UTILIZAÇÃO DE CPGS E TÉCNICAS DE INTELIGÊNCIA COMPUTACIONAL NA GERAÇÃO DE MARCHA EM ROBÔS HUMANOIDES

RAFAEL CORTES DE PAIVA

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UTILIZAÇÃO DE CPGS E TÉCNICAS DE INTELIGÊNCIA COMPUTACIONAL NA GERAÇÃO DE MARCHA EM ROBÔS HUMANOIDES

RAFAEL CORTES DE PAIVA

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA.

APROVADA POR:

_____

Prof. Alexandre Ricador Soares Romariz, ENE/FT/UnB
(Orientador)

_____

Prof. Marco Antonio Assfalk de Oliveira, EEEC/UFG
Examinador Externo

_____

Prof. Antônio Padilha L. Bó, ENE/FT/UnB
Examinador Interno

BRASÍLIA, 18 DE AGOSTO DE 2014.

FICHA CATALOGRÁFICA

PAIVA, RAFAEL CORTES DE
Utilização de CPGs e técnicas de inteligência computacionalna geração de marcha em robôs humanoides [Distrito Federal] 2014.
xi, 83p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2014).
DISSERTAÇÃO DE MESTRADO – Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Central Pattern Generator (CPG)
2. Algoritmo Genético (AG)
3. Particle Swarm Optimization (PSO)
4. Aprendizagem por reforço

I. ENE/FT/UnB
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

PAIVA, R.C. (2014). Utilização de CPGs e técnicas de inteligência computacionalna geração de marcha em robôs humanoides, DISSERTAÇÃO DE MESTRADO em Engenharia Elétrica, Publicação PPGENE.TD-570/2014, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 83p.

CESSÃO DE DIREITOS

AUTOR: Rafael Cortes de Paiva
TÍTULO: Utilização de CPGs e técnicas de inteligência computacionalna geração de marcha em robôs humanoides.
GRAU: Mestre ANO: 2014

Rafael Cortes de Paiva
Departamento de Eng. Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, o autor da vida, por me permitir e ajudar a chegar até aqui. Aos meus pais e parentes por me darem suporte durante toda minha vida. Ao meu professores orientador Alexandre Romariz e professor co-orientador Geovany Borges por terem me orientado durante esse trabalho. Aos meus companheiros de curso e companheiros do LARA (Laboratório de robótica e automação) pelo companheirismo e ajudas dadas durante o curso e no LARA. E também aos companheiros do NVC (Núcleo de Vida Cristã) pelo companheirismo e também demonstrar o Reino do Senhor na universidade e na vida.

RESUMO

UTILIZAÇÃO DE CPGS E TÉCNICAS DE INTELIGÊNCIA COMPUTACIONAL NA GERAÇÃO DE MARCHA EM ROBÔS HUMANOIDES

Autor: Rafael Cortes de Paiva

Orientador: Prof. Alexandre Ricador Soares Romariz, ENE/FT/UnB

Programa de Pós-graduação em Engenharia de Sistemas Eletrônicos e Automação

Brasília, 18 de agosto de 2014

Nesse trabalho foi realizado o estudo de técnicas bio-inspiradas para gerar a marcha de um robô bípede. Foi utilizado o conceito de CPG, *Central Pattern Generator* (CPG), que é uma rede neural capaz de produzir respostas rítmicas. Elas foram modeladas como osciladores acoplados chamados de osciladores neurais. Para tanto foram utilizados alguns modelos de osciladores, o modelo de Matsuoka, o modelo de Kuramoto e o modelo de Kuramoto com acoplamento entre a dinâmica do oscilador e a dinâmica da marcha. Foram usados dois modelos de robôs, o Bioloid e o NAO. Para otimizar os parâmetros dos osciladores foram utilizados o Algoritmo Genético (AG), o *Particle Swarm Optimization* (PSO) e o *Nondominated sorting Genetic Algorithm II* (NSGA-II). Foi utilizada uma função de custo que através de determinadas condições tem como objetivo obter uma marcha eficiente. No NSGA-II, além dessa função de custo, foi utilizada outra função de custo que considera o trabalho realizado pelo robô. Além disso, também foi utilizada a aprendizagem por reforço para treinar um controlador que corrige a postura do robô durante a marcha. Foi possível propor um *framework* para obter os parâmetros dos osciladores e através dele obter uma marcha estável em ambas as plataformas. Também foi possível propor um *framework* utilizando aprendizagem por reforço para treinar um controlador para corrigir a postura do robô com a marcha sendo gerado pelo oscilador de Kuramoto com acoplamento. O objetivo do algoritmo foi minimizar a velocidade do ângulo de arfagem do corpo do robô, dessa forma, a variação do ângulo de arfagem também foi minimizada consequentemente. Além disso, o robô andou mais "cautelosamente" para poder manter a postura e dessa forma percorreu uma distância menor do que se estivesse sem o controlador.

Palavras chaves: *Central Pattern Generator* (CPG), Algoritmo Genético (AG), *Particle Swarm Optimization* (PSO), *Nondominated sorting Genetic Algorithm II* (NSGA-II), Aprendizagem por reforço

ABSTRACT

USING CPGS AND COMPUTATIONAL INTELLIGENCE TECHNIQUES IN THE GAIT GENERATION OF HUMANOID ROBOTS

Author: Rafael Cortes de Paiva

Supervisor: Prof. Alexandre Ricador Soares Romariz, ENE/FT/UnB

Eletronic System and Automation Engineering Graduate Program

Brasília, 18th August 2014

This document describes computational optimized bipedal robot gait generators. The gaits are applied by a neural oscillator, composed of coupled central pattern generators (CPG), which are neural networks capable of producing rhythmic output. The models of the oscillators used were the Matsuoka model, Kuramoto model and Kuramoto model with coupling between the dynamics of the oscillator and dynamics of the gait. Two bipedal robots, a NAO and a Bioloid, were used. The neural oscillators were optimized with three algorithms, a Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Nondominated sorting Genetic Algorithm II (NSGA-II). It was used a fitness function that has the objective to obtain an efficient gait through some conditions. In NSGA-II, besides this fitness function, another one was used that has the objective to minimize the work done by the robot. Additionally, reinforcement learning techniques were used to train a controller that corrects the robots gait posture. It was proposed a framework to obtain the parameters of the oscillators used and obtain efficient gaits in both robots. Also, it was proposed a framework using reinforcement learning to train a controller to correct the robots gait posture. The objective of the algorithm was to minimize the pitch angular velocity, consequently the pitch angle standard deviation was minimized. Additionally, the robot moved with more "caution" and walked less compared with the walk without the posture controller.

Keyword: Central Pattern Generator (CPG), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Nondominated sorting Genetic Algorithm II (NSGA-II), Reinforcement learning

SUMÁRIO

Lista de Figuras

Lista de Tabelas

LISTA DE SÍMBOLOS

Notação

Foi adotada a notação para vetor com uma barra em cima do simbolo. Por exemplo $\overline{x}$.

Símbolos Latinos

x	Taxa de disparo de um neurônio do oscilador de Matsuoka/ Parente do AG/ / Posição do PSO
v	Fadiga de um neurônio do oscilador de Matsuoka/ Velocidade do PSO
g	Coeficiente de inibição mútua entre a taxa de disparo de dois neurônios do oscilador de Matsuoka
y	Saída/ Parente do AG
u	Entrada
w	Pesos utilizados nos osciladores de Matsuoka/ Peso de inercia do PSO
f	Frequência
A	Amplitude/ Conjunto de ações da aprendizagem por reforço
k	Constante de acoplamento para oscilador de Kuramoto/
a	Estado da amplitude/ Ação da aprendizagem por reforço
o	Estado de offset
O	Offset
D	Fração do tempo em que a perna move para trás, do ciclo completo do movimento
N	Quantidade
c	Amplitude de harmônico da Série de Fourier / Constante acelerativa para o PSO/ Centro
$\Re$	Conjunto de números reais
n	Contador
S	Conjunto de soluções para o NSGA-II/ Conjunto de estados do ambiente da aprendizagem por reforço
$\mathcal{F}$	Frente de Pareto
p	Solução p para o NSGA-II/ melhor posição uma partícula achou ou conhece no PSO/Probabilidade de sequência de eventos ocorrer na aprendizagem por reforço
q	Solução q para o NSGA-II
Q	Conjunto de soluções para o NSGA-II
P	Conjunto de soluções para o NSGA-II
R	Conjunto de soluções para o NSGA-II/ Retorno da aprendizagem por reforço

t	tempo, contínuo ou discreto
s	Estado do ambiente da aprendizagem por reforço
s'	Próximo estado do ambiente da aprendizagem por reforço
r	Recompensa da aprendizagem por reforço
T	Ultimo passo de tempo da aprendizagem por reforço
$\mathcal{P}$	Probabilidades de transição da aprendizagem por reforço
$\mathcal{R}$	Valor esperado da próxima recompensa da aprendizagem por reforço
V	Função valor da aprendizagem por reforço
Q	Função ação-valor da aprendizagem por reforço
e	Rastro de elegibilidade da aprendizagem por reforço por TD(λ)
b	Base
J	Matriz Jacobiana
h	Altura
W	Trabalho
K	Constante

Símbolos Gregos

τ	Constante de tempo
β	Representa a fadiga na taxa de disparo de cada neurônio para o oscilador de Matsuoka/Constante de tamanho de passo da atualização da política da aprendizagem por reforço
ϕ	Fase do oscilador de Kuramoto/ Valor referente a características de estados da aprendizagem por reforço/ Ângulo de arfagem
π	Número pi/ Política da aprendizagem por reforço
φ	Defasagem desejada entre osciladores de Kuramoto/ Constante para calcular o fator de restrição do PSO
α	Constante da exponencial para oscilador de Kuramoto/ Número randômico entre 0 e 1 do AG/Constante de tamanho de passo da atualização da função valor da aprendizagem por reforço
Γ	Comando do motor
Θ	Fase
γ	Saída de uma aproximação cúbica/ Taxa de desconto da aprendizagem por reforço
θ	Fase de harmônicos da Série de Fourier/ Parâmetros ajustáveis de uma função aproximadora (no caso função valor da aprendizagem por reforço)
$\prec_n$	Operador comparador de lotação
ϵ	Números randômicos no PSO/ Probabilidade de escolher uma ação randômica na aprendizagem por reforço
χ	Fator de restrição do PSO
Δ	Variável auxiliar usada na aprendizagem por reforço/ Diferença
δ	Erro de diferença temporal da aprendizagem por reforço por TD
λ	Parâmetro de decaimento de rastro da aprendizagem por reforço por TD(λ)
σ	Desvio padrão
ψ	Parâmetros ajustáveis de uma função aproximadora (no caso da política da aprendizagem por reforço)/ Ângulo de rolagem
μ	Média
Σ	Matriz de covariância
$\mathscr{S}$	Trajetória
ω	Velocidade angular

