CENTRO UNIVERSITÁRIO FEI

FELIPE PAGLIUCA DE MORAES

OTIMIZAÇÃO DA LOCOMOÇÃO EM ROBÔS MÓVEIS HUMANOIDES UTILIZANDO INTELIGÊNCIA DE ENXAME

São Bernardo do Campo
2018

FELIPE PAGLIUCA DE MORAES

OTIMIZAÇÃO DA LOCOMOÇÃO EM ROBÔS MÓVEIS HUMANOIDES UTILIZANDO INTELIGÊNCIA DE ENXAME

Defesa de Mestrado, apresentada ao Centro Universitário da FEI para obtenção do título de Mestre em Engenharia Elétrica. Orientado pelo Prof. Dr. Reinaldo Bianchi.

São Bernardo do Campo
2018

Pagliuca de Moraes, Felipe.
Otimização da Locomoção em Robôs Móveis Humanoides Utilizando Inteligência de Enxame / Felipe Pagliuca de Moraes. São Bernardo do Campo, 2018.
80 f. : il.

Dissertação - Centro Universitário FEI.
Orientador: Prof. Dr. Reinaldo Augusto da Costa Bianchi.

1. Robô Humanoide. 2. Gerador de Caminhada. 3. Otimização por Enxame de Partícula. 4. Osciladores Lineares Acoplados. I. Augusto da Costa Bianchi, Reinaldo, orient. II. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

centro universitário FEI

APRESENTAÇÃO DE DISSERTAÇÃO
ATA DA BANCA EXAMINADORA

Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica

Mestrado

PGE-10

Aluno: Felipe Pagliuca de Moraes **Matrícula:** 115311-3

Título do Trabalho: Otimização da locomoção em robôs móveis humanoides utilizando inteligência de enxame.

Área de Concentração: Inteligência Artificial Aplicada à Automação e Robótica

Orientador: Prof. Dr. Reinaldo Augusto da Costa Bianchi

Data da realização da defesa: 27/09/2018

ORIGINAL ASSINADA

Avaliação da Banca Examinadora:

São Bernardo do Campo, / / .

MEMBROS DA BANCA EXAMINADORA	
Prof. Dr. Reinaldo Augusto da Costa Bianchi	Ass.: _____
Prof. Dr. Flavio Tonidandel	Ass.: _____
Profª. Drª. Esther Luna Colombini	Ass.: _____

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒ REPROVADO ☐

VERSÃO FINAL DA DISSERTAÇÃO

APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

A minha família e amigos.

AGRADECIMENTOS

Agradeço a minha familia e meus amigos pelo apoio em momentos de dificuldade.

Agradeço ao Prof. Dr. Reinaldo Augusto da Costa Bianchi pela ajuda e suporte em todos os momentos que passei durante o período de desenvolvimento deste trabalho. Agradeço por acreditar no meu trabalho, por dar criticas construtivas, e por dividir comigo seu conhecimento tanto durante as aulas quanto fora delas.

Agradeço aos professores do Centro Universitário da FEI pelas incentivo e pelas aulas ministras, sem as quais este trabalho não poderia ser realizado: Prof. Dr. Carlos Eduardo Thomaz, Prof. Dr. Flavio Tonidandel, Prof. Dr. Paulo E. Santos e Prof. Dr. Plinio Thomaz Aquino Júnior.

Agradeço aos colegas Aislan Cesar de Almeida, Claudio de Oliveira Vilão Junior e Isaac Jesus da Silva pelo apoio e suporte.

Agradeço a Deus por esta oportunidade.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

“Nós podemos somente ver uma pequena distancia a frente, mas podemos ver lá muito do que se deve ser feito.”
Alan Turing

RESUMO

É comum encontrarmos robôs capazes de se locomover utilizando trilhos, esteiras e rodas. Estes meios de locomoção são eficientes para robôs em alguns casos como por exemplo um galpão ou locais com pisos lisos e continuo. Entretanto, em locais com muitos desníveis ou descontinuidades, como por exemplo montanhas, escombros de um prédio ou uma escada, a locomoção utilizando rodas, trilhos ou esteiras se torna prejudicada. Já a locomoção utilizando pernas é capaz de obter um ótimo desempenho em locais com ou sem desníveis e com ou sem continuidade, porém, este tipo de locomoção é mais complexo, o que torna mais desafiador obter uma caminhada eficiente.

Esta dissertação propõem o uso de um algoritmo de otimização chamado *Particle Swarm Optimization* (PSO) para encontrar os valores de parâmetros que gerem um caminhar rápido e estável para um robô móvel humanoide. Estes parâmetros serão utilizados pelo robô humanoide desenvolvido pelo Centro Universitário da FEI para participar de competições de futebol de robôs como a RoboCup, na categoria KidSize da liga humanoide. É abordado neste trabalho as principais variações de PSO, as principais técnicas de geração de caminhada em robôs humanoides, cinemática e arquitetura de controle chamada de *Arquitetura em Cruz*. Os resultados obtidos neste trabalho demonstraram que o PSO é capaz de melhorar a caminhada do robô humanoide.

Palavras-chave: Robô Humanoide, Gerador de Caminhada, Otimização por Enxame de Partícula, Osciladores Lineares Acoplados

ABSTRACT

Is usual to find robots capable of move using rails, continuous track and wheels. These kinds of locomotion are efficient for robots in same cases like an industrial shed or places with flat and continuous floors. However, in places with uneven or discontinuous floors, like mountains, buildings debris or ladders, the locomotion using rails, continuous track or wheels become impaired. The locomotion using legs is capable to obtain a great performance in places with or without unevenness and with or without continuity. However, this kind of locomotion is more complex, making more challenging to obtain an efficient gait.

This dissertation propose the use of an optimization algorithm called *Particle Swarm Optimization* (PSO) to find the parameters values that make a fast and stable gait for a humanoid mobile robot. This parameters is going to be used in the humanoid robot developed by the Centro Universitário da FEI to compete in football competitions like the RoboCup's KidSize category of the humanoid league. In this work will be approached the main PSO variations, the main humanoid robot gait generation technique, humanoid robot cinematic and a controller architecture called *Architecture in Cross*. The results obtain by this work shows that the PSO is capable of improve the humanoid robot gait.

Keywords: Humanoid Robots, Gait Generator, Particle Swarm Optimization, Oscillator Parameters

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Atualmente robôs fazem cada vez mais parte do dia a dia dos seres humanos recebendo a responsabilidade de atuar em situações perigosas, complexas e arriscadas. Estas máquinas foram escolhidas para tais tarefas pois conseguem executar certas funções com extrema precisão e resiliência.

Com o crescimento da atuação dos robôs começou a se mostrar necessário que estes se locomovessem por qualquer ambiente que estivessem inseridos, seja um galpão contendo demarcações para este robô seguir, seja uma casa com diversos cômodos, seja os escombros de um prédio. Para atingir tal objetivo foi identificado que robôs que utilizam rodas como meio de locomoção não são adequados, pois estes possuem dificuldades para se movimentarem em terrenos desnivelados, acidentados e/ou com pequenos obstáculos como por exemplo pedras. Estas tarefas são adequadas para robôs que possuem pernas como por exemplo um robô humanoide.

Devido a necessidade do desenvolvimento de robôs que se locomovem utilizando pernas ao invés de rodas, este trabalho irá propor a utilização de técnicas para otimizar os valores dos parâmetros do sistema de controle do andar de um robô humanoide visando um andar rápido e estável. Para tal será utilizando um simulador 3D e um algoritmo de *Particle Swarm Otimization*.

1.1 MOTIVAÇÃO

Apesar da importância da utilização de robôs humanoides capazes de caminhar, ainda é necessário desenvolvimento nesta área para que estes robôs sejam capazes de caminhar com a mesma eficiência de um ser humano. Devido a este fato existem diversas pesquisas sendo feitas utilizando técnicas de inteligência artificial.

Os robôs humanoides do Centro Universitário da FEI possuem parâmetros que são utilizados pelo controle para gerar o caminhar. Estes parâmetros eram ajustados manualmente e verificando a estabilidade do andar do robô através de observação. Há pouco tempo, foi realizado um trabalho de automatização do ajuste destes parâmetros utilizando aprendizado por reforço (SILVA, 2015), motivando assim a continuidade do desenvolvimento desta automação. Portanto este trabalho propõem uma nova abordagem para esta automatização utilizando o Particle Swam Ootimization.

1.2 OBJETIVO

Este trabalho visa melhorar a estabilidade e velocidade do caminhar de robôs humanoides Darwin utilizando a técnica de otimização PSO. Este estudo é destinado às pesquisas na área e a competição de futebol de robôs realizadas pela RoboCup. A técnica de locomoção

utilizada pelo robô humanoide deste trabalho será a técnica proposta por Ha, Tamura e Asama (2011) que utiliza osciladores lineares acoplados.

Serão utilizados os robôs humanoides do Centro Universitário da FEI que atendem os requisitos da RoboCup. Apesar destes robôs estarem sendo utilizados para o desenvolvimento no domínio do futebol de robôs, eles podem ser utilizados para pesquisas do Centro Universitário da FEI em outros ramos da robótica que utilizem robôs humanoides.

1.3 A ROBOCUP

A RoboCup é uma competição que visa promover as pesquisas nas áreas da robótica e da Inteligência Artificial oferecendo um desafio formidável, mas que também seja publicamente atraente. Este desafio consiste em realizar, até a metade do século 21, uma partida de futebol, utilizando as regras oficiais da FIFA, entre um time completo de robôs humanoides autônomos e o time campeão da Copa do Mundo mais recente e o time de robôs deve vencer (ROBOCUP, s.d.).

A ideia de fazer robôs jogarem futebol teve sua primeira menção no artigo do professor Alan Mackworth intitulado ”On Seeing Robots” de 1992. Independentemente, um grupo de pesquisadores japonês estava organizando um *Workshop* sobre grandes desafios da Inteligência Artificial e este *Workshop* direcionou as discussões para a ideia de colocar robôs para jogar futebol com a finalidade de promover ciência e tecnologia. Foram estas discussões que foram responsáveis pela criação da *Robot J-League*, porém devido a reclamações de pesquisadores fora do Japão, este projeto foi estendido para uma iniciativa internacional (ROBOCUP, s.d.).

Foi durante o IJCAI-95 que foi anunciado que seria realizado o primeiro *Robot World Cup Soccer Games and Conferences* em conjunto com o IJCAI-97. Foi decido que deveria ser realizado um *Pre-RoboCup-96* em 1996 para identificar os problemas associados a realização

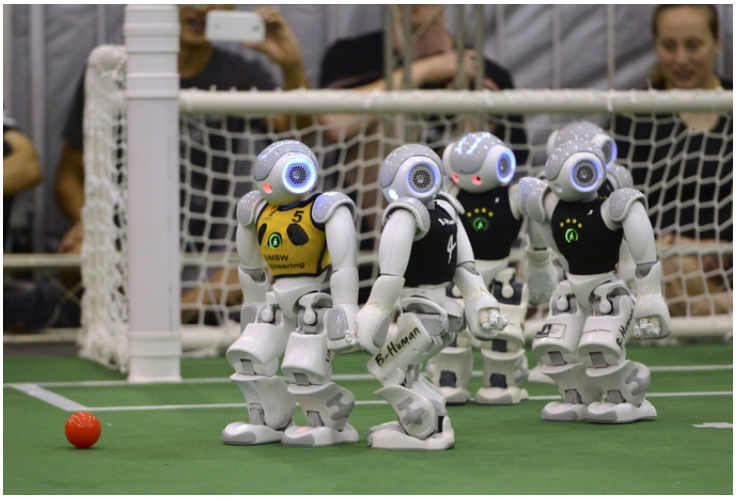


Figura 1 – Partida de futebol de robôs da *Standard Platform league* da RoboCup de 2015

Fonte: RoboCup,.

da RoboCup em larga escala. O *Pre-RoboCup-96* aconteceu durante a *International Conference on Intelligence Robotics and Systems* (IROS-96) em 1996 e contou com oito times competindo na *simulation league* e na demonstração do robô real para a *middle size league* (ROBOCUP, s.d.).

As competições da RoboCup tiveram início em 1997, mesmo ano que o *Deep Blue* da *IBM* derrotou o campeão mundial de xadrez daquela época, o russo Garry Kasparov e a *Pathfinder* obteve êxito em pousar em Marte permitindo que o *rover Sojourner*, um sistema robótico autônomo, pudesse explorar a superfície de Marte. Esta primeira edição da RoboCup foi considerada um sucesso e contou com a participação de mais de 40 times (contando as competições de robôs reais e de simulações) e mais de 5000 espectadores (ROBOCUP, s.d.).

É possível afirmar que a RoboCup está para a robótica assim como a Fórmula 1 está para o automobilismo, ou seja, ambas as competições visam promover o desenvolvimento de suas respectivas áreas utilizando um desafio para acelerar o desenvolvimento de tecnologias e de inovações, e utilizam a competição para atrair o interesse do público.

1.4 ORGANIZAÇÃO DO TRABALHO

Neste primeiro capítulo foi realizado uma introdução e a descrição da motivação e objetivos deste trabalho.

O restante deste trabalho está dividido em 6 capítulos. O Capítulo 2 aborda os conceitos de Cinemática, caminhada em robôs humanoides e *PSO* que serão utilizados neste trabalho. O capítulo 3 aborda trabalhos que possuem temas relacionados com o tema deste trabalho. O capítulo 4 discorre sobre a proposta deste trabalho detalhando os softwares, hardwares, algoritmos e métodos. O capitulo 5 aborda a metodologia utilizada e os resultados obtidos nos experimentos. O capitulo 6 aborda as conclusões que puderam ser feitas a partir dos resultados demonstrados no capitulo 5.

2 TEORIA

Neste capítulo serão abordadas as teorias sobre a Cinemática e a caminhada de robôs humanoides e sobre o algoritmo de otimização *PSO*.

2.1 CINEMÁTICA E CAMINHADA

"Locomoção, a habilidade de um corpo de se mover de um lugar para outro, é uma característica definidora da vida animal. Ela é atingida pela manipulação do corpo com relação ao ambiente." (WESTERVELT et al., 2007). Tanto os animais como os objetos feitos pelo homem, possuem diversos tipos de locomoção, como por exemplo, nadar, voar, andar utilizando pernas ou rodas. No caso de um ambiente com terreno irregular e\ou descontinuo, a locomoção utilizando pernas se mostra a mais apropriada e versátil, pois desta forma é possível ignorar certas descontinuidades e irregularidades no terreno.

O interesse pelo estudo de robôs bípedes em especifico, se deve ao interesse sociológico e comercial de substituir os humanos em tarefas perigosas, como por exemplo mineração, inspeção de uma usina nuclear, intervenções militares, etc, e ao interesse de recuperar o movimento de pessoas com dificuldade de movimentação desenvolvendo próteses capazes de executar um movimento mais similar aos membros humanos, simulações neurais funcionais e exoesqueletos que auxiliem a movimentação de um membro (WESTERVELT et al., 2007).

2.1.1 Cinemática

O corpo do robô humanoide é dividido em 3 planos, seguindo a ideia de um plano cartesiano, como mostra a Figura 2. Estes planos são Frontal, Sagital e Transversal. O plano Sagital é o plano longitudinal que divide o corpo em esquerda e direita. O plano Frontal é o plano paralelo ao eixo longitudinal do corpo e perpendicular ao plano Sagital, separando o corpo em dianteira e traseira. O plano Transversal é o plano perpendicular aos outros dois planos e divide o corpo exatamente no quadril, criando uma parcela acima e uma abaixo do quadril (WESTERVELT et al., 2007).

Robôs possuem uma característica chamada *Grau de Liberdade*. Grau de Liberdade indica quantos movimentos este determinado robô pode realizar. Esta característica é definida pelos atuadores disponíveis no robô, onde cada um destes atuadores pode realizar movimentos em uma, duas ou três dimensões. O HUBO2+ utilizado no trabalho do O'Flaherty et al. (2013), por exemplo, possui 27 Graus de Liberdade, como mostra a figura 4.

O cálculo da Cinemática de um robô permite que o controle manipule as partes do corpo do robô, como por exemplo uma mão, apontando uma posição no espaço ao invés de identificar em qual ângulo θ o atuador posicionado nas juntas deve estar.

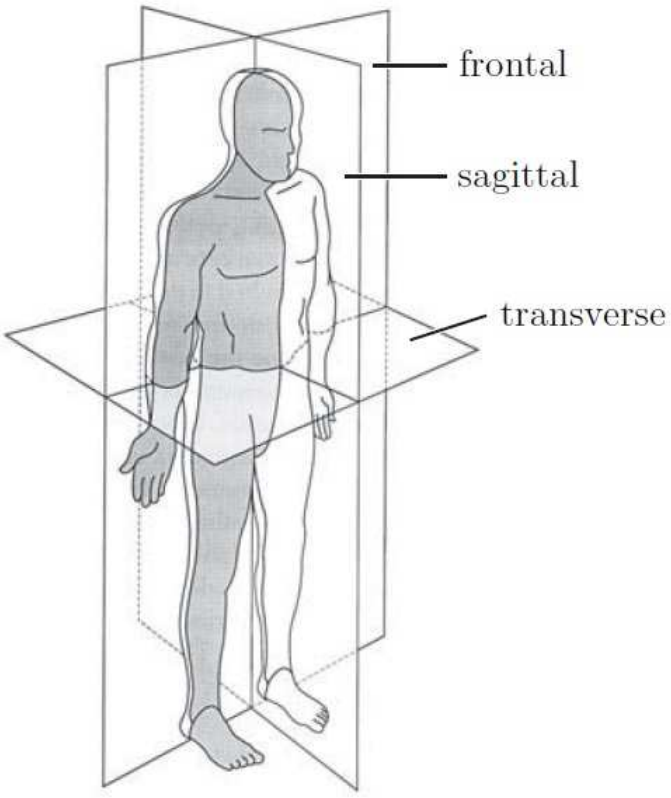


Figura 2 – Figura representando os planos do corpo de um robô humanoide

Fonte: Westervelt et al., 2007.

Figura 3 – Robô HUBO2+

Fonte: O'Flaherty et al., 2013.

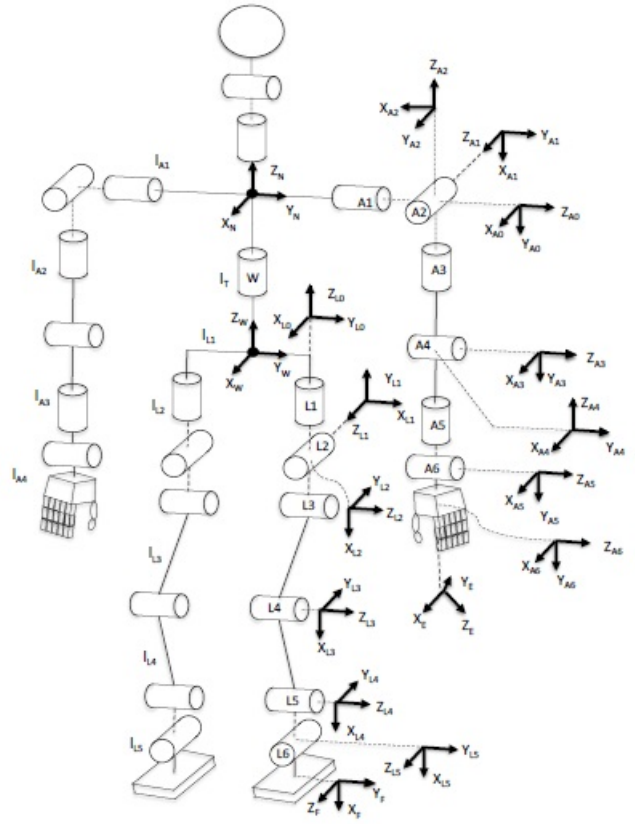


Figura 4 – Graus de Liberdade do Robô HUBO2+

Fonte: O'Flaherty et al., 2013.

No trabalho do O'Flaherty et al. (2013) foi utilizado os parâmetros *Denavit-Hartenberg* (DH) para a realização dos cálculos de cinemática para as pernas e para os braços do robô. Neste trabalho foi feito o cálculo da *Cinemática Direta* e da *Cinemática Inversa*. A *Cinemática Direta* é responsável por encontrar a posição e orientação da parte do corpo do robô sendo movimentada dado o ângulo dos atuadores das juntas. Já a *Cinemática Inversa* é responsável por encontrar o ângulo dos atuadores das juntas dado a posição e orientação da parte do robô sendo movimentada.

A figura 6 mostra a resposta do cálculo da cinemática inversa. É possível observar que os ângulos dos atuadores mostrados, *bacia*, *joelho* e *canela*, são representados utilizando os ângulos de Euler devido ao fato dos atuadores serem motores e somente realizarem movimentos circulares, assim como as juntas dos seres humanos. A figura 5 mostra como os ângulos de Euler são definidos em um robô humanoide.

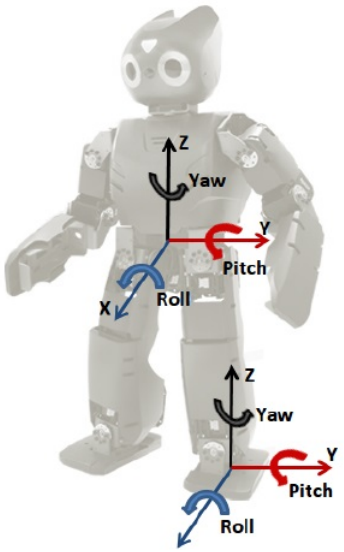


Figura 5 – Ângulos de Euler em um robô humanoide

Fonte: Silva, 2015.

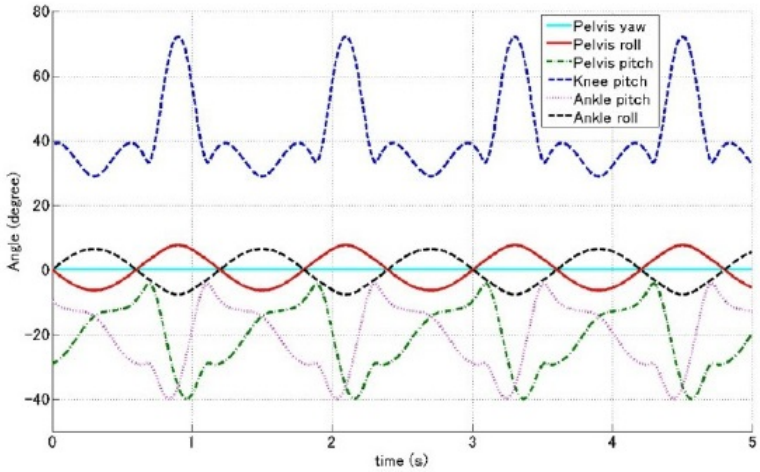


Figura 6 – Valor do ângulo das juntas calculado utilizando a Cinemática Inversa

Fonte: Ha; Tamura; Asama, 2011.

Uma das ferramentas para realizar o cálculo da cinemática é a notação de *Denavit-Hartenberg* (DH). Esta notação se baseia no fato de que para determinar a posição relativa de duas retas no espaço, somente será necessário definir dois parâmetros (HOLLERBACH, 1985):

a) Distância entre as duas retas, medida utilizando a reta normal comum a ambas as retas.

b) Ângulo de rotação em torno da reta normal comum a ambas as retas (regra da mão direita), necessário para que ambas as retas se tornem paralelas.

