MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
(Real Academia de Artilharia, Fortificação e Desenho/1792)
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Jan Segre
Victor Bramigk

Heurística Estática para Times Cooperativos de Robôs

Rio de Janeiro
Março de 2014

Instituto Militar de Engenharia

Heurística Estática para Times Cooperativos de Robôs

Iniciação à Pesquisa apresentada ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.
Orientador: Paulo F. F. Rosa - Ph.D

Rio de Janeiro
Março de 2014

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

xxxx Segre, J., Bramigk, V.
Heurística Estática para Times Cooperativos de Robôs / Jan Segre, Victor Bramigk : Instituto Militar de Engenharia, 2014.

Iniciação à Pesquisa (IP) - Instituto Militar de Engenharia - Rio de Janeiro, 2014.

1. Engenharia da computação. 2. Redes Neurais. 3. Lógica Nebulosa. 4. Otimização da Colônia de Formigas. Instituto Militar de Engenharia.

Instituto Militar de Engenharia

Jan Segre
Victor Bramigk

Heurística Estática para Times Cooperativos de Robôs

Iniciação à Pesquisa apresentada ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.

Orientador: Paulo F. F. Rosa - Ph.D

Aprovado em 27 de março de 2014 pela seguinte Banca Examinadora:

Paulo F. F. Rosa - Ph.D
Orientador

Julio Cesar Duarte - D.Sc. do IME

Ricardo Choren Noya - Ph.D do IME

Rio de Janeiro
Março de 2014

Resumo

O objetivo deste trabalho é prever como um time de futebol de robôs irá se comportar baseado somente nas posições e orientações de um conjunto discreto de amostras. Para isso, foram estudados os métodos da ACO (Ant Colony Optimization), SA (Simulated Annealing), Algorítimo Genético, Lógica Nebulosa e Redes Neurais. A partir do estudo detalhado desse algoritmos definiu-se duas linhas principais de ação para a solução do problema: uma baseada em Logica Nebulosa e a outra baseada em Redes Neurais. Após um estudo mais aprofundado deseja-se implementar um processo de otimização em ambos os algorítimos para que o resultado seja refinado.

Abstract

The main objective of this work is predicting how a robot soccer team whill behave, based on a set of positions and orientations of a discrete sample. For that, some heuristics were studied: ACO (Ant Colony Optimization), SA (Simulated Annealing), GA (Genetic Algorithm), and Neural Networks. Based on the detailed study of these algorithms two branches were defined as candidate solutions: one based on the Neural Network heuristics, and the other based on Fuzzy logic. After a deeper study on of these methods it's desirable to implement an algori- thm which will refine the result.

Sumário

Lista de ilustrações

Lista de abreviaturas e siglas

ACO	*Ant Colony Optimization*
IA	Inteligência artificial
RoboCup	*Robot World Cup Initiative*
SA	*Simulated Annealing*
SAM	*Standard Addition Method*
SSL	*Small Size League*

1 Introdução

Cooperação é uma relação de ajuda mútua entre indivíduos e/ou entidades, no sentido de alcançar objetivos em comum, utilizando métodos mais ou menos consensuais. Projetar robôs para trabalhar juntos não é uma tarefa trivial. A tarefa se torna mais difícil ainda quando os robôs devem ser autônomos. Nos trabalhos em que o domínio da aplicação é o futebol de robôs, é comum a idéia de se distribuir papéis dinamicamente entre os membros que compõem a equipe. Este tipo de modelo em um ambiente cooperativo se torna menos complexo quando todos os membros tem características em comum, não havendo especialização entre eles. Adicionalmente, é comum a adoção de objetivos globais e locais ao sistema.

1.1 *RoboCup Small Size League*

A idéia de robôs jogando futebol foi mencionada pela primeira vez pelo professor Alan Mackworth (*Univercity of British Columbia*, Canadá) em um artigo intitulado *"On Seeing Robots"*, apresentado no *Vision Interface 92* e posteriormente publicado em um livro chamado *Computer Vision: System, Theory and Applications*. Independentemente, um grupo de pesquisadores japoneses organizou um *Workshop* no *Ground Challange in Artificial Inteligence*, em Outubro de 1992, Tóquio, discutindo e propondo problemas que representavam grandes desafios. Esse *Workshop* os levou a sérias discussões sobre usar um jogo de futebol para promover ciência e tecnologia. Estudos foram feitos para analisar a viabilidade dessa idéia. Os resultados desses estudos mostram que a idéia era viável, desejável e engloba diversas aplicações práticas. Em 1993 um grupo de pesquisadores incluindo Minoru Asada, Yasuo Kuniyoshi e Hiroaki Kitano, lançaram uma competição robótica chamada de Robot *J-League* (fazendo uma analogia à *J-League*, nome da Liga Japonesa de Futebol Profissional). Em um mês vários pesquisadores já se pronunciavam dizendo que a iniciativa deveria ser estendida ao âmbito internacional. Surgia então a *Robot World Cup Initiative* (RoboCup).

RoboCup é uma competição destinada a desenvolver os estudos na área de robótica e inteligência artificial (IA) através de um competição amigável. Além disso, ela tem como objetivo até 2050, desenvolver uma equipe de robôs humanóides totalmente autônomos capazes de derrotar a equipe campeã mundial de futebol humano. A competição possui várias modalidades. Neste trabalho será analizada a *Small Size Robot League* (SSL), também conhecida como F180. De acordo com as regras da SSL, as equipes devem ser compostas por 6 robôs, sendo um deles o goleiro. O goleiro deve ser designado antes do

início do jogo. Durante o jogo nenhuma interferência humana é permitida com o sistema de controle dos robôs. É fornecido aos times um sistema de visão global e esses controlam seus robôs com máquinas próprias. O sistema de controle dos robôs geralmente é externo, recebe os dados de um conjunto de câmeras localizada acima do campo, processa os dados, determina qual comando deve ser executado em cada robô e envia este comando através de ondas de rádio aos robôs. Embora seja permitido que as equipes utilizem sistemas próprios de visão, a maioria das equipes utiliza a visão centralizada.

1.2 Objetivos

A pesquisa tem por finalidade realizar um estudo sobre métodos para que agentes controláveis possam interagir de maneira eficiente com agentes do time inimigo, que não são controláveis. A pesquisa se propoem a realizar esse aprendizado dos agentes inimigos baseado em gravações coletadas da visão e juiz de partidas, também conhecidas como *logs*, para posteriormente serem incorporados ao sistema de inteligência da equipe de futebol de robôs do Laboratório de Robótica, denominada RoboIME.

1.3 Justificativa

O futebol de robôs, problema padrão de investigação internacional, reúne grande parte dos desafios presentes em problemas do mundo real a serem resolvidos em tempo real. As soluções encontradas para o futebol de robôs podem ser estendidas, possibilitando o uso da robótica em locais de difícil acesso para humanos, ambientes insalubres e situações de risco de vida iminente.

Há diversas novas áreas de aplicação da robótica, tais como exploração espacial e submarina, navegação em ambientes inóspitos e perigosos, serviço de assistência médica e cirúrgica, além do setor de entretenimento, podem beneficiar-se do uso de sistemas multi-robôs. Nestes domínios de aplicação, sistemas de multirobôs deparam-se sempre com tarefas muito difíceis de serem efetuadas por um único robô. Um time de robôs pode prover redundância e contribuir cooperativamente para resolver o problema em questão. Com efeito, eles podem resolver o problema de maneira mais confiável, mais rápida e mais econômica, quando comparado com o desempenho de um único robô.

Devido a grande complexidade do problema de interação com humanos, faz-se necessário que os robôs sejam dotados de uma capacidade de aprendizado para facilitar a interação desses com o mundo real. Isso é relevante tanto para aplicações industriais quanto para aplicações em resgates e militares. Isso diminui a necessidade de modelagem exata dos ambientes em que os sistemas robóticos serão introduzidos e permite que a

adaptação a ambientes complexos seja realizada através da exposição destes sistemas às possíveis situações de trabalho. Através da incorporação do sistema de aprendizagem, situações não consideradas podem ser incorporadas ao algoritmo de controle dos robôs e permitir que esses reajam de maneira mais eficiente em futuras cituações semelhantes.