Subscritos

$\{e, f\}$	Extensor e flexor para o oscilador de Matsuoka
i	Equivale ao número utilizado
j	Equivale ao número utilizado
vir	Virtual
k	Equivale ao número utilizado
n	Dimensão
p	Solução p para o NSGA-II
q	Solução q para o NSGA-II
$distancia$	*Crowding-distance* para o NSGA-II
$rank$	Rank (uma medida que indica qual frente a solução está) para o NSGA-II
g	A melhor posição que uma partícula do PSO conhece
max	Máximo
s	Estado do ambiente da aprendizagem por reforço
s'	Próximo estado do ambiente da aprendizagem por reforço
$'$	Referente ao próximo estado do ambiente da aprendizagem por reforço
$*$	Referente a política ótima da aprendizagem por reforço
π	Referente a política π da aprendizagem por reforço
θ	Referente aos parâmetros θ da função de aproximação usada
ψ	Referente aos parâmetros ψ da função de aproximação usada
lh	*left hip* (quadril esquerdo)
rh	*right hip* (quadril direito)
lk	*left knee* (joelho esquerdo)
rk	right knee (joelho direito)
aux	Auxiliar
$pitch$	Arfagem
$roll$	Rolagem
pl	Posição lateral
$total$	Total
osc	Oscilador
r	Robô
z	Eixo z

Sobrescritos

h	Quadril (do inglês *hip*)/ harmônicos
k	Joelho (do inglês *knee*)
st	Stance
sw	Swing
$+$	Conjunto de estado mais os estados terminais da aprendizagem por reforço
a	Ação da aprendizagem por reforço
n	N-passos na aprendizagem por reforço
λ	Retorno-λ na aprendizagem por reforço
D	Dimensão
e	Esquerdo
d	Direito
p	Posição
$filtro$	Referente a filtro
qa	Quadril arfagem
ja	Joelho arfagem

Siglas

AG	Algoritmo Genético
CdM	Centro de Massa
CdP	Centro de Pressão
CPG	Central Pattern Generator
MC	Monte Carlo
MDP	Processo de decisão Markoviano (do inglês *Markov Decision Process*)
NSGA-II	Nondominated sorting Genetic Algorithm II
PSO	Particle Swarm Optimization
TD	Diferença temporal (do inglês Temporal-Difference)
ZMP	ponto de momento nulo (do inglês *zero moment point*)

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O termo robô tem origem na palavra checa *robota*, que significa "trabalho forçado". O termo foi introduzido ao público pelo escritor Karel Capek em sua peça R.U.R (*Rossum's Universal Robot*), publicada em 1920[1]. Não há uma única definição para o termo. Pode-se dizer que um robô é um dispositivo automático que possui conexões de realimentação (*feedback*) entre seus sensores, atuadores e o ambiente, dispensando a ação do controle humano direto para realizar determinadas tarefas, podendo também haver robôs parcialmente controlados por pessoas.

Na robótica há a área da robótica móvel que estuda a locomoção de robôs e envolve conhecimentos de diversas áreas. Dentre elas, pode-se citar a programação, o controle, a matemática, a computação gráfica, a cinemática e a dinâmica. Dentro da robótica móvel há o estudo de robôs com pernas.

O estudo de robôs com pernas tem como principal motivação a necessidade de locomoção em terrenos irregulares e com obstáculos, pois eles possuem vantagens nessa área sobre os robôs com rodas. Assim, pode-se enviar um robô com patas para uma região com um terreno inapropriado para um robô com rodas. Além disso, no caso de robôs bípedes e humanoides há o interesse de se obter características próximas do ser humano. Uma das dificuldades dos robôs com pernas é a geração do movimento que não leve à instabilidade do mesmo.

Nesse sentido há o estudo de várias formas de controle para a locomoção de robôs com patas, como métodos baseados no ponto de momento nulo, ZMP, métodos baseados em ciclo limite, métodos biologicamente inspirados, entre outros.Tratando-se de métodos biologicamente inspirados, há um interesse crescente nessa abordagem, pois animais são capazes de se locomover sem grandes dificuldades, mesmo em terrenos irregulares. Animais devem se locomover para buscar alimentos, fugir de predadores, procurar abrigo, entre outros. Outro fato interessante é que mesmo animais recém nascidos em um pequeno intervalo de tempo já conseguem se locomover [23, 59].

Dessa forma, diversas pesquisas foram feitas sobre o funcionamento da locomoção em animais e em como se implementar isso em robôs com patas. Uma delas é o conceito de osciladores neurais e CPG, *central pattern generator.* Os CPGs são redes neurais, encontradas na espinha dorsal de animais vertebrados, capazes de produzir respostas rítmicas. Porque as respostas são rítmicas e pela locomoção ser um movimento em geral periódico, uma forma de modelagem para os CPGs é através de osciladores, chamados de osciladores neurais [23,59].

[1]Como visto no dicionário online: `http://dictionary.reference.com/browse/robotic` (Última vez acessado dia 20/03/2014)

1.2 OBJETIVOS DESTE TRABALHO

O objetivo deste trabalho é implementar um sistema de geração de marcha em robôs bípedes, utilizando osciladores bio-inspirados. Para tanto, foram usados osciladores neurais, baseado no conceito de CPG, *Central Pattern Generator [12, 13].*

1.3 ALGUNS TRABALHOS CORRELACIONADOS

CPGs já foram utilizadas em diversas tipos de robôs para gerar marchas. Por exemplo, utilizando osciladores acoplados gerar marcha em robôs quadrúpedes, como em [29–32, 39, 43], em bípedes, como em [22, 54], em hexápodes, como em [3, 35].

Diversos modelos de osciladores também já foram utilizados. Exemplos de modelos são o modelo de Matsuoka [12,13,17,24,27,29–31,33,37,38,48,53,54,56,58], oscilador de fase também chamado de modelo de Kuramoto (ou oscilador de fase) [1,2,5,10,11,36,40–42,50], modelo cíclico inibitório [34], modelo de Wilson-Cowan [7, 45, 46, 57], oscilador de Hopf [6, 9, 47, 49, 60, 61], oscilador de Van der Pol [18, 55, 62], oscilador de Rayleigh [18, 19, 55], entre outros.

Huang [22], por exemplo, utilizou o modelo de Matsuoka [37, 54] para gerar a marcha em um robô bípede. Foi proposto um método para obter a coordenação entre os osciladores com o intuito de diminuir erros de fase gerados por pertubações nos osciladores. Além disso, nesse trabalho o sinal de saída dos osciladores é utilizado como referencia para a trajetória dos pés e por cinemática inversa os ângulos das juntas são obtidos.

Um outro exemplo é o trabalho de Sproewitz [50] que utilizou o oscilador de Kuramoto em um robô quadrupede onde a forma de onda de saída foi uma forma de onda específica modelada por ele. Ele utilizou o sinal de saída como ângulos das juntas e otimizou os parâmetros por meio do PSO para obter uma marcha.

Mais um exemplo é o trabalho de Morimoto [40–42] que utilizou o oscilador de Kuramoto para realizar a marcha em um robô humanoide. Como sinal de saída ele utilizou senoides que foram usadas como ângulos das juntas. Ele também propôs uma forma de se realizar um acoplamento entre o oscilador e a dinâmica do movimento.

1.4 DESCRIÇÃO DO PROBLEMA

Nesse trabalho são utilizados osciladores neurais para gerar a locomoção de um robô humanoide. Inicialmente o oscilador de Matsuoka foi utilizado na plataforma Bioloid para gerar a marcha [13]. Observou-se que uma dificuldade de obter os parâmetros que produzem a marcha para o robô e o oscilador serem estáveis. Então, para otimizar os parâmetros dos

osciladores da CPG que geram a marcha do robô foram utilizados o algoritmo genético (AG) e o *particle swarm optimization* (PSO) [12].

Depois foram utilizados os mesmos algoritmos na plataforma NAO usando o oscilador de Matsuoka e o oscilador de Kuramoto, cuja saída é dada por uma Série de Fourier truncada. Nesse ultimo também foi utilizado o oscilador acoplado com a dinâmica do movimento. Também foi proposto utilizar o *nondominated sorting genetic algorithm* II (NSGA-II) para se obter parâmetros através da função de custo usada no AG e PSO e outra que considera o trabalho realizado pelo robô NAO.

Além disso, uma outra questão sobre os osciladores é em como se utilizar entradas sensoriais. Para isso foi utilizado a aprendizagem por reforço para corrigir a postura do robô durante a marcha.

1.5 RESULTADOS OBTIDOS

Foi possível gerar a marcha do robô Bioloid usando o oscilador de Matsuoka com parâmetros otimizados pelo AG e pelo PSO. Foram usados de quatro a seis osciladores e observou-se que o desempenho da marcha com seis osciladores foi melhor.

Também foi possível gerar a marcha no robô NAO usando os osciladores de Matsuoka, de Kuramoto e de Kuramoto com acoplamento entre o oscilador e a dinâmica do movimento. Os parâmetros dos osciladores foram otimizados pelo AG, PSO e NSGA-II. Observou-se que no caso do oscilador de Kuramoto com acoplamento os resultados foram melhores que nos outros casos.

E por fim, também foi possível treinar um controlador de postura utilizando a aprendizagem por reforço ao se utilizar o oscilador com acoplamento para poder corrigir a postura em uma dada marcha.

1.6 APRESENTAÇÃO DO MANUSCRITO

Este manuscrito encontra-se organizado da seguinte forma: no capítulo 2 é feita uma fundamentação teórica sobre CPG, osciladores neurais, algoritmo genético (AG), *particle swarm optimization* (PSO), *nondominated sorting genetic algorithm* II (NSGA-II) e aprendizagem por reforço. No capítulo 3 é feita uma descrição sobre as plataformas bioloid e NAO, como as marchas foram geradas nas plataformas, como foram otimizados os parâmetros dos osciladores usando AG e PSO e como foi realizado o treinamento de postura usando a aprendizagem por reforço. No capítulo 4 são apresentados os resultados experimentais para o bioloid e NAO. Finalmente, no capítulo 5 encontra-se a conclusão.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INTRODUÇÃO

A locomoção de animais com patas e animais bípedes é bastante complexa e requer coordenação de vários músculos e membros do animal para manter a postura estável e realizar a marcha. Além disso, os animais movem-se com eficiência em ambientes complexos e com obstáculos. Ou seja, eles conseguem adaptar sua marcha de acordo com o ambiente e isso de maneira rápida. Eles devem ser capazes de fugir de predadores, buscar alimentos, entre outros. E também é possível observar que animais recém-nascidos, após um pequeno período já são capazes de se movimentar. Desse modo, técnicas inspiradas na biologia dos animais podem utilizadas e adaptadas para a robótica.

Assim, aspectos da morfologia de animais são bastante explorados para serem utilizados em robótica. Nesse trabalho foi utilizado o conceito de CPG, *Central Pattern Generator*, para gerar padrões rítmicos para gerar a marcha do robô. Uma dificuldade do método é em se obter os parâmetros do osciladores usados no CPG, para isso foram utilizados o Algoritmo Genético(AG), *Nondominated Sorting Genetic Algorithm* II (NSGA-II) e *Particle Swarm Optimization* (PSO). Assim, nesse capítulo foi feita uma revisão bibliográfica sobre CPG, AG e PSO.

Além disso, foi utilizado o aprendizado por reforço para treinar um controlador de postura para determinada marcha. Nesse sentido, também será feita uma revisão bibliográfica sobre o aprendizado por reforço.