Desta forma, se para determinar a posição relativa de duas retas são necessários os dois parâmetros descritos anteriormente, para determinar a posição relativa de um sistema de coordenadas, será necessário utilizar quatro parâmetros. Isso se deve ao fato de que em um espaço

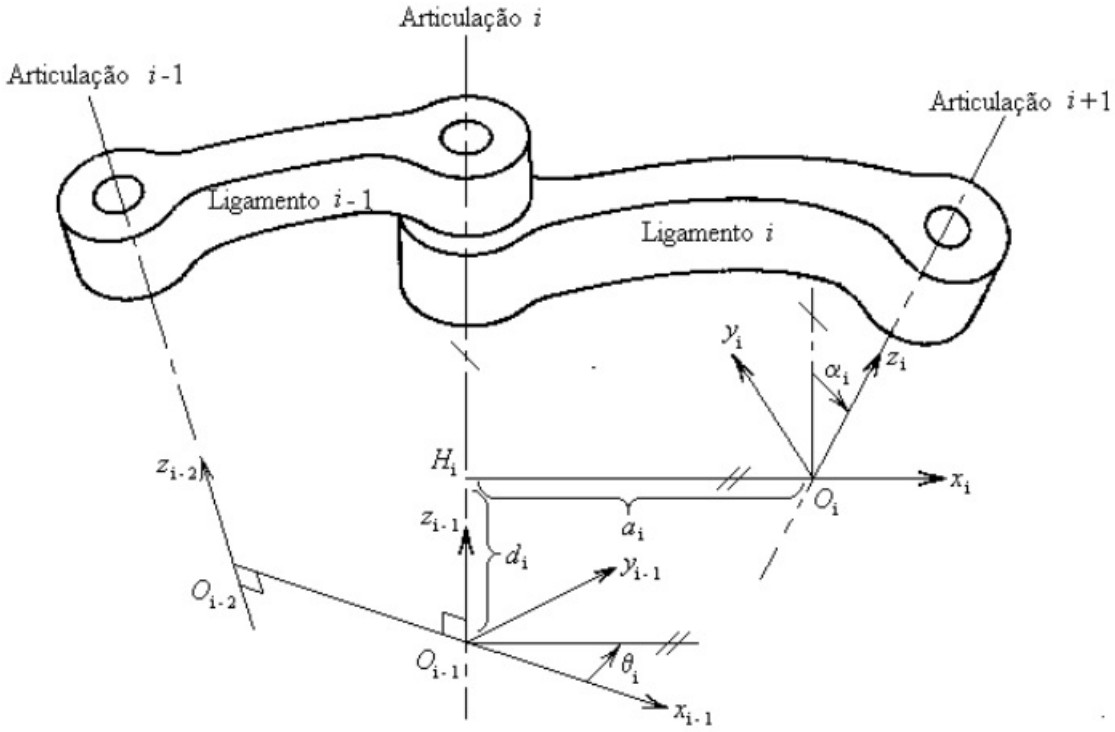


Figura 7 – Notação *Denavit-Hartenberg* (DH)

Fonte: Baglioni et al., 2013.

tridimensional é possível definir o terceiro eixo de um sistema de coordenadas conhecendo somente dois dos eixos devido as condições de ortogonalidade.

Portanto a notação DH consiste de quatro parâmetros que são dependentes da geometria dos conectores e atuadores de um robô. A figura 7 mostra um par de ligamentos adjacentes (i e $i-1$) e suas respectivas articulações ($i-1$, i e $i+1$). Nesta representação, os eixos z dos sistemas de coordenadas são os eixos ao longo das articulações e a reta H_iO_i é a reta normal entre as articulações i e $i+1$ e portanto normal aos eixos z_{i-1} e z_i.

Os parâmetros DH representados na figura 7 são (CRAIG, 2005):

a) a_i: é a distância, em módulo, entre os eixos z_{i-1} e z_i ao longo da normal comum entre esses eixos (no caso equivalente ao eixo x_i), ou seja, o tamanho da reta H_iO_i;

b) α_i: é o ângulo, com sinal, entre o eixo z_{i-1} e o eixo z_i, medido sem torno do eixo x_i (normal comum a ambos os eixos) utilizando a regra da mão direita, que torna os eixos z_{i-1} e z_i paralelos;

c) d_i: é a distância, com sinal, entre os eixos x_{i-1} e x_i medido ao longo do eixo z_{i-1}, que é a normal comum entre os eixos x_{i-1} e x_i. A distância é medida a partir da origem O_{i-1} e é positiva se acompanha a direção do eixo z_{i-1} e negativa se acompanha o sentido oposto ao eixo z_{i-1};

d) θ_i: é o ângulo, com sinal, entre os eixos x_{i-1} e x_i, medido em torno do eixo z_{i-1} (normal comum entre os eixos x_{i-1} e x_i) utilizando a regra da mão direita, que torna os eixos x_{i-1} e x_i paralelos;

Com estes quatro parâmetros é possível definir a posição e orientação do sistema de coordenadas i em relação ao sistema $i-1$ utilizando a sequência de transformações representada pela equação 1,

$$A_{i-1}^{i} = Rot(z,\theta_i)Trans(z,d_i)Trans(x,a_i)Rot(x,\alpha_i), \tag{1}$$

onde Rot representa uma transformação de rotação e $Trans$ representa uma transformação de translação e A_{i-1}^{i} representa uma transformação homogênea (CRAIG, 2005).

Do efetuador à base existem n transformações homogênea consecutivas, assim a posição e orientação do efetuador em relação a base é dada por (CRAIG, 2005):

$$A_0^n = A_0^1(q_1)A_1^2(q_2)...A_{n-1}^n(q_n), \tag{2}$$

onde q representa a posição da articulação, se a articulação for de revolução será o θ, se a articulação for prismática será o d.

Como os quatro parâmetros da notação DH dependem da geometria das articulações, estes parâmetros podem ser variáveis ou constantes. No caso da figura 7, os parâmetros a_i e α_i são constantes e dependem somente da geometria do ligamento i, enquanto que os parâmetros d_i e θ_i são variáveis e dependem da movimentação das articulações.

Desta forma, é possível seguir o algoritmo 1 para calcular a cinemática direta (CABRAL, s.d.):

Algoritmo 1 – Algoritmo para o cálculo da cinemática direta

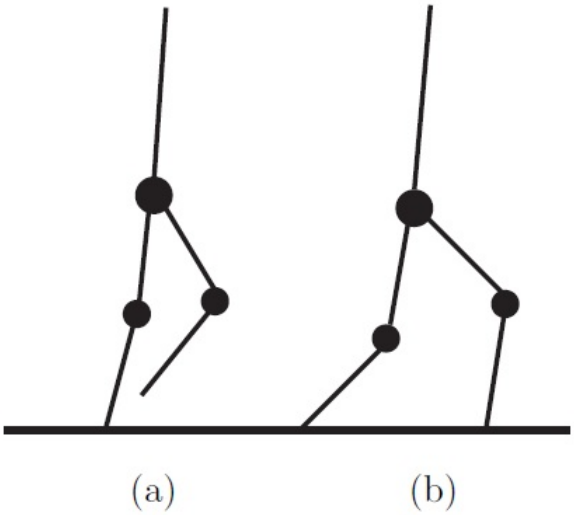
Dados: n = número de articulações
; **para** i *de 0* **até** $n-1$ **faça**
  Localizar o eixos da articulação i, ou seja, o eixo z_i
fim
Estabelecer o sistema de coordenadas da base. A origem deste sistema pode ser escolhida em qualquer ligar do eixo z_0.
para i *de 1* **até** $n-1$ **faça**
  Localizar a origem do sistema i (O_i). O ponto O_i será o ponto onde a normal comum aos eixos z_i e z_{i-1} intercepta o eixo z_i. Caso o eixo z_i intercepte o eixo z_{i-1}, o ponto O_i estará localizado na interseção. Caso os eixos z_i e z_{i-1} forem paralelos, o ponto O_i estará localizado na articulação i.
  Estabelecer o eixo x_i ao longo da normal comum entre os eixos z_i e z_{i-1} e partindo do ponto O_i. O sentido do eixo x_i é do eixo z_{i-1} para o eixo z_i. Caso os eixos z_i e z_{i-1} se cruzem, o eixo x_i será normal a ambos.
  Tendo os eixos z_i e x_i, definir o eixo y_i através da condição de ortogonalidade.
fim
Estabelecer o sistema de coordenadas do efetuador, sistema $x_n y_n z_n$ com origem no ponto O_n escolhido de forma arbitrária. Geralmente o ponto O_n é definido como o centro da garra ou outro ponto de interesse do efetuador. Os eixos deste sistema também são definidos de forma arbitrária, porém é comum definir o eixo x_n como o eixo perpendicular ao eixo z_{n-1}. O eixo z_n normalmente é posicionado paralelo ao eixo z_{n-1}.
Criar uma tabela com os parâmetros de *Denavit-Hartenberg* referente a cada um dos ligamentos e articulações.
Montar as matrizes de transformação homogênea, $A_{i-1}^{i}(q_i)$, a partir dos parâmetros de *Denavit-Hartenberg*.
Obter a matriz de transformação homogênea $A_0^n(q_i,...,q_n)$, a partir da equação 2 que relaciona a posição e orientação do efetuador em relação ao sistema da base.
retorna

2.1.2 Caminhada

Analogamente a um ser humano, um robô humanoide durante a caminhada passa por duas fases, a Single Support Phase (Swing Phase) e a Double Support Phase, como mostra a Figura 8.

Utilizando estas fases, é possível definir a diferença entre o ato de **Correr** e o ato de **Caminhar**. A **Caminhada** é a sequência cíclica da fase de suporte único seguida da fase de suporte duplo e assim por diante. A **Corrida** é uma sequência de fases de suporte único (WESTERVELT et al., 2007).

Figura 8 – Fases do andar de um robô bípede com pês pontudos. Em (a) a fase de suporte único (*Single Support Phase*) e em (b) a fase de suporte duplo (*Double Support Phase*)



Fonte: Westervelt et al., 2007.

Uma das técnicas para gerar o movimento de caminhada em um robô humanoide é chamada de *Zero Moment Point* (ZMP). O *ZMP* foi proposto por Vukobratović e Borovac (2004) e consiste em um ponto no qual as componentes horizontais da força de reação (gerada pelo atrito com o chão) serão balanceadas com a força realizada pelo tornozelo do robô, fazendo assim com que os momentos horizontais sejam:

$$\begin{aligned} M_x &= 0, \\ M_y &= 0. \end{aligned} \tag{3}$$

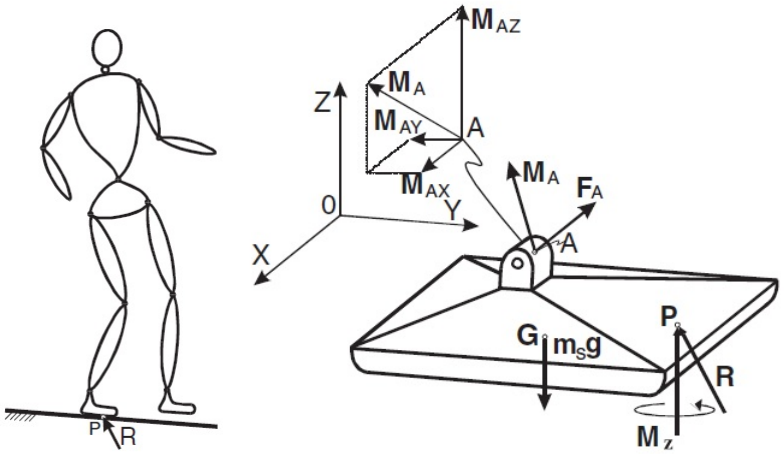
Em certas situações o ZMP coincide com o ponto P mostrado na figura 9. O ponto P é o ponto ao qual a força resultante de reação do chão, devido ao atrito e normal, será aplicada ao polígono de suporte (pé do robô). O ZMP pode ser calculado utilizando a fórmula

$$\overrightarrow{OP} \times \vec{R} + \overrightarrow{OG} \times m_s g + M_A + M_Z + \overrightarrow{OA} \times F_A = 0, \tag{4}$$

Onde $\overrightarrow{OP}$, $\overrightarrow{OG}$ e $\overrightarrow{OA}$ são os vetores de posição do ponto de ação da força de reação (ponto P), do centro de massa do pé (ponto G) e da junta do calcanhar do robô (ponto A) respectivamente.

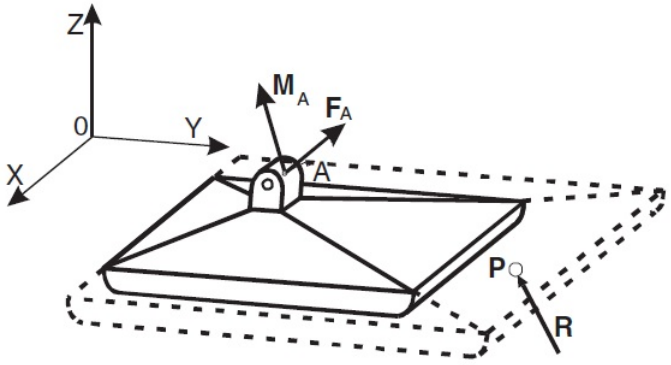
Ao se calcular a posição do ponto ZMP, é possível descobrir que este se encontra fora da área do polígono de suporte do robô como no caso da Figura 10. Neste caso, o ponto P não coincidirá com o ZMP, pois a força resultante da soma das forças de reação do chão deve ser aplicada ao polígono de suporte, sendo assim, este ZMP encontrado será um ZMP fictício (FZMP). Desta forma, o ponto P se encontrará na borda do polígono de suporte e os momentos horizontais Mx e My não serão mais iguais a zero, resultando em um momento resultante que

Figura 9 – Identificação do ponto P em um robô humanoide e as forças de reação no pé deste



Fonte: Vukobratović; Borovac, 2004.

Figura 10 – ZMP calculado encontrado fora do polígono de suporte, portanto instável. Polígono de suporte em pontilhado seria estável



Fonte: Vukobratović; Borovac, 2004.

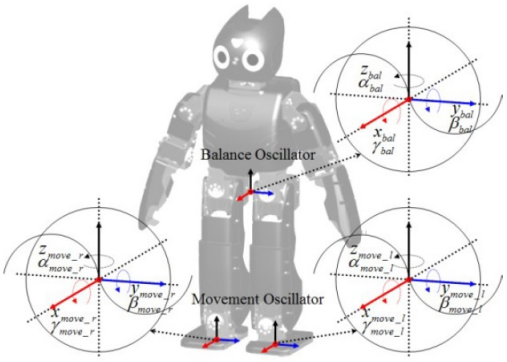
fará o robô rotacionar em torno do ponto P, tornando o andar instável (VUKOBRATOVIĆ; BOROVAC, 2004).

No caso do ZMP calculado se encontrar dentro do polígono de suporte, o ZMP coincidirá com o ponto P, ou seja, os momentos horizontais M_x e M_y serão iguais a zero fazendo com que o robô não rotacione em torno do ponto P ficando estável (VUKOBRATOVIĆ; BOROVAC, 2004).

É importante destacar que o ZMP muda de posição conforme a força F_A e momento M_A são aplicados ao pé, portanto o robô precisa manter o ZMP dentro do polígono de suporte para se manter estável na fase de suporte único da caminhada.

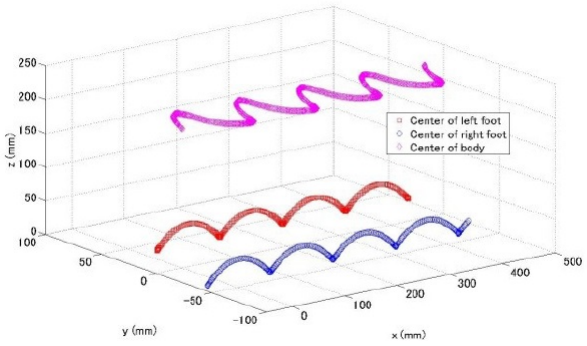
Righetti e Ijspeert (2006) propuseram outra técnica utilizada para realizar a caminhada, a criação de um sistema de osciladores não-linear acoplados para serem usados como um análogo programável de um Gerador de Padrões Central (CPG). Um CPG é uma rede neural biológica que pode ser encontrada em animais vertebrados e invertebrados e quando ativada, produz pa-

Figura 11 – Robô DARwIn-OP e o modelo de Osciladores acoplados



Fonte: Ha; Tamura; Asama, 2011.

Figura 12 – Trajetoria dos pés e do Centro de Massa



Fonte: Ha; Tamura; Asama, 2011.

drões motores ritmados sendo capaz de controlar o andar, a respiração, o voo e o nadar destes animais na ausência de estímulos que carregam informação em tempo real, como por exemplo um estímulo sensorial (MARDER; BUCHER, 2001).

Entretanto, devido ao tamanho do espaço de parâmetros do CPG e da relação entre os parâmetros do CPG e as condições do ambiente, se mostra difícil a implementação desta abordagem em um robô humanoide real (HA; TAMURA; ASAMA, 2011).

Com base na abordagem utilizando CPG, Ha, Tamura e Asama (2011) propuseram a utilização de um modelo de osciladores lineares acoplados simplificado dividido em um oscilador de movimento e um oscilador de equilíbrio. Esta abordagem permite um caminhar estável e não necessita do cálculo em tempo real do *Zero Moment Point* pois é possível definir parâmetros para o oscilador de equilíbrio que garantam a estabilidade.

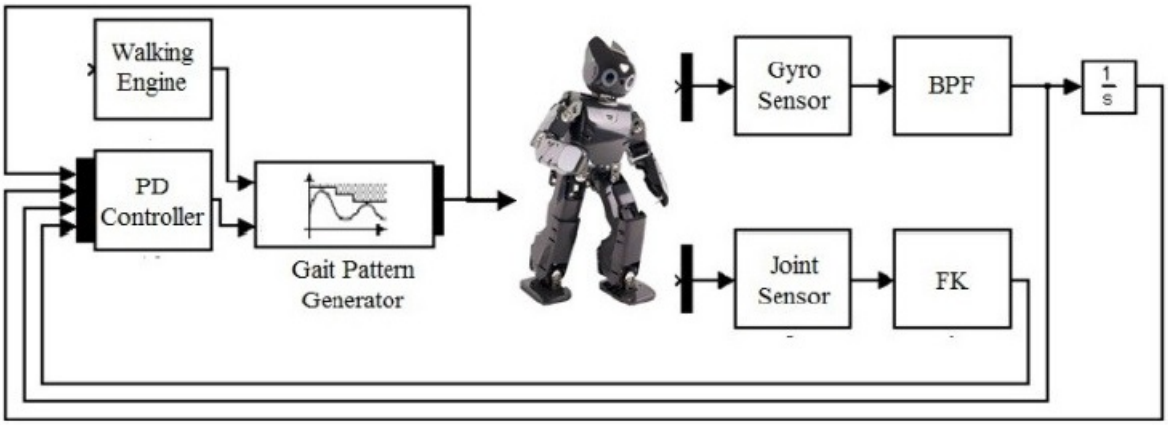
Esta abordagem foi testada em um DARwIn-OP e a estabilidade foi análise calculando o ZMP. Nesta analize foi observado que é possível ajustar a trajetória do ZMP alterando os valores dos parâmetros do oscilador de equilíbrio (osc_{bal}).

A figura 11 mostra que para cada grau de liberdade foi criado um oscilador de movimento e um oscilador de equilíbrio. O resultado da soma de cada um destes osciladores para os pés (direito e esquerdo) e para o Centro de Massa (quadril) podem ser observadas na figura 12. Estes osciladores são controlados por parâmetros como amplitude ρ, velocidade angular ω, fase δ, offset μ, razão da fase de suporte duplo r, seguindo as equações 5, 6 e 7 (HA; TAMURA; ASAMA, 2011).

$$osc_{total} = osc_{move} + osc_{bal} \tag{5}$$

$$osc_{bal} = \rho_{bal} \sin(\omega_{bal} t + \delta_{bal}) + \mu_{bal} \tag{6}$$

Figura 13 – Malha de controle da caminhada do Robô DARwIn-OP utilizando a técnica de Osciladores acoplados



Fonte: Ha; Tamura; Asama, 2011.

$$osc_{move} = \begin{cases} \rho_{move} & \left[0, \frac{rT}{4}\right) \\ \rho_{move} \sin(\omega_{move} t + \delta_{move}) & \left[\frac{rT}{4}, \frac{T}{2} - \frac{rT}{4}\right) \\ -\rho_{move} & \left[\frac{T}{2} - \frac{rT}{4}, \frac{T}{2} + \frac{rT}{4}\right) \\ \rho_{move} \sin\left\{\omega_{move}\left(t - \frac{rT}{2}\right) + \delta_{move}\right\} & \left[\frac{T}{2} + \frac{rT}{4}, T - \frac{rT}{4}\right) \\ \rho_{move} & \left[T - \frac{rT}{4}, T\right) \end{cases} \tag{7}$$

Os autores propuseram a utilização de uma malha de controle fechada para manter a estabilidade da caminhada de robô humanoide que utiliza a técnica de osciladores acoplados. Essa malha de controle é mostrada na figura 13. Para isso foi considerado que o modelo de dinâmico de um robô humanoide é um modelo de pêndulo invertido. Assim, a correção do erro desta malha de controle foi desenvolvida para gerar um toque de cancelamento necessário para manter a estabilidade (HA; TAMURA; ASAMA, 2011).

O torque de compensação é calculado utilizando a formula 8 e então este torque é adicionado ao oscilador de balanço através do parâmetro μ_{bal}. Mesmo com o significado físico do parâmetro μ_{bal} não sendo um torque está sendo considerado que μ_{bal} opera como um gerador de torque, isso porque os atuadores do robô DARwIn-OP somente suportam o controle de suas posições e velocidades. Assim, o oscilador osc_{bal} pode ser escrito como na formula 9 (HA; TAMURA; ASAMA, 2011).

$$\tau = k_p \theta_r + k_d \dot{\theta}_r \tag{8}$$

$$osc_{bal} = \rho_{bal} \sin(\omega_{bal} t + \delta_{bal}) + k_p \theta_r + k_d \dot{\theta}_r \tag{9}$$

Onde: θ_r é o ângulo das juntas, o $\dot{\theta_r}$ é o dado do giroscópio, k_p e k_d são ganhos e são encontrados por métodos experimentais.

Os parâmetros utilizados pelos osciladores para gerar a caminhada podem ser divididos em 3 grupos conforme a tabela 1:

Tabela 1 – Grupo de parâmetros da técnica de caminhada utilizando osciladores lineares acoplados

Parâmetros de Ajuste (offset)	**Parâmetros dos Osciladores**	**Parâmetros da Realimentação**
X_offset	$period\_time$	$balance\_knee\_gain$
Y_offset	dps_ratio	$balance\_ankle\_pitch\_gain$
Z_offset	$step\_forward\_back\_ratio$	$balance\_hip\_roll\_gain$
$roll\_offset$	$foot\_height$	$balance\_ankle\_roll\_gain$
$pitch\_offset$	$swing\_right\_left$	
yaw_offset	$swing\_top\_down$	
hip_pitch_offset	arm_swing_gain	
$pelvis\_offset$		

Fonte: Silva, 2015.

O grupo *Parâmetros de ajuste (offset)* contem parâmetros que podem ser configurados enquanto o robô está estático, pois estes parâmetros não necessitam de testes dinâmicos. O grupo *Parâmetros de Osciladores* é responsável por gerar as oscilações responsáveis por realizar a caminhada do robô humanoide. Já o grupo *Parâmetros de Realimentação* contém os parâmetros relacionados ao ganho da realimentação do giroscópio (SILVA, 2015). A tabela 2 mostra em detalhes cada um dos parâmetros.

Devido ao baixo consumo computacional da técnica utilizando os osciladores lineares acoplados quando comparada com a técnica do cálculo dinâmico do *Zero Moment Point* (ZMP), esta é a técnica utilizada pela equipe RoboFEI do Centro Universitário da FEI e por diversas outras equipes participantes da RoboCup.

Tabela 2 – Descrição dos parâmetros da técnica de caminhada utilizando osciladores lineares acoplados

Parâmetro	**Descrição**
X_offset	Deslocamento dos pés na direção x em milímetros
Y_offset	Deslocamento dos pés na direção y em milímetros
Z_offset	Faz com que o robô agache na direção do eixo z em milímetros
roll_offset	Deslocamento no ângulo dos pés em torno do eixo x em graus, inclinando os pés para a direita ou para a esquerda em conjunto
pitch_offset	Deslocamento no ângulo dos pés em torno do eixo y em graus, inclinando os pés para frente ou para trás em conjunto (essa rotação ocorre nos servos do tornozelo)
yaw_offset	Deslocamento no ângulo da perna em torno do eixo z em graus
hip_pitch_offset	Inclinação do corpo na altura do tronco. Usa-se uma unidade especial referente ao motor que vale 2.85 graus
period_time	Tempo necessário para completar um passo, onde levanta e abaixa o pé esquerdo e o direito. A unidade está em milissegundo
dsp_ratio	Razão entre o tempo em que ambos os pés estão no chão para quando apenas um pé (esquerdo ou direito) está no chão
step_forward_back_ratio	Distância na direção X da diferença entre o pé esquerdo e o direito durante a caminhada. A unidade está em milímetro
foot_height	Altura máxima que o pé levanta durante a caminhada. Unidade em milímetros
swing_right_left	Balanço da cintura que o robô realiza durante a caminhada na direção do eixo y, balançando entre a direita e a esquerda. Unidade em milímetros
swing_top_down	Balanço da cintura que o robô realiza durante a caminhada na direção do eixo z, subindo e descendo o corpo. Unidade em milímetros
pelvis_offset	Ângulo de deslocamento da pélvis ao longo do eixo X. Usa-se uma unidade especial referente ao motor que vale 2.85 graus
arm_swing_gain	Ganho na oscilação do braço em relação ao comprimento do passo para frente ou para trás, durante o andar
balance_knee_gain	Ganho correspondente ao valor de correção que ocorre no joelho, usado durante a realimentação do giroscópio
balance_ankle_pitch_gain	Ganho correspondente ao valor de correção que ocorre no tornozelo no pitch, usado durante a realimentação do giroscópio
balance_hip_roll_gain	Ganho correspondente ao valor de correção que ocorre no quadril no roll, usado durante a realimentação do giroscópio
balance_ankle_roll_gain	Ganho correspondente ao valor de correção que ocorre no tornozelo no roll, usado durante a realimentação do giroscópio
p_gain	Ganho correspondente ao P (Proporcional) do controle de PID
i_gain	Ganho correspondente ao I (Integral) do controle de PID
d_gain	Ganho correspondente ao D (Derivativo) do controle de PID

Fonte: Silva, 2015.