1.4 Metodologia

Para atingir os objetivos propostos será seguida a seguinte metodologia. Inicialmente o problema a ser investigado será definido formalmente, utilizando-se de definições e teoremas.

Posteriormente a bibliografia é revisada. São evidenciados os métodos comumente utilizados para a abordagem do problema mais geral de classificação. Também são analisádos trabalhos aplicados especificamente à SSL.

A seguir são analisadas as heurísticas levantadas durante a revisão da bibliografia. Cada uma delas é descrita. Após descrição sumária de de cada heurística, são apresentados os respectivos exemplos de cada aplicação. Ao fim de cada exemplo são apresentados como as respectivas heurísticas se encaixam na resolução do problema.

A seguir são apresentadas abordagens envolvendo as heurísticas introduzidas anteriormente. Cada abordagem relaciona no mínimo uma dessas heurísticas. Ao final de cada seção é apresentado uma metodologia a ser aplicada na resolução do problema. Uma análise das características de cada abordagem é feita ao final da descrição de cada abordagem.

Posteriormente, utilizando-se das análises de cada abordagem feitas anteriormente, uma das abordagens é selecionada. Essa análise é estudada mais profundamente. São descritos possíveis algorítmos para implementar essa abordagem. Também são analisados os *Frameworks* mais comumente utilizados, levando os pontos positivos e negativos dos mais utilizados em problemas de IA. Também são desenvolvidos métricas para análise da abordagem selecionada.

1.5 Estrutura do Trabalho

No capítulo 2 o problema a ser resolvido é definido formalmente.

No capítulo 3 a bibliografia é revisada.

No capítulo 4 são apresentados tutoriais relacionados às heurísticas comumente utili-

zadas em problemas de classificação.

No capítulo 5 são descritas possíveis abordagens a serem seguidas para resolução do problema.

No capítulo 6 o cronograma de estudo é apresentado.

No capítulo 7 são apresentadas as principais conclusões atingidas até o momento.

2 Definição do Problema

O problema do time de futebol de robôs que será modelado neste capítulo é baseado no modelo de planejador apresentado em (ZICKLER, 2010) e na teoria de agentes descrita em (RUSSELL; NORVIG, 2004).

2.1 Arquitetura

A arquitetura as ser considerada é baseada em (AMARAL; ALMEIDA, 2011). A *RoboCup Small Size League* (SSL) envolve problemas de diversas áreas da engenharia. Logo, como o objetivo de facilitar compreensão do problema, a arquitetura a ser considerada é apresentada na figura 1. Essa arquitetura é composta pelos seguintes sistemas:

- Cameras Visão: Conjunto de câmeras firewire que captura as imagens do campo e as envia para a SSL-Vision
- Comunicação: Módulo responsável por receber os parâmetros dos motores e enviar o comando via onddas de radio para os robôs
- Execução: Módulo responsável por realizar a tomada de decisões em baixo nível de quais ações os robôs devem realizar a partir da decisão tomada pelo módulo de Inteligência
- Inteligência: Módulo responsável por realizar a tomada de decisão em alto nível de quais ações os robôs devem realizar tendo auxílio de um módulo de Simulação
- Mundo Real: Mundo real, onde os agentes agem
- Referee-Box: *Software* padronizado pela Robocup para que as regras da competição sejam cumpridas sem que haja intervenção humana excessiva durante uma partida
- Simulação: Módulo do software do time responsável por simular o ambiente da partida, tendo como entrada os parâmetros do mundo real
- Software Time 1/2: *Software* do time 1/2
- SSL-Vision: *Software* padronizado pela Robocup que permite a integração com uma conjunto de câmeras *firewire* que capturam imagens do campo e as processa, extraindo informações sobre os objetos na imagem

- Time 1/2: Time de robôs que executa os comandos recebidos pelo sistema de transmissão do time 1/2
- Transmissão 1/2: sistema de transmissão do time 1/2
- World Model: Módulo responsável por modelar o mundo e dar confiabilidade aos dados que serão enviados ao módulo de Inteligência e são oriundos do Referee-Box e SSL-Vision

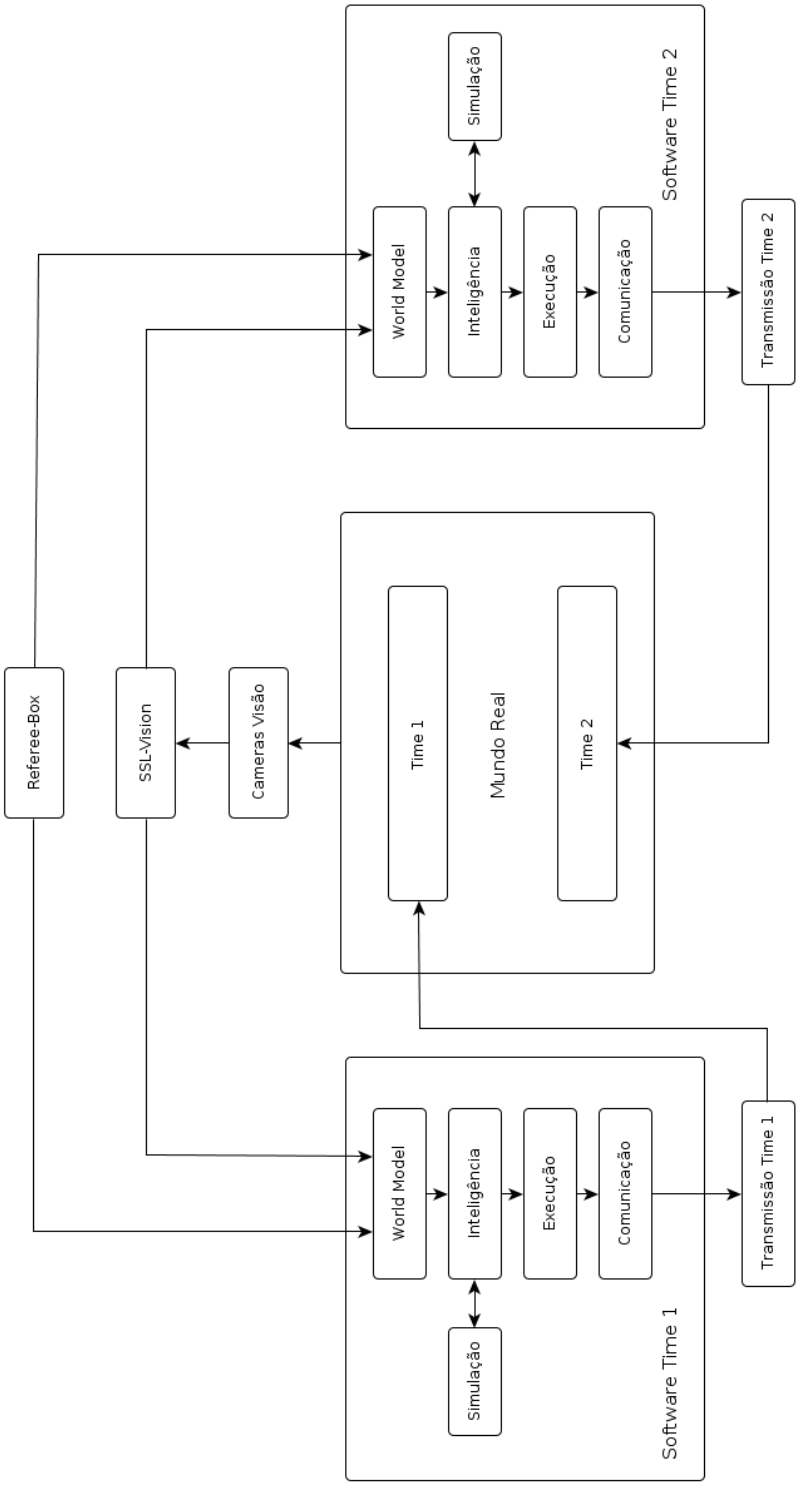

Figura 1 – Arquitetura básica da SSL

Apesar de modelar a maioria dos sistemas atualmente empreganos atualmente na SSL, existem alguns problemas na arquitetura descrita anteriormente. Entretanto, são necessárias algumas definições apresentadas nas próximas secções para que eles possam ser discutidos.

