que_zero*	*baixa*	*ataque*	0,5
atacante	*maior_que_zero*	*baixa*	*ataque*	0,3
goleiro	*maior_que_zero*	*media*	*ataque*	0,1
zagueiro	*maior_que_zero*	*media*	*ataque*	0,5
atacante	*maior_que_zero*	*media*	*ataque*	0,4
goleiro	*maior_que_zero*	*alta*	*ataque*	0,0
zagueiro	*maior_que_zero*	*alta*	*ataque*	0,5
atacante	*maior_que_zero*	*alta*	*ataque*	0,5