2.2 CENTRAL PATTERN GENERATOR (CPG)

Os CPGs são redes neurais capazes de produzir respostas rítmicas e não necessitam necessariamente de uma entrada proveniente de sensores ou uma entrada proveniente de um controlador em um nível acima. Como apontado por Yu [59] atividades periódicas como locomoção, respiração e mastigação são produzidas por CPGs. Elas estão presentes em animais invertebrados, presente nos gânglios segmentários, e em animais vertebrados, nesse caso ela encontra-se na espinha dorsal do animal. Normalmente centros de nível maior, como o córtex, cerebelo e alguns gânglios, são responsáveis por modular o padrão de movimento de acordo com as condições do ambiente [23, 59].

Conforme visto no trabalho de Ijspeert [23], foram feitos estudos que comprovam que as redes neurais não necessitam de uma entrada sensorial para gerar respostas rítmicas. No caso, a espinha dorsal de uma lampreia foi extraída e isolada do corpo e mesmo desse modo foi

capaz de produzir padrões de locomoção, chamados de locomoção fictícia. Mas, embora não sejam necessárias entradas sensoriais para o funcionamento do CPG, elas são importantes no funcionamento do movimento, pois por meio delas é possível obter-se coordenação nos movimentos. Essa característica pode ser observada na revisão feita por Ijspeert [23], onde é citado um experimento realizado em um gato descerebrado, em que foi utilizada uma esteira acionada mecanicamente e desse modo era possível induzir caminhada normal.

Uma outra característica dos CPGs é o fato de que a partir de um sinal simples é possível induzir atividade da CPG. Em muitos animais vertebrados estímulos elétricos na Região Locomotora Mesencefálica induzem certos comportamentos locomotivos. O nível de estimulo modula a velocidade do movimento e também pode causar a mudança de marcha. Um exemplo é citado novamente por Ijspeert [23]. Novamente foi utilizado um gato descerebrado, e um aumento do estímulo causava o chaveamento da caminhada para o trote e do trote para o galope. Já na salamandra, que também é citada, um aumento do estimulo causa o chaveamento de caminhada para o movimento de natação.

Há vários meios de se modelar os CPGs dependendo do objeto de estudo. Podem ser empregados modelos biofísicos, modelos conexionistas e um sistema de osciladores acoplados.

Modelos biofísicos são baseados no neurônio de Hodgkin-Huxley, que computa como bombas de íons e canais de íons influenciam o potencial das membranas e a geração de ações potenciais. Modelos conexionistas usam modelos de neurônios simplificados, chamados, do inglês, *leaky-integrator* e *integrate-and-fire neuron.* O foco destes modelos é descrever como a geração de ritmo é gerada por propriedades da rede e como diferentes circuitos de osciladores neurais são sincronizados por meio de conexões intraneurais. O modelo de osciladores acoplados são modelos matemáticos de osciladores não-lineares acoplados. Nesse caso um oscilador representa a atividade completa de um centro oscilatório, em vez de apenas um neurônio ou um pequeno circuito.

O modelo matemático de osciladores acoplados é amplamente usado para robótica, onde normalmente os osciladores são representados por meio de equações diferenciais que são integradas numericamente por meio de processadores e microcontroladores. Esse modelo é utilizado para gerar marcha em robôs quadrúpedes, como em [29–32, 39, 43], em bípedes, como em [22, 54], em hexápodes, como em [3, 35] . Em alguns casos como no trabalho de Nakada, Asai e Amemiya [44], o oscilador foi implementado por um circuito analógico.

Como apontado por Yu [59] há diversos modelos de osciladores não lineares usados para se modelar CPGs. Exemplos de modelos são o modelo de Matsuoka [12, 13, 17, 24, 27, 29–31, 33, 37, 38, 48, 53, 54, 56, 58], oscilador de fase também chamado de modelo de Kuramoto [1, 2, 5, 10, 11, 36, 40–42, 50], modelo cíclico inibitório [34], modelo de Wilson-Cowan [7, 45, 46, 57], oscilador de Hopf [6, 9, 47, 49, 60, 61], oscilador de Van der Pol [18, 55, 62], oscilador de Rayleigh [18, 19, 55], entre outros.

Como visto em alguns dos trabalhos citados, o modelo de Matsuoka [37, 54] é bastante utilizado e também foi utilizado nesse trabalho. Outro modelo utilizado foi o de oscilador de fase. Desse modo, serão mostrados a seguir as características e funcionamento desses modelos.

2.2.1 Oscilador neural de Matsuoka

O oscilador neural de Matsuoka [37, 54] é um oscilador não-linear composto de dois neurônios que possuem uma inibição mútua, na qual um neurônio, chamado extensor, inibe a ação do outro, chamado flexor, e vice versa. A Figura 2.1 representa o oscilador. O oscilador também pode ser descrito por meio das seguintes Equações:

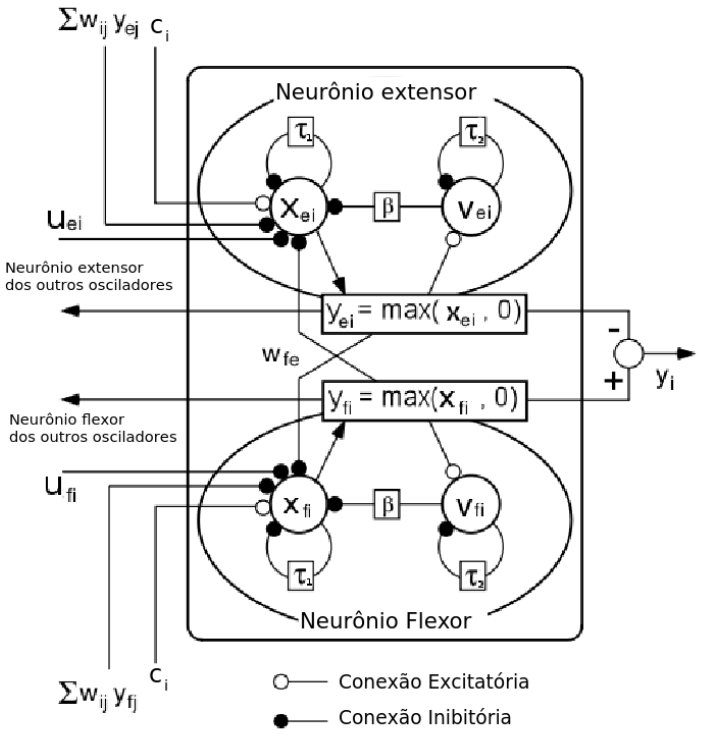


Figura 2.1: Modelo de oscilador utilizado, adaptado de [31].

$$\tau_1 \dot{x}_{\{e,f\}i} = c_i - x_{\{e,f\}i} - \beta v_{\{e,f\}i} - g y_{\{f,e\}i} - u_{\{e,f\}i} - \sum_{j=1}^{n} w_{ij} y_{\{e,f\}j}, \tag{2.1}$$

$$y_{\{e,f\}i} = max(x_{\{e,f\}i}, 0), \tag{2.2}$$

$$u_{\{e\}i} = max(u_i, 0), \tag{2.3}$$

$$u_{\{f\}i} = min(u_i, 0), \tag{2.4}$$

$$\tau_2 \dot{v}_{\{e,f\}i} = -v_{\{e,f\}i} + y_{\{e,f\}i}, \tag{2.5}$$

$$y_i = y_{\{e\}i} - y_{\{f\}i}. \tag{2.6}$$

O índice $\{e, f\}$ representa extensor e flexor, respectivamente e o índice $\{f, e\}$ flexor e extensor. O índice i representa um oscilador específico, para o caso de acoplamento com outros osciladores e o índice n o número total de osciladores. As variáveis $x_{\{e,f\}}$ e $v_{\{e,f\}}$, representam estados do neurônio, $x_{\{e,f\}}$ corresponde a taxa de disparo e $v_{\{e,f\}}$ a fadiga de cada neurônio, de acordo com Bailey [3]. A saída $y_{\{e,f\}}$ de cada neurônio é dada por $x_{\{e,f\}}$ limitado a um mínimo de zero e a saída inibe a ação do outro neurônio e também dos neurônios dos outros osciladores acoplados. A entrada sensorial do oscilador u é utilizada da seguinte forma, se u é positivo então ele inibe a ação do neurônio extensor,$u_{\{e\}}$, se u é negativo então ele inibe a ação do neurônio flexor, $u_{\{f\}}$. A saída do oscilador y é dada pela diferença entre a saída do neurônio extensor $y_{\{e\}}$e a saída do neurônio flexor $y_{\{f\}}$.

Os parâmetros τ_1 e τ_2 representam a constante de tempo de primeira ordem da taxa de disparo, x, e da fadiga,v , respectivamente. A constante c é um estimulo constante, chamada de entrada tônica. Ela representa as entradas de padrão geradas por circuitos de centros maiores. Nesse modelo, um aumento da constante não causa um aumento da frequência de movimento, mas apenas um aumento da amplitude de oscilação. O parâmetro β representa o efeito da fadiga na taxa de disparo de cada neurônio. O parâmetro g é o coeficiente de inibição mútua entre a taxa de disparo dos dois neurônios. Os parâmetros w_{ij} são os pesos dados para o acoplamento entre osciladores, no qual a taxa de disparo do neurônio extensor (ou flexor) de um oscilador inibe a taxa de disparo do neurônio extensor (ou flexor) de outro oscilador. Na Figura 2.2 são mostradas a saída, taxas de disparo e fadiga para o oscilador com determinados parâmetros.

De acordo com Bailey [3], quando o ciclo se inicia a taxa de disparo está menor que zero e a fadiga (escalonada por β) é menor que a entrada constante c, causando o aumento da taxa de disparo (pela taxa determinada por τ_1). Depois de determinado tempo, a taxa de disparo se torna maior que o nível de fadiga, causando o aumento da fadiga (pela taxa determinada por τ_2). A fadiga cresce até ficar acima do efeito da entrada constante, causando a queda da taxa de disparo. A taxa de disparo decresce até ficar menor que a fadiga e eventualmente negativa, causando o decrescimento da fadiga, em direção ao inicio do ciclo. O quão rápido a taxa de disparo decai para o fim do ciclo depende da ação do outro neurônio e do coeficiente de inibição mútua g. A parte positiva da saída de cada neurônio inibe o outro, causando que a taxa de disparo de cada neurônio oscile fora de fase um com o outro.