2.2 Definições

Definição 2.2.1 (Corpo Rígido)**.** *Um corpo rígido r é definido por dois subconjuntos disjuntos de parâmetros $r = \langle \hat{r}, \bar{r} \rangle$ onde:*

- $\hat{r} = \langle \alpha, \beta, \gamma, \omega \rangle$*, que são os parâmetros de estado mutaveis, respectivamente: posição ($\mathbb{R}^3$), orientação($\mathbb{R}^3$), velocidade linear($\mathbb{R}^3$), velocidade angular ($\mathbb{R}^3$)*
- $\bar{r}$ *: parâmetros imutaveis corpo que descrevem sua natureza fixa e que permanece constante ao longo do curso de planejamento.*

Exemplos de parâmetros considerados nesta modelagem imutaveis são: coeficiente de atrito estático e dinâmico, descrição $3D$ do corpo (por exemplo, através de um conjunto de primitívas $3D$), centro de massa no referencial do corpo, coeficiente de restituição, coeficiente de amortecimento linear e angular. Note que a matrix de rotação $R \in \mathbb{R}^{3\times3}$) gerada a partir da rotação em torno de um vetor unitário direção $d \in \mathbb{R}^3$ de $\theta \in \mathbb{R}$ radianos satisfaz a equação $R = exp\,(\beta)$ (onde $\beta = d.\theta \in \mathbb{R}^3$) (MURRAY; LI; SASTRY, 1994).

Definição 2.2.2 (Bola)**.** *Bola é um corpo rígido $\hat{b}$ no qual somente as componentes $\langle x, y \rangle$ do o parâmetro $\hat{b}.\alpha$ são observáveis. De acordo com acordo com a aquitetura considerada para a SSL descrita na figura 1, tem-se que a partir de uma sequênicia de quadros é possível obter um valor estimado para o parâmetro $\hat{b}.\gamma$ a partir do intervalo entre os dados recebidos da SSL-Vision e da equação $\gamma \approxeq \frac{\Delta\alpha}{\Delta t}$. Entretanto, uma vez que a componente z de $\hat{b}.\alpha$ não é observável, $\hat{b}.\gamma.z$ não pode ser estimada a partir do intervalo entre os dados recebidos da SSL-Vision. Semelhantemente, uma vez que o parâmetro $\hat{b}.\beta$ também não pode ser observado, não se pode estimar o valor de $\hat{b}.\omega$ com exatidão.*

Definição 2.2.3 (Robô)**.** *Robô rob é um conjunto de sistemas compostos de corpos rígidos, hardware e firmware. São eles:*

- *Drible: imprime um torque a bola*
- *Chute baixo: imprime uma força a bola $\hat{b}$ e possivelmente um torque, com o objetivo de alterar as componentes $\langle x, y \rangle$ do parâmetro $\hat{b}.\gamma$*

- *Chute alto: Chute baixo: imprime uma força a bola* $\hat{b}$ *e possivelmente um torque, com o objetivo de alterar as componentes* $\langle x, y, z \rangle$ *do parâmetro* $\hat{b}.\gamma$*, com* $\hat{b}.\gamma_z \neq 0$
- *Receptor: rebe comandos enviados pelo sistema de transmissão de seu respectivo time*
- *Sistema de movimentação: imprime uma força e um torque ao centro de massa global do rob*

Através desses sistemas rob pode executar um conjunto de ações A_{rob}*.*

Apesar de o modelo descrito acima abranger a mairia dos robôs utilizados atualmente por equipes da SSL, é importante ressaltar que o robô pode ter um conjunto de sensores que poderiam coletar informações adicionais ás transmitidas pela SSL-Vision juntamente com um sistema de transmissão para envia-las ao software do seu respectivo time. Isso é interessante, pois, conforme observado na definição 2.2.2, o parâmetro $\hat{b}.\beta$ *não é observavel. Como o sitema de drible impoem um torque a bola, através de um sensor é possível estimar o valor de* $\hat{b}.\omega$*. Sem esse sensor não é possível prever com exatidão a tragetória da bola através do módulo de simulação apenas com os dados da SSL-Vision.*

Definição 2.2.4 (Time)**.** *Sejam os seguintes parâmetros:*

Rob_c *o conjunto dos robôs controlados*

Rob_i *o conjunto dos robôs inimigos, i.e., não controlados*

X *o espaço de estado de todos os corpos rígidos envolvidos na partida considerada*

$x_{init} \in X$ *o estado inicial*

$X_{goal} \subset X$ *o conjunto de estados objetivo*

x^i_{ob} *os estados observados pelo módulo SSL-Vision no instante i*

$X^i_{ob} = \{x^0_{ob} = x_{init}, ..., x^i_{ob}\}$

$Sk \subset A_{rob}$ *um conjunto de skills*

$prob : X \longrightarrow [0, 1]$ *uma distribuição de probabilidade cujo argumento é* $x \in X$

$tk = G(V \in Sk, E \in prob)$ *o conjunto de todas as táticas possíveis formados a partir de grafos orientados onde os vértices são skills* $sk \in Sk$ *e as arestas são prob, associadas a possibilidade de ocorrerem as transições entre uma skill e outra*

$A_c = A_{rob1} \cup ... \cup A_{robn_c}$ *o conjunto das ações possíveis de* Rob_c

$A_i = A_{rob1} \cup ... \cup A_{robn_i}$ *o conjunto das ações possíveis de* Rob_i

$A = A_c \cup A_i$ *o conjunto das ações possíveis de* $Rob_c \cup Rob_i$

e *a função de transição de estado que pode aplicar uma ação* $a \in A$ *em um estado particular* $x \in X$ *e compute o estado seguinte* $x^{'} \in X$*, i.e.:* $e : \langle x, a \rangle \longrightarrow x^{'}$.

$f_U : X \longrightarrow \mathbb{R}^+ \cup \{0\}$ *uma função utilidade tal que* $f_U(x)$ *mede a utilidade do estado* $x \in X$ *um entre estados do mundo dado os estados.*

$r_i : \langle A_i, X^i_{ob} \rangle \longrightarrow a^{'}_i$ *o modelo de reação dos robôs inimigos dado* X^i_{ob}

$AB = \{V \subset X, E \subset A\}$ *uma árvore de busca*

$e_b : \langle X^i_{ob}, e, f_U, r_i, AB \rangle \longrightarrow AB^{'}$ *uma estratégia de busca*

Então um time T *é definido por:*

$$T : \langle A, X^i_{ob}, e, e_b, r_i \rangle \longrightarrow a^{i+1}_c$$

Assim, utilizando-se de e*,* T *simula várias sequência de ações a dado* X^i_{ob}*, gerando a partir de* f_U*,* e_b *e.*

Definição 2.2.5 (Partida)**.** *Dado dois times* T_1 *e* T_2*. Uma partida* p *é definida por:*

$$p = \{T_1, T_2, \Delta t, \delta t, \langle Ref^0, X^0_{ob}, A^0_1, A^0_2 \rangle, ..., \langle Ref^N, X^N_{ob}, A^N_1, A^N_2 \rangle\}$$

uma sequência , onde:

Δt *é o tempo de duração da partida*

δt *é o tempo médio entre cada frame enviado pela SSL-Vision ao longo de* Δt

$N \approxeq \frac{\Delta t}{\delta t}$ *é número total de frames enviados pela SSL-Vision ao longo de* Δt.

Ref^i *são os comandos envidos pelo módulo Referee-Box no instante i*

X^i_{ob} *são os dados enviados pelo módulo SSL-Vision no instante i*

A^i_1 *são as ações executadas por* T_1 *no instante i*

A^i_2 *são as ações executadas por* T_2 *no instante i*

Definição 2.2.6 (Logs)**.** *Dada uma partida* p*. O log de* p *é definidor por:*

$$log(p) = \{p.\langle Ref^0, X^0_{ob} \rangle, ..., p.\langle Ref^N, X^N_{ob} \rangle\}$$

2.3 Enunciado do Problema

A partir das definições apresentadas, pode-se enunciar o problema abordado por este trabalho:

Dado do um *log* de uma partida p. Determinar a função

$$F : \langle A, X_{ob}^{i}, e, e_b, r_i \rangle \longrightarrow a_c^{i+1}$$

que minimiza o erro

$$erro = \|F - p.T_1\|$$

Onde $\| \bullet \|$ denota uma norma a ser definida.