2.2 OTIMIZAÇÃO POR ENXAMEDE PARTICULAS (PSO)

O *PSO* é um algoritmo de otimização capaz de otimizar funções não lineares continuas. Ele foi proposto em 1995 por Eberhart e J. Kennedy (1995) e utiliza como base o comportamento de um bando de pássaros, um cardume de peixes e a teoria de enxame e possui semelhança com a teoria de computação evolutiva, pois o PSO possui características do *Genetic Algotithm* e de *Evolutionary Programming*.

O algoritmo inicia com uma população de resultados aleatória e cada elemento desta população é chamado de Partícula. Cada partícula contém a sua posição atual no hiperespaço das soluções, um valor de adaptação (valor de um determinado ponto da função que se deseja otimizar) e a posição e valor da melhor adaptação (maior valor, no caso de uma maximização e menor valor no caso de uma minimização) encontrada por esta partícula. Esta versão do PSO é chamada de Local, representada pela figura 14, porém neste trabalho será utilizado como base a versão chamada de Global, representada pela figura 15, onde todas as partículas conhecem qual é a posição e o valor da melhor adaptação encontrada durante a execução do algoritmo (EBERHART; J. KENNEDY, 1995).

A otimização utilizando o *PSO* ocorre da seguinte forma, a cada passo (ou geração) as partículas calculam seu valor de adaptação, atualizam a melhor adaptação local ($pbest$) se necessário, atualizam a melhor adaptação global ($gbest$) se necessário e alteram sua velocidade utilizando uma fórmula que se vale de números aleatórios "acelerando" a partícula em direção a $pbest$ e $gbest$ (EBERHART; J. KENNEDY, 1995).

Todas as partículas possuem a mesma dimensão do problema a ser resolvido, e a velocidade em cada uma destas dimensões são calculadas seguindo a seguinte fórmula (EBERHART; J. KENNEDY, 1995),

$$v_{x_{i+1}} = v_{x_i} + c_1 * rand() * (pbest_x - present_x) + c_2 * rand() * (gbest - present_x), \quad (10)$$

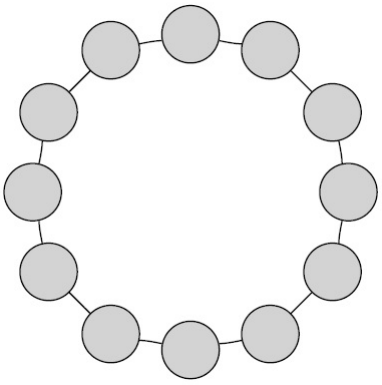


Figura 14 – Topologia Local

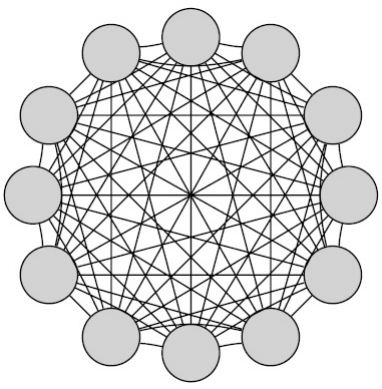


Figura 15 – Topologia Global

Fonte: Bratton; J. Kennedy, 2007.

onde, $v_{x_{i+1}}$ é a nova velocidade na dimensão em questão, v_{x_i} é a velocidade atual na dimensão em questão, c_1 e c_2 são constantes, $rand()$ são números aleatórios entre 0 e 1 e $present_x$ é a posição atual da partícula na dimensão em questão.

Para calcular a nova posição new_x desta partícula é utilizada a velocidade calculada pela equação 10 somado com a posição anterior como mostra a equação 11 (EBERHART; J. KENNEDY, 1995),

$$new_x = present_x + v_{x_{i+1}} \tag{11}$$

A figura 16 ilustra o cálculo utilizado para definir a nova velocidade e posição das partículas. Nesta está representada a soma vetorial da velocidade atual, o $\vec{ct}$, com as distancias entre a melhor posição encontrada pela partícula e a posição atual , o $p\vec{best}$, e entre a melhor posição encontrada pelo enxame e a posição atual, o $p\vec{best}$. Em seguida a posição atual da partícula é somada com o valor da velocidade, o X^{t+1}.

Este ciclo continuará a ser executado até que o critério de parada seja satisfeito, como descrito pelo algoritmo 2 e pelo fluxograma da figura 17. Este critério pode ser o número de gerações (passos) ou a proximidade entre as partículas, dado que se as partículas se aglomerarem em um ponto em especifico, isso significa que o algoritmo convergiu e a resposta foi encontrada (SHI et al., 2001).

Algoritmo 2 – Algoritmo do *Particle Swarm Optimization*.

Dados: vetor X de partículas de dimensão D
início
  Iniciar todas as partículas $\in X$ em posições aleatórias
fim
enquanto *Condição de parada não atingida* **faça**
  para cada *partícula* $x_i \in X$ **faça**
    Avaliar o valor da função a qual se deseja maximizar
    se $valordafunçãoatual < pbest_{va}$ **então**
      $pbest_{va} = valordafunçãoatual$
    fim
    se $valordafunçãoatual < gbest$ **então**
      $gbest = valordafunçãoatual$
    fim
    para cada D **faça**
      Mudar velocidade usando a fórmula: $v_{D_{i+1}} = v_{D_i} + c_1 * rand() * (pbest_D - present_D) + c_2 * rand() * (gbest - present_D)$
    fim
    para cada D **faça**
      Mover a partícula x_i usando a fórmula: $new_D = present_D + v_{D_{i+1}}$
    fim
  fim
fim
retorna *Posição nas D dimensões do $gbest$*

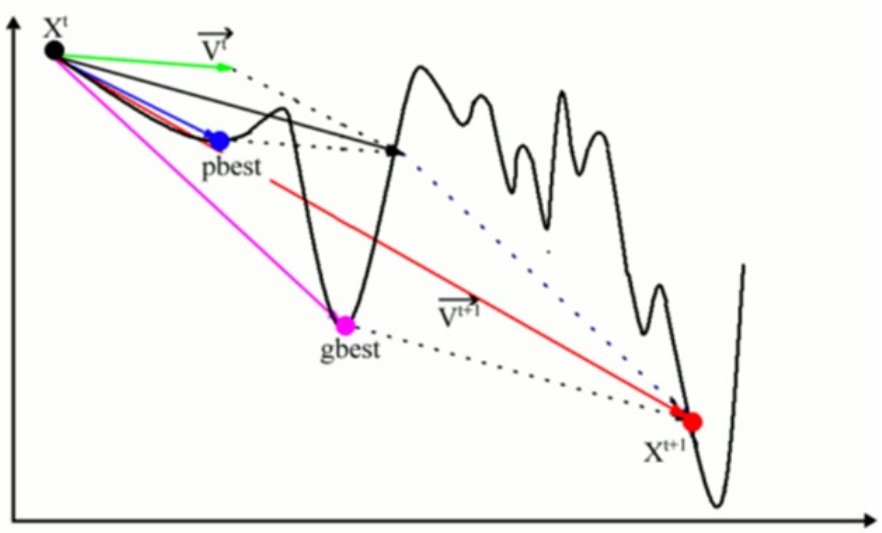


Figura 16 – Exemplo do cálculo da nova posição da partícula

Fonte: Wang et al., 2010.

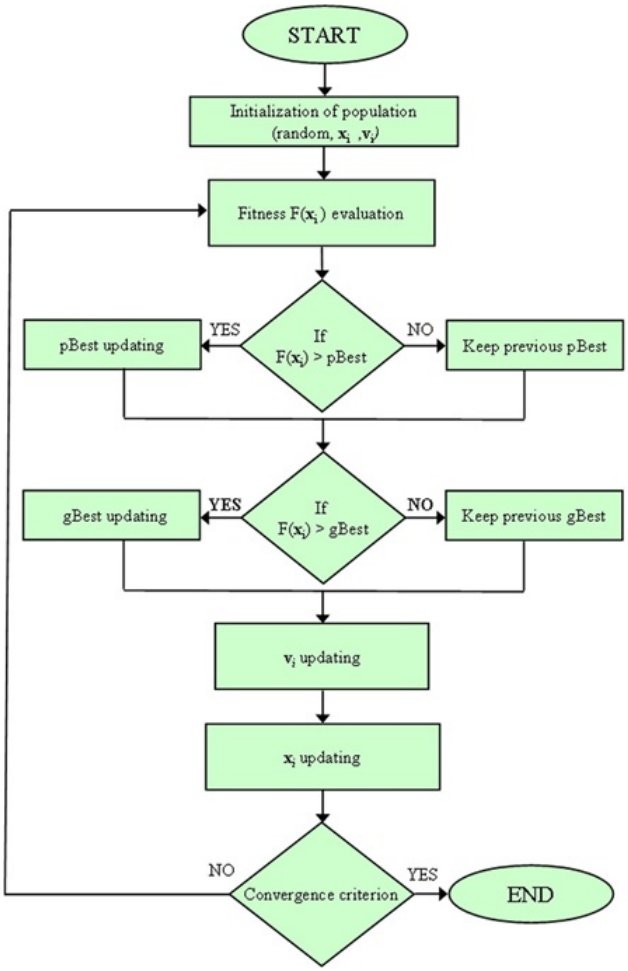


Figura 17 – Fluxograma do *Particle Swarm Optimization*

J. Kennedy e Eberhart (2011) realizaram experimentos com o algoritmo buscando o valor mais adequado para as constantes c_1 e c_2 da equação 10 e concluíram que ao atribuir o valor 2 para as constantes, o algoritmo superava em performance as versões anteriores deste. O valor 2 das constantes faz com que os agentes ultrapassam o alvo metade das vezes. Portanto a equação 10 para calcular a velocidade passa a ser:

$$v_{x_{i+1}} = v_{x_i} + 2 * rand() * (pbest_x - present_x) + 2 * rand() * (gbest - present_x). \quad (12)$$

Em um trabalho, J. Kennedy e Eberhart (2011) demonstra que foram propostas outras versões do PSO. Uma das versões propunha reduzir os dois termos da equação 10 em um único termo, utilizando o ponto médio entre o $pbest$ e o $gbest$, porém nesta versão as partículas convergiam para este ponto médio, sendo ele um ponto ótimo (máximo ou mínimo) ou não.

Outra versão propunha que as partículas fossem separadas em dois grupos, um grupo seria responsável por vasculhar o espaço de resposta buscando um possível $gbest$, enquanto que o outro grupo seria responsável por realizar uma Subida da Encosta ou micro explorar regiões que se mostrassem promissoras de conterem o ponto ótimo. Esta abordagem não apresentou nenhuma melhora na performance se comparada com a performance da versão mais simples representada pela equação 12, portanto, utilizando o princípio da Navalha de Occam, esta versão foi descartada.

Uma terceira versão propunha remover o momento v_{x_i} da equação 12, fazendo com que a equação passasse a ser (J. KENNEDY; EBERHART, 2011),

$$v_{x_{i+1}} = 2 * rand() * (pbest_x - present_x) + 2 * rand() * (gbest - present_x). \quad (13)$$

Entretanto esta versão, apesar de mais simplificada, se mostrou ineficiente para encontrar um ótimo global (J. KENNEDY; EBERHART, 2011).

Trelea (2003) realizou um estudo para identificar a influência do número de partículas no desempenho do algoritmo. Neste estudo foi possível observar que, como esperado, quanto maio o número de partículas, maior será a precisão do algoritmo. Isto ocorre pois com um grande número de partículas é possível varrer o espaço de estados mais completamente, porém esta grande quantidade de partículas faz com que sejam necessários uma elevada quantidade de cálculos durante cada uma das iterações, tornando assim o algoritmo mais lento e, portanto, não ideal para aplicações em tempo real. Os testes utilizando 15 e 30 partículas foram os que apresentaram melhores resultados.

Uma das razões do *PSO* ser atrativo, é o fato deste possuir um número pequeno de parâmetros a ser ajustado. Pequenas ou nenhuma variação são necessárias para que este algoritmo funcione de forma satisfatória em uma ampla variedade de aplicações (TRELEA, 2003).

Entretanto, existem trabalhos como o do Afshinmanesh, Marandi e Rahimi-Kian (2005) que visam utilizar outras técnicas de otimização junto com o PSO para melhorar a performance do PSO e/ou reduzir as deficiências que o PSO apresenta.

No caso do trabalho do Afshinmanesh, Marandi e Rahimi-Kian (2005), foi proposta a unificação de uma das técnicas do algoritmo de **Sistemas Imunológicos artificiais** (CASTRO; TIMMIS, 2003) com o PSO, com a finalidade de melhorar a performance do PSO em problemas com soluções discretas, como por exemplo uma solução binária. A técnica utilizada foi a **Teoria da Seleção Negativa**, que tem como inspiração a capacidade do sistema imunológico de permitir que somente anticorpos que reagem com antígeno externos.

Para isso os anticorpos são ligados com os antígenos do próprio corpo dentro do Timo, glândula que faz parte do sistema imunológico localizada próximo a traqueia. Os anticorpos que reagirem com algum dos antígenos do próprio corpo são eliminados. O anticorpos que não reagirem, saem do Timo e são liberados na corrente sanguínea para tentar reagir com o antígeno externo e elimina-lo (AFSHINMANESH; MARANDI; RAHIMI-KIAN, 2005).

Afshinmanesh, Marandi e Rahimi-Kian (2005) propõe implementar a Seleção Negativa utilizando a velocidade calculada no PSO como sendo o antígeno, ou seja, a velocidade calculada no PSO é confrontada com uma velocidade limite, caso a velocidade seja maior que a velocidade limite, esta velocidade calculada pelo algoritmo do PSO é descartada e uma velocidade aleatória é utilizada como substituta.

Neste trabalho em questão (AFSHINMANESH; MARANDI; RAHIMI-KIAN, 2005), é apresentado evidencias que a mescla do algoritmo de *Imunidade Artificial*, e do algoritmo de *PSO* resultou na melhoria da taxa de convergência e na eliminação de máximos locais.

No trabalho de Li, Zhan e Hao (2017) é proposto a unificação do PSO com outro aspeto de *Sistemas Imunológicos artificiais*, a **Seleção de Clones** (*Clonal Selection*). Esta abordagem proposta foi denominada de *Immune Particle Swarm Optmization* e tem como inspiração o método como o sistema imunológico do ser humano se comporta ao enfrentar um antígeno externo utilizando uma célula chamada de *B-cells*, célula que produz anticorpos com uma configuração especifica para antígenos específicos (GREENSMITH; WHITBROOK; AICKELIN, 2010). Esta técnica altera a população calculada pelo PSO utilizando a *Seleção de Clones* com a finalidade de melhorar a performance do algoritmo de PSO e impedir uma convergência prematura do algoritmo em um ótimo local.

Observando o problema de balancear a exploração na direção do $pbest$ e na direção do $gbest$, Maurice Clerc e James Kennedy (CLERC; J. KENNEDY, 2002) propuseram uma abordagem conhecida como constrição. Nesta abordagem a fórmula tradicional do cálculo da velocidade de uma partícula foi multiplicada por um fator chamado de *fator de constrição* (χ) e calculado utilizando a seguinte fórmula (BRATTON; J. KENNEDY, 2007):

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \varphi = c_1 + c_2$$

Foi identificado que quando $\varphi < 4$ o enxame possui uma tendência a se mover como uma espiral ao redor da melhor solução encontrada sem a garantia de convergência. Quando o $\varphi > 4$ o enxame passa a convergir de forma rápida e garantida (CLERC; J. KENNEDY, 2002) (BRATTON; J. KENNEDY, 2007).

Esta abordagem também tem como benefício impedir que o algoritmo exploda sem a necessidade de utilizar um valor máximo para a velocidade (v_max) de cada partícula. A utilização de um v_max é vista como artificial e difícil de se balanceada, pois é muito difícil encontrar o valor adequado para cada dimensão do problema. Problemas com um espaço de solução muito grande precisam de valores de v_max mais elevados para permitir uma exploração mais adequada, enquanto que problemas com espaço de solução pequenos precisam de valores menos elevados para prevenir que o algoritmo se comporte como se estivesse explodido.

Estas 3 abordagens mostram que embora o PSO se mostre como um algoritmo de otimização promissor, é possível utilizar outros algoritmos de otimização para melhorar a performance do PSO, fazendo com que ele convirja mais rápido para o resultado ótimo e/ou tenha uma maior capacidade de eliminar os ótimos locais aumentando assim sua eficiência.

O PSO está sendo utilizado na evolução dos pesos de Redes Neurais e na estrutura destas (YAO, 1999; J. F. KENNEDY et al., 2001 **apud** TRELEA, 2003). Outras aplicações a qual o PSO vem sendo utilizado são: otimização multidimensional de operações em ambiente de manufatura (TANDON, 2000 **apud** TRELEA, 2003), otimização do controle de tensão e potência reativa de uma unidade elétrica japonesa (FUKUYAMA; NAKANISHI, 1999 **apud** TRELEA, 2003) e uma empresa americana vem utilizando o PSO para a otimização da mistura de ingredientes, conseguindo assim obter resultados duas vezes melhores utilizando o PSO em comparação com métodos mais tradicionais de otimização industrial.

3 TRABALHOS RELACIONADOS

Esta seção apresenta 4 trabalhos que buscam a otimização da caminhada de um robô humanoide através da otimização dos parâmetros do oscilador de caminhada proposto por Righetti e Ijspeert (2006). O primeiro trabalho utiliza um algoritmo de aprendizado por reforço para realizar a otimização, enquanto que o segundo trabalho utiliza o PSO, o terceiro utiliza o PSO e o GA e compara estes. O quarto trabalho utiliza o CPG para gerar a caminhada e o NEPSO para otimiza-la.

No trabalho do Silva (2015) foi realizada a otimização do caminhar de um robô humanoide utilizando o algoritmo de aprendizado por reforço. Os robôs utilizados neste trabalho foram os robôs B1 e B2 pertencentes ao Centro Universitário da FEI que participam de competições como a RoboCup. Além do robô físico, foi utilizado o simulador de robótica Webots.

Para a otimização dos parâmetros foi utilizado o algoritmo TD(λ) (SUTTON; BARTO, 1998) e uma função $V(s)$ de recompensa que usa como base a velocidade atingida pelo robô e pune o robô quando este sofre uma queda. Após o aprendizado, foi possível encontrar os parâmetros que faziam com que o robô possuísse um caminhar rápido e estável, andando a uma velocidade de $19.68\ cm/s$ e não caindo durante o experimento. Durante os experimentos também foi possível identificar os parâmetros que geravam o caminhar mais rápido ($28.79\ cm/s$), porém sem muita estabilidade.

Entretanto, os parâmetros do caminhar do robô otimizados utilizando o aprendizado por reforço foram os parâmetros *period_time*, *swing_right_left* e o *setXAmplitude* do oscilador de caminhada. Isto porque quanto maior for o tamanho do conjunto de parâmetros que se deseja aprender, mais lento se torna o processo de aprendizagem.

Um segundo trabalho que realizou a otimização da caminhada de um robô humanoide foi o trabalho do Niehaus, Röfer e Laue (2007), onde foi feita a otimização da caminhada de um robô Kondo KHR-1 modificado utilizando o algoritmo de *Particle Swarm Optimizaion* para otimizar alguns dos valores do oscilador de caminhada deste robô.

Neste trabalho, foi primeiramente realizado experimentos utilizando o PSO em um simulador de robótica afim de encontrar os paramentos do PSO, tamanho do enxame (quantidade de partículas) e a forma da vizinhança, mais adequados para o trabalho realizado. A forma da vizinhança define quantas partículas influenciam as demais, pode ser *full* ou *k-random*, onde *full* significa que todas as partículas serão influenciada por todas e *k-random* significa que uma partícula será influenciada por k partículas aleatórias. Após tais experimentos foi definido que a combinação que menos consome *Hardware* e retorna resultados satisfatórios é a combinação de 12 partículas e uma vizinhança com o formato *full*.

Em seguida a este experimento, foi realizada a otimização dos parâmetros do oscilador de caminhada do robô Kondo KHR-1 utilizando o robô físico. Para tal foram definidos alguns parâmetros para serem constantes e os limites para os paramentos que seriam otimizados, foi utilizado o algoritmo PSO e para a função de adaptação do PSO foi utilizada a distância per-

corrida pelo robô durante um tempo limite de 20 segundos e regras que levam em consideração a ocorrência de uma queda, da não movimentação do robô e de um erro no ângulo da caminhada. Os detalhes referentes a função de adaptação foram coletados e inseridos no algoritmo manualmente.

Desta forma, foram necessárias entre 3 e 3,5 horas para realizar 25 iterações afim de otimizar os parâmetros para uma única direção. Após realizar este processo para as caminhadas com os ângulos 0º, 90º, 180º, -45º e -135º os valores foram interpolados resultando em uma caminhada com velocidade de 17 cm/sec em qualquer direção.

O trabalho de Chaohong Cai e Hong Jiang (CAI; JIANG, 2013) compara a performance dos algoritmos Algoritmo Genético (GA), Estratégia Evolutiva da Adaptação da Matriz de Covariância (CMA-ES), Otimização por Enxame de Partícula (PSO) e Evolução Diferencial (DE) para otimizar os parâmetros de caminhada de um robô humanoide utilizando um ambiente de simulação de futebol 3D.

Para realizar a comparação entre os algoritmos foi realizada 3 tarefas de otimização, uma tarefa visando um andar rápido, a segunda visando um andar estável e a terceira visando um andar flexível. Cada tarefa possui uma fórmula de adaptação diferente (CAI; JIANG, 2013).

A tarefa do andar rápido possui como objetivo encontrar os parâmetros que resultem em uma caminhada com a máxima velocidade. A simulação consiste no robô tentando atingir a maior distância possível em 15 segundos. A equação de adaptação desta tarefa é:

$$fit_{fast} = dis_{walked} - P_{fall},$$

onde, fit_{fast} é o valor de adaptação da primeira tarefa, $dist_{walked}$ é a distância percorrida pelo robô e P_{fall} é a penalidade que deve ser adicionada caso o robô caia durante a tarefa. A penalidade P_{fall} possui valor igual a 2.5 (CAI; JIANG, 2013).

A tarefa responsável por encontrar os parâmetros que geram uma caminhada estável, mantém a simulação como a tarefa anterior, porem a fórmula de adaptação é:

$$fit_{stable} = dis_{walked} - P_{fall} - P_{var},$$

onde, fit_{stable} é o valor de adaptação, P_{var} é um termo de penalidade que possui valor 2.0 e deve ser incluído se a soma dos ângulos de *roll* e *pitch* ultrapassam um limite (CAI; JIANG, 2013).

Na última tarefa, a simulação passa a ser executada por 30 segundos e o robô passa a ter o objetivo de levar a bola para o gol. Esta simulação faz com que o robô experimente viradas rápidas, passada lateral e caminhada rápida. A função de adaptação segue a fórmula:

$$fit_{flexble} = dis_{ball} - P_{fall} - P_{var} + R_{goal},$$

onde, $fit_{flexble}$ é o valor de adaptação, dis_{ball} é a distância que a bola andou em direção ao gol e R_{goal} é o termo de recompensa que deve ser adicionado quando a bola entrar no gol durante a simulação, seu valor é 2.0 (CAI; JIANG, 2013).

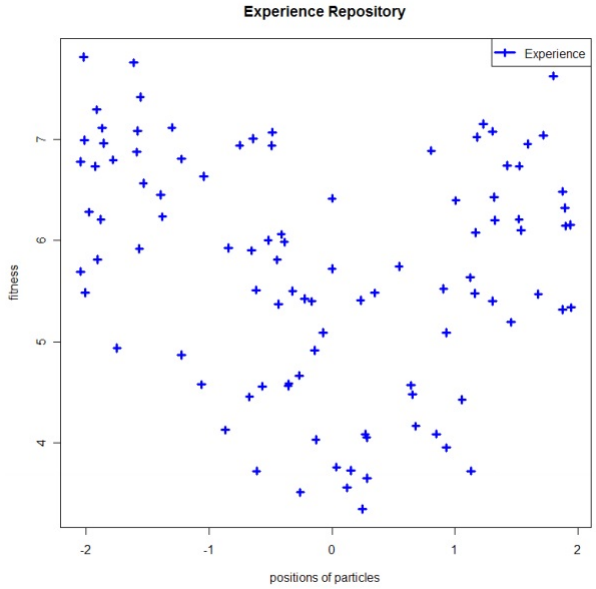


Figura 18 – Repositório de Experiências de tamanho 100

Fonte: Kim; J.-W. Lee; J.-J. Lee, 2009.