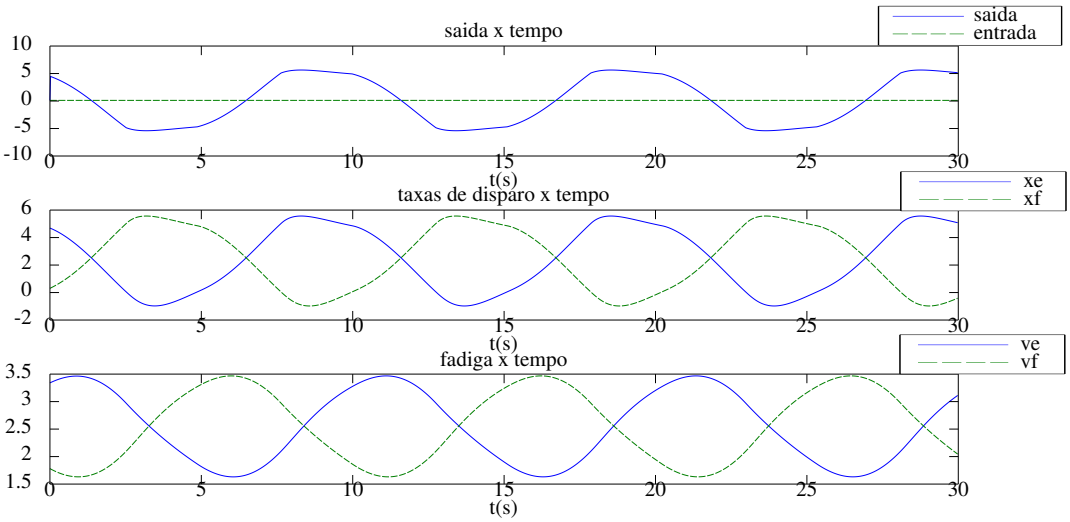


Figura 2.2: Saída, taxa de disparo e fadiga do oscilador de Matsuoka [37] para os seguintes parâmetros:$c = 8,5; \beta = 1,2; g = 1,2; \tau_1 = 0,5033; \tau_2 = 5,0329$.

Foi demonstrado por Huang [22] que o oscilador possui um único ciclo limite se:

$$g - 1 - \frac{\tau 1}{\tau 2} > 0, \tag{2.7}$$

$$g - 1 - \beta > 0. \tag{2.8}$$

Também foi mostrado que caso não haja uma entrada externa a frequência do oscilador é:

$$f = \frac{1}{2\pi\tau_1}\sqrt{\frac{\tau_1}{\tau_2}\left(1 + \left(\frac{\beta}{g} - 1\right)\left(1 + \frac{\tau_1}{\tau_2}\right)\right)}. \tag{2.9}$$

Dessa forma, a amplitude aproximada será:

$$A = \frac{2c}{1 + \beta + g}. \tag{2.10}$$

Dessa forma as Equações 2.7, 2.8, 2.9 e 2.10 foram usadas para se obterem os parâmetros utilizados nesse trabalho e foram utilizadas as mesmas considerações feitas por Huang [22] que foram:

- Como visto na Equação 2.8, o parâmetro g deve ser maior do que 1.
- O valor de $\frac{\tau_1}{\tau_2}$, chamado de b, como visto na Equação 2.7, foi escolhido pequeno para que $g - 1 - \frac{\tau 1}{\tau 2} > 0$, tipicamente entre 0.04-0.1.

- O valor de β foi escolhido igual ao valor de g, $\beta = g$, para simplificar a Equação 2.9 para

$$f = \frac{1}{2\pi\tau_1}\sqrt{\frac{\tau_1}{\tau_2}\left(1+\left(\frac{g}{g}-1\right)\left(1+\frac{\tau_1}{\tau_2}\right)\right)} = \frac{1}{2\pi\tau_1}\sqrt{\frac{\tau 1}{\tau 2}}. \tag{2.11}$$

- Assim a Equação 2.11 foi usada para se obter a constante τ_1 e consequentemente a constante τ_2.

- Assim a constante c foi obtida por meio da Equação 2.10.

2.2.2 Oscilador de fase ou oscilador de Kuramoto

O modelo de Kuramoto [1] é um modelo de oscilador de fase que consiste de N osciladores acoplados que possuem fase $\phi_i(t)$ com frequência natural ω_i e e cuja dinâmica é governada pela Equação abaixo:

$$\dot{\phi_i} = \omega_i + \sum_{j=1}^{N} k_{ij}\text{sen}\left(\phi_j - \phi_i\right),\ i = 1, ..., N. \tag{2.12}$$

Cada oscilador tenta se desenvolver independentemente em sua própria frequência, enquanto o acoplamento tenta sincronizá-lo com os outros.

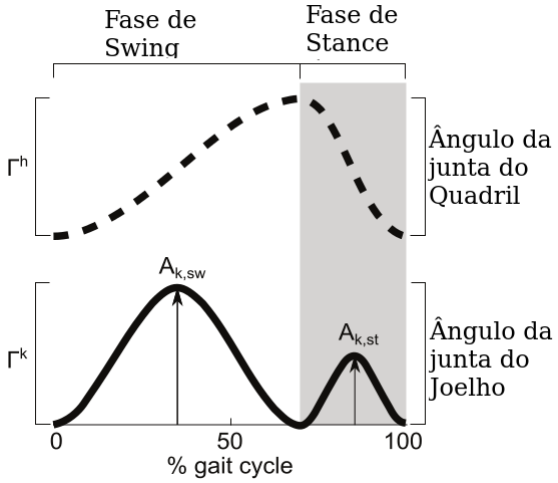


Figura 2.3: Forma de onda do trabalho de Sproewitz [50].(Adaptado de [50])

Como o acoplamento desse oscilador está na fase somente a saída do oscilador pode ter uma forma diferente. Morimoto [40–42] por exemplo usou essa forma de oscilador e as saídas dos osciladores eram senoides com fase $\phi(t)$. Outro exemplo é o trabalho de Sproewitz [50] que obteve a forma de onda dada na Figura 2.3 utilizando as seguintes Equações para o oscilador:

$$\dot{\phi}_i = 2\pi f + \sum_{j \neq i} k_{ij} \text{sen}\left(\phi_j - \phi_i - \varphi_{ij}\right), \tag{2.13}$$

$$\dot{a}_i^h = \alpha\left(A_i^h - a_i^h\right), \tag{2.14}$$

$$\dot{o}_i^h = \alpha\left(O_i^h - o_i^h\right), \tag{2.15}$$

$$\Theta_i^h = \begin{cases} \frac{\phi_i}{2D_{vir}}, & 0 \leq \phi_i \leq 2\pi D_{vir} \\ \frac{\phi_i + 2\pi(1-2D_{vir})}{2(1-D_{vir})} \end{cases}, \tag{2.16}$$

$$\Gamma_i^h = a_i^h \cos\left(\Theta_i^h\right) + o_i^h, \tag{2.17}$$

$$\Theta_i^k = \Theta_i^h + \varphi_{hk}, \tag{2.18}$$

$$a_i^k = \begin{cases} A_{st}^k, & \Theta_i^k < \pi \\ A_{sw}^k & \Theta_i^k \geq \pi \end{cases}, \tag{2.19}$$

$$\dot{o}_i^k = \alpha\left(O_i^k - o_i^k\right), \tag{2.20}$$

$$\theta_i^{'} = \frac{\Theta_i^k}{2\pi}, \tag{2.21}$$

$$\theta_i = 2\theta_i^{'}, \tag{2.22}$$

$$\gamma_i = \begin{cases} -16\theta_i^3 + 12\theta_i^2, & \theta_i < 0,5 \\ 12\left(\theta_i - 0,5\right)^3 - 12\left(\theta_i - 0,5\right)^2 + 1 \end{cases}, \tag{2.23}$$

$$\Gamma_i^k = a_k^i \gamma_i + o_i^k, \tag{2.24}$$

onde ϕ_i é a fase do oscilador i, f é a frequência dos osciladores, k_{ij} é a constante de acoplamento entre os osciladores i, j, $\varphi_{i,j}$ é a defasagem desejada entre os osciladores i e j, a_i^h é o estado da amplitude do quadril e A_i^h a amplitude desejada, o_i^h é o estado de offset do quadril e O_i^h o offset desejado. Note que a forma de onda do estado da amplitude e do estado do offset é uma exponencial e α é uma constante dessa exponencial. D_{vir} é a fração do tempo em que a perna move-se para trás, do ciclo completo do movimento. Γ_i^h é o comando do

motor do quadril. Θ_i^h é a fase do quadril. Θ_i^k é a fase do joelho. φ_{hk} é a defasagem desejada entre o quadril e joelho. a_i^k é o estado da amplitude do joelho. A_{st}^k é a amplitude desejada para a fase de stance do joelho. A_{sw}^k é a amplitude desejada para a fase de swing do joelho. o_i^k é o estado do offset do joelho e O_i^k o offset desejado. As Equações 2.21, 2.22 e 2.23 são usadas para se obter um formato de onda cúbica por partes. Γ_i^k é o comando do motor do joelho.

Inspirado em Sproewitz [50] nesse trabalho, como a saída do oscilador foi utilizada uma Série de Fourier truncada com o intuito de se obter diversas formas de onda. As Equações da saída podem ser vistas abaixo:

$$\dot{a_i} = \alpha \left(A_i - a_i \right), \tag{2.25}$$

$$\dot{o_i} = \alpha \left(O_i - o_i \right), \tag{2.26}$$

$$y_i = o_i + a_i \sum_{k=1}^{N_h} c_{ik} \cos\left(\phi_i - \theta_{ik} \right), \tag{2.27}$$

onde a_i é o estado da amplitude do oscilador i. Note que a amplitude evolui de acordo com uma exponencial tendendo a A_i, amplitude desejada. O valor o_i é o estado do offset do oscilador i.Note que o offset evolui de acordo com uma exponencial tendendo a O_i,offset desejado. A constante α é a constante da exponencial do estado de amplitude e do estado de offset. O valor y_i é a saída do oscilador i. N_h é a quantidade de harmônicos menos um (o menos um surge por conta do k=1 ser a fundamental e não um harmônico) utilizados, c_k é a amplitude de cada harmônico e ϕ_k a fase de cada harmônico. Note que a Série de Fourier está na forma polar.

2.3 ALGORITMO GENÉTICO (AG)

O AG é um método para resolver problemas de otimização com e sem restrições. Ele é baseado em seleção natural, o processo que controla a evolução biológica de acordo com Charles Darwin. A cada passo, o algoritmo seleciona indivíduos, de acordo com uma função de custo, também chamada de função de adequação, da população atual para serem parentes que irão produzir filhos para a próxima geração. Depois de sucessivas gerações, a população evolui em direção à solução ótima [15].

Cada indivíduo é chamado de cromossomo e cada cromossomo possui genes, no caso desse trabalho os parâmetros dos osciladores. Há regras para seleção de parentes, regras de cruzamento para combinar genes de parentes e assim produzir uma nova geração, e também

há regras de mutação para alterar genes da nova geração randomicamente. O algoritmo pode ser observado no Algoritmo 2.1.

Algoritmo 2.1 Algoritmo Genético.

```
Inicia randomicamente a população
Faça enquanto uma condição de parada não foi obtida
   calcula a função de custo de cada indivíduo
   seleciona parentes para próxima geração baseado
      em sua função de custo
   obtém a nova geração utilizando as regras de cruzamento
   realiza a mutação dos genes da nova geração
   Troca a população atual pela nova população gerada
fim do loop
```

Também se pode utilizar filhos elites, que são os melhores indivíduos dos pais, sem alterá-los a fim de mantê-los na próxima geração.

A regra de seleção utilizada foi a regra da roleta. Ele seleciona indivíduos simulando uma roleta, na qual a área de cada seção da roleta é proporcional ao valor da função de custo desse indivíduo. Como pode ser visto na Figura 2.4. Nela foi sorteado um número $r \in [0, F)$ e no caso o valor sorteado estava na região B, dessa forma o individuo correspondente a essa região foi selecionado.