3 Revisão Bibliográfica

Em (RUSSELL; NORVIG, 2004), (HAYKIN, 2001), (KOSKO; BART, 1997), (PASSOS; GOLDSCHMIDT, 2005), (DORIGO; STÜTZLE, 2004) e (BERTSIMAS; TSITSIKLIS, 1993) são descritas as heurísticas mais comumente aplicadas à problemas envolvendo Inteligência Artificial (IA).

Em (JR.; YONEYAMA, 2004) é apresentado uma abordagem de problemas de controle utilizando de IA. Destaca-se a abordagem utilizando algoritmos de otimização.

Em (AMARAL; ALMEIDA, 2011) é apresentado uma arquitetura para o ambiente da SSL/F180. Essa arquitetura foi base para a arquitetura descrita no capítulo 2.

Em (ZICKLER, 2010) é apresentado uma modelagem de um planejador robótico baseado em física para ambientes dinâmicos. Essa modelagem foi base para a enunciação do problema descrita no capítulo 2, juntamente com arquitetura orientada a *Skills, Tatics and Plays* descrita em (BOWLING et al., 2003) e com a teoria dos agentes apresentada em (RUSSELL; NORVIG, 2004).

Em (VAIL; VELOSO, 2008) é apresentada uma modelagem do problema considerado neste trabalho utilizando-se *Conditional Random Fields*. É interesante resaltar que essa modelagem foi realizada utilizando-se um time T_1 (definição 2.2.4) conhecido no qual os dados de treinamento foram coletados da partida p sabendo-se $p.A_1^i$, e não somente X_{ob}^i.

Em (SHENG et al., 2005) é apresentada uma modelagem considerando apenas as X_{ob}^i de um time T. Essa modelagem utiliza redes neurais de três camadas. Foi utilizada uma função de ativação sigmoidal para a camada oculta e linear para a função de ativação da saída. O algorítimo de treinamento utilizado na rede neural foi o *standard back-propagation algorithm*, descrito em (HAYKIN, 2001).

4 Heurísticas para Solução do Problema

4.1 Lógica Nebulosa

Sistemas nebulosos aproximam funções. Eles são aproximadores universais se usarem regras suficientes. Neste sentido sistemas difusos podem modelar qualquer função ou sistema contínuos. Aqueles sistemas podem vir tanto da física quanto da sociologia, bem como da teoria do controle ou do processamento de sinais. Uma arquiterura funcional generica de um Sistema Nebuloso proposta por (PASSOS; GOLDSCHMIDT, 2005) é apresentada na figura 4.1.

A qualidade da aproximação difusa depende da qualidade das regras. Na prática especialistas sugerem regras difusas ou aprendem-nas através de esquemas neurais através de dados e ajustam as regras com novos dados. Os resultados sempre aproximam alguma função não linear desconhecida que pode mudar com o tempo. Melhores cérebros e melhores redes neurais resultam em melhores aproximações (KOSKO; BART, 1997).

4.1.1 Modelo Aditivo Padrão (SAM)

O sistema difuso $F : \Re^n \rightarrow \Re^p$ é em si uma árvore de regras rasa e extensa. É um aproximador por antecipação. Existem m regras da forma "Se X é conjunto difuso A então Y é conjunto difuso B". A relação de pertinência "X é conjunto difuso A"é descrita por uma função de pertinência $\mu_A : x \longrightarrow [0, 1]$. No diagrama da figura 4.1 esse processo é

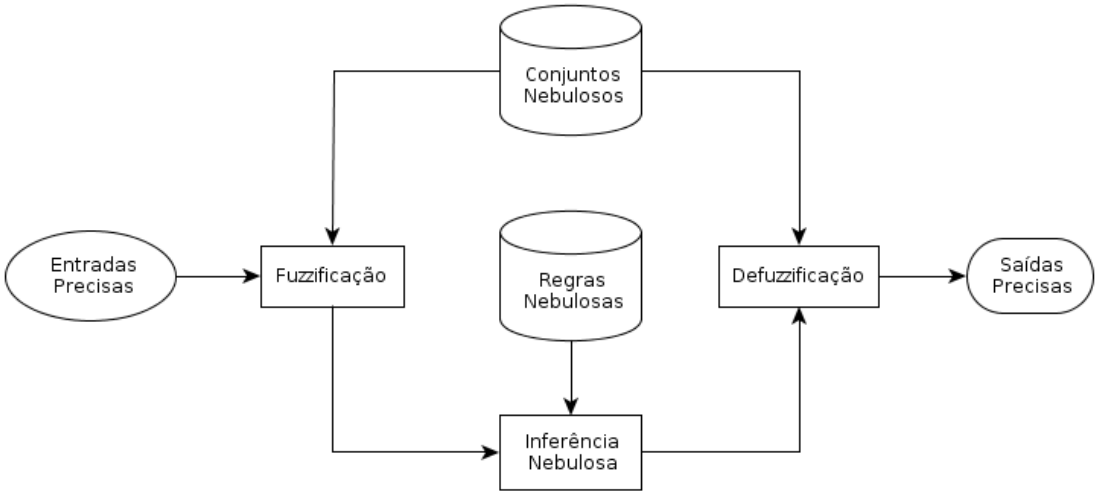

Figura 2 – Arquitetura funcional genérica de um Sistema Nebuloso (PASSOS; GOLDSCHMIDT, 2005)

representado pela caixa *Fuzzificação*. A regra nebulosa "Se <antecedente nebuloso> então <consequente nebuloso>"dispara o processo de *Defuzzificação* descrito no <consequente nebuloso>. Esse processo de conecção é a *Inferência Nebulosa*.

Cada entrada x aciona parcialmente todas as regras em paralelo. Então o sistema age como um processador associativo a medida que calcula a saída $F(x)$.

Essas regras relacionam os conjuntos A_j e B_j, gerando o caminho difuso $A_j x B_j$. Na prática, é utilizado o produto para definir $a_j x b_j(x, y) = a_j(x).b_j(y)$. Esta é a parte "padrão"no SAM. A parte "aditiva"se refere ao fato de a entrada x acionar a j-ésima regra em um grau $a_j(x)$ e o sistema soma os acionamentos ou partes escaladas dos conjuntos escalados $a_j(x)B_j$, (KOSKO; BART, 1997):

$$F(x) = \frac{\sum w_i.a_i(x).V_i.c_i}{\sum w_j.a_j(x).V_j} \tag{4.1}$$

Com o volume/área V_j e o centroide c_j são dados por:

$$V_j = \int b_j(y_1, ..., y_p)_{\Re^p}.dy_1...dy_p > 0 \tag{4.2}$$

$$c_j = \frac{\int y.b_j(y_1, ..., y_p)_{\Re^p}.dy_1...dy_p}{V_j} \tag{4.3}$$

4.1.1.1 Exemplo de aplicação

Este exemplo foi retirado de (PASSOS; GOLDSCHMIDT, 2005). Deseja-se neste exemplo de terminar qual o valor da apólice de seguro a ser pago pelo cliente João a partir dos valores de idade e de pressão deste cliente. Considerando-se as Regras Nebulosas

- *SE* idade é *meia-idade E* pressão é *baixa ENTÂO* seguro é *baixo*
- *SE* idade é *jovem E* pressão é *alta ENTÂO* seguro é *alto*

Considere que João tem 35 anos e pressão $(130, 70)$. Suponha que as funções de pertinência dão como resultado:

$\mu_{meia-idade}(35) = 0.8$

$\mu_{jovem}(35) = 0.6$

$\mu_{Alta}(130, 70) = 0.5$

$\mu_{Baixa}(130, 70) = 0.6$

Logo, para a regras 1 e 2, considerando-se o modelo de Mandani descrito em (PASSOS; GOLDSCHMIDT, 2005), tem-se que

$0.8\ E\ 0.6 = Min0.8, 0.6 = 0.6 = \mu_{baixo}(X)$

$0.6\ E\ 0.5 = Min0.6, 0.5 = 0.5 = \mu_{alto}(Y)$

A partir de um processo de defuzzificação, das as distribuições μ_{baixo} e μ_{alto}, tem-se que

$X = 800$ e $Y = 700$

$Seguro = \frac{(0.5\times800)+(0.6\times700)}{0.5+0.6} = 745.45$

Assim a apólice de seguro a ser paga por João será de 745.45 reais.