Os resultados deste trabalho mostram que o PSO foi capaz de gerar uma caminhada rápida e estável nas tarefas de caminhar rápido e estável. Já o GA gerou uma caminhada mais lenta e menos estável que o PSO (CAI; JIANG, 2013).

Alguns trabalhos utilizam um modelo matemático do *Gerador de padrões Central* (CPG) para gerar o movimento de robôs, sejam eles movimentos de caminhada em um robô bípede ou movimentos de ritmados de um braço mecânico (KIM; J.-W. LEE; J.-J. LEE, 2009), (AVRIN et al., 2016). No caso do trabalho do Kim, J.-W. Lee e J.-J. Lee (2009), foi utilizado o CPG para gerar a caminhada e uma variação do *PSO* para encontrar os parâmetros do CPG que gerassem um caminhar otimizado.

A variação do PSO proposta neste trabalho (KIM; J.-W. LEE; J.-J. LEE, 2009) é denominada de *nonparametric estimation based Particle Swarm Optimization* (NEPSO). Sua principal diferença com relação ao PSO canónico é o fato dele conter um repositório de experiências e utilizar uma estimativa da melhor posição.

O repositório de experiências consiste em um local onde os dados das experiências das partículas são armazenados. No caso do PSO canónico, estes dados de experiências são os melhores valores obtidos por cada partícula. No caso do NEPSO, os dados de experiências armazenados são os dados das M iterações anteriores de todas as partículas. As informações contidas neste repositório são utilizadas para atualizar a velocidade de cada uma das partículas. A figura 18 mostra um repositório de experiências com tamanho 100, ou seja, se o tamanho do enxame for de 10 este repositório armazena as informações de cada partícula durante 10 iterações (AVRIN et al., 2016).

A estimativa da melhor posição é realizada da seguinte forma, K posições dentro do espaço de busca são selecionadas como amostras, em seguida cada uma destas amostras terão

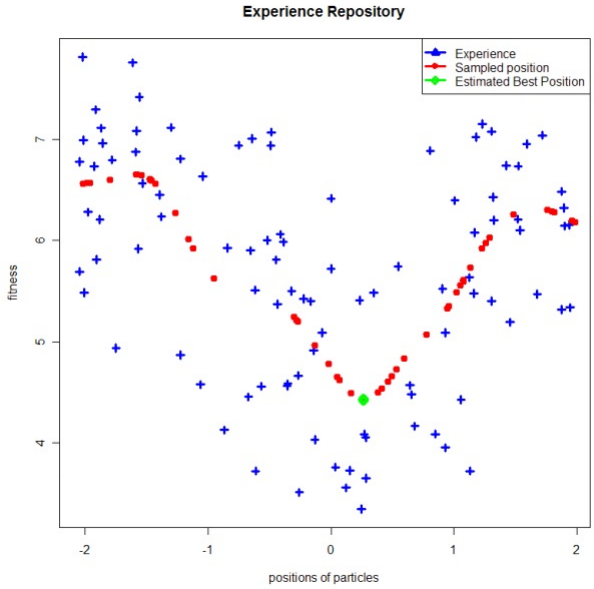


Figura 19 – Repositório de Experiências, amostras e melhor posição estimada

Fonte: Kim; J.-W. Lee; J.-J. Lee, 2009.

seus valores de afinidades estimados utilizando uma regressão não paramétrica e em seguida a amostra com o melhor valor de adaptação estimada é escolhida como a melhor posição. Com isso, a velocidade das partículas serão calculadas utilizando a equação de velocidade baseada no gerador de números aleatórios da distribuição Gaussiana representada pela equação 14 (AVRIN et al., 2016).

$$|randn|(\vec{p}_{best} - \vec{X}(t)) + |randn|(\vec{g}_{best} - \vec{X}(t)), \tag{14}$$

onde $|randn|$ é um numero aleatório positivo gerado de acordo com os valores absolutos da distribuição de probabilidade Gaussiana, $\vec{g}_{best}$ é a melhor posição encontrada pelo enxame, $\vec{X}(t)$ é a posição da partícula na iteração t e$\vec{\epsilon}$é a melhor posição estimada.

A simulação foi realizada utilizando um simulador dinâmico 3D e os resultados obtidos utilizando o NEPSO foi comparado com o PSO canônico e três outras variantes do PSO. Nestas simulações cada um dos parâmetros encontrados pelos PSO foram testados e avaliados gerando assim um valor de adaptação (AVRIN et al., 2016).

O NEPSO se mostrou eficiente na busca dos parâmetros ótimos do *Gerador de padrões Central*(CPG) convergindo mais rapidamente que o PSO canônico, porém esta técnica demanda mais poder computacional, tanto em espaço de memoria, devido ao repositório de experiências, quanto em tempo de processamento, quando comparado ao PSO canônico (AVRIN et al., 2016).

É possível afirmar que o PSO proposto por J. Kennedy e Eberhart (2011) se mostra vantajoso devido o seu baixo custo computacional, alta velocidade de convergência e efetividade. Este algoritmo de otimização se assemelha ao *Algoritmo Genético* e em testes (HASSAN et al., 2005) o PSO se mostrou tão efetivo quanto o GA, pois o poder computacional necessário para

implementar o PSO é menor que o necessário para a implementação do GA. Além disso, o PSO pode ser combinado com outras técnicas de otimização para melhorar sua performance.

4 PROPOSTA

A proposta deste trabalho é utilizar o algoritmo do PSO para otimizar os parâmetros do oscilador de caminhada do robô humanoide Darwin que possui as mesmas características do robô do Centro Universitário da FEI mostrados na tabela 2, visando atingir um caminhar veloz e robusta, ou seja, um caminhar com a maior velocidade possível e com o menor número de quedas possível e comparar os resultados obtidos utilizando a *Otimização por Enxame de Partículas* com os resultados obtidos utilizando o Aprendizado por Reforço no trabalho do Silva (2015).

Este trabalho será executado no simulador de robótica Webots, permitindo que as medições dos dados necessários para o cálculo do valor de adaptação sejam recolhidas de forma automática, permitindo que se executem mais iterações e permitindo aumentar a dimensão do espaço de soluções, ou seja, permitindo a otimização de um número maior de parâmetros.

O PSO foi escolhido pois se trata de um algoritmo de otimização que vem se mostrando promissor devido a sua rápida convergência e o seu baixo custo computacional, o que no caso, poderia permitir a otimização dos parâmetros da caminhada durante uma competição como a da RoboCup.

Os dados obtidos neste trabalho serão comparados aos dados obtidos no trabalho do Silva (2015), que utiliza um algoritmo de aprendizado por reforço para a otimização dos parâmetros da caminhada, pois ambos os trabalhos utilizaram o mesmo simulador e os mesmos robôs. Os dados obtidos também serão comparado com o trabalho do Niehaus, Röfer e Laue (2007) que realizou a otimização da caminhada de um robô Kondo KHR-1 físico utilizando o PSO.

4.1 SOFTWARE

Para a execução deste trabalho será necessário utilizar os seguintes softwares:

a) Simulador de robótica Webots;

b) Linguagem de Programação C++;

c) Software de Controle do Robô desenvolvido utilizando Arquitetura em Cruz

O algoritmo do *PSO* será implementado no ambiente de desenvolvimento do simulador de robótica Webots utilizando a linguagem de programação C++.

4.1.1 Webots

O Webots é um ambiente de desenvolvimento que permite modelar, programar e simular robôs moveis. O controlador do robô pode ser programado utilizando a IDE disponível no Webots e pode ser transferido para robôs disponíveis no mercado. O comportamento do robô

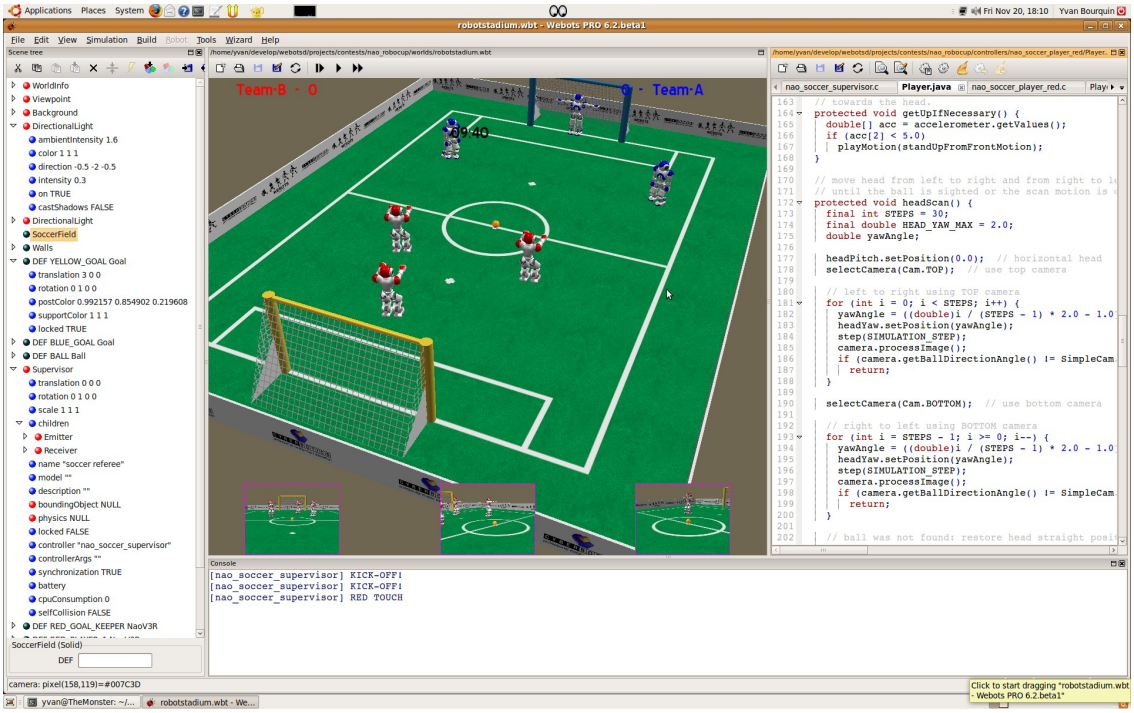


Figura 20 – GUI do ambiente de desenvolvimento e simulação do Webots

Fonte: CYBERBOTICS, 2017.

pode ser testado em um ambiente virtual fisicamente realista (CYBERBOTICS, 2017c). A figura 20 mostra a interface de desenvolvimento do Webots.

Devido aos inúmeros sensores e atuadores simulados presentes no Webots, é possível criar modelos de diversos robôs, sejam eles comerciais ou não, para a execução de teste de comportamento destes perante a diversos ambientes. Estes ambientes podem ser definidos pelo usuário, o que permite que estes sejam personalizados. Eles podem ser simples como um simples plano ou complexos como um campo de futebol contendo todas as marcações e as traves como nas figuras 21 e 22.

No Webots é possível realizar a simulação do comportamento de múltiplos robôs em paralelo, como por exemplo um time inteiro de futebol de robôs. Desta forma, é possível realizar teste sem a necessidade de robôs físicos, o que torna os experimentos mais baratos, rápidos e replicáveis. Todos esses testes podem ser utilizados como uma primeira etapa do desenvolvimento de um software ou hardware para um robô, o que faz com que ocorra uma diminuição na quantidade de ajustes necessários para que tanto software quanto hardware se adeque ao robô físico.

Na versão mais recente do Webots existe um modelo de robô simulado similar ao robô utilizado pela equipe RobôFEI. Este modelo será utilizando neste trabalho para realizar a simulação da caminhada e obter os dados necessários para o cálculo do valor de adaptação que será abordado futuramente neste documento. Desta forma, os parâmetros de caminhada ótimo encontrados serão valores bem aproximados dos valores ótimos para o robô físico.

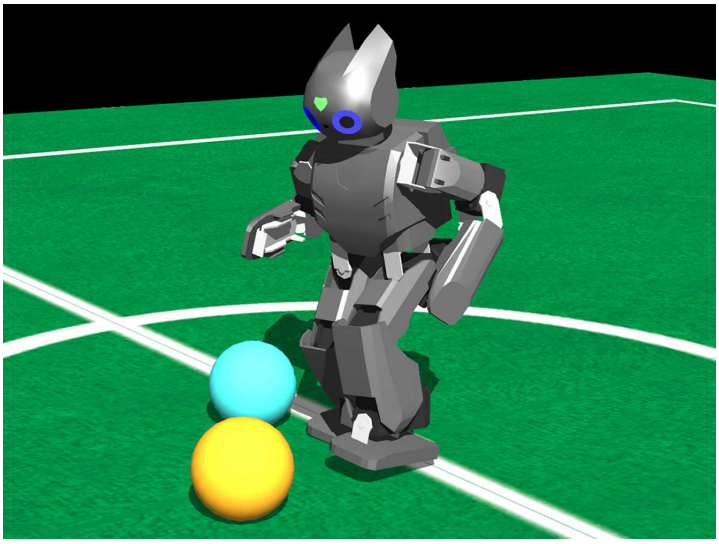


Figura 21 – Robô DARwIn-OP simulado no Webots

Fonte: CYBERBOTICS, 2017.

Figura 22 – Simulação de futebol de robôs utilizando dois times e mostrando o ponto de vista de alguns robôs

Fonte: CYBERBOTICS, 2017.

Esta versão do Webots já possui um ambiente que contém um campo de futebol com as traves, as marcações e paredes nas linhas laterais e de fundo do campo e possui uma bola. Neste ambiente será possível verificar o desempenho dos parâmetros de caminhada ótimos encontrados em uma situação similar a qual este robô será submetido.

O ambiente de programação do Webots suporta as linguagens de programação C, C++, Java, Python, porém existe uma API para integração com o MATLAB. Também é possível utilizar o framework ROS para o desenvolvimento. A instalação e utilização de cada uma destas APIs está descrita no manual do usuário (CYBERBOTICS, 2017b) presente no site oficial do Webots.

4.1.2 Arquitetura em Cruz

Foi necessário utilizar o software desenvolvido pela equipe RoboFEI do Centro Universitário da FEI para os robôs humanoides B1 e B2. Este software utiliza uma arquitetura híbrida chamada de *Arquitetura em Cruz* que é descrita pela figura 23. Esta é definida como hibrida pois os processos IMU e de controle seguem um paradigma reativo enquanto que os processos como visão, localização e decisão seguem paradigmas hierárquicos (PERICO et al., 2015).

Na figura 23 cada caixa representa um processo independente, visão, localização, decisão, comunicação, planejamento, sentidos (IMU), controle. Estes processos foram divididos em *clusters* devido ao seus custos computacionais e cada um destes *clusters* foi atribuído para um núcleo do processador utilizado que contem 4 núcleos. A divisão foi feita da seguinte forma:

a) Cluster 1: Visão (núcleo 0);

b) Cluster 2: Localização (núcleo 1);

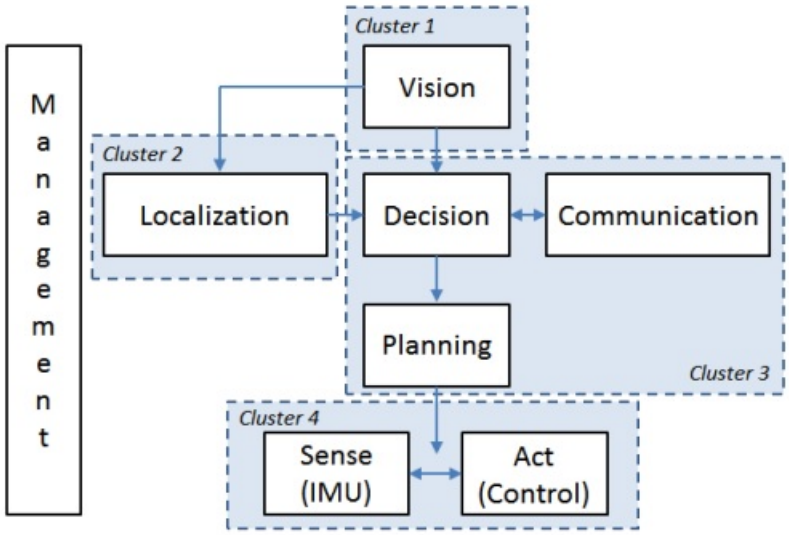


Figura 23 – Arquitetura em Cruz usada nos robôs B1 e B2

Fonte: Perico et al., 2015.

c) Cluster 3: Decisão, Comunicação e Planejamento (núcleo 2);

d) Cluster 4: Sentidos (IMU) e Controle (núcleo 3);

Embora independentes, os processos podem trocar informações através de uma memória compartilhada que atua como *Blackboard*. Desta forma, qualquer processo pode publicar as suas variáveis compartilhadas fazendo que qualquer um dos outros processos possam ler o valor desta variável.

4.2 HARDWARE

O computador no qual o Webots e por sua vez o algoritmo de *PSO* serão executados possui a seguinte especificação:

a) Processador Intel Core I7-4790 3.60 GHz

b) 16.0 GB de memória RAM

c) Sistema Operacional:

- Windows 10 Pro
- Ubuntu 16.04.2

d) Placa de vídeo Nvidia GTX1070 G1 Gigabyte

e) 120.0 GB SSD + 2.0 TB HDD

A otimização da caminhada será realizada utilizando o robô Darwin, porem devido à similaridade, os robôs humanoides B1 e B2 pertencentes ao Centro Universitário da FEI e utilizado em competições de futebol como a RoboCup KidSize League World Competition pela equipe RoboFEI poderão utilizar os resultados deste trabalho. A tabela 3 apresenta as especificações dos robôs B1 e B2 (similares as especificações do Darwin) das figuras 24 e 25.

Tabela 3 – Especificações dos robôs B1 e B2

Altura	520 mm
Peso	3.0 kg
Velocidade de Caminhada	70 cm/min
Graus de Liberdade	22: 6 por perna, 3 por braço, 2 na cabeça e 2 no quadril
Motores	Dynamixel RX-28
Sensores	UM6 Ultra-Miniature Orientation Sensor Câmera Logitech HD Pro Webcam C920
CPU	Intel NUC i5-4250U 8GB DDR3 SDRAM 120GB SDD

Figura 24 – Robôs do Centro Universitário da FEI

Fonte: Equipe RoboFEI.

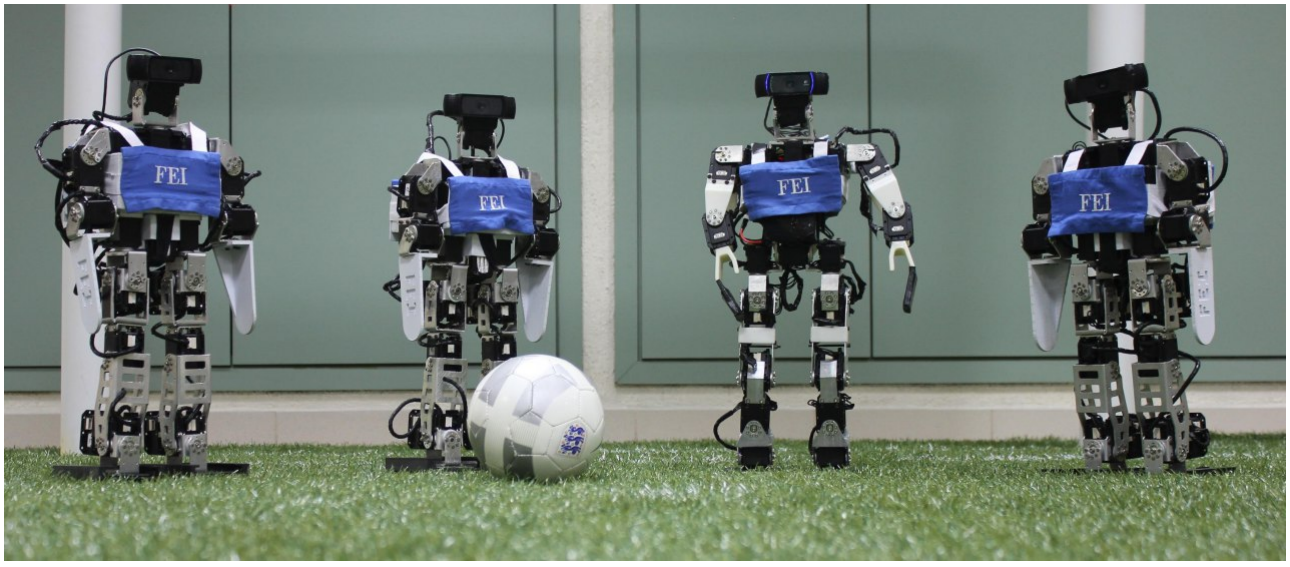


Figura 25 – Robôs do Centro Universitário da FEI com bola no padrão da RoboCup

Fonte: Equipe RoboFEI.

Os robôs do Centro Universitário da FEI foram desenvolvidos utilizando critérios de equilíbrio como o *Zero Moment Point* e *Center of Pressure* e os motores foram projetados considerando o movimento relativo entre eles, garantindo a mobilidade necessária para cada junta, evitando colisões e interferências pelo movimento simultâneo.

Os robôs foram projetados para concentrar todo o processamento em um único computador que se comunica com os motores e outros periféricos através da porta USB. Desta forma foi possível eliminar o micro controlador que era utilizado como intermédio entre o computador

e os motores. Atualmente os robôs utilizam um sensor de orientação, o *UM6 Ultra-Miniature Orientation Sensor* (LLC, s.d.), um sensor que possui giroscópio, acelerômetro, magnetômetro e a capacidade de se conectar com um modulo externo de GPS.

Entretanto como as suas características se são as mesmas do robô Darwin, a simulação do Darwin foi utilizada neste trabalho para realizar as otimizações. Isso permite uma execução mais rápida dos experimentos. A equipe RobôFEI observou que os dados obtidos utilizando a simulação do robô Darwin se assemelham muito dos resultados obtidos o robô, o que permite que os resultados obtidos neste trabalho possam ser utilizados nos robôs do Centro universitário da FEI.

4.3 ARQUITETURA

O algoritmo de *Particle Swarm Optimization* será desenvolvido e executado no ambiente de desenvolvimento do Webots, isto permite integrar a execução do algoritmo com a execução da simulação da caminhada do robô. A arquitetura do sistema proposto por este trabalho está ilustrada pela figura 26 e a figura 27 mostra a sequência de chamadas dos componentes deste sistema.

É possível alterar o programa para inicializar todas ou algumas das partículas da primeira geração com valores definidos com a finalidade de garantir que o algoritmo explore soluções próximas a estas soluções definidas inicialmente. Também é possível alterar o número máximo de iterações, o número de partículas e o número da iteração inicial.

O *Otimizador* alterará os valores dos parâmetros de caminhada do robô chamando o *Configurador* e passando os valores de cada um destes parâmetros. O *Configurador* irá efetuar a alteração no *Repositório de Parâmetros*, que neste caso é um arquivo **.ini**. Os parâmetros definidos no *Repositório de Parâmetros* serão utilizados pelo *Simulador* para executar a simulação da caminhada do robô.