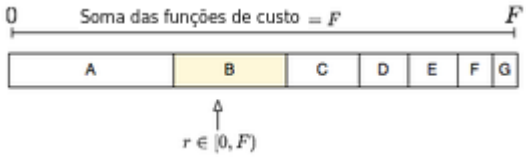


Figura 2.4: Regra da roleta do AG.

Outra regra de seleção usada foi a regra do torneio onde um número fixo de indivíduos é sorteado aleatoriamente e disputam um torneio. O vencedor do torneio, aquele que possui melhor valor da função de custo, é selecionado para ser um parente da próxima geração.

Para a regra de cruzamento utilizamos o cruzamento convexo. Nesse método os filhos dos parentes $\overline{x}$ e $\overline{y}$ são gerados pelas seguintes Equações:

$$\text{Filho1} = \alpha\overline{x} + (1-\alpha)\,\overline{y}, \tag{2.28}$$

$$\text{Filho2} = (1-\alpha)\,\overline{x} + \alpha\overline{y}, \tag{2.29}$$

onde α é uma número randômico entre 0 e 1. As Equações 2.28 e 2.29 são aplicadas em cada gene do cromossomo.

E para a regra de mutação foi utilizada a mutação Gaussiana que, com uma probabilidade p_{mut}, adiciona um valor aleatório de uma distribuição Gaussiana com média 0 ao gene.

2.3.1 Nondominated sorting Genetic Algorithm II (NSGA-II) - Um Algoritmo Genético Multiobjetivo

O NSGA-II é um algoritmo genético multiobjetivo proposto por Deb [14]. Esse algoritmo é utilizado caso os objetivos sejam conflitantes entre si. No caso em que os objetivos estão relacionados e possuem uma dependência direta, de forma que a maximização de um objetivo implica a maximização do outro também, então o problema pode ser resolvido com uma otimização mono objetivo. Para o caso dos objetivos serem conflitantes é utilizado o conceito de dominância de Pareto.

Um ponto $\overline{x} = (x_1, x_2, ..., x_n) \in \Re^n$ domina, no sentido de Pareto, um outro ponto $\overline{y} = (y_1, y_2, ..., y_n) \in \Re^n$ se, e somente se, $\forall i\ x_i \geq y_i$ e $\exists i\ x_i > y_i,\ 1 \leq i \leq n$.

Um exemplo para n=2 pode ser visto na Figura 2.5. Nela a solução B domina fortemente a solução C, pois é maior em todos os objetivos, e a solução A domina fracamente a solução C pois é maior apenas em um objetivo.

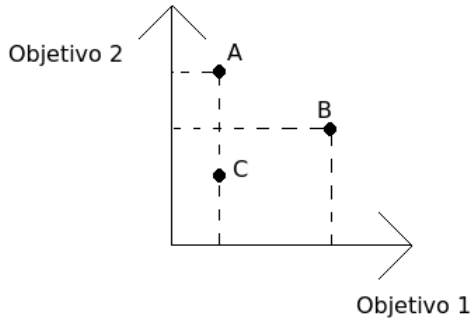


Figura 2.5: Exemplo de dominância de Pareto.

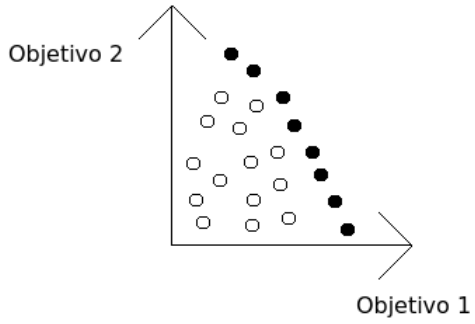


Figura 2.6: Exemplo de frente de Pareto.

Assim é utilizado o conceito de frente de Pareto, que é o conjunto de soluções que não dominam umas às outras, mas que dominam as outras soluções restantes. Um exemplo pode ser visto na Figura 2.6, onde os pontos preenchidos são da frente de Pareto.

Algoritmo 2.2 Algoritmo de ordenação de frentes.

```
FAST NONDOMINATED SORT(P)
para cada p ∈ P faça
    S_p = ∅
    n_p = 0
    para cada q ∈ P faça
        se p domina q então
         S_p = S_p ∪ {q}      /*Adiciona q a S_p*/
        se não se q domina p então
         n_p = n_p + 1 /*Incrementa o contador
         de dominância de p*/
        fim do se
    fim do loop para
    se n_p = 0 então /*p pertence a primeira frente*/
         rank(p) = 1/*o rank indica em qual frente a
         solução está*/
         F_1 = F_1 ∪ {p}/*Adiciona p ao conjunto da
         primeira frente*/
    fim do se
fim do loop para
i = 1
Enquanto F_i ≠ ∅ faça
    Q = ∅
    /*Usado para guardar os membros da próxima frente*/
    para cada p ∈ F_i faça
        para cada q ∈ S_p faça
            n_q = n_q − 1
            se n_q = 0 então
                rank(q) = i + 1
                Q = Q ∪ {q}
            fim do se
        fim do loop para
    fim do loop para
    i = i + 1
    F_i = Q
fim do loop enquanto
```

O NSGA-II ordena o conjunto de soluções nas diferentes frentes que existem, sendo a primeira frente a frente de Pareto, a segunda frente a frente de Pareto do conjunto de soluções sem os elementos da frente de Pareto, e assim sucessivamente. Deb [14] propôs um algoritmo de ordenação de frentes o qual ele chamou de *fast nondominated sort* que utiliza duas entidades:

1. Contador de dominância n_p, que é o número de elementos que dominam a solução p.
2. S_p, que é o conjunto de soluções que a solução p domina.

O algoritmo de ordenação de frentes pode ser visto no Algoritmo 2.2. Inicialmente o algoritmo calcula n_p e S_p para todo p. As soluções que possuem $n_p = 0$ não são dominadas por ninguém, portanto elas são da primeira frente $\mathcal{F}_1$ e é atribuído um rank (uma medida que indica em qual frente a solução está) de 1. Agora para cada solução p com $n_p = 0$, visitamos cada membro q do conjunto S_p e reduzimos por 1 o contador de dominância n_q. Dessa forma, para qualquer membro q se $n_q = 0$ ele é colocado em uma lista separada Q. Esses são os membros da segunda frente. Agora esse processo é repetido utilizando os membros da segunda frente ao invés da primeira, dessa forma a terceira frente é identificada e o algoritmo continua dessa forma até que todas as frentes sejam identificadas.

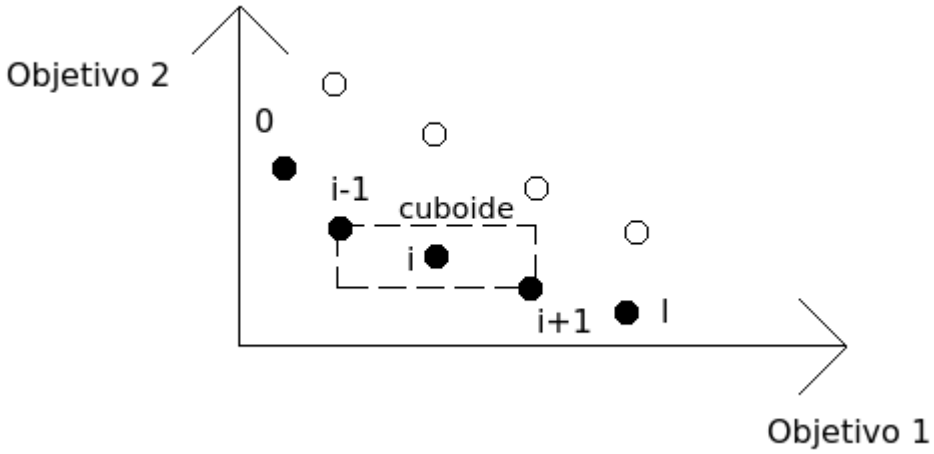


Figura 2.7: Exemplo de medição de *crowding-distance*.

Para manter um bom espalhamento das soluções no conjunto de soluções obtido o NSGA-II utiliza outros dois conceitos:

1. O *crowding-distance*. Ele é uma estimativa da densidade de soluções adjacentes a uma particular solução. Ele é calculado como a média da distância dos pontos dos vizinhos superior e inferior de um dado ponto ao longo de cada objetivo. Essa quantidade $i_{distancia}$ é chamada de *crowding-distance* e é uma estimativa do perímetro do cuboide formado usando os vizinhos mais próximos como vértices. Essa métrica só é calculada

Algoritmo 2.3 Algoritmo de medição de crowding distance.

```
CROWDING DISTANCE ASSIGNMENT(I)
N = |I|     /*Número de soluções de I*/
para cada i de 1 a N faça
    I[i]_distancia = 0   /*Inicializa a distancia*/
fim do loop para
para cada objetivo m faça
   I = ordena(I, m) /*Ordena usando cada
   valor do objetivo*/
   I[1]_distancia = I[N]_distancia = ∞\*é atribuído
   infinito para borda*\
   para i = 2 até (N − 1) faça
        I[i]_distancia =
I[i]_distancia + (I[i + 1].m − I[i − 1].m) / (f_m^max − f_m^min)
/*I[i].m refere-se ao valor da m-ésima função de
objetivo(custo) do individuo i*/
   fim do loop para
fim do loop para
```

para indivíduos de uma mesma frente. Para os elementos das bordas essa estimativa é infinita. Na Figura 2.7, um exemplo bidimensional de calculo de *crowding-distance* de um i-ésimo individuo de uma frente (marcada como círculos preenchidos) é a média do tamanho do lado do cuboide (marcado em pontilhado). O algoritmo que calcula o *crowding-distance* de um conjunto I pode ser visto no Algoritmo 2.3.

2. *Crowded-Comparison Operator* (Operador comparador de lotação): O *crowded-comparison operator* ($\prec_n$) guia a o processo de seleção de parentes do algoritmo genético em direção a uma frente de Pareto ótima uniforme e espalhada. Se cada individuo i de uma população possui rank i_{rank} e *crowding-distance* $i_{distancia}$, $\prec_n$ é definido como:

$$i \prec_n j \text{ se } (i_{rank} < j_{rank}) \quad \text{ou} \quad ((i_{rank} = j_{rank}) \text{ e } (i_{distancia} > j_{distancia})).$$

Isso significa que entre duas soluções com diferentes ranks, a que tiver menor rank é preferível. Caso as duas tenham o mesmo rank, então a solução preferível é aquela que está numa região com menos soluções adjacentes.