4.1.1.2 Aplicação ao problema

Para aplicar esse método ao problema analisádo neste trabalho é necessário definir as regras e as distribuições das variáveis. A partir de então, bastaria encontrar os pesos para minimizar o erro.

4.2 Otimização da Colônia de Formigas

Na busca por alimento, as formigas utilizam de feromônios para encontrar o melhor caminho. Isso acontece da seguinte maneira: cada formiga deposita feromônio ao se deslocar. A partir da avaliação da quantidade de feromônio depositada por formigas que já passaram pelo local, formigas subsequentes tem mais probabilidade de se mover em rotas que tem mais feromônios. Ao decorrer do tempo os feromônios vão evaporando, apagando rastros que não foram reforçados. Com isso, caminhos que são percorridos por mais formigas tem mais chance de serem percorridos por outras formigas do que aqueles que foram percorridos por menos formigas e caminhos que foram percorridos á pouco tempo tem mais chance de serem percorridos que caminhos percorridos a muito tempo. A quantidade de feromônio depositado é mais intensa no trajeto de volta, quando a comida foi encontrada. Outro fator que é levado em consideração é a qualidade da comida encontrada, de maneira que mais feromônio é depositado quanto melhor for a

fonte de alimento encontrada. A medida que mais formigas exploram o local e encontram alimento, esse procedimento tende a otimizar o trajeto entre a fonte de alimento e a colônia.

Apesar dessa heurística utilizada pelas formigas ser interessante para se resolver problemas combinatórios do tipo NP(i.e., com complexidade não polinomial), são necessários algumas adaptações na construção de um algoritmo computacional.

A seguir é apresentado a meta-heurística do ACO(*Ant Colony Optimization*) algoritmo, juntamente com observações relacionadas as diferenças entre a heurística do ACO e o comportamento natural das formigas descrito anteriormente.

4.2.1 Meta-heurística do ACO

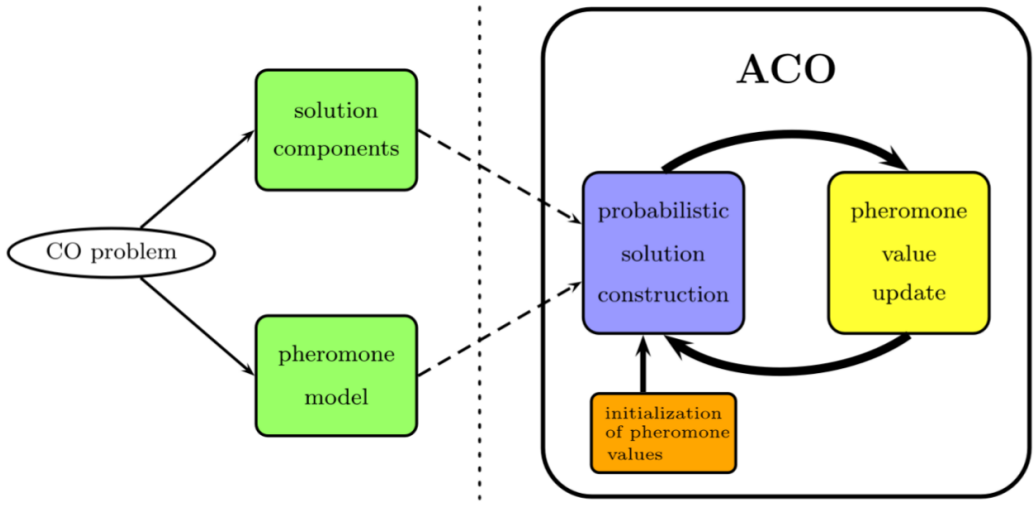

Figura 3 – Diagrama de funcionamento da meta-heurística do ACO (BLUM, 2005)

Procedimento
  enquanto $n < N_{MAX\_IT}$ **faça**
    AgendarAtividade
      ConstruirSolucoesFormigas
      AtualizarFeromonios
      `// opcional:`
      AcoesGlobais
    fim
  fim
fim

Algoritmo 1: Pseudo código da meta-heurística do ACO

A meta-heurística do ACO pode ser subdividida em três partes, conforme proposto por (DORIGO; STÜTZLE, 2004): *ConstruirSolucoesFormigas*, *AtualizarFeromonios* e

AcoesGlobais.

ConstruirSolucoesFormigas gerencia a movimentação de uma colônia de formigas em torno dos nós vizinhos. A escolha do próximo nó é feita através de uma decisão estocástica que é função da quantidade de feromônio no nós vizinhos e informação heurística. Quando uma formiga encontra uma solução, ou enquanto a solução é construída, esta avalia a qualidade da solução(completa ou parcial) que será utilizada pelo procedimento *AtualizarFeromonios* para decidir a quantidade de feromônio que será depositada. Outro procedimento relevante na construção da solução é a eliminação de possíveis ciclos, utilizado por exemplo, no problema do caixeiro viajante.

AtualizarFeromonios é o processo que atualiza os traços de feromônio depositados pelas formigas no espaço de busca. Os traços de feromônio podem aumentar, caso uma formiga tenha visitado o nó/conexão em questão, ou diminuir, devido ao processo de evaporação do feromônio. Esse procedimento faz com que nós/conexões que foram visitados por muitas formigas ou por uma formiga e que tenha levado em uma solução boa aumentem a probabilidade de serem visitados por futuras formigas. Semelhantemente, reduz a probabilidade de que nós que não foram visitados por novas formigas por muitas iterações sejam visitados novamente. Logo, este procedimento evita a convergência a caminhos sub ótimos, favorecendo também a exploração de novas regiões do espaço de busca.

Por fim, o procedimento *AcoesGlobais* é utilizado para centralizar ações que não podem ser executadas pelas formigas individualmente. Um exemplo de ações desse tipo é a filtragem de soluções ou o favorecimento de regiões por meio de informações globais.

O procedimento *AgendarAtividade* não necessariamente é uma instrução sequencial. Pode-se, portanto, implementá-lo de maneira sequencial ou paralela, síncrona ou assincronamente. O tipo de abordagem que será utilizada depende das características do problema que se deseja resolver.

4.2.2 Exemplo de utilização

Definition 1. In the TSP is given a completely connected, undirected graph $G = (V, E)$ with edge-weights. The nodes V of this graph represent the cities, and the edge weights represent the distances between the cities. The goal is to find a closed path in G that contains each node exactly once (henceforth called a tour) and whose length is minimal. Thus, the search space S consists of all tours in G. The objective function value $f(s)$ of a tour $s \in S$ is defined as the sum of the edge-weights of the edges that are in s. The TSP can be modelled in many different ways as a discrete optimization problem. The most common model consists of a binary decision variable X_e for each edge in G. If in a solution