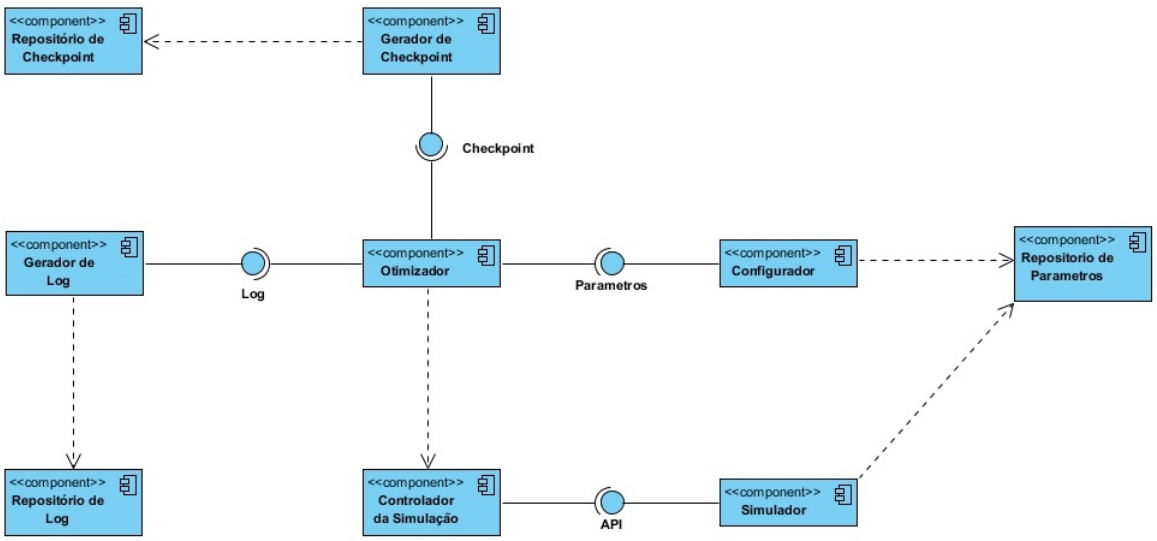


Figura 26 – Desenho da solução

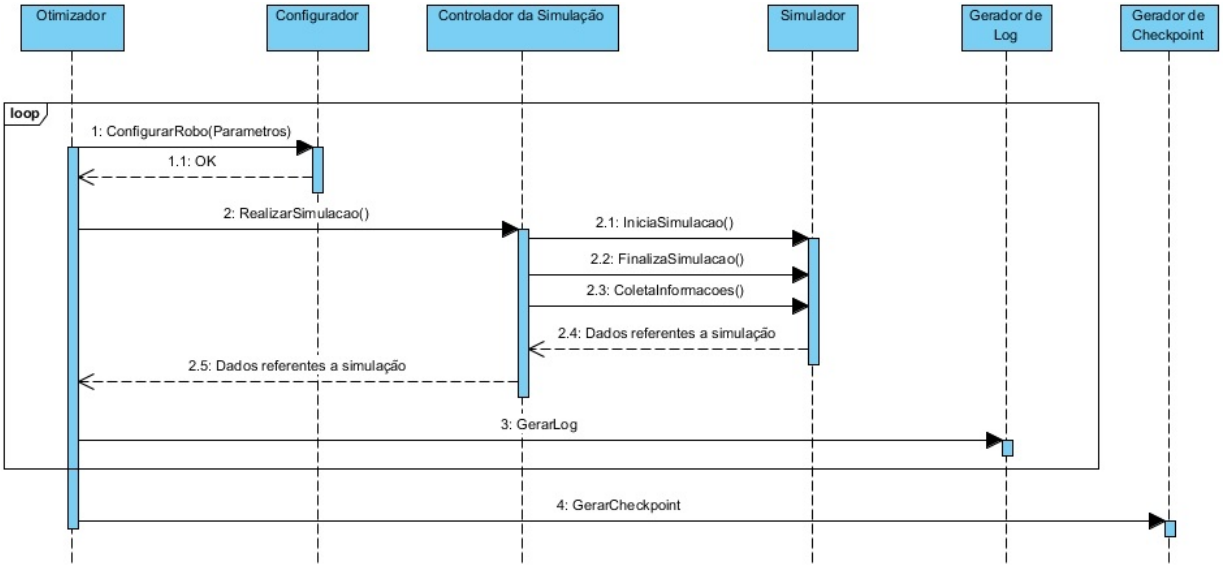


Figura 27 – Diagrama de sequencia da solução

O *Otimizador* chama o *Controlador da Simulação*, que será responsável por inicializar a execução da simulação e por coletar os dados necessários para o cálculo do valor de adaptação de cada uma das partículas. Isto permitirá que o algoritmo possa ser executado de forma automática, ou seja, dispensando a necessidade de intervenção humana no processo. Para controlar o *Simulador*, o *Controlador da Simulação* utilizará a API do Webots.

Ao finalizar a execução do algoritmo, os parâmetros definidos no *Repositório de Parâmetros* serão os parâmetros com o maior valor de adaptação encontrados pelo algoritmo ($gbest$).

Para fins de análise dos resultados, todas as partículas terão os seus valores das dimensões, de adaptação e do $pbest$ salvos em LOG durante toda a execução do *Otimizador*, assim como o valor do $gbest$. Desta forma, será possível observar o comportamento do algoritmo durante toda a execução e a existência de conjuntos de parâmetros com valores diferentes que resultam em uma solução (valor de adaptação) similar. Também será salvo em LOG a duração da execução do *Otimizador* e de cada uma das simulações feitas. Isto permitirá que seja estimado a duração da execução com um número diferente de parâmetros e permitirá a identificação de erros durante a execução de uma simulação.

Para gerar o LOG, o *Otimizador* irá chamar o *Gerador de Log*, que irá receber os dados a serem salvos e criará um arquivo de LOG com o formato CSV, ou seja, cada dado separado por uma virgula. Este arquivo poderá ser importado em outros programas para a realização do tratamento dos dados.

Serão salvos, no *Repositório de Checkpoint*, os valores das dimensões, de adaptação e do $pbest$ de todas as partículas, iniciais e da última iteração, os valores das dimensões, de adaptação do $gbest$ e o número da última iteração realizada com sucesso. Desta forma, será possível utilizar este arquivo para retomar a execução caso ocorra alguma interrupção externa, como por exemplo, falta de energia ou uma falha no computador.

Para a criação destes **checkpoints**, o *Otimizador* envia para o *Gerador de Checkpoint* os dados que devem ser inseridos no **checkpoint**. O *Gerador de Checkpoint* cria um arquivo de texto em que cada dado ocupa uma linha do arquivo, que será o **checkpoint**, e o salva no *Repositório de Checkpoint*.

4.4 ALGORITMO DE PARTICLE SWARM OPTIMIZATION

Após testar diversas formas do algoritmo de *PSO*, J. Kennedy e Eberhart (2011) observaram que a versão simples do PSO obteve resultados satisfatórios, portanto, esta versão será utilizada neste trabalho para a realização da otimização dos parâmetros do oscilador de caminhada do robô humanoide Darwin, que se assemelha os robôs utilizados pelo Centro Universitário da FEI.

Para evitar definir uma velocidade máxima para as partículas, será utilizada a abordagem de constrição mostrada no trabalho do Bratton e J. Kennedy (2007). Desta forma o algoritmo 3 se assemelha ao algoritmo 2 porém $c_1 = c_2 = 2.05$, o número de partículas igual a 25 e 100, vizinhança definida como *full* e dimensão igual ao número de parâmetros que serão otimizados.

Algoritmo 3 – Algoritmo do *PSO* utilizado.

Dados: vetor X contendo p partículas de dimensão D,
$D = parametrosOtimizados$, $p = 25$
início
  Iniciar todas as partículas $\in X$ em posições aleatórias
fim
enquanto *Convergência ou limite de iterações não atingidos* **faça**
  para cada *partícula* $x_i \in X$ **faça**
    Avaliar o valor da função a qual se deseja maximizar
    se $valordafunçãoatual < pbest_{va}$ **então**
      $pbest_{va} = valordafunçãoatual$
    fim
    se $valordafunçãoatual < gbest_{va}$ **então**
      $gbest_{va} = valordafunçãoatual$
    fim
    para cada D **faça**
      Mudar velocidade usando a fórmula: $v_{D_{i+1}} = 0.72984 * (v_{D_i} + 2.05 * rand() * (pbest_D - present_D) + 2.05 * rand() * (gbest - present_D))$
    fim
    para cada D **faça**
      Mover a partícula x_i usando a fórmula: $new_D = present_D + v_{D_{i+1}}$
    fim
  fim
fim
retorna *Posição nas* D *dimensões do* $gbest$

4.5 CÁLCULO DO VALOR DE ADAPTAÇÃO

Para que o algoritmo de *PSO* possa fazer os cálculos dos valores de adaptação ele irá alterar os valores do arquivo *.ini* de configuração dos osciladores responsáveis pelo caminhar do robô e em seguida executará a simulação da caminhada.

O arquivo *.ini* contém os valores de todos os parâmetros necessários para que o sistema de controle do robô execute ações como chutar a bola e andar. Este arquivo é lido no início da simulação, se tornando um valor estático durante esta.

Após a simulação, os valores de distância, velocidade e número de quedas será retornado para o algoritmo de PSO que calculará o valor de adaptação desta partícula. Este ciclo pode ser observado na figura 27 e será repetido para todas as partículas, até que o algoritmo convirja ou atinja o número de iterações limite.

Este trabalho propõe uma fórmula para o cálculo do valor de adaptação que leva em consideração não somente o evento da queda, mas como também a quantidade de quedas, fazendo assim com que partículas que causem uma única queda tenham um valor de adaptação maior que uma partícula que cause diversas quedas. Entretanto, o valor da punição não é muito alto nas primeiras quedas porem o seu valor cresce exponencialmente, isso se deve a componente $q \times \sqrt[2]{q}$ da formula 15, que foi adotada com a finalidade de não descartar soluções que gerem uma velocidade elevada porem fazem com que o robô caia uma única vez. O valor da Adaptação será atribuído pela fórmula 15.

$$va = d - (P \times q \times \sqrt[2]{q}), \tag{15}$$

onde, va é o Valor de Adaptação, d é a distância percorrida em metros, P é uma constante do peso que a queda tem para o cálculo do Valor de Adaptação (neste trabalho o valor será igual a 1), q é a quantidade de quedas sofridas pelo robô.

O valor de adaptação (va) será utilizado no algoritmo do PSO para definir a qualidade da solução representada por uma partícula, será utilizada para definir a melhor solução encontrada por cada partícula ($pbest$) e será usada para definir a melhor solução encontrada por todo enxame ($gbest$) que ao final da execução do PSO será a resposta do algoritmo

Este capítulo apresentou a proposta deste trabalho descrevendo o algoritmo do PSO que será implementado no simulador, o programa responsável por realizar os experimentos, o simulador Webots (utilizado neste trabalho) e o hardware e software dos robôs humanoide do Centro Universitário da FEI. O capítulo seguinte abordará a metodologia utilizada nos experimentos e os resultados obtidos utilizando o simulador Webots.

5 EXPERIMENTOS

Este capítulo aborda os experimentos realizados neste trabalho e está dividido em duas seções. A primeira seção descreve a metodologia utilizada durante a execução dos experimentos que compõem este trabalho e os objetivos de cada um destes experimentos. A segunda seção irá abordar cada uma das simulações realizadas e mostrará os resultados e informações coletadas de cada uma destas simulações.

5.1 METODOLOGIA

Os experimentos foram realizados utilizando o simulador de robótica Webots (seção 4.1.1), o que permitiu que o experimento não necessitasse de intervenção humana. A figura 28 mostra o robô durante um dos experimentos.

O PSO utilizado foi o descrito no capítulo 4, onde cada uma das partículas do PSO contém os 19 parâmetros. Afim de entender a influência da configuração do PSO nos resultados, alguns dos experimentos realizados tiveram alterações nas configurações do PSO, como por exemplo a quantidade de partículas, tempo de execução da simulação e número máximo de iterações.

O programa inicia fazendo a leitura do arquivo temporário de checkpoint. A primeira linha deste arquivo se refere ao número de iterações já executadas. Caso este número seja igual a 0, o programa inicia as partículas aleatoriamente no espaço de possibilidades. Caso este número

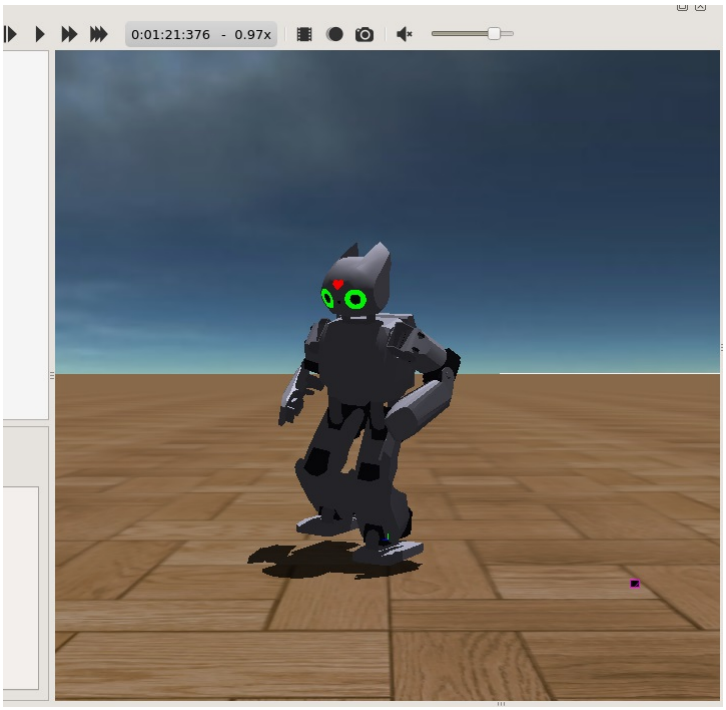


Figura 28 – Robô caminhando durante experimento

Fonte: autor.

seja diferente de 0, o programa faz a leitura deste arquivo de checkpoint e recupera informações como posição, valor de adaptação, $pbest$ (valor e posição) de cada partícula, posição e valor do $gbest$, caminho do arquivo de log.

Tal mecanismo de inicialização se mostrou importante pois para evitar erros durante a simulação após cada uma das iterações a simulação foi reiniciada, de forma que para retomar a execução anterior foi necessário implementar tal mecanismo de checkpoint e inicialização.

No início de cada uma das interações as partículas têm o seu valor de adaptação calculado, para tal, o valor atual dos parâmetros é alterado, o robô é reposicionado para a posição inicial e o programa responsável por executar a caminhada é reiniciado para que os novos valores dos parâmetros sejam utilizados. O robô permanece andando para frente durante um tempo predeterminado. Após estes tempo a distância percorrida é calculada, o número de quedas sofridas pelo robô é contabilizado e a fórmula 15 é utilizada para calcular o valor de adaptação.

Em seguida cada partícula verifica se o valor atual possui um valor de adaptação maior que o seu $pbest$. Se o valor atual for maior que o valor do $pbest$, o $pbest$ é atualizado. Caso contrário nada é feito. Após cada partícula verificar seu $pbest$ o programa verifica se o $gbest$ deve ser atualizado comparando o valor do $gbest$ e o valor atual das partículas.

Por fim, as novas posições das partículas são calculadas utilizando as equações de velocidade e posição presentes no algoritmo 3. O arquivo de checkpoint é zerado e escrito novamente com as informações desta última iteração executada e é adicionado no arquivo de log uma nova linha posições e valores de adaptação de cada partícula, assim como a posição e valor de adaptação do $gbest$.

O número da iteração é comparado com o máximo de iterações. Se o número da iteração for menor a simulação é reiniciada junto com o programa que executa a caminhada do robô e o programa descrito. Caso o número da iteração seja igual ou maior, o experimento é encerrado e o valor atual do $gbest$ é definido como a resposta da otimização.

Utilizando o arquivo de log gerado pelo programa é possível observar a posição e valor de adaptação de cada uma das partículas durante toda execução do algoritmo e a evolução do $gbest$.

Este trabalho foi dividido em 6 experimentos. Estes experimentos são caracterizadas pelo número de parâmetros que estão sendo otimizados, variando de 1 parâmetro até os 19, e pelo tamanho do passo, que nos dois últimos experimentos foi definido com um valor 1,5 vezes maior do que o dos anteriores. Estes experimentos foram:

a) Simulação com 1 Parâmetro;

b) Simulação com 3 Parâmetros;

c) Simulação com 6 Parâmetros;

d) Simulação com 19 Parâmetros;

e) Simulação com 6 Parâmetros e com passo 1.5 vezes maior;

f) Simulação com 19 Parâmetros e com passo 1.5 vezes maior;

Os experimentos foram realizados nesta ordem com a finalidade de obter mais informações sobre o comportamento do PSO na otimização dos parâmetros da caminhada do robô humanoide, como por exemplo influência das configurações do PSO (número de partículas, número de iterações máxima e tempo máximo da execução da simulação responsável pelo valor de adaptação). Cada experimento também possuía como objetivo a análise da viabilidade do aumento da complexidade, ou seja, do aumento do número de parâmetros otimizados.

O simulador Webots possui um conjunto de exemplo dos parâmetros da caminhada do robô humanoide Darwin que é capaz de gerar uma caminhada estável. Com a finalidade de garantir que as alterações no comportamento do robô são causadas pela alteração no valor dos parâmetros, este conjunto de exemplo foi utilizado como os valores nominais de cada parâmetro, ou seja, durante os experimentos somente os parâmetros sendo otimizados sofreram alteração. Já os parâmetros que não estavam sendo otimizados foram mantidos no valor nominal durante o experimento.

Para a realização dos experimentos foi identificado a necessidade de se limitar os valores possíveis para o posicionamento aleatório das partículas durante a primeira iteração do PSO. Esta medida foi adotada para diminuir o espaço de respostas inicial, impedindo assim que as partículas ficassem muito distantes umas das outras.

Também foi possível notar a necessidade de impedir que as partículas pudessem atingir valores negativos para alguns parâmetros, portanto foi colocado um limitador no cálculo da nova posição das partículas que faz com que as posições negativas de determinados parâmetros fossem considerados igual a posição 0 destes parâmetros. Um exemplo de parâmetro ao qual foi introduzida esta limitação foi o $period\_time$, pois com este parâmetro negativo o robô caminha para o sentido oposto ao desejado, no caso, o robô passaria a andar de costas na direção oposta a desejada.

Em espaço de respostas maiores, foi necessário adicionar um limite ao ângulo final do robô em relação ao ângulo inicial com a finalidade de garantir uma caminhada em linha reta. Esta medida foi adotada pois ao realizar o comando para o robô andar em linha reta, este deve realizar esse movimento o melhor possível, para que os erros durante a movimentação sejam os menores possíveis.

Desta forma, foi imposto um limite ao ângulo final do robô em relação ao ângulo inicial que se ultrapassado resultaria na atribuição do valor 0 para a distância percorrida no momento de realizar o cálculo da função de adaptação. Desta forma a penalidade devido as quedas ainda seria atribuída ao resultado final, fazendo com que parâmetros que realizassem uma caminhada que ultrapassassem o limite porem não resultassem em uma queda do robô fossem considerados melhores que o conjunto de parâmetros que cause uma queda.

Os ângulos limites escolhidos foram -45^{o} e 45^{o}, ou seja, uma simulação de caminhada que tivesse o seu ângulo final, em relação ao ângulo inicial, entre -45^{o} e 45^{o} era considerada como uma caminhada em linha reta e teria o seu valor de adaptação calculado normalmente.

Como os valores dos parâmetros são iniciados e gerados de forma aleatória, alguns conjuntos de parâmetros faziam com que um ou alguns motores do robô precisassem ir para uma posição que não seria possível devido as juntas do robô. Na simulação este erro faz com que a movimentação do robô não seja natural, podendo ele ser capaz de andar mesmo deitado. No robô real, este erro poderia ser capaz de danificar as juntas ou os motores. Para mitigar esta situação foi implementado um verificador que atribui um valor de adaptação igual a -99 para todo conjunto de parâmetros que resultem em pelo menos um erro de junta. Isso faz com que o algoritmo evite a qualquer custo se dirigir para áreas do espaço de soluções que resultem neste comportamento.

O cálculo de física do simulador pode conter imprecisões em seus cálculos e em uma simulação de longa duração resultar em erros. Estes erros são identificados pelo simulador que resulta em um aviso, porem a simulação não é parada ou reiniciada. Para impedir este comportamento, foi implementado um sistema que realizava a reinicialização da simulação ao final de cada iteração e utilizava o conceito de checkpoint, para retomar os valores anteriores a reinicialização permitindo que uma nova iteração fosse realizada.

Esta técnica se mostrou eficiente, porém não era capaz de garantir a integridade do cálculo da física para configurações que resultavam em intervalos muito grandes entre o reinicio da simulação, como o caso da configuração utilizando 100 partículas e 30 segundos de execução da simulação. Este erro no cálculo da física pode resultar em dados incoerentes, como por exemplo a queda do robô independente dos parâmetros utilizados.

5.2 RESULTADOS

Neste trabalho foi utilizado o simulador de robótica Webots para realizar simulações utilizando diversas configurações para o PSO. Estas configurações utilizadas são variações das seguintes configurações:1, 3, 6 e 19 parâmetros, 25, 50 e 100 partículas 15, 30 e 45 segundos para o tempo de execução da simulação. Esta seção será dividida em subseções que irão abordar os resultados e informações coletadas em cada um dos experimentos.

5.2.1 Simulação com 1 Parâmetro

Neste experimento o parâmetro $period\_time$ foi o alvo da otimização, portanto ele era o único parâmetro que teve seu valor alterado e era a única dimensão das partículas. Cada partícula do PSO teve seu valor inicial do parâmetro $period\_time$ definido de forma aleatória com valores entre 0 e 1000. Os demais parâmetros foram mantidos nos valores definidos pelo simulador. Para permitir a comparação com o trabalho do Silva (2015) o parâmetro $swing\_right\_left$ foi mantido em 20.

Devido a influencia na velocidade da caminhada do parâmetro $period\_time$, ele foi escolhido para ser otimizado neste experimento. Essa influência se deve ao fato de que este

parâmetro corresponder ao T na formula 7 do cálculo dos osciladores acoplados, ou seja ele corresponde ao tempo total do passo.

Um valor de $period\_time$ muito pequeno pode fazer com que o passo seja muito rápido e a distância percorrida durante este passo não seja adequada. Já um valor grande para este parâmetro pode fazer com que o passo demore de mais causando instabilidade na caminhada do robô. Portanto este experimento visa encontrar um valor para $period\_time$ que seja pequeno a ponto de gerar um passo rápido, aumentando a estabilidade e a velocidade da caminhada, porém não tão pequeno a ponto de fazer com que a distância percorrida a cada passo seja muito pequena.

No total foram feitas 6 experimentações utilizando o processo descrito sendo que 4 delas utilizaram 25 partículas e 2 delas utilizaram 100. Os resultados apresentados pelas execuções contendo 25 partículas se mostraram similares.

O gráfico da figura 29 mostra a evolução do $gbest$ ao longo das iterações para as configurações utilizando 25 e 100 partículas. Ambos os dados são provenientes de 4 experimentos. Neste gráfico, é possível observar que utilizando 100 partículas, o algoritmo foi capaz de encontrar uma região no espaço de soluções que possui valor de adaptação mais elevado do que a região encontrada utilizando 25 partículas. A partir destes gráficos conclui-se que um enxame maior torna o algoritmo mais resiliente a máximos locais, entretanto, um enxame maior também faz com que o tempo de execução do algoritmo seja mais elevado como pode ser observado nas figuras 30 e 31.

Na figura 31. É possível observar uma rápida alteração no posicionamento do valor médio do parâmetro $period\_time$ das partículas assim como o desvio médio. Isso indica que uma das partículas que consistiu o enxame foi capaz de encontrar um valor de $period\_time$ que resultasse em um valor de adaptação melhor. Isso fez com que as demais partículas do enxame passassem a oscilar ao redor do novo valor de $period\_time$.

O resultado obtido na otimização utilizando 100 partículas foi o valor 599 para o parâmetro $period\_time$ contendo um valor de adaptação de 1.645, enquanto que o resultado da otimização utilizando 25 partículas foi 575 com valor de adaptação de 1.364. Definindo o ponto utilizando ambos os valores do $period\_time$ e $swing\_right\_left$ igual a 20, é possível observar que ambos os pontos estão contidos na área ótima identificada pelo Silva (2015) e mostrada na figura 32.

Como foi possível replicar o resultado obtido pelo trabalho utilizando Aprendizado por reforço (SILVA, 2015), é possível afirmar que o PSO é capaz de realizar a otimização dos parâmetros de caminhada do robô humanoide.