Portanto o NSGA-II é descrito da seguinte forma: Inicialmente uma população P_0 randômica é criada. A população é ordenada em suas frentes e o rank de cada solução é atribuído. Inicialmente, a seleção por torneio é utilizada e regras de cruzamento e mutação são utilizadas para gerar a população de filhos Q_0 com tamanho N. O eliticismo é introduzido ao

comparar a população atual com a população de parentes, por conta disso o processo é diferente para as próximas gerações. Portanto para as próximas gerações t o algoritmo se torna: Primeiro uma população combinada é formada $R_t = P_t \cup Q_t$. R_t é ordenada de acordo com as frentes que possui. Depois cada frente é ordenada de acordo com o *crowding-distance* $i_{distancia}$. Então os N primeiros membros de R_t são selecionados para serem a nova população P_{t+1}, ou seja os membros com menor rank, e dentre os com mesmo rank os com menor *crowding-distance*. Assim a nova população P_{t+1} é utilizada para as regras de seleção, cruzamento e mutação e é obtido Q_{t+1}, lembrando que $\prec_n$ é utilizado para guiar a regra de seleção. O algoritmo original proposto por Deb [14] utiliza como regra de seleção o torneio, mas o autor sugere que outros métodos podem ser utilizados. Os procedimentos do NSGA-II para uma época diferente da época inicial podem ser visto no Algoritmo 2.4.

Algoritmo 2.4 Algoritmo NSGA-II.

$R_t = P_t \cup Q_t$
/*Combina a população de parentes e filhos*/
$\mathcal{F}$ =**FAST NONDOMINATED SORT**(R_t)
/*$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, ...)$ todas as frentes de R_t*/
$P_{t+1} = \emptyset$ e $i = 1$
Enquanto $|P_{t+1}| + |\mathcal{F}_i| \leq N$ /*Até que a população seja completa*/
    CROWDING DISTANCE ASSIGNMENT$(\mathcal{F}_i)$ /*Calcula *crowding-distance**/
    $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$/*Inclui a i-ésima frente na população*/
    $i = i + 1$ /*checa a próxima frente para inclusão*/
fim do loop enquanto
Ordena$(\mathcal{F}_i, \prec_n)$ /*Ordena em ordem decrescente de acordo com $\prec_n$*/
$P_{t+1} = P_{t+1} \cup$
$\mathcal{F}_i[1 : (N - |P_{t+1}|)]$ /*Escolhe os primeiros$(N - |P_{t+1}|)$ elementos de $\mathcal{F}_i$*/
$Q_{t+1} =$
faz nova população(P_{t+1})/*Usa seleção, cruzamento e mutação para ter Q_{t+1}*/
$t = t + 1$ /*Incrementa o contador de épocas*/

2.4 PARTICLE SWARM OPTIMIZATION (PSO)

O PSO é um algoritmo proposto por Kennedy [16, 28] para a otimização de funções não lineares. Ele foi proposto a partir de simulações de modelos sociais simplificados e foi inspirado no comportamento de sociedades tais como bando de pássaros e cardume de peixes.

Basicamente, ele possui uma população de tamanho N, chamada de enxame (no inglês *swarm*), de possíveis candidatos a solução se movendo em um espaço de busca D-dimensional de acordo com regras simples. Cada partícula possui sua posição, $\overline{x}_i = (x_{i1}, x_{i2}, ..., x_{id})$, e velocidade, $\overline{v}_i = (v_{i1}, v_{i2}, ..., v_{id})$. Durante o processo de otimização cada partícula guarda a melhor posição onde esteve, $\overline{p}_i = (p_{i1}, p_{i2}, ..., p_{id})$, e a melhor posição conhecida até agora, $\overline{p}_{gi} = (p_{gi1}, p_{gi2}, ..., p_{gid})$, normalmente sendo a melhor posição global. Uma função de custo é usada para avaliar as posições das partículas. A atualização da posição e velocidade dá-se pelas seguintes Equações:

$$\overline{v}_i = \overline{v}_i + c_1\epsilon_1\left(\overline{p}_i - \overline{x}_i\right) + c_2\epsilon_2\left(\overline{p}_g - \overline{x}_i\right), \tag{2.30}$$

$$\overline{x}_i = \overline{x}_i + \overline{v}_i, \tag{2.31}$$

onde c_1 e c_2 são constantes aceleративas, ϵ_1 e ϵ_2 são números randômicos, gerados a cada passo, entre 0 e 1. O processo de atualização pode ser visto no Algoritmo 2.5. No algoritmo original a velocidade das partículas era limitada por um valor máximo v_{max}, para evitar o algoritmo de entrar em um estado instável onde a atualização da velocidade possa crescer rapidamente em determinada situação, tendendo a infinito.

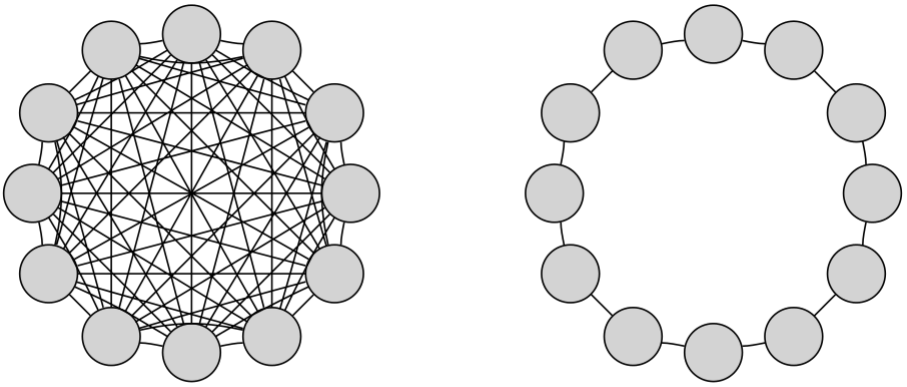
Algoritmo 2.5 Algoritmo do PSO original.

```
para cada passo t faça
 para cada partícula i do enxame faça
   atualiza posição x̄_i
   calcula a função de custo da partícula f(x̄_i)
   atualiza p̄_i e p̄_g
  fim do loop
fim do loop
```

Há variantes para o algoritmo PSO original [4] que serão mostradas a seguir.

2.4.1 Topologia de comunicação das partículas

Uma variante é na topologia de comunicação entre as partículas onde $\overline{p}_{gi}$ ao invés de ser a melhor posição global é a melhor posição que as partículas vizinhas acharam ou a que



(a) *Topologia gbest* (b) *Topologia lbest*

Figura 2.8: Topologia de comunicação no PSO.(Adaptado de [4].)

própria partícula achou. A primeira topologia é chamada de gbest e a segunda de lbest, como pode ser visto na Figura 2.8.

De acordo com Bratton [4] a vantagem da topologia lbest está na sua taxa de convergência menor do que a topologia gbest. Ter uma taxa de convergência mais rápida é benéfica quando o enxame achou o local ótimo global ou está próximo dele, mas não é desejado quando ocorre a convergência em um local sub-ótimo, um mínimo local não desejado. A topologia lbest permite uma exploração maior do espaço, em troca da taxa de convergência, e permite o algoritmo de fugir de certos mínimos locais. Dessa forma inicialmente a topologia gbest pode obter resultados melhores, mas há um longo prazo, a topologia lbest pode obter resultados melhores do que o gbest.

2.4.2 Peso de inércia e restrição

Para obter um melhor balanço entre a taxa de exploração global e a taxa de exploração local (do inglês *global exploration* e *local exploitation*) enquanto se evita a limitação da velocidade por v_{max}, um novo parâmetro foi introduzido no método. Foi verificado que para certos problemas um valor inadequado para v_{max} pode causar um desempenho ruim para o algoritmo, embora achar um valor apropriado fosse difícil e muitas vezes sendo utilizada a tentativa e erro [4].

O novo parâmetro é o peso de inércia w usado ao invés de v_{max}. Assim a equação de atualização da velocidade se torna:

$$\overline{v}_i = w\overline{v}_i + c_1\epsilon_1\left(\overline{p}_i - \overline{x}_i\right) + c_2\epsilon_2\left(\overline{p}_g - \overline{x}_i\right). \tag{2.32}$$

Ao ajustar esse parâmetro o enxame tem uma chance maior de se restringir na área contendo o melhor valor para a função de custo. Também é sugerido pelos autores do método de se utilizar um valor dinâmico para w, começando com um valor maior do que 1 para

encorajar a exploração no inicio e depois diminui-lo para um valor menor que 1 para focar a busca na melhor área achada na exploração.

Outro método tentando obter esse balanço entre a taxa de exploração global e a taxa de exploração local foi o método de constrição [4]. Nele um novo parâmetro foi utilizado, χ conhecido como fator de restrição. χ é obtido das constantes presentes das equações de atualização de velocidade como visto na Equação abaixo:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \ \varphi = c_1 + c_2. \tag{2.33}$$

Foi descoberto que para $\varphi < 4$ o enxame iria evoluir lentamente em forma de espiral em direção e em volta da melhor solução do espaço de busca, mas sem garantia de convergência. Enquanto para $\varphi > 4$ a convergência seria rápida e garantida [4]. A equação de atualização da velocidade se torna:

$$\overline{v}_i = \chi \left(\overline{v}_i + c_1 \epsilon_1 \left(\overline{p}_i - \overline{x}_i \right) + c_2 \epsilon_2 \left(\overline{p}_g - \overline{x}_i \right) \right). \tag{2.34}$$

Os efeitos da restrição são similares ao do peso de inércia. De fato, o método da restrição é um caso especial do método de inércia em que os valores dos parâmetros foram obtidos analiticamente.

2.4.3 Questões referentes ao limite do espaço de busca

Uma das questões referentes ao algoritmo está no fato do que fazer quando uma partícula está fora da região de busca. Bratton [4] sugere que para evitar qualquer efeito na desempenho do algoritmo, o método mais simples é deixar que as partículas cruzem a borda e deixar a atualização de velocidade e de posição ocorrerem normalmente, mas a atualização da função de custo não é feita para evitar que uma posição fora do espaço de busca possa se tornar a melhor posição que a partícula conheça $\overline{p}_g$. Em alguns casos raros também é possível que a partícula desenvolva velocidades altas estando próxima da borda da região de busca, para isso pode-se utilizar um fator v_{max} para evitar que a partícula vá para regiões muito distantes da região de busca.

2.5 APRENDIZADO POR REFORÇO

Como dito por Sutton [51] o aprendizado por reforço é um método onde se aprende por iterações com o intuito de obter determinado objetivo. Nele há um agente que aprende a tomar decisões para interagir com determinado ambiente. A aprendizagem do agente é de forma ativa, por ações diretas no ambiente e a observação da forma como esse responde,

diferentemente das técnicas de aprendizagem supervisionadas.

O agente e o ambiente interagem em passos discretos de tempo, $t = 0, 1, 2, 3....$ A cada passo t, o agente recebe uma representação do estado do ambiente, $s_t \in S$, onde S é o conjunto de possíveis estados, e com base nesse estado escolhe uma ação, $a_t \in A(s_t)$, onde $A(s_t)$ é o conjunto de ações disponíveis no estado s_t. Um passo de tempo depois, como consequência da ação utilizada o agente recebe uma recompensa numérica, $r_{t+1} \in \Re$, e se encontra em um novo estado s_{t+1}. Essa interação pode ser vista na Figura 2.9.