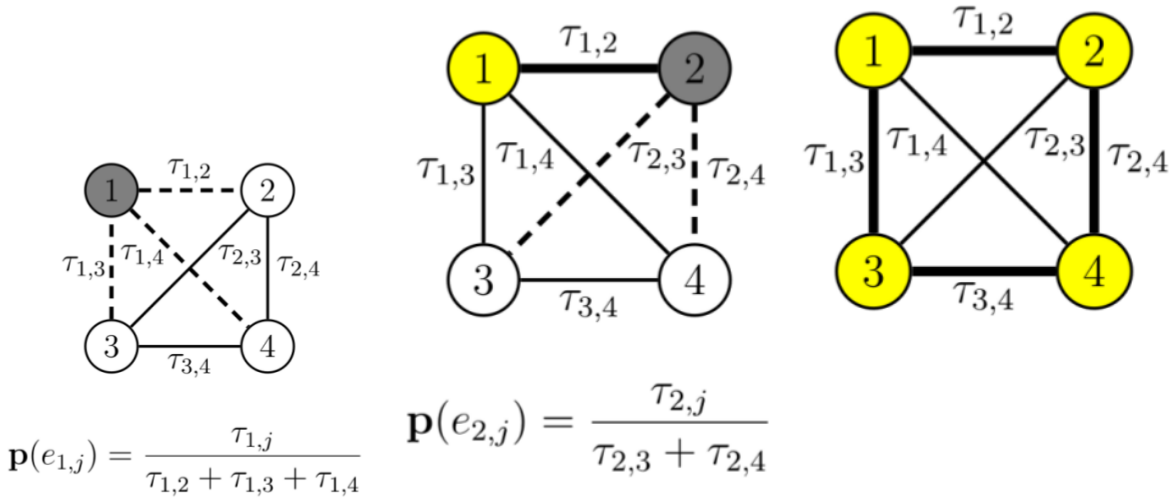

Figura 4 – Example of the solution construction for a TSP problem

$X_e = 1$, then edge e is part of the tour that is defined by the solution. Concerning the AS approach, the edges of the given TSP graph can be considered solution components, i.e., for each $e_{i,j}$ is introduced a pheromone value $\tau_{i,j}$. The task of each ant consists in the construction of a feasible TSP solution, i.e., a feasible tour. In other words, the notion of task of an ant changes from "choosing a path from the nest to the food source" to "constructing a feasible solution to the tackled optimization problem". Note that with this change of task, the notions of nest and food source loose their meaning. Each ant constructs a solution as follows. First, one of the nodes of the TSP graph is randomly chosen as start node. Then, the ant builds a tour in the TSP graph by moving in each construction step from its current node (i.e., the city in which she is located) to another node which she has not visited yet. At each step the traversed edge is added to the solution under construction. When no unvisited nodes are left the ant closes the tour by moving from her current node to the node in which she started the solution construction. This way of constructing a solution implies that an ant has a memory T to store the already visited nodes. Each solution construction step is performed as follows. Assuming the ant to be in node v_i , the subsequent construction step is done with probability:

$$p(e_{i,j}) = \frac{\tau_{i,j}}{\sum_{k \in \{1,...,|V|\}, v_k \notin T} \tau_{i,k}}, \forall j \in \{1, ..., |V|\}, v_j \notin T \tag{4.4}$$

For an example of such a solution construction see Fig. 4.2.2. Once all ants of the colony have completed the construction of their solution, pheromone evaporation is performed as follows:

$$\tau_{i,j} \leftarrow (1-\rho) \cdot \tau_{i,j}, \forall \tau_{i,j} \in \mathcal{T} \tag{4.5}$$

where $\mathcal{T}$ is the set of all pheromone values. Then the ants perform their return trip. Hereby, an ant—having constructed a solution s— performs for each $e_{i,j} \in s$ the following pheromone deposit:

$$\tau_{i,j} \leftarrow \tau_{i,j} + \frac{Q}{f(s)} \tag{4.6}$$

where Q is again a positive constant and $f(s)$ is the objective function value of the solution s. As explained in the previous section, the system is iterated—applying n_a ants per iteration—until a stopping condition (e.g., a time limit) is satisfied. Even though the AS algorithm has proved that the ants foraging behaviour can be transferred into an algorithm for discrete optimization, it was generally found to be inferior to state-of-the-art algorithms. Therefore, over the years several extensions and improvements of the original AS algorithm were introduced. They are all covered by the definition of the ACO metaheuristic, which we will outline in the following section.

4.2.3 Aplicação ao Problema

4.3 Recozimento Simulado

No processo de recozimento de um metal, a quantidade de energia interna livre esta intrinsecamente relacionada ao processo de resfriamento em que o metal é submetido. Quanto mais rápido se resfriam um metal mais energia é armazenada internamente. Isso pode ser explicado considerando que o tempo que a estrutura leva para atingir o estado de menor energia é maior que o disponível devido a redução da mobilidade dos átomos com o decaimento da temperatura. Com efeito, quanto maior a taxa de resfriamento maior o número de defeitos na estrutura do sólido e menor o tamanho médio dos grãos. Quando se reduz a taxa de resfriamento, há uma maior chance de se atingir configurações mais estáveis. Como resultado, a energia interna é reduzida. De acordo com (BERTSIMAS; TSITSIKLIS, 1993), pode-se modelar a probabilidade p_{ij} de uma configuração atômica $\{r_i\}$ com energia $E\{r_i\}$ passar para a configuração $\{r_j\}$ com energia $E\{r_j\}$ na temperatura T como:

$$p_{ij} = \begin{cases} 1 & \text{se } E\{r_j\} \leq E\{r_i\} \\ exp\left\{-\frac{(E\{r_j\}-E\{r_i\})}{k_B.T}\right\} & \text{se } E\{r_j\} > E\{r_i\} \end{cases} \tag{4.7}$$

Onde k_B é a constante de Boltzmann. Para se reduzir a energia livre, é necessário que uma rotina de resfriamento seja escolhida de acordo com o tipo de material a ser resfriado.

Conforme proposto por Kirikpartrick, Gellett e Vechin (1983) e Cerny (1985), pode-se desenvolver uma heurística probabilística para se encontrar o mínimo global de uma função custo que possua vários mínimos locais fazendo-se uma analogia com o fenômeno físico descrito acima. A meta-heurística induzida por este processo é chamada de meta-heurística *Simulated Annealing*(Recozimento Simulado), ou SA, apresentado a seguir.

4.3.1 Meta-heurística do SA

De acordo com (BERTSIMAS; TSITSIKLIS, 1993), os elementos básicos da meta-heurística do SA para a resolução de um problema combinatório são:

1. Um conjunto finito S.
2. Um função custo J de imagem real definida em S. Seja $S^* \subset S$ o conjunto de todos os mínimos globais da função J, suposto subconjunto próprio.
3. Para cada $i \in S$ um conjunto $S(i) \subset S - \{i\}$, chamado de conjunto das vizinhos de i.
4. Para cada i, uma coleção de coeficientes positivos q_{ij}, $j \in S(i)$, tal que $\sum_{j \in S(i)} qij = 1$.
5. Uma função não crescente $T : \mathbf{N} \rightarrow (0, \infty)$, chamada de rotina de resfriamento. Aqui $\mathbf{N}$ representa o conjunto de inteiros positivos, e $T(t)$ é chamada de *temperatura* no tempo t.
6. Um estado inicial $x(0) \in S$.

Com base na definições acima, tem-se o seguinte pseudo código para a meta-heurística do SA:

Procedimento
  SetarValoresInicias;
  para $n = 1$ *até* N_{MAX_IT} *ou* $J(x^*) \leq TOL$ **faça**
    para $k = 1$ *até* N_{MAX_IT} *ou a solução convergir* **faça**
      EscolherVizinho
        selecionar algum $j \in S(i)$;
      fim
      CalcTransicao
        $\Delta J \leftarrow J(j) - J(i)$;
        se $DeltaJ \leq 0$ **então**
          $x(t+1) \leftarrow j$;
          $x^* \leftarrow j$;
        fim
        senão
          $q_{ij} \leftarrow exp^{-\frac{\Delta J}{T(t)}}$;
          se $random() < q_{ij}$ **então** $x(t+1) \leftarrow j$;
          senão $x(t+1) \leftarrow i$;
        fim
      fim
    fim
    AtualizarTemperetura;
  fim
fim

Algoritmo 2: Pseudo código da meta-heurística do SA

No algorítimo 2, o procedimento *AtualizarTemperetura* executa a rotina de resfriamento através da função $T(t)$ definida anteriormente. Já o procedimento *EscolherVisinho* escolhe aleatoriamente um dos elementos da vizinhança do vértice atual i.

4.4 Algorítimo Genético

Um *algorítimo genético* é uma heurística de busca que procura imitar a seleção natural que ocorre no processo evolucionário dos organismos vivos.