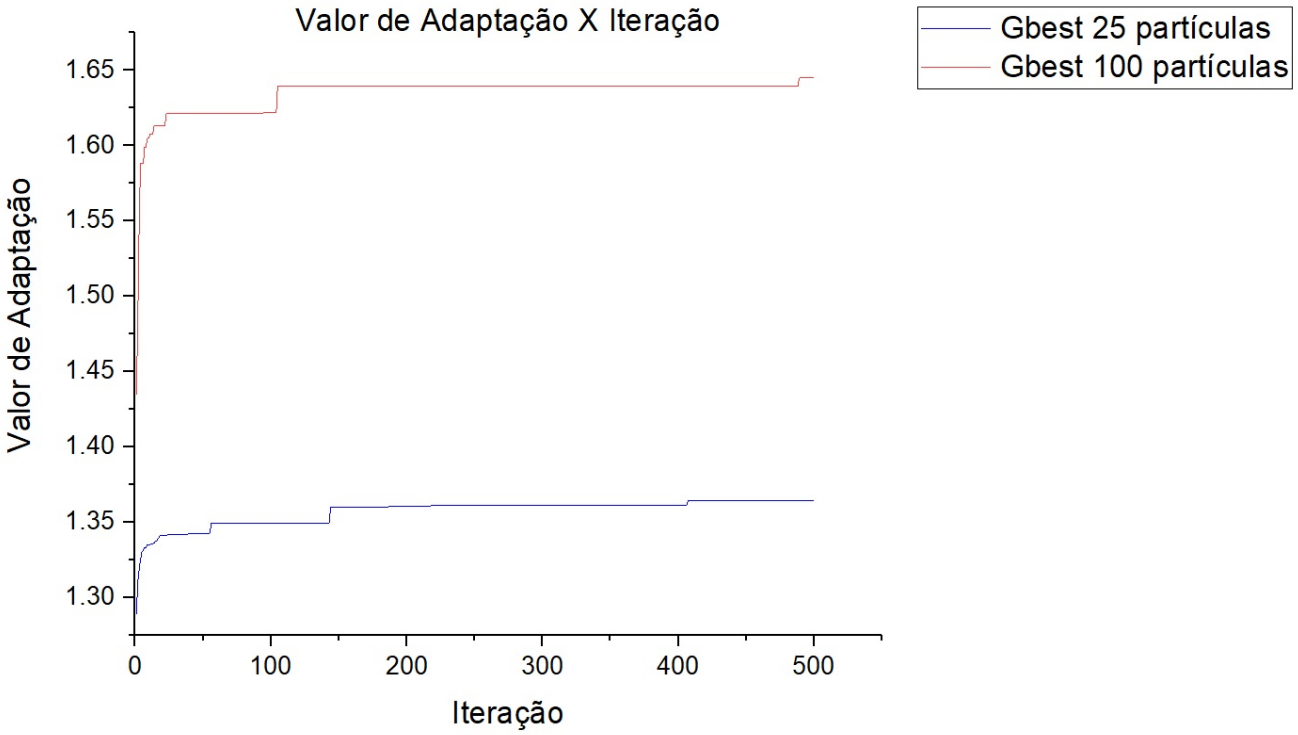


Figura 29 – Valor de adaptação do $gbest$ utilizando 25 e 100 partículas

Fonte: autor.

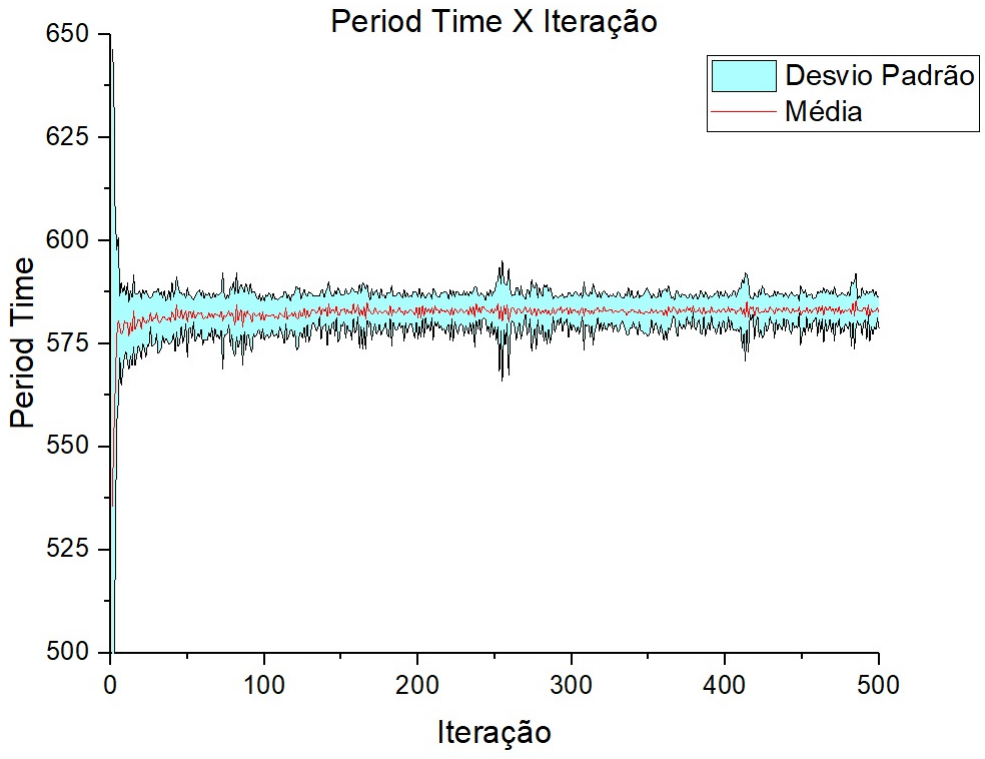


Figura 30 – Média e desvio padrão do parâmetro Period Time utilizando 25 partículas

Fonte: autor.

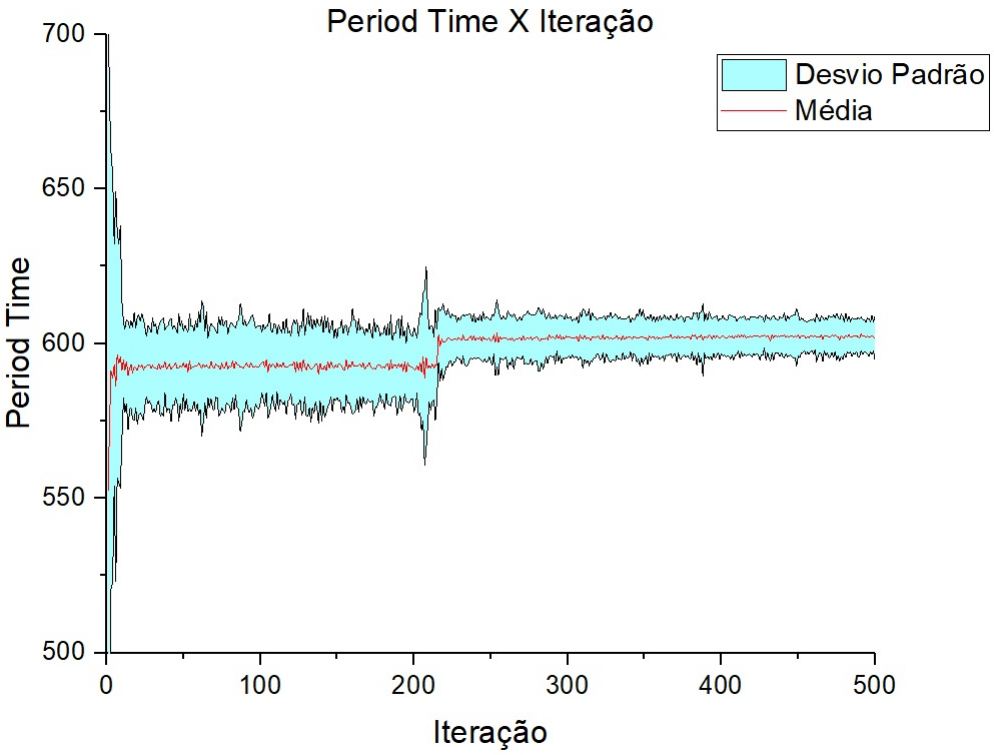


Figura 31 – Média e desvio padrão do parâmetro Period Time utilizando 100 partículas

Fonte: autor.

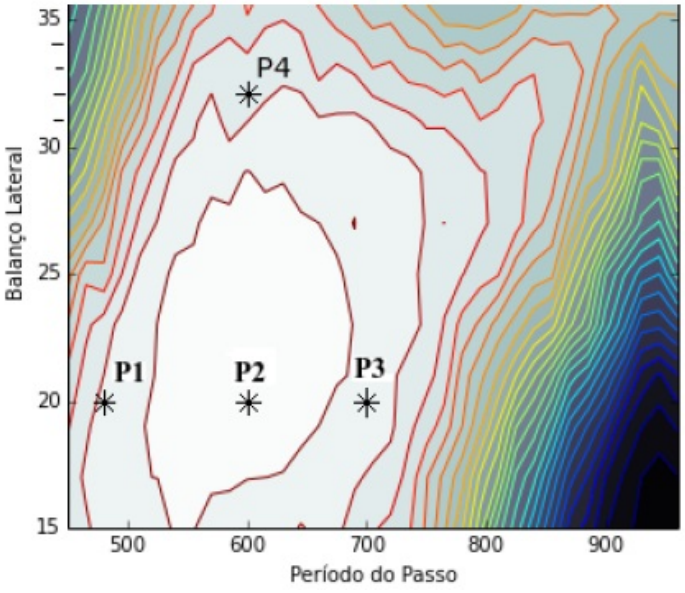


Figura 32 – Gráfico dos pontos encontrados pelo algoritmo de aprendizado por reforço

Fonte: Silva, 2015.

5.2.2 Simulação com 3 Parâmetros

Neste experimento foram realizadas simulações utilizando 3 parâmetros e tinham como objetivo observar a capacidade de otimizar o caminhar do robô do algoritmo PSO e verificar se o número de partículas e tempo de execução podem influenciar o resultado.

Além disso estes resultados foram comparados com os resultados obtidos pelo Silva (2015) utilizando a técnica de aprendizado por reforço. Para tal, foram utilizados os mesmos parâmetros escolhidos no trabalho do Silva (2015), ou seja, o $period\_time$, $foot\_height$ e $swing\_right\_left$.

Os dois parâmetros adicionais ao experimento anterior, $foot\_height$ e $swing\_right\_left$, foram escolhidos pois o $swing\_right\_left$ é o parâmetro responsável por fazer com que o centro de massa, ou o quadril, balance entre a direita e a esquerda, eixo y, com a finalidade de fazer com que o robô humanoide mantenha o equilíbrio. Este parâmetro corresponde ao ρ_{bal} na formula 6 do oscilador de balanço. Portanto este parâmetro é importante para compensar o desequilíbrio causado pela retirada de um dos pés do chão e pelo posicionamento deste pé a frente do corpo do robô, movimentando o centro de massa. Dessa forma valores pequenos podem não ser suficientes para realizar essa compensação e valores muito elevados podem contribuir para a instabilidade da caminhada.

Já o parâmetro $foot\_height$ é responsável por definir a altura máxima que o robô humanoide deve erguer o pé durante o passo. Este parâmetro corresponde ao ρ_{move} no eixo z na formula 6. Portanto este parâmetro é importante para o equilíbrio do robô durante a caminhada pois quanto mais se eleva o pé mais instável a fase de apoio único fica. Entretanto levantar pouco o pé pode dificultar a locomoção em diferentes tipos de terreno. Por exemplo, um plano liso não requer que o robô eleve muito o pé durante o passo, entretanto um campo de grama sintética necessita que o pé seja elevado para diminuir o atrito entre o pé e a grama durante a movimentação do pé na fase de apoio único.

Neste experimento foram utilizadas as seguintes configurações para as simulações:

a) 25 partículas, 15 segundos de execução e 500 iterações;

b) 25 partículas, 45 segundos de execução e 500 iterações;

c) 100 partículas, 15 segundos de execução e 500 iterações;

d) 50 partículas, 30 segundos de execução e 500 iterações;

As figuras 33, 34, 35, 36 mostram o gráfico do valor de adaptação médio (va) versus iteração de duas amostras. Estas figuras são referentes a diversas configurações diferentes do PSO.

Nas figuras 33, 34, 35, 36 é possível observar que aumentar tanto a quantidade de partículas quanto o tempo de execução fez com que o algoritmo fosse capaz de encontrar uma

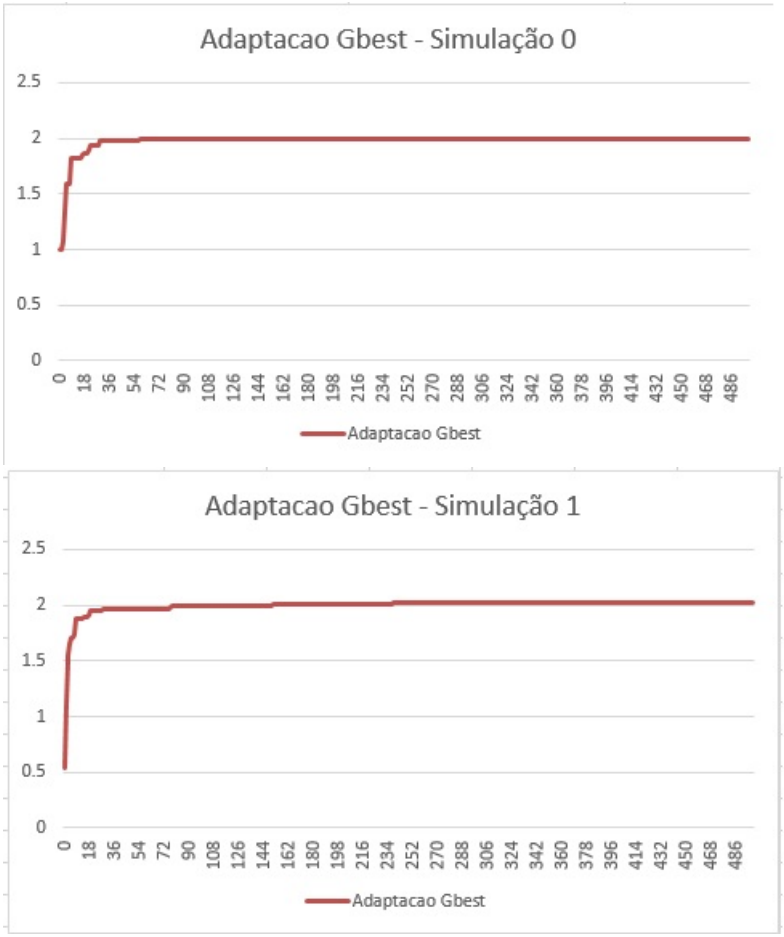


Figura 33 – Valor de adaptação do $gbest$ durante a otimização de 3 parâmetros, utilizando 25 partículas e 15 segundos de execução da simulação

Fonte: autor.

combinação de parâmetros com valor de adaptação ligeiramente maior, entretanto ambos aumentaram o número de iterações necessárias para atingir o valor otimizado de adaptação.

Mesmo ao realizar um aumento menos significativo no número de partículas e no tempo de execução, ambos em uma mesma simulação, foi possível encontrar valores ligeiramente melhores que a simulação de controle, porem valores muito próximos das simulações descritas anteriormente. Entretanto, ainda ocorreu um aumento na quantidade de iterações necessárias para se obter um resultado ligeiramente menor que nas simulações anteriores.

Desta forma é possível concluir que a alteração do número de partículas e do tempo de duração da simulação não está ligado diretamente a qualidade do resultado obtido pelo PSO, pois mesmo dobrando e triplicando estas configurações, os resultados tenham sido maiores do que a simulação de controle, estes resultados foram pouco diferentes entre si. Já a quantidade de iterações necessárias para se obter o resultado sofreu uma influência significativa a alteração das configurações.

As figuras 37 e 38 mostram a trajetória percorrida pelo robô durante 30 segundos e 10 minutos, respectivamente, obtidas realizando simulações com conjuntos de parâmetros do iní-

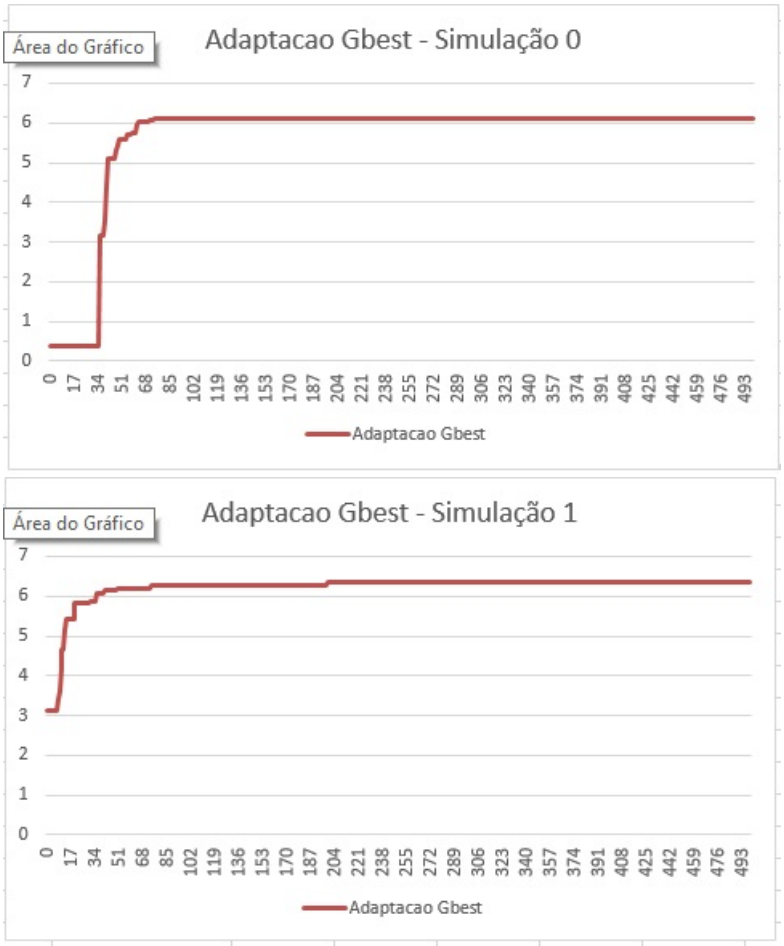


Figura 34 – Valor de adaptação do $gbest$ durante a otimização de 3 parâmetros, utilizando 25 partículas e 45 segundos de execução da simulação

Fonte: autor.

cio, meio e final da execução do PSO e com o conjunto de parâmetros definidos pelo simulador Webots considerado neste trabalho como o valor nominal. A figura 39 destaca a diferença entre a trajetória do conjunto nominal e o conjunto encontrado pelo algoritmo de PSO.

Nas figuras 37, 38 e 39 as foi utilizada a seguinte nomenclatura:

a) Trajetória LOW: Indica a trajetória do conjunto de parâmetros coletados no início da execução do algoritmo de PSO;

b) Trajetória MID: Indica a trajetória do conjunto de parâmetros coletados no meio da execução do algoritmo de PSO;

c) Trajetória HIGH: Indica a trajetória do conjunto de parâmetros coletados ao termino da execução do algoritmo de PSO, sendo este conjunto a resposta do PSO;

d) Trajetória Nominal: Indica a trajetória do conjunto de parâmetros definidos pelo simulador Webots;

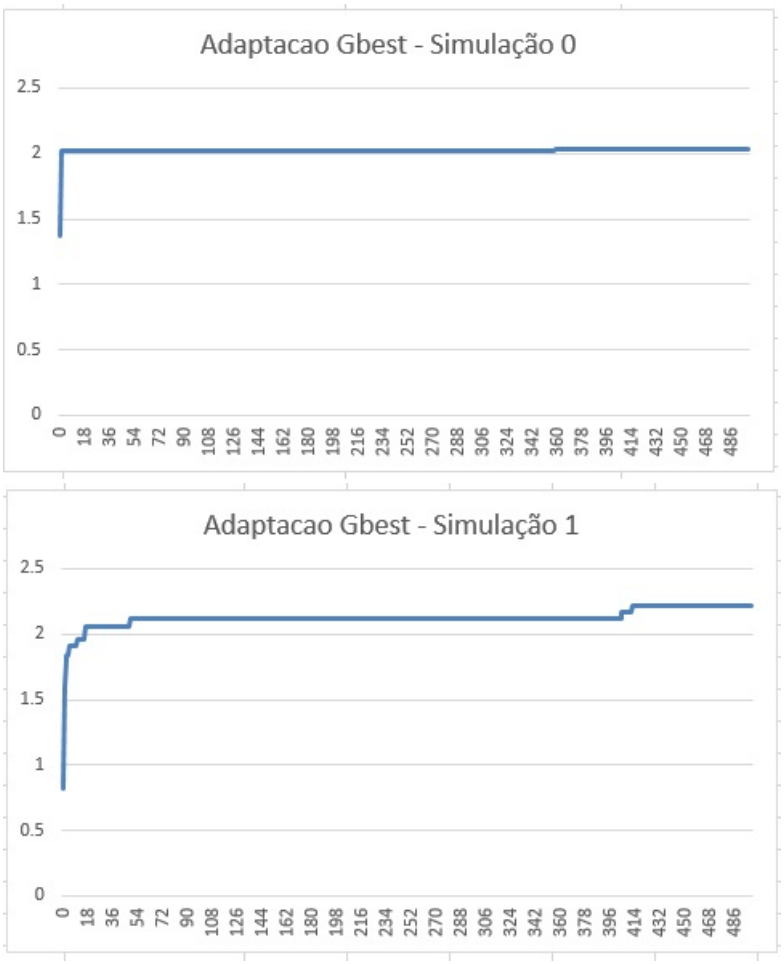


Figura 35 – Valor de adaptação do $gbest$ durante a otimização de 3 parâmetros, utilizando 100 partículas e 15 segundos de execução da simulação

Fonte: autor.

Para obter esses valores foram realizadas quatro simulações utilizando conjuntos de parâmetros do início, meio e final da execução do PSO e o conjunto de parâmetros nominal. Nestas simulações, a cada 1 segundo a posição atual do robô foi anotada, resultando nos quatro caminhos apresentados nas figuras 37, 38 e 39.

Na figura 37 é possível observar a evolução da qualidade e velocidade da caminhada gerada pelo PSO. Na trajetória LOW o robô é capaz de se deslocar uma distância muito curta, enquanto que na trajetória MID o robô foi capaz de se deslocar por uma distância maior porem por causa da instabilidade a trajetória resultante não é retilínea. A trajetória HIGH é a trajetória mais estável e rápida encontrada pelo PSO. Também é possível notar que o conjunto de parâmetros encontrados pelo PSO gera uma caminhada próxima e mais rápida do conjunto definido pelo simulador (nominal). Com velocidades de $12{,}85\ cm/s$ para o conjunto do PSO e $9{,}49$ cm/s para o conjunto de parâmetros nominal.

Como os parâmetros otimizados neste experimento foram os mesmos utilizados pelo Silva (2015) é possível realizar uma comparação direta com os resultados obtidos por ele utilizando a técnica de aprendizado por reforço.

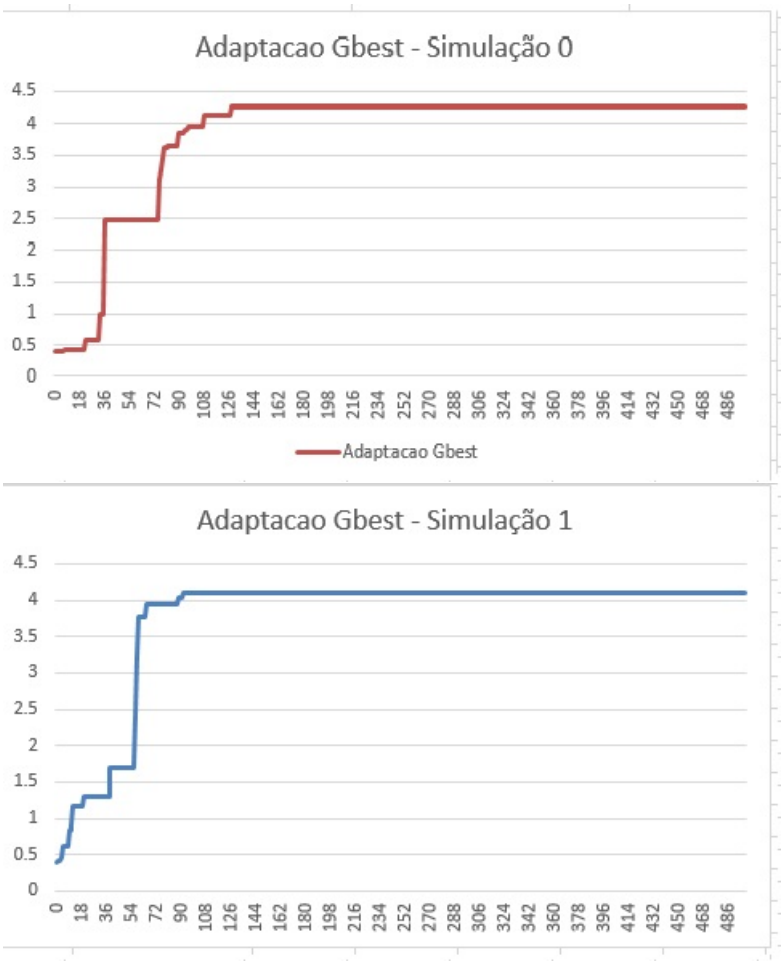


Figura 36 – Valor de adaptação do $gbest$ durante a otimização de 3 parâmetros, utilizando 50 partículas e 30 segundos de execução da simulação

Fonte: autor.

A figura 40 apresenta o caminho realizado pelo robô humanoide utilizando os parâmetros encontrados pelo algoritmo de aprendizado por reforço proposto no trabalho do Silva (2015). Ao comparar ambos os resultados, foi possível observar que embora o PSO tenha sido capaz de encontrar um conjunto de parâmetros capaz de gerar uma caminhada um pouco mais retilínea, o algoritmo de aprendizado por reforço foi capaz de encontrar conjuntos de parâmetros que geram uma velocidade maior do robô, com velocidades de até $28{,}79\ cm/s$.