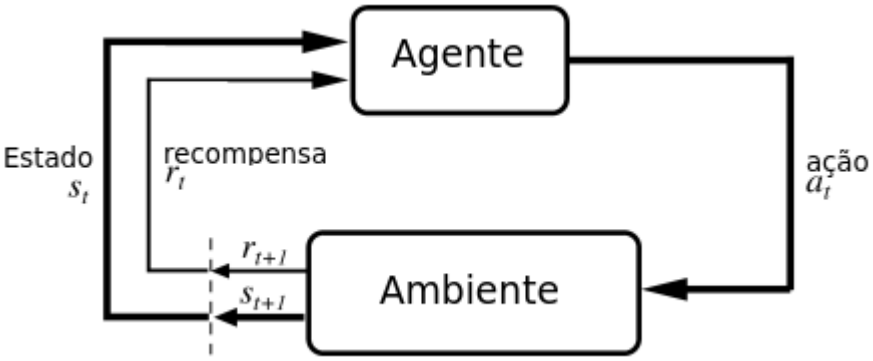


Figura 2.9: Interação entre agente e ambiente.

A cada passo o agente usa um mapeamento de estados para a probabilidade de se selecionar determinada ação, esse mapeamento é chamado de política π_t, onde $\pi_t(s, a)$ é a probabilidade de $a_t = a$ se $s_t = s$.

O objetivo do aprendizado por reforço é maximizar o retorno esperado, R_t, que é definido como:

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}, \tag{2.35}$$

onde γ é chamado de taxa de desconto, $0 \leq \gamma \leq 1$. A taxa de desconto representa o peso de futuras recompensas, ou seja, uma recompensa no instante k passos de tempo no futuro vale γ^{k-1} vezes o que ela valeria no estado imediato. E T é o último passo de tempo. Quando T é um número finito, significa que há um estado terminal seguido de uma reinicialização para um estado inicial. Quando isso ocorre temos um episódio e quando uma tarefa é dessa forma chamamos a tarefa de tarefa episódica. Para tarefas periódicas, algumas vezes é importante distinguir o conjunto de estado não terminais, S, com o conjunto de estado mais os estados terminais, S^+. Quando $T = \infty$, significa que a tarefa não possui um estado terminal e nem episódios, então chamamos chamamos tarefas dessa forma de tarefas continuas. Para o caso de tarefas contínuas $0 \leq \gamma < 1$, pois caso $\gamma = 1$ e $T = \infty$ podemos ter um valor infinito de R_t.

2.5.1 Processos de decisão Markovianos

Problemas de aprendizado por reforço são, em sua maioria, tratados como processos de decisão Markovianos (MDP, do inglês *Markov Decision Process*). Se os espaço de estados e espaço de ações são finitos, eles são chamados de processos de decisão Markovianos finitos, MDP finito.

Um MDP satisfaz a propriedade de Markov, que para um problema de aprendizado por reforço pode ser definido como se segue. Para deixar a matemática simples, assumimos um número finito de estados e valores de recompensa, dessa forma podemos trabalhar em termos de somas e probabilidade ao invés de integrais e densidade de probabilidade, mas os argumentos usados podem ser estendidos para incluírem estados contínuos e recompensas contínuas. Considerando um ambiente geral, a resposta de um ambiente causal, em geral, depende de tudo o que ocorreu anteriormente. Ou seja, a dinâmica só pode ser definido ao se especificar a distribuição de probabilidade completa:

$$Pr\left\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, ..., r_1, s_0, a_0\right\}, \tag{2.36}$$

para todo s', r, e todos os possíveis valores dos eventos passados: $s_t, a_t, r_t, ..., r_1, s_0, a_0$. Para o caso em que o ambiente tenha a propriedade de Markov, então a resposta do ambiente no momento $t+1$ só dependerá do estado e ação do momento passado t, no caso a dinâmica pode ser definida como:

$$Pr\left\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\right\}, \tag{2.37}$$

para todo s', r, s_t e a_t. Dessa forma, podemos prever o que ocorrerá no próximo estado apenas com as informações do estado e ação atuais, não precisando do histórico de estados, ações e recompensas.

A propriedade de Markov é importante para o aprendizado por reforço, pois decisões e valores são assumidas por serem função apenas do estado atual [51].

Definimos, para um MDP finito, para qualquer estado e ação, s e a, a probabilidade do possível próximo estado, s', como

$$\mathcal{P}^a_{ss'} = Pr\left\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\right\}. \tag{2.38}$$

Essas quantidades são chamadas de probabilidades de transição. De forma análoga, para qualquer estado e ação, s e a, com próximo estado qualquer, s', o valor esperado da próxima recompensa é dado por

$$\mathcal{R}^a_{ss'} = E\left\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\right\}. \tag{2.39}$$

2.5.2 Função valor

Quase todos algoritmos de aprendizado por reforço são baseados na estimação de funções valor, funções de estado (ou par estado-ação) que estimam o quão bom é para o agente estar em um estado (ou quão bom é realizar determinada ação em um estado) [51]. O termo quão bom é usado referente ao retorno esperado. Funções valor são definidas com respeito a políticas particulares. A função valor de estado para uma determinada política, $V^{\pi}(s)$, é o valor esperado de retorno quando começamos em um estado s seguindo uma política π após ele. Para MDPs, $V^{\pi}(s)$ pode ser definida como:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right\}. \tag{2.40}$$

Analogamente, a função ação-valor para uma política π, $Q^{\pi}(s,a)$, é o valor esperado de retorno começando do estado s, tomando ação a e seguindo a política π daí em diante. Para MDPs, $Q^{\pi}(s,a)$ é definida como

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a \right\}. \tag{2.41}$$

Uma propriedade interessante da função valor é que ela satisfaz uma relação recursiva particular, que será usada para o aprendizado por reforço. Essa propriedade pode ser visto abaixo:

$$\begin{aligned}
V^{\pi}(s) &= E_{\pi}\{R_t | s_t = s\} \\
&= E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right\} \\
&= E_{\pi}\left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s \right\} \\
&= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_{t+1} = s' \right\} \right] \\
&= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s') \right],
\end{aligned} \tag{2.42}$$

onde está implícito que a ação a é uma ação do conjunto $A(s)$ e o próximo estado s' é um estado do conjunto S, ou S^{+} em caso de tarefas periódicas.

A Equação 2.42 é chamada de equação de Bellman para V^{π}. Ela expressa a relação entre o valor de um estado e o valor dos estados subsequentes.

Também podemos definir a politica ótima, π^*, que é a política que obtém o máximo do retorno esperado. Dessa forma, temos a função valor de estado ótima, V^*, como:

$$V^*(s) = \max_{\pi} V^{\pi}(s), \tag{2.43}$$

para todo $s \in S$.

Analogamente podemos definir a função ação-valor ótima, Q^*, como:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a), \tag{2.44}$$

para todo $s \in S$ e $a \in A(s)$. Também podemos escrever Q^* em termos de V^* como segue:

$$Q^*(s,a) = E\left\{r_{t+1} + \gamma V^*(s_{t+1}) \,|s_t = s, a_t = a\right\}. \tag{2.45}$$

Como V^* é uma função valor para uma política, no caso a política ótima, ela deve satisfazer a equação de Bellman dada pela Equação 2.42. Mas como V^* é a função valor ótima a sua condição de consistência pode ser escrita numa condição de consistência especial, sem se especificar alguma política. Essa equação é chamada de equação de Bellman ótima que pode ser vista a seguir:

$$\begin{aligned}
V^*(s) &= \max_{a \in A(s)} Q^{\pi^*}(s,a) \\
&= \max_{a \in A(s)} E_{\pi^*}\left\{R_t |\, s_t = s, a_t = a\right\} \\
&= \max_{a \in A(s)} E_{\pi^*}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a\right\} \\
&= \max_{a \in A(s)} E_{\pi^*}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \,\middle|\, s_t = s, a_t = a\right\} \\
&= \max_{a \in A(s)} E\left\{r_{t+1} + \gamma V^*(s_{t+1}) \,|s_t = s, a_t, a\right\} \qquad (2.46) \\
&= \max_{a \in A(s)} \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right]. \qquad (2.47)
\end{aligned}$$

As equações 2.47 e 2.46 são as duas formas da equação de Bellman ótima para V^*. Analogamente, a equação de Bellman ótima para Q^* é dada como se segue:

$$\begin{aligned}
Q^*(s,a) &= E\left\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \,\middle|\, s_t = s, a_t = a\right\} \\
&= \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')\right]. \qquad (2.48)
\end{aligned}$$

2.5.3 Programação dinâmica

O termo programação dinâmica refere-se a uma coleção de algoritmos que podem ser usados para computar a política ótima dado um modelo perfeito do ambiente como um MDP [51]. Ela evita a solução explícita das equações de Bellman utilizando métodos iterativos, basicamente ela transforma as equações de Bellman em regras de atualização para melhorar aproximações do valor desejado da função valor.

Inicialmente temos a avaliação de política, na qual inicialmente atribui-se zero aos valores da função de valor, e em seguida são feitas iterações da equação de Bellman até a convergência. O algoritmo pode ser visto no Algoritmo 2.6.

Algoritmo 2.6 Avaliação de política programação dinâmica.

AVALIAÇÃO DE POLÍTICA(π)
Coloque π a política a ser avaliada
Inicialize $V(s) = 0$, para todo $s \in S$
Repita
    $\Delta = 0$
    Para cada $s \in S$:
        $v = V(s)$
        $V(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
        $\Delta = \max(\Delta, |v - V(s)|)$
Até que $\Delta < \theta$ (um valor pequeno positiva)
Saída $V \approx V^\pi$

Após calcularmos a função valor de uma política, devemos saber se devemos mudar a política. Para isso calculamos, a cada estado, a política gulosa, que é a política que escolhe a próxima ação que leva a um valor máximo de $V(s)$ subsequente, de acordo com a função valor recém-calculada. Se for diferente da política atual, modificá-la. Isso é chamado de melhoramento de política.

As duas ideias anteriores levam a um método sistemático de melhoramento de política, chamado de iteração de política. Ele avalia $V(s)$ na política atual, verifica se $V(s)$ pode ser melhorada por uma política gulosa e repete a operação até convergência. O algoritmo pode ser visto no Algoritmo 2.7.

Uma das desvantagens da programação dinâmica é o custo computacional para realizar a iteração de políticas. Outra desvantagem é que ela precisa do conhecimento do modelo completo do ambiente, a probabilidades de transição, $\mathcal{P}$, e do modelo de recompensa, $\mathcal{R}$. Usualmente, em práticas de aprendizado por reforço essas características não estão disponíveis, sendo necessários outros algoritmos que não necessitem desses dados, como o método de Monte Carlo e aprendizagem por diferença temporal. Mas embora ocorram essas desvantagens, a ideia de iteração de política é utilizada pelos outros algoritmos.

Algoritmo 2.7 Iteração de política programação dinâmica.

1. INICIALIZAÇÃO

   $V(s) \in \Re$ e $\pi(s) \in A(s)$ arbitrário para todo $s \in S$

2. AVALIAÇÃO DE POLÍTICA

   Repita

   $\Delta = 0$

   Para cada $s \in S$:

   $v = V(s)$

   $V(s) = \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

   $\Delta = \max(\Delta, |v - V(s)|)$

   Até que $\Delta < \theta$ (um valor pequeno positiva)

3. MELHORAMENTO DE POLÍTICA

   $politica - estavel \leftarrow Verdadeiro$

   Para cada $s \in S$:

   $b = \pi(s)$

   $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

   Se $b \neq \pi(s)$, então $politica - estavel \leftarrow falso$

   Se $politica - estavel = Verdadeiro$, então pare;

   Caso contrário vá para 2.

2.5.4 Método de Monte Carlo

O método de Monte Carlo (MC) baseia-se em obter uma média das amostras de retorno obtidos por experiência. Para tanto, para garantir que retornos bem definidos estejam disponíveis, nós iremos definir os métodos de MC apenas para tarefas periódicas. Apenas após um episodio completo ocorre a atualização da função valor e da política. Dessa forma o método de MC é iterativo episodio por episodio e não passo a passo.

Da mesma forma que a programação dinâmica, o método de MC realiza a avaliação da política, mas armazena o valor do retorno obtido e utiliza a média dos retornos obtidos no estado s, como aproximação para $V(s)$. Há dois tipos de métodos de MC, o método MC toda visita, em que a estimativa de $V(s)$ é obtida pela média do valor de retorno de todas as visitas a s em um episodio. E há o MC primeira visita, que utiliza apenas o dado de retorno da primeira visita. Nos dois métodos se um número de visitas (ou primeira visita) infinitas ocorre a função valor estimada irá convergir para $V^\pi(s)$. A avaliação de política para o MC primeira visita pode ser vista no Algoritmo 2.8.

Quando não há um modelo, a função $Q(s,a)$ normalmente é mais importante que a função $V(s)$, pois não se conhece a priori como as ações resultam em mudança de estados. Basicamente, a técnica é a mesma: fazer uma média dos retornos obtidos quando se visitou o estado s e se tomou a ação a. O problema é garantir que todos os estados e ações relevantes

sejam visitados, pois em uma interação real com o ambiente o agente começa rapidamente a melhorar sua política e tomar ações específicas. Com isso, é cada vez menor a possibilidade de explorar novas ações ou visitar novos estados, que pode levar a um resultado melhor. Esse é o problema da exploração global e a exploração local (do inglês *global exploration* e *local exploitation*) no aprendizado por reforço.

Algoritmo 2.8 Avaliação de política MC.

Inicializa:
  $\pi =$política a ser avaliada
  $V =$Uma função estado valor
  $Retornos\,(s) =$uma lista vazia, para todo $s \in S$
Repita para sempre:
  (a) Gera um episódio usando π
  (b) Para cada estado s aparecendo no episódio:
    $R =$retorno seguido da primeira visita a s
    Anexa R a $Retornos\,(s)$
    $V\,(s) =$ média $(Retornos\,(s))$

Algoritmo 2.9 Controle do MC.

Inicializa, para todo $s \in S, a \in A\,(s)$:
  $Q\,(s,a) =$arbitrário
  $Retornos\,(s,a) =$uma lista vazia
  $\pi\,(s) =$política arbitraria ou política ϵ-gulosa
Repita sempre:
  (a) Gera um episódio usando hipótese do início
   explorador e π ou Gera um episódio usando π
  (b) Para cada par s,a aparecendo no episódio:
    $R =$retorno seguido a primeira ocorrência de s,a
    Anexa R a $Retornos\,(s,a)$
    $Q\,(s,a) =$ média $(Retornos\,(s,a))$
  (c) Para cada s no episódio:
    Se $\pi\,(s)$ é uma política arbitraria
      $\pi\,(s) =$ arg $\max_a Q\,(s,a)$
    Se $\pi\,(s)$ é uma política ϵ-gulosa
      $a^* =$ arg $\max_a Q\,(s,a)$
      Para todo $a \in A\,(s)$:

$$\pi\,(s,a) = \begin{cases} 1 - \epsilon + \epsilon / |A\,(s)| & \text{se } a = a^* \\ \epsilon / |A\,(s)| & \text{se } a \neq a^* \end{cases}$$

Duas soluções típicas são:

- Hipótese do início explorador: Todo par (s,a) tem probabilidade não-nula de ser o primeiro de um episódio. Hipótese pouco razoável.
- Políticas parcialmente estocásticas (ϵ-gulosa) :Com probabilidade $1-\epsilon$ escolhe a melhor ação aparente (gulosa). Com probabilidade ϵ escolhe uma ação aleatória (exploração). Normalmente ϵ tem valor baixo, e pode diminuir com o tempo (ϵ nulo torna a política estritamente gulosa).

Pode-se mostrar que a iteração de política também funciona para políticas parcialmente estocásticas (ϵ-gulosa) como visto em [51].

Dessa forma o MC pode ser usado em controle, isto é, para aproximar políticas ótimas, como segue o Algoritmo 2.9.

No MC descrito até agora temos procurado melhorar uma política a partir da experiência de seguir esta mesma política. Estes são chamados métodos *on-policy*. Também há o método *off-policy* que melhora uma política (por exemplo, gulosa) com base na experiência adquirida por uma política diferente (por exemplo, ϵ-gulosa).

Para isso basta ajustar os pesos das estimativas de acordo com as diferentes probabilidades de ocorrência dos eventos de acordo com cada política. Seja π a política a melhorar e π' a política usada para gerar episódios, com a condição $\pi(s,a) > 0 \rightarrow \pi'(s,a) > 0$, R_i o retorno acumulado após a primeira visita ao estado i, no i-ésimo episódio em que há visita a s, p_i a probabilidade de esta sequência de eventos ocorrer sob a política π, e p'_i a probabilidade sob a política π', n_s o número total de episódios com visitas a s, então a estimativa desejada do MC é:

$$V(s) = \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)} R_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)}}. \tag{2.49}$$

Sendo que a quantidade $\frac{p_i(s)}{p'_i(s)}$ depende apenas das probabilidades de transição do ambiente (iguais para ambos) e das políticas, pois considerando $T_i(s)$ o tempo de finalização do i-ésimo episódio envolvendo s, obtemos que:

$$\frac{p_i(s)}{p'_i(s)} = \frac{\prod_{k=t}^{T_i(s)-1} \pi(s_k,a_k) \mathcal{P}^{a_k}_{s_k s_{k+1}}}{\prod_{k=t}^{T_i(s)-1} \pi'(s_k,a_k) \mathcal{P}^{a_k}_{s_k s_{k+1}}} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k,a_k)}{\pi'(s_k,a_k)}. \tag{2.50}$$

Dessa forma o algoritmo de controle para um MC *off-policy* pode ser visto no Algoritmo 2.10.

Algumas limitações do método de MC são que dependendo da aplicação o método pode levar a episódios muito longos. Em outras palavras, é difícil dividir a atuação do agente em episódios. E além disso, uma ação exploratória ruim pode contaminar todo o episódio, tornando o aprendizado muito ruidoso.

Algoritmo 2.10 Controle do MC *off-policy*.

Inicializa, para todo $s \in S$, $a \in A(s)$:
$Q(s,a) =$arbitrário
$N(s,a) = 0$; Numerador e
$D(s,a) = 0$; Denominador de $Q(s,a)$
$\pi(s) =$política arbitraria determinística
Repita sempre:
(a) Seleciona a politica π' e a usa para gerar o episódio:
$s_0, a_0, r_1, s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T, s_T$
(b) $\tau =$último tempo em que $a_\tau \neq \pi(s_\tau)$
(c) Para cada par s, a aparecendo no episódio no tempo τ ou depois:
$t =$tempo da primeira ocorrência de s, a, tal que $t \geq \tau$
$w = \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)}$
$N(s,a) = N(s,a) + wR_t$
$D(s,a) = D(s,a) + w$
$Q(s,a) = \frac{N(s,a)}{D(s,a)}$
(d) Para cada $s \in S$:
$\pi(s) = \arg\max_a Q(s,a)$

2.5.5 Aprendizagem por diferença temporal

Aprendizagem por diferença temporal (TD, do inglês *Temporal-Difference*) é uma combinação das ideias de MC e programação dinâmica. Como o método de MC, métodos TD permitem a aprendizagem por experiência sem um modelo do ambiente. Como programação dinâmica, métodos TD possuem a atualização da estimativa baseada nas outras estimativas aprendidas, sem esperar uma saída final como no método de MC.

Como visto na Equação 2.42 podemos escrever $V^\pi(s)$ como:

$$V^\pi(s) = E_\pi \left\{ r_{t+1} + \gamma V^\pi(s_{t+1}) \middle| s_t = s \right\}. \tag{2.51}$$

Essa equação sugere um estimativa de valor da seguinte forma:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right], \tag{2.52}$$

onde α é uma constante de tamanho de passo e $r_{t+1} + \gamma V(s_{t+1})$ é chamado de atualização TD. Dessa forma a avaliação de política é dada pela equação acima.

Para o TD há dois modos de controle usados, o SARSA e o Q-Learning. Também há um método que possui uma estrutura de memória separada que explicitamente representa a política independente da função valor. Esse método é chamado de Ator-Crítico.

2.5.5.1 SARSA: Controle TD *on-policy*

Da mesma forma que estimamos $V(s)$ a partir da recompensa atual apenas, podemos procurar estimar $Q(s,a)$ de acordo com

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right]. \tag{2.53}$$

Para isso precisamos, a partir de um dado estado atual, tomar uma ação, verificar a recompensa e o novo estado, e escolher a ação no novo estado pela mesma política. Ele então precisa dos valores de S,A,R,S,A, dai seu nome. O algoritmo pode ser visto no Algoritmo 2.11.

Algoritmo 2.11 SARSA.

```
Inicializa, Q(s,a), ∀s ∈ S, a ∈ A(s), arbitrário
Repete (para cada episodio):
  Inicializa s
  Escolhe a de s usando uma política derivada de Q
  Repete (para cada passo do episódio)
    Toma ação a, observa r, s'
    Escolhe a' de s' usando uma política derivada de Q
    Q(s,a) = Q(s,a) + α[r + γQ(s',a') − Q(s,a)]
    s = s'; a = a';
  até s seja terminal
```

Algoritmo 2.12 Q-Learning.

```
Inicializa, Q(s,a), ∀s ∈ S, a ∈ A(s), arbitrário
Repete (para cada episodio):
  Inicializa s
  Repete (para cada passo do episódio)
    Escolhe a de s usando uma política derivada de Q
    Toma ação a, observa r, s'
    Q(s,a) = Q(s,a) + α[r + γ max_a' Q(s',a') − Q(s,a)]
    s = s';
  até s seja terminal
```

2.5.5.2 Q-Learning: Controle TD *off-policy*

A estimação do Q-Learning para a função $Q(s,a)$ pode ser vista na equação abaixo:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right]. \tag{2.54}$$

Nesse caso, a função $Q(s,a)$ aprendida aproxima diretamente a $Q^*(s,a)$ independentemente da política sendo seguida, pois o Q-Learning estima diretamente o melhor valor Q de cada estado. O algoritmo pode ser visto no Algoritmo 2.12.

2.5.5.3 Métodos Ator-Crítico

O método do ator-crítico possui uma estrutura de memória separada que explicitamente representa a política independente da função valor, como dito anteriormente. A estrutura da política é conhecida como ator, pois seleciona ações, e a função valor estimada é conhecida como crítico, pois critica a ação tomada pelo ator como pode ser visto na Figura 2.10. A aprendizagem é sempre *on-policy*, o crítico precisa aprender sobre e criticar a política que está sendo seguida.