Nessa heurística, uma população de soluções (também chamadas de indivíduos ou fenótipos) para problemas de otimização é evoluída para conseguir soluções melhores. Cada solução possui um conjunto de propriedades (cromossomos ou genótipos) que podem

ser mutados ou alterados.

Os requerimentos são, tipicamente:

- uma representação genética da solução
- uma função de aptidão para avaliação da solução

4.4.1 O processo

O processo é iniciado com uma população com propriedades geradas aleatoriamente.

A iteração da heurística se da em 3 etapas:

- procriação: indivíduos são pareados e é aplicada a operação de cruzamento (*crossover*)
- mutação: alguns indivíduos são selecionados e é aplicada a operação de mutação (*mutation*)
- seleção: é usada a função de aptidão para descartar os indivíduos menos aptos restando as soluções que de fato trouxeram alguma melhora.

As condições mais comuns para terminação do processo são as seguintes:

- encontrada uma solução que atende os requisitos mínimos
- número fixo de gerações alcançado
- recursos alocados (tempo ou dinheiro) alcançados
- a melhor solução alcançou um patamar estável em que mais iterações não produzem soluções melhores
- inspeção manual

4.4.2 Limitações

As limitações mais comuns no emprego de um algorítimo genético são:

- Funções de avaliação computacionalmente caras tornam essa heurística ineficiente.

- Não escala bem com a complexidade, isto é, quando o número de elementos expostos a mutação é grande o espaço de busca cresce exponencialmente. Por isso, na prática algorítimos genéticos são usados para, por exemplo, projetar uma hélice e não um motor.

- A melhor solução é relativa às outras soluções, por isso o critério de parada não é muito claro em alguns problemas.

- Em muitos problemas os algorítimos genéticos tendem a convergir para um ótimo local ou as vezes pontos arbitrários em vez do ótimo global.

- É difícil aplicar algorítimos genéticos para conjunto de dados dinâmicos. Pois as soluções podem começar a convergir para um conjunto de dados que já não é mais válido.

- Algorítimos genéticos não conseguem resolver eficientemente problemas em que a avaliação é binária (certo/errado), como em problemas de decisão. Nesse caso buscas aleatórias convergem tão rápido quanto essa heurística.

- Para problemas mais específicos existem outras heurísticas que encontram a solução mais rapidamente.

4.4.3 Pseudo código de um Algorítimo Genético

Procedimento
  $k \leftarrow 0$;
  $P_k \leftarrow$ população de n indivíduos escolhidos aleatoriamente;
  enquanto *a avaliacao*(i) *de cada* i *em* P_k *não for boa o suficiente* **faça**
    Selecionar os $(1 - \chi) \times n$ membros com maior *avaliacao*(i) de P_k e inserir em P_{k+1};
    Selecionar $\chi \times n$ membros de P_k, pareá-los e inserir a cria em P_{k+1};
    Selecionar os $\mu \times n$ membros de P_{k+1} com maior *avaliacao*(i) e inverter um bit aleatório de cada membro;
    $k \leftarrow k + 1$;
  fim
  melhor $\leftarrow$ o membro i em P_k com maior *avaliacao*(i);
  retorna *melhor*
fim

4.4.4 Exemplo

Suponha uma fábrica com duas máquinas, e chegam quatro pedidos, em ordem, para serem produzidos. Cada pedido consiste irá levar um tempo fixo para ser produzido. O

problema a ser resolvido é determinar qual máquina processa cada pedido, de tal modo que o tempo total (até o último pedido ser atendido) seja mínimo. Concretizando o exemplo supõe-se os seguintes pedidos: A: 1h, B: 5h, C: 2h, D: 3h.

Primeiro deve ser modelado a representação genética, ou fenótipo. Uma proposta simples é uma string de 4 bits, em que cada bit representa qual das duas máquinas produz o pedido, e a ordem dos bits é a ordem dos pedidos.

Segundo uma função de aptidão, que nesse caso é determinística: o maior entre a soma dos tempos dos pedidos de bits 0 e a soma dos de bit 1. Isto é para uma sequencia 0110 teríamos o maior entre $1h + 3h$ e $5h + 2h$, ou seja $7h$.

Terceiro uma regra de cruzamento, que nesse caso pode simplesmente ser a troca aleatória de alguns bits. Por exemplo, o cruzamento de 1100 e 0011 pode trocar apenas o primeiro bit, resultando em 0100 e 1011.

Por último uma regra de mutação, nesse caso é sufiente a inversão de um bit.

Com esses quatro requisitos é possível aplicar a iteração genética sobre uma população inicial.

Esse exemplo demonstra a modelagem de uma otimização usando um algorítimo genético e foi baseado em (VIEIRA; SOARES; JUNIOR, 2002).

4.4.5 Aplicação ao problema

O problema descrito neste trabalho pode se beneficiar de algorítimos genéticos para otimizar conjuntos de parâmetros de outras heurísticas.

4.5 Rede Neural

O termo mais apropriado é rede neural artificial, já que apenas rede neural pode se referir ao sistema biológico de nervos, no antando dado o contexto desse texto e o uso consagrado do termo “rede neural”, esse será usado no lugar da versão mais explícita “rede neural artificial”.

Uma rede neural é um sistema inspirado no sistema nervoso central (em especial o cérebro) encontrado em muitos animais. A ideia básica é ter um grafo em que cada nó abstrai um neurônio e é representado como uma função, alguns desses nós são responsáveis pela observação e outros pela saída e os nós de entrada alimentam os próximos nós até chegar nos nós de saída. (HAYKIN, 2001)

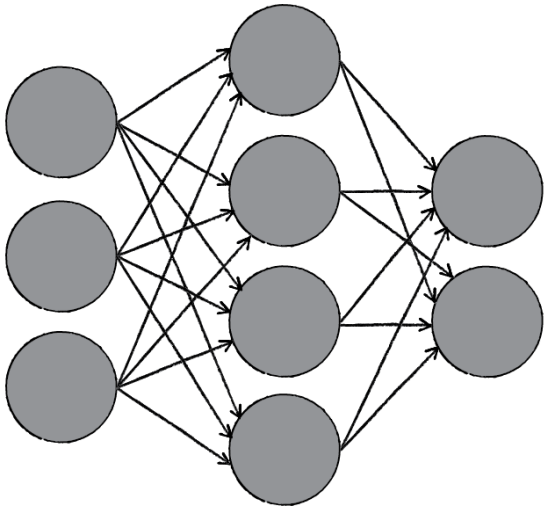

Figura 5 – Rede neural com três camadas.

A figura 5 exemplifica uma rede neural *feedforward*, que é baseada num grafo direcionado acíclico, em que podem ser vistas 3 camadas a primeira é chamada de camada de entrada, a última, de saída e as intermediárias, de escondidas. (SHIFFMAN; FRY; MARSH, 2012)

4.5.1 O *Perceptron*

O bloco de construção básico de uma rede neural são os neurônios. Um *perceptron* é a rede neural mais simples possível: é formada por apenas um neurônio.

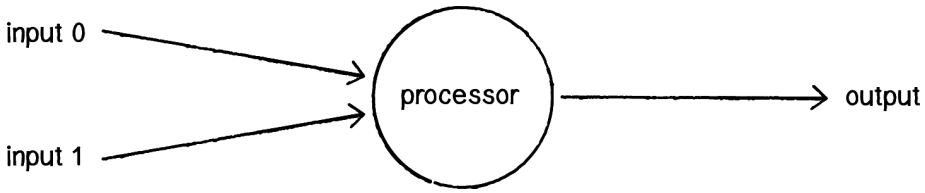

Figura 6 – *Perceptron* de duas entradas e uma saída.

O funcionamento de um *perceptron* pode ser descrito nos seguintes passos:

- Receber e armazenar as entradas;
- Pesar as valores armazenados: consistem em multiplicar cada um pelo peso correspondente àquela entrada;
- Enviar o saída calculara anteriormente.

4.5.2 Exemplo

Considere uma reta em $\Re^2$, que separa o plano em duas regiões A e B. O problema a ser resolvido pelo *perceptron* é dizer se um ponto (x, y) está em A ou B. As entradas do problema são as coordenadas x e y, e a saída é um escalar cujo o valor é um escalar positivo para sinalizar o conjunto A e negativo para sinalizar o B.