Além disso é possível observar que utilizando o aprendizado por reforço o aumento na velocidade e estabilidade da caminhada é gradativa, enquanto que utilizando o PSO este aumento pode ser mais abrupto, como pode ser observado na figura 36. Isso acontece devido ao fato do PSO não possuir um passo definido e possuir um elemento aleatório para alterar a posição das partículas. Por causa disso, elas podem se movimentar mais livremente pelo espaço de soluções facilitando a exploração.

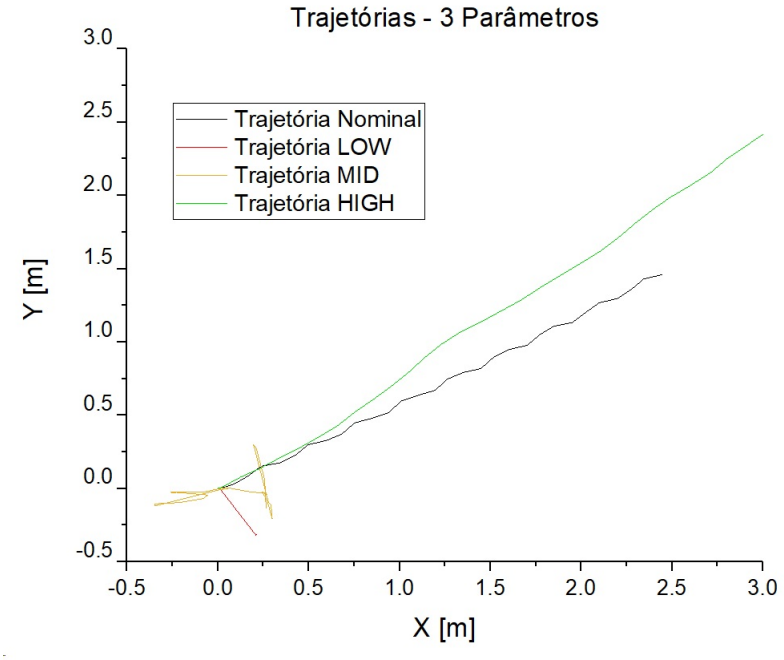


Figura 37 – Trajetórias de 30 segundos utilizando conjunto de parâmetros nominal e coletados da execução do PSO utilizando 3 parâmetros

Fonte: autor.

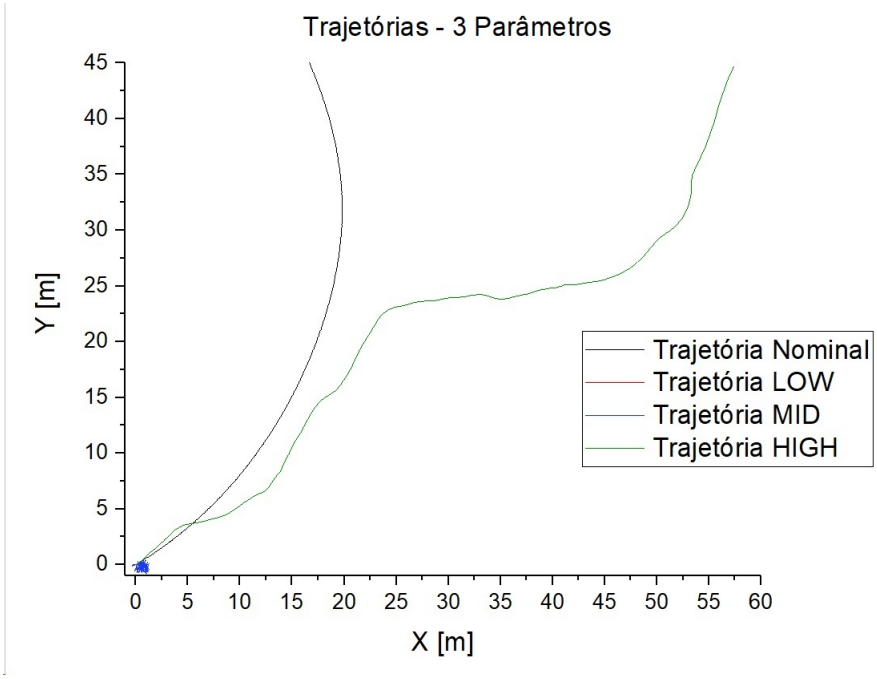


Figura 38 – Trajetórias de 10 minutos utilizando conjunto de parâmetros nominal e coletados da execução do PSO utilizando 3 parâmetros

Fonte: autor.

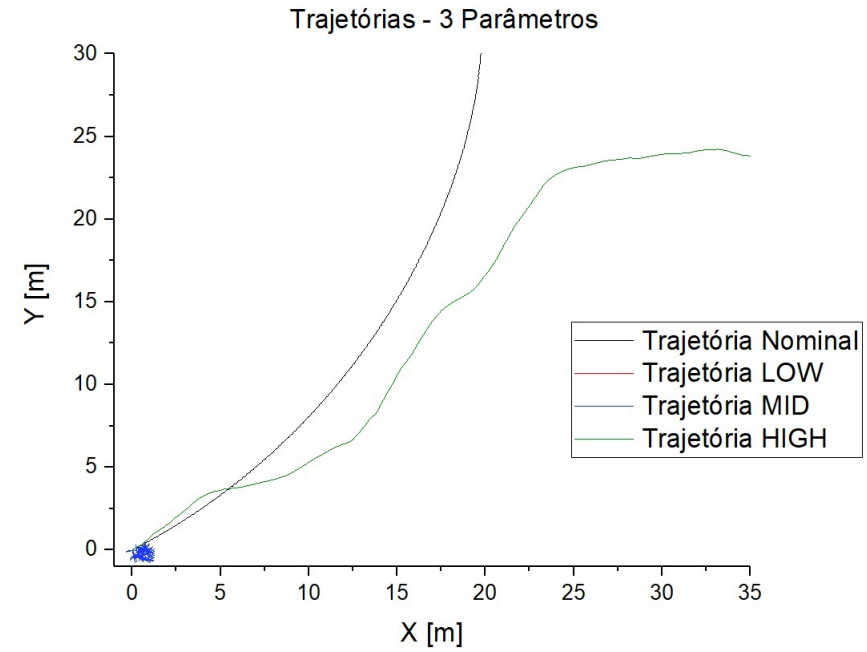


Figura 39 – Vista aproximada da figura 38

Fonte: autor.

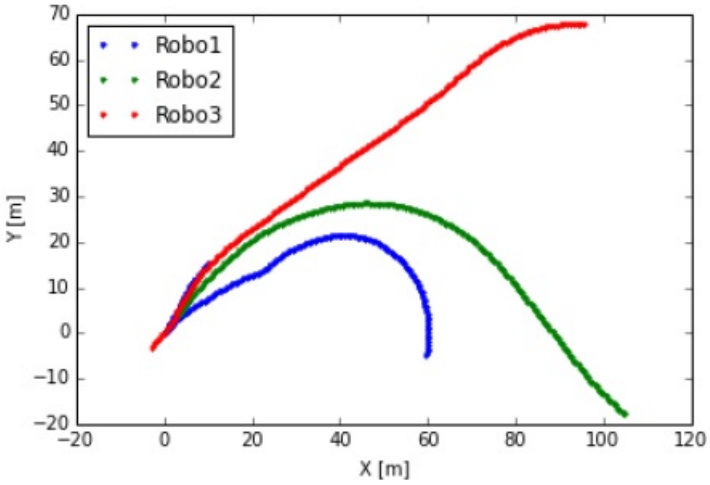


Figura 40 – Gráfico da trajetória percorrida pelo robô no simulador utilizando o algoritmo de Aprendizado por Reforço

Fonte: Silva, 2015.

5.2.3 Simulação com 6 Parâmetros

Neste experimento foram realizadas as simulações utilizando os 6 parâmetros dos osciladores que geram o caminhar do robô. Os parâmetros de Ajuste e Realimentação foram mantidos fixados nos valores definidos pelo simulador como padrão, ao qual este trabalho denomina como valores nominais, durante todas as simulações deste experimento.

Os Três parâmetros adicionados neste experimentos são os restantes dos parâmetros que influenciam diretamente nos osciladores acoplados que geram a caminhada e que são definidos pelas formulas 5, 6 e 7. Estes parâmetros são dps_ratio, $step\_forward\_back\_ratio$ e $swing\_top\_down$.

O parâmetro dps_ratio é a Razão entre o tempo em que ambos os pés estão no chão para quando apenas um pé (esquerdo ou direito) está no chão. Este parâmetro corresponde ao r na formula 7. Desta forma, este parâmetro influencia na velocidade e estabilidade da caminhada, pois um valor baixo faz com que o tempo que o robô passa com um único pé no chão seja maior podendo causar instabilidade, já um valor mais alto faz com que o tempo com um único pé no chão seja pequeno, entretanto é nesta fase que o robô movimenta o pé para frente e, portanto, se esta fase for muito breve o robô não será capaz de se deslocar com eficiência.

O parâmetro $step\_forward\_back\_ratio$ é a distância, no eixo x, da diferença entre o pé esquerdo e o pé direito. Na formula 7 dos osciladores acoplados o parâmetro representa o ρ_{move} no eixo x. Este parâmetro define o quanto o robô irá esticar o pé que está sendo lançado para a frente durante o passo e portanto um valor pequeno faz com que o passo seja longo aumentado o tamanho do passo e portanto a velocidade, porem gerando instabilidade. Já um valor menor garante a estabilidade da caminhada porem diminui a distância percorrida em um passo e consequentemente a velocidade.

O parâmetro $swing\_top\_down$ é o balanço na direção do eixo z, subindo e descendo o corpo, que a cintura do robô faz durante a caminhada. Este parâmetro corresponde ao ρ_{bal} no eixo z da formula 6 . Como os demais parâmetros relacionados ao oscilador de balanço, este parâmetro define a quantidade de movimento que será aplicada no quadril do robô, centro de massa, para balancear os movimentos no eixo z causados pela execução da caminhada, ou seja, tornar a caminhada mais estável. Um valor pequeno pode causar um movimento do quadril insuficiente para gerar estabilidade e um valor grande deste parâmetro pode causar uma movimentação do quadril desnecessário gerando instabilidade na caminhada.

Para as simulações deste experimento, foram utilizadas as seguintes configurações:

a) 50 partículas, 30 segundos de execução e 500 iterações;

b) 100 partículas, 30 segundos de execução e 500 iterações;

A simulação utilizando a primeira configuração teve duração em torno de 14horas, enquanto que a simulação utilizando a segunda configuração durou em torno de 28 horas. Foi observado que nesta situação o aumento no número de partículas não resultou em nenhuma alteração na qualidade do resultado obtido.

Também foi possível verificar que realizar a otimização dos 19 parâmetros da caminhada do robô humanoide é viável, dado que um aumento no número de parâmetros não reflete diretamente em um aumento do tempo de execução do algoritmo.

As figuras 41, 42 mostram a evolução do valor de adaptação ao longo das iterações das simulações realizadas com ambas as configurações deste experimento.

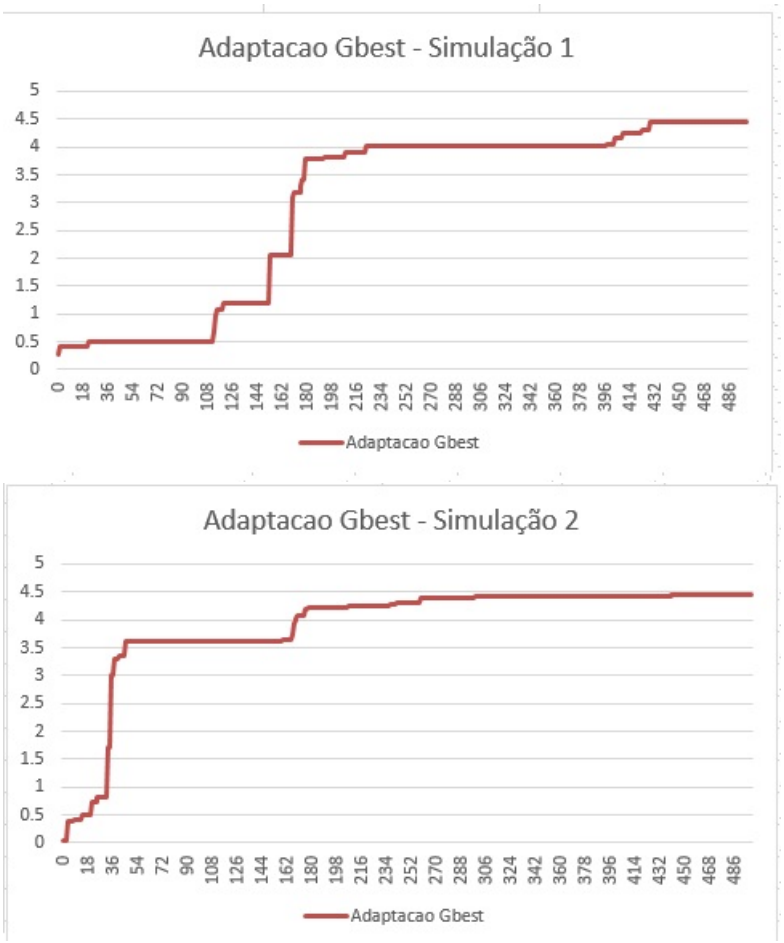


Figura 41 – Valor de adaptação do $gbest$ durante a otimização de 6 parâmetros, utilizando 50 partículas e 30 segundos de execução da simulação

Fonte: autor.

Através das figuras 41, 42 é possível observar que o algoritmo foi capaz de realizar a otimização dos 6 parâmetros, sendo capaz até de encontrar valores de adaptação melhores do que no experimento anterior. Estes resultados mostram que é possível realizar a otimização dos 19 parâmetros do caminhar do robô humanoide utilizando a configuração de 50 partículas e 30 segundos de tempo de execução da simulação, porem será necessário aumentar a quantidade máxima de iterações.

Neste experimento também foi realizada a impressão da trajetória realiza pelo robô humanoide utilizando conjuntos de parâmetros coletados no início, meio e no termino do algoritmo de PSO e o conjunto nominal. A figura 43 mostra estas quatro trajetórias coletadas através de uma simulação com duração de 30 segundos.

Assim como na subseção 5.2.2, a figura 43 utiliza a nomenclatura *LOW*, *MID*, *HIGH* e *Nominal* para diferenciar as trajetórias realizadas por conjuntos de parâmetros diferentes. Os nomes são referências ao momento da execução do PSO que esses conjuntos de parâmetros foram coletados, exceto *Nominal* que se refere ao conjunto de parâmetros definido pelo simulador. As trajetórias da figura 43 são:

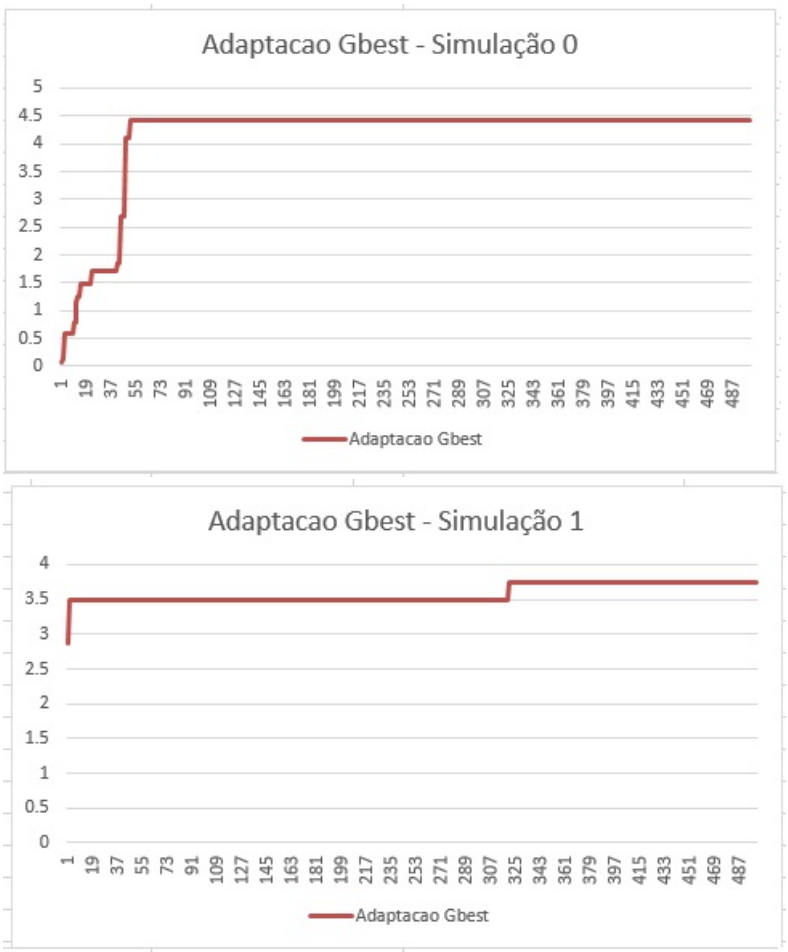


Figura 42 – Valor de adaptação do $gbest$ durante a otimização de 6 parâmetros, utilizando 100 partículas e 30 segundos de execução da simulação

Fonte: autor.

a) Trajetória LOW: Indica a trajetória do conjunto de parâmetros coletados no início da execução do algoritmo de PSO;

b) Trajetória MID: Indica a trajetória do conjunto de parâmetros coletados no meio da execução do algoritmo de PSO;

c) Trajetória HIGH: Indica a trajetória do conjunto de parâmetros coletados ao termino da execução do algoritmo de PSO, sendo este conjunto a resposta do PSO;

d) Trajetória Nominal: Indica a trajetória do conjunto de parâmetros definidos pelo simulador Webots;

Através da figura 43 é possível afirmar que o PSO foi capaz de encontrar um conjunto de parâmetros capaz de realizar uma caminhada com uma trajetória próxima a do conjunto nominal, porem com uma velocidade superior. Com velocidades de 14,03 cm/s para o conjunto de parâmetros encontrado pelo PSO e 9,49 cm/s para o conjunto nominal.

Dado que a execução deste experimento durou 14 horas por simulação, a previsão do tempo de execução para uma configuração contendo o máximo de iterações igual a 2000 é de

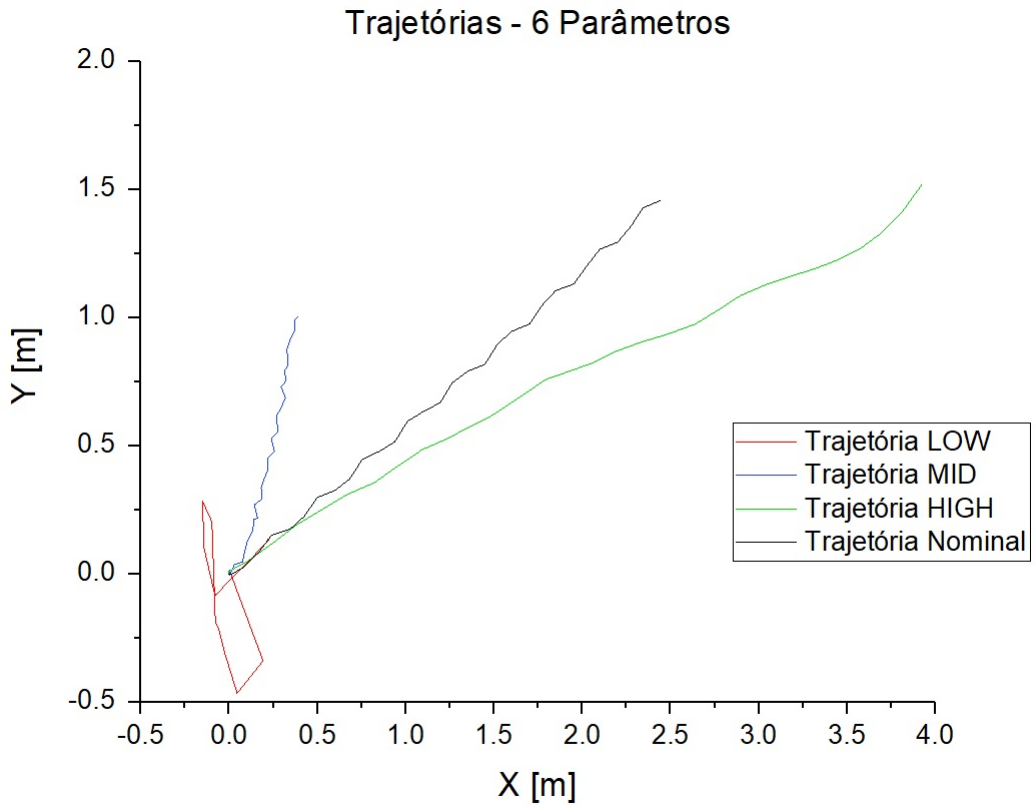


Figura 43 – Trajetórias de 30 segundos utilizando conjunto de parâmetros nominal e coletados da execução do PSO utilizando 6 parâmetros

Fonte: autor.

56 horas. Ao se comparar este tempo de execução com os 7 dias necessários para otimizar 3 parâmetros utilizando o aprendizado por reforço, mostra que o PSO é uma técnica adequada para realizar a otimização dos parâmetros da caminhada do robô humanoide.

5.2.4 Simulação com 19 Parâmetros

Neste experimento foi realizada a otimização da caminhada do robô humanoide utilizando todos os 19 parâmetros, tanto os parâmetros dos osciladores, quanto os parâmetros de Ajuste, quanto os parâmetros de realimentação. Devido as informações coletadas nos experimentos anteriores, neste experimento foi mantido as alterações como por exemplo o limite do ângulo final do robô após a caminhada, o verificador de erros de junta e a limitação a números negativos no parâmetro $period\_time$ e foi escolhida a configuração com 50 partículas, 30 segundos de duração para a simulação e número total de 2000 iterações.

A tabela 4 mostra os valores de quatro conjunto de parâmetros obtidos pelo algoritmo de PSO e seus respectivos valores de adaptação. A tabela também compara os valores dos parâmetros obtidos utilizando o PSO com os parâmetros definidos com padrão pelo simulador, denominado como “Valor Nominal” na tabela.

Também pode ser observado na tabela 4 que cada uma das execuções resultou em um conjunto de parâmetros diferentes, porem com valores de adaptação muito próximos. Isso ocorreu porque o algoritmo do PSO foi capaz de encontrar diversas áreas de ótimo contendo conjuntos de parâmetros diferentes que fossem capazes de produzir uma caminhada ótima.

A figura 44 mostra o valor de adaptação do $gbest$ durante cada uma das iterações. Nesta figura é possível observar que o algoritmo foi capaz de vasculhar o espaço de soluções e encontrar o conjunto de parâmetros capaz de fazer com que o robô humanoide realizasse uma caminhada ótima.

A partir dos resultados mostrados na figura 44 é possível afirmar que o algoritmo do PSO é capaz de identificar um conjunto de parâmetros responsável por gerar uma caminhada otimizada em um espaço de soluções aproximadamente infinito, dado que somente foi implementado um limite na dimensão $period\_time$ com a finalidade de impedir o robô de andar de costas. Para chegar no resultado o algoritmo necessitou de 675 iterações, ou 3,25 dias.

O algoritmo foi capaz de encontrar alguns conjuntos de parâmetros. Com isso é possível concluir que no ambiente em que foi realizada a otimização, no caso da simulação utilizada uma superfície lisa, é possível utilizar diversos conjuntos de parâmetros para obter uma caminhada mais rápida e mais estável. Isso ocorre, pois, diversos parâmetros podem ser alterados para compensar a instabilidade gerada pela modificação de um parâmetro. Fazendo assim com que diversas regiões de valores capazes de gerar uma caminhada rápida e estável surjam no espaço de soluções, o que faz com que o algoritmo retorne respostas diferentes para cada execução.