O *perceptron* poderia ser modelado com duas entradas, porém note que desse modo o ponto $(0, 0)$ sempre irá resultar numa saída igual a 0. Para evitar isso são usadas três entradas no *perceptron* sendo que uma delas é sempre igual a 1 e é chamada de *bias*, que serve basicamente para adicionar um peso aditivo e não só multiplicativo. O funcionamento desse *perceptron* ser observado na figura 7.

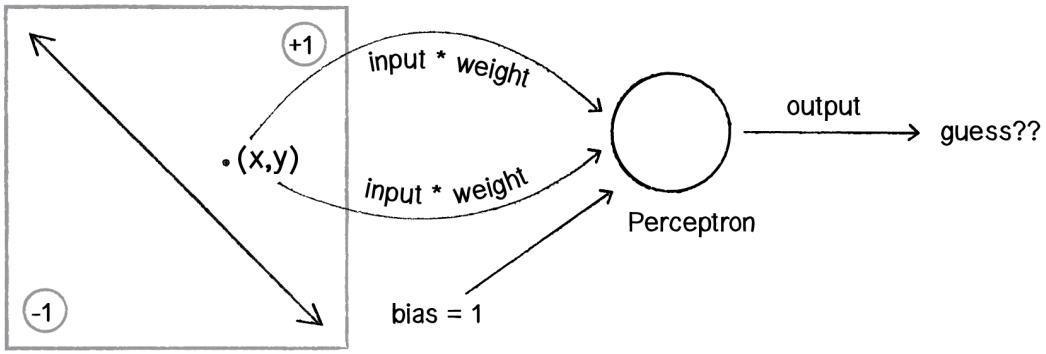

Figura 7 – *Perceptron* para decidir região de um ponto no plano.

Esse exemplo usará o método supervisionado de aprendizado, que funciona da seguinte maneira:

- Alimentar o *perceptron* com uma entrada para o qual se conhece a resposta;
- Pedir a saída ao *perceptron*;
- Computar o erro;
- Ajustar os pesos de acordo com o erro;
- Repetir o processo.

O ajuste dos pesos pode ser feito calculando o erro e incrementando o peso um fator de aprendizado vezes o erro vezes a entrada daquele peso.

Assim o *perceptron* é capaz de ser treinado e sua saída ser cada vez mais próxima da desejada.

4.5.3 Aplicação ao problema

No contexto do problema proposto uma rede neural poderia ser utilizada em duas vertentes. A primeira, e mais simples, é na otimização genérica de funções eventuais. A segunda, e mais complexa, é na detecção de padrões. O diferencial dessa técnica é a detecção de padrões, por isso a segunda vertente é mais apropriada.

Concretizando melhor a aplicação os cenários mais provável é alimentar uma rede neural com os dados de estado do jogo e treiná-la para identificar padrões do time inimigo. Mais formalmente isso se traduz a classificar os estados do jogo em relação ao subconjunto de informações desse estado associado ao time inimigo.

5 Análise das Possíveis Abordagens

Como os métodos AG (Algorítimo Genético), SA e ACO são heurísticas de otimização, é necessário que o problema da aproximação de uma função seja reduzido a um problema de otimização. Uma abordagem é se escolher uma base finita de funções ortonormadas e minimizar a soma dos quadrados da norma da diferença entre os valores de treinamento e uma combinação linear das funções da base escolhida. Foi suposto, por simplicidade, que o espaço dos vetores um espaço vetorial com norma. Isso restringe o espaço solução, uma vez que apenas funções que são combinação linear das funções da base serão solução.

O método da Lógica Fuzzy, conforme exposto anteriormente, necessita que um conjunto de regras seja definido. Essas regras poder ser geradas a partir de uma análise mais detalhada do problema, mas também podem ser aprendidas via rede neural.

Já a Rede Neural define implicitamente a estrutura interna que minimiza a diferença entre a saída real e a saída desejada. Entretanto, é necessário definir a topologia mais adequada para o problema em questão. Também, apesar de não ser necessário, pode-se decompor o problema em subproblemas e "atribuir redes neurais um subconjunto de tarefas que coincidem com suas capacidades inerentes"(HAYKIN, 2001, pag. 29) com o objetivo de aumentar a adaptabilidade da rede. Apesar de ser uma modelagem para a arquitetura da inteligência a ser mapeada, não representa uma restrição tão considerável quando a das abordagens citadas anteriormente.

Com base nisso, as seguintes abordagens foram estabelecidas:

1. Utilizar Lógica Fuzzy com regras geradas através de uma rede neural;
2. Utilizar Rede Neural com uma topologia mista.

Na primeira abordagem, pretende-se utilizar os métodos de otimização para se encontrar a melhor topologia de rede neural que melhor otimiza o conjunto de regras buscado.

Pretende-se também, na segunda abordagem, utilizar os métodos de otimização descritos anteriormente para se encontrar a melhor topologia que aproxima da melhor maneira.

6 Cronograma

7 Conclusão

Referências

AMARAL, T. A. N.; ALMEIDA, D. F. de. *Robôs Autônomos Cooperativos: Small Size League*. [S.l.]: Instituto Militar de Engenharia, 2011. Monografia de Iniciação à Pesquisa ao Curso de Graduação em Engenharia de Computação. Citado 2 vezes nas páginas 14 e 21.

BERTSIMAS, D.; TSITSIKLIS, J. Simulated annealing. *Statistical Science*, JSTOR, p. 10–15, 1993. Citado 3 vezes nas páginas 21, 28 e 29.

BLUM, C. *Ant colony optimization: Introduction and recent trends*. [S.l.]: Elsevier, 2005. Citado 2 vezes nas páginas 8 e 25.

BOWLING, M. et al. *Plays as Team Plans for Coordination and Adaptation*. 2003. Citado na página 21.

DORIGO, M.; STÜTZLE, T. *Ant Colony Optimization*. [S.l.]: Bradford Book, 2004. ISBN 0262042193. Citado 2 vezes nas páginas 21 e 25.

HAYKIN, S. *Redes neurais*. [S.l.]: Grupo A, 2001. Citado 3 vezes nas páginas 21, 33 e 37.

JR., C. L. N.; YONEYAMA, T. *Inteligência Artificial em Controle e Automação*. segunda. [S.l.]: FAPESP, 2004. Citado na página 21.

KOSKO; BART. *Fuzzy engineering*. [S.l.]: Prentice-Hall, Inc., 1997. Citado 3 vezes nas páginas 21, 22 e 23.

MURRAY, R. M.; LI, Z.; SASTRY, S. S. *A Mathematical Introduction to Robotic Manipulation*. [S.l.]: CRC Press, 1994. Citado na página 17.

PASSOS, E.; GOLDSCHMIDT, R. *Data Mining: Um Guia Prático*. [S.l.]: Elsevier, 2005. Citado 5 vezes nas páginas 8, 21, 22, 23 e 24.

RUSSELL, S. J.; NORVIG, P. *Inteligência Artificial*. quinta reimpressão. [S.l.]: Elsevier, 2004. Citado 2 vezes nas páginas 14 e 21.

SHENG, Y. et al. Motion prediction in a high-speed, dynamic environment. *Proceedings of 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2005. Citado na página 21.

SHIFFMAN, D.; FRY, S.; MARSH, Z. *The Nature of Code*. [S.l.]: D. Shiffman, 2012. Citado na página 34.

VAIL, D. L.; VELOSO, M. M. *Feature Selection for Activity Recognition in Multi-Robot Domains*. 2008. Citado na página 21.

VIEIRA, G. E.; SOARES, M. M.; JUNIOR, O. G. Otimização do planejamento mestre da produção através de algorítmos genéticos. *XXII Encontro Nacional de Engenharia de Produção*, 2002. Citado na página 33.

ZICKLER, S. *Physics-Based Robot Motion Planning in Dynamic Multi-Body Environments*. Tese (Doutorado) — Carnegie Mellon University, May 2010. Citado 2 vezes nas páginas 14 e 21.