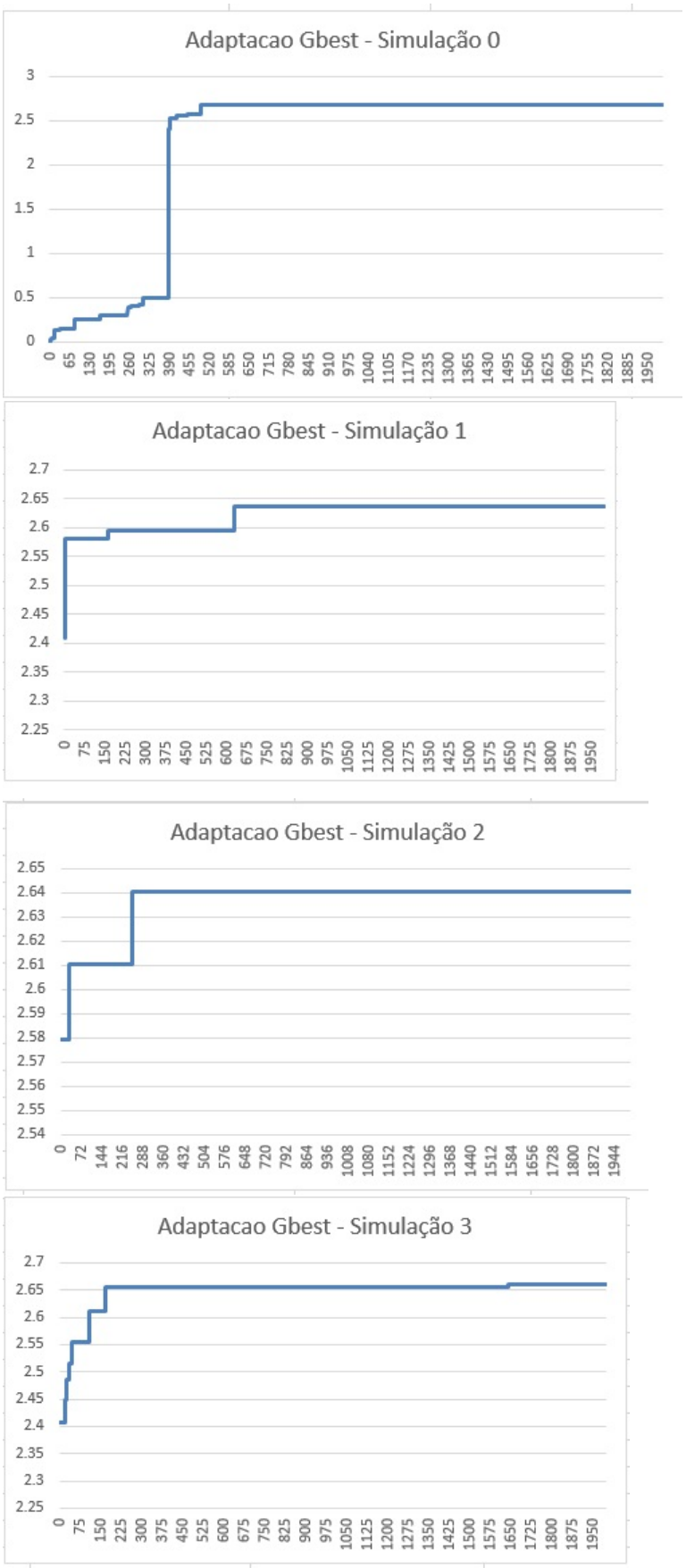


Figura 44 – Valor de adaptação do $gbest$ durante a otimização de 19 parâmetros, utilizando 50 partículas e 30 segundos de execução da simulação

Fonte: autor.

Tabela 4 – Resultados das simulações dos 19 parâmetros

Numero da Simulação	Sim 0	Sim 1	Sim 2	Sim 3	Valor Nominal
Posicao x_offset Gbest	13.619	−197.209	11.803	0.906078	−10
Posicao y_offset Gbest	12.9465	−95.017	38.7835	−102.655	5
Posicao z_offset Gbest	10.7187	−2.86318	−11.2987	−61.6729	20
Posicao roll_offset Gbest	200.996	−44.4763	159.06	−69.1137	0
Posicao pitch_offset Gbest	15.8781	−45.6018	−63.7896	109.309	0
Posicao yaw_offset Gbest	−12.2359	3.93024	−306.595	−43.4101	0
Posicao hip_pitch_offset Gbest	−71.0751	−32.2717	0.545947	92.7896	13
Posicao period_time Gbest	398.565	673.477	1075.24	525.257	600
Posicao dsp_ratio Gbest	2.653	6.46438	20.8131	6.25901	0.1
Posicao step_forward_back_ratio Gbest	5.54768	5.95633	5.23852	1.69775	0.28
Posicao foot_height Gbest	176.321	159.315	164.565	125.651	40
Posicao swing_right_left Gbest	73.7961	145.341	101.381	173.358	20
Posicao swing_top_down Gbest	−65.6819	20.4283	64.5123	211.516	5
Posicao pelvis_offset Gbest	−486.708	84.549	−10.7283	13.6351	3
Posicao arm_swing_gain Gbest	−4.97245	−2.08146	2.88007	6.62693	1.5
Posicao balance_knee_gain Gbest	−0.0160437	5.64968	6.11142	7.3987	0.3
Posicao balance_ankle_pitch_gain Gbest	3.75611	4.55549	7.33153	7.35562	0.9
Posicao balance_hip_roll_gain Gbest	9.88561	3.14559	4.56306	5.77076	0
Posicao balance_ankle_roll_gain Gbest	0.165656	2.416	7.23975	4.61591	0
Adaptacao Gbest	2.68272	2.63543	2.64054	2.66123	2.67899

5.2.5 Simulação com 6 Parâmetros e com passo 1.5 vezes maior

Neste experimento foi realizada a otimização dos 6 parâmetros dos osciladores da caminhada do robô humanoide utilizando um tamanho de passo 1.5 vezes maior do que o das simulações anteriores e possuía o objetivo de analisada a capacidade de adaptação do algoritmo, obrigando que o robô realize um passo 1,5 vezes maior do que os dos experimentos anteriores. A configuração escolhida foi a com 50 partículas, 30 segundos de duração para a simulação e número total de 500 iterações.

Para ajustar o passo do robô para 1,5 vezes maior foi necessário alterar o controlador. No simulador Webots este controlador permite que seja solicitado ao robô que execute um passo no tamanho necessário, entretanto este controlador limita essa requisição para que somente seja possível solicitar a execução de um passo menor ou igual a 1 vez o tamanho do passo gerado pela técnica de osciladores acoplados. Ao alterar este limite para 1,5, é possível solicitar que o robô execute um passo 1,5 vezes maior do que o passo gerado pela técnica dos osciladores acoplados.

A figura 45 apresenta um gráfico de valor de adaptação versus iterações de duas amostras.

A figura 45 mostra que o algoritmo foi capaz de realizar a otimização mesmo com a dificuldade de manter a estabilidade da caminhada do robô humanoide realizando uma passada maior. Foi necessário um número maior de iterações para encontrar o resultado porem estes resultados possuem valores de adaptação próximos.

O fato de ser possível realizar esta otimização mostra que o PSO é capaz de realizar a otimização do caminhar do robô humanoide em diversos ambientes, permitindo que seja possível encontrar o conjunto de parâmetros que gera uma caminhada ótima tanto para um piso liso como para uma grama sintética, e independente das características do robô, ou seja, se alterarmos o robô utilizado para um outro robô humanoide que utilize os mesmos 19 parâmetros para gerar sua caminhada, o algoritmo ainda será capaz de realizar a otimização destes parâmetros. Desta forma caso ocorra uma alteração na regra do futebol de robôs que obrigue os robôs a jogarem em um campo com grama de verdade ao invés de grama sintética, será possível realizar a otimização desde que tal ambiente possa ser inserido no simulador.

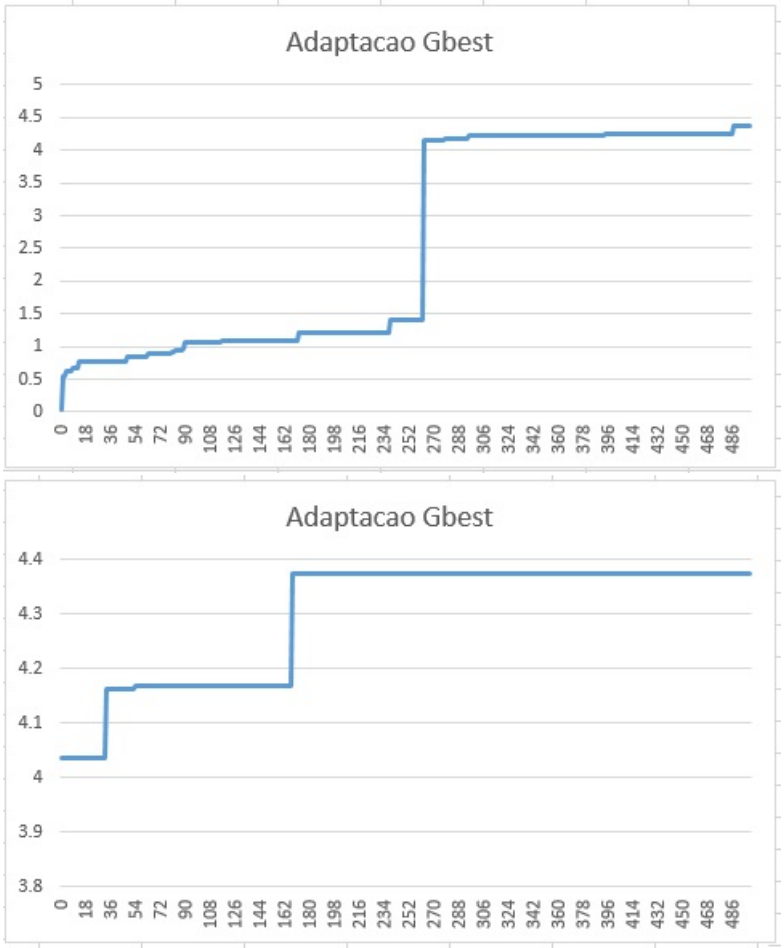


Figura 45 – Valor de adaptação do $gbest$ durante a otimização de 6 parâmetros, utilizando 50 partículas, 30 segundos de execução da simulação e 1.5 de tamanho do passo

Fonte: autor.

5.2.6 Simulação com 19 Parâmetros e com passo 1.5 vezes maior

Dado que foi possível realizar a otimização dos 6 parâmetros do oscilador da caminhada com um passo 1,5 vezes maior, foram realizados experimentos com o objetivo de otimizar todos os 19 parâmetros.

Este experimento tinha por objetivo realizar uma simulação com condições diferentes das condições das primeiras simulações para verificar a capacidade do algoritmo de otimizar os todos os 19 parâmetros em condições extremas.

Apesar de o experimento anterior sugerir que seria possível realizar tal otimização, o algoritmo não foi capaz de encontrar nenhum conjunto de parâmetros que fossem capazes de realizar a caminhada.

Após a execução das 2000 iterações o algoritmo não foi capaz de encontrar um conjunto de parâmetros que fosse capaz de gerar uma caminhada ótima. Este fato mostra que embora este algoritmo seja capaz de realiza tal otimização, é necessário realizar alterações para que este se torne mais preciso. Isso pois no momento que se utiliza um passo 1,5 vezes maior a caminhada

do robô humanoide se torna mais instável e como consequência as áreas ótimas do espaço de soluções passam a serem mais estreitas, aumentando a dificuldade para o algoritmo encontrar tais áreas ótimas.

Como o algoritmo não possui um passo fixo para a geração de novos valores para as dimensões das partículas e este passo possui em seu cálculo um elemento aleatório, o fato das áreas de máximo serem muito estreitas pode fazer com que o algoritmo fique perdido em um vale de resultados não ótimos. O que ocorreu nas otimizações realizadas neste experimento

6 CONCLUSÃO

Este trabalho apresentou a utilização do algoritmo evolutivo PSO para a otimização da caminhada de um robô humanoide. O PSO utiliza uma função de adaptação para avaliar a qualidade de cada possível resultado e utiliza uma função para calcular os novos possíveis resultados levando em consideração os valores de adaptação de cada um dos resultados. No caso deste trabalho, o PSO precisava encontrar o conjunto de parâmetros que seria capaz de realizar uma caminhada ótima para um robô humanoide. A função de adaptação utilizada possuía como variável a quantidade de quedas e a distância percorrida pelo robô durante um período de tempo determinado.

Este capitulo está dividido em duas seções que apresentam as contribuições deste trabalho e que apresentam sugestões de trabalhos futuros a este.

6.1 ANALISE DOS RESULTADOS

Este trabalho contribuiu com a análise da utilização do PSO para realizar a otimização de parâmetros de caminhada em robôs humanoides. Foi possível observar que além do PSO ser capaz de realizar a otimização, ele foi capaz de realizar esta tarefa em uma quantidade de tempo menor do que o algoritmo de aprendizado por reforço utilizando no trabalho de Silva (2015). Enquanto que o aprendizado por reforço precisou de uma semana para realizar a otimização de três parâmetros o PSO foi capaz de realizar esta mesma otimização e, obtendo valores parecidos, somente utilizando, no pior dos casos, 24 horas.

No caso da otimização dos 19 parâmetros o algoritmo necessitou de 675 iterações para chegar ao resultado, porem a simulação foi finalizada com 2000 iterações o que demorou 3,25 dias. Mesmo com o tempo de execução sendo suscetível ao hardware em qual o simulador está sendo executado, o tempo se mostra viável para que este algoritmo seja utilizado para otimizar a caminhada de um robô humanoide mesmo que durante uma competição. Principalmente se forem definidos alguns limites para o espaço de soluções de forma a impedir que o algoritmo tenha a necessidade de avaliar partículas contendo valores de parâmetros que não são capazes de realizar uma caminhada ou que gerem movimentos anormais ou impossíveis (por causa das juntas).

Além disso, o tempo de execução do algoritmo de aprendizado por reforço está diretamente relacionado com o número de parâmetros a serem otimizados, enquanto que o tempo de execução do PSO está vinculado ao número de partículas, número máximo de iterações e, no caso deste trabalho, o tempo de execução da simulação. Isso quer dizer que o tempo de execução do algoritmo de aprendizado por reforço irá aumentar se aumentarmos o número de parâmetros enquanto que o tempo de execução do PSO somente será aumentado se houver a necessidade de alterar o número de partículas, ou o número máximo de iterações devido ao aumento do espaço de soluções.

Por outro lado, foi possível observar que o PSO utilizado neste trabalho teve dificuldades para encontrar o conjunto de parâmetros ótimos quando o espaço de possibilidade era muito grande ou quando as áreas de máximo eram muito pequenas. Devido a característica aleatória e ao fato de existirem diversas áreas de máximo espalhadas pelo espaço de soluções, não existe uma garantia de que duas execuções do algoritmo terão o mesmo conjunto de parâmetros como resposta, porém é possível afirmar que a resposta encontrada pelo algoritmo irá possuir o valor de adaptação otimizado. Outra contribuição deste trabalho é que o algoritmo desenvolvido pode ser utilizado pela equipe RobôFEI para realizar a otimização dos parâmetros de caminhada dos robôs humanoides que participam do futebol de robôs. O benefício é que na eventualidade de uma alteração nas características do campo utilizado, como por exemplo de um tapete verde para grama sintética ou de grama sintética para grama tradicional, a equipe RobôFEI poderá encontrar o conjunto de parâmetros ótimos para estas situações.

Como este algoritmo utiliza os dados coletados da execução da caminhada do robô humanoide para gerar um valor de adaptação e somente altera os valores dos parâmetros que geram a caminhada do robô, este algoritmo não está vinculado ao robô utilizado neste trabalho. Desta forma é possível utilizar este mesmo algoritmo em robôs com características completamente diferentes desde que este outro robô utilize parâmetros para gerar uma caminhada.

Isso acontece pois durante a execução da caminhada as características do robô impactam no comportamento do robô e, portanto, impactam o valor de adaptação já que este depende da distância percorrida pelo robô e a quantidade de quedas sofridas durante a caminhada. Portanto, o algoritmo não precisa ser alterado para se adaptar as características do robô utilizado.

Entretanto, para utilizar este algoritmo é necessário adequá-lo a quantidade e limites dos parâmetros que geram a caminhada do robô e adequá-lo à coleta de dados da caminhada para atribuição do valor de adaptação, pois os dados podem ser providos por um simulador ou pela caminhada de um robô real

6.2 TRABALHOS FUTUROS

Esta seção trata de alguns pontos deste trabalho que ao final, foram observados que poderiam ser temas de trabalhos futuros.

Embora os valores obtidos neste trabalho através do simulado de robótica Webots possam ser utilizados no robô real, em um robô Newton, ou similar, a execução deste algoritmo em um robô real não foi abordada e, portanto, pode ser abordada por trabalhos futuros. O algoritmo utilizado pode ser o mesmo utilizado neste trabalho, porem existe a dificuldade de mitigar a extrema necessidade da repetitiva execução do cálculo de adaptação, uma vez que a excessiva repetição desta tarefa pode deixar a execução deste algoritmo inviável.

Um outro ponto que pode ser explorado por trabalhos futuros é a elaboração de uma função de adaptação que seja capaz de levar em conta os balanços paralelos ao chão para assim

além de garantir uma caminhada livre de quedas, garantir uma caminhada que mantenha um balanço que seja ideal para a estabilidade.

Existem alguns algoritmos que são variações do PSO que podem trazer benefício ao algoritmo utilizado neste trabalho. Um exemplo é o algoritmo de Otimização por Enxame de Partículas autorregulado (SRPSO) que utiliza mecanismos de autor regulação e auto percepção para melhorar a exploração das possibilidades. Também é possível realizar a junção de outras técnicas com o PSO como por exemplo juntar o algoritmo de *Sistemas Imunológicos artificiais* baseado na Seleção Clonal para auxiliar o PSO a se balancear entre diversidade individual e convergência do enxame.

REFERÊNCIAS

AFSHINMANESH, Farzaneh; MARANDI, Alireza; RAHIMI-KIAN, Ashkan. A novel binary particle swarm optimization method using artificial immune system. Em: IEEE. COMPUTER as a Tool, 2005. EUROCON 2005. The International Conference on. 2005. vol. 1, pp. 217–220.

AVRIN, Guillaume et al. Particle Swarm Optimization of Matsuoka's oscillator parameters in human-like control of rhythmic movements. Em: IEEE. AMERICAN Control Conference (ACC), 2016. 2016. pp. 342–347.

BAGLIONI, Stefano et al. Parametric Multibody Modeling of Anthropomorphic Robot to Predict Joint Compliance Influence on End Effector Positioning. Em: AMERICAN SOCIETY OF MECHANICAL ENGINEERS. ASME 2013 International Mechanical Engineering Congress and Exposition. [**sinelocosinenomine**], 2013. v04at04a091–v04at04a091.

BRATTON, Daniel; KENNEDY, James. Defining a standard for particle swarm optimization. Em: IEEE. SWARM Intelligence Symposium, 2007. SIS 2007. IEEE. 2007. pp. 120–127.

CABRAL, Eduardo Lobo Lustosa. **Análise de Robôs**. cap. 5. Acessado: 2017-05-07. URL: <http://sites.poli.usp.br/p/eduardo.cabral/Cinem%C3%A1tica%20Direta.pdf>.

CAI, Chaohong; JIANG, Hong. Performance comparisons of evolutionary algorithms for walking gait optimization. Em: IEEE. INFORMATION Science and Cloud Computing Companion (ISCC-C), 2013 International Conference on. 2013. pp. 129–134.

CASTRO, LN De; TIMMIS, Jon I. Artificial immune systems as a novel soft computing paradigm. **Soft Computing-A Fusion of Foundations, Methodologies and Applications**, Springer, vol. 7, n.º 8, pp. 526–544, 2003.

CLERC, Maurice; KENNEDY, James. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. **IEEE transactions on Evolutionary Computation**, IEEE, vol. 6, n.º 1, pp. 58–73, 2002.

CRAIG, John J. **Introduction to robotics: mechanics and control**. Pearson Prentice Hall Upper Saddle River, 2005. vol. 3.

CYBERBOTICS, Ltd. **Webots Screenshots page**. Acessado: 2017-05-15. 2017. URL: <http://www.cyberbotics.com/screenshots>.

_____. **Webots User Guide release 8.5.4**. 2017. Acessado: 2017-05-01. URL: <https://www.cyberbotics.com/doc/guide/menu>.

_____. **Webots Webpage**. Acessado: 2017-05-01. 2017. URL: <https://www.cyberbotics.com/webots.php>.

EBERHART, Russell; KENNEDY, James. A new optimizer using particle swarm theory. Em: IEEE. MICRO Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on. 1995. pp. 39–43.

FUKUYAMA, Y; NAKANISHI, Yosuke. A particle swarm optimization for reactive power and voltage control considering voltage stability. Em: PROC. 11th IEEE Int. Conf. Intell. Syst. Appl. Power Syst. 1999. pp. 117–121.

GREENSMITH, Julie; WHITBROOK, Amanda; AICKELIN, Uwe. Artificial immune systems. Em: HANDBOOK of Metaheuristics. Springer, 2010. pp. 421–448.

HA, Inyong; TAMURA, Yusuke; ASAMA, Hajime. Gait pattern generation and stabilization for humanoid robot based on coupled oscillators. Em: IEEE. INTELLIGENT Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. 2011. pp. 3207–3212.

HASSAN, Rania et al. A comparison of particle swarm optimization and the genetic algorithm. Em: 46TH AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference. 2005. p. 1897.

HOLLERBACH, John M. Optimum kinematic design for a seven degree of freedom manipulator. Em: CAMBRIDGE, MIT PRESS. ROBOTICS research: The second international symposium. 1985. pp. 215–222.

KENNEDY, James F et al. **Swarm intelligence**. Morgan Kaufmann, 2001.

KENNEDY, James; EBERHART, Russell. Particle swarm optimization. Em: ENCYCLOPEDIA of machine learning. Springer, 2011. pp. 760–766.

KIM, Jeong-Jung; LEE, Jun-Woo; LEE, Ju-Jang. Central pattern generator parameter search for a biped walking robot using nonparametric estimation based particle swarm optimization. **International Journal of Control, Automation and Systems**, Springer, vol. 7, n.º 3, pp. 447–457, 2009.

LI, Rui; ZHAN, Weida; HAO, Ziqiang. Artificial Immune Particle Swarm Optimization Algorithm Based on Clonal Selection. **Boletín Técnico**, vol. 55, n.º 1, pp. 158–164, 2017.

LLC, CH Robotics. **UM6 Ultra-Miniature Orientation Sensor Datasheet**. Acessado: 2017-05-30. URL: <http://www.chrobotics.com/docs/UM6_datasheet.pdf>.

MARDER, Eve; BUCHER, Dirk. Central pattern generators and the control of rhythmic movements. **Current biology**, Elsevier, vol. 11, n.º 23, r986–r996, 2001.

NIEHAUS, Cord; RÖFER, Thomas; LAUE, Tim. Gait optimization on a humanoid robot using particle swarm optimization. Em: PROCEEDINGS of the Second Workshop on Humanoid Soccer Robots in conjunction with the. 2007. pp. 1–7.

O'FLAHERTY, Rowland et al. **Kinematics and Inverse Kinematics for the Humanoid Robot HUBO2+**. 2013.

PERICO, Danilo H. et al. RoboFEI Humanoid Team 2015 Team Description Paper for the Humanoid KidSize League. Em:

RIGHETTI, Ludovic; IJSPEERT, Auke Jan. Programmable central pattern generators: an application to biped locomotion control. Em: IEEE. ROBOTICS and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on. 2006. pp. 1585–1590.

ROBOCUP, Federation. **RoboCup**. Acessado: 2017-05-15. URL: <http://www.robocup.org>.

SHI, Yuhui et al. Particle swarm optimization: developments, applications and resources. Em: IEEE. EVOLUTIONARY computation, 2001. Proceedings of the 2001 Congress on. 2001. vol. 1, pp. 81–86.

SILVA, Isaac Jesus da. USO DE APRENDIZADO POR REFORÇO PARA OTIMIZAÇÃO DO CONTROLE DE CAMINHADA DE UM ROBÔ MÓVEL HUMANOIDE, 2015.

SUTTON, Richard S.; BARTO, Andrew G. **Introduction to Reinforcement Learning**. 1st. Cambridge, MA, USA: MIT Press, 1998.

TANDON, V. Closing the gap between CAD/CAM and optimized CNC end milling. **Master's thesis, Purdue School of Engineering and Technology, Indiana University Purdue University Indianapolis**, 2000.

TRELEA, Ioan Cristian. The particle swarm optimization algorithm: convergence analysis and parameter selection. **Information processing letters**, Elsevier, vol. 85, n.º 6, pp. 317–325, 2003.

TURING, Alan M. Computing machinery and intelligence. **Mind**, JSTOR, vol. 59, n.º 236, pp. 433–460, 1950.

VUKOBRATOVIĆ, Miomir; BOROVAC, Branislav. Zero-moment point—thirty five years of its life. **International Journal of Humanoid Robotics**, World Scientific, vol. 1, n.º 01, pp. 157–173, 2004.

WANG, Yanchao et al. Crystal structure prediction via particle-swarm optimization. **Physical Review B**, APS, vol. 82, n.º 9, p. 094116, 2010.

WESTERVELT, Eric R et al. **Feedback control of dynamic bipedal robot locomotion**. CRC press, 2007. vol. 28.

YAO, Xin. Evolving artificial neural networks. **Proceedings of the IEEE**, IEEE, vol. 87, n.º 9, pp. 1423–1447, 1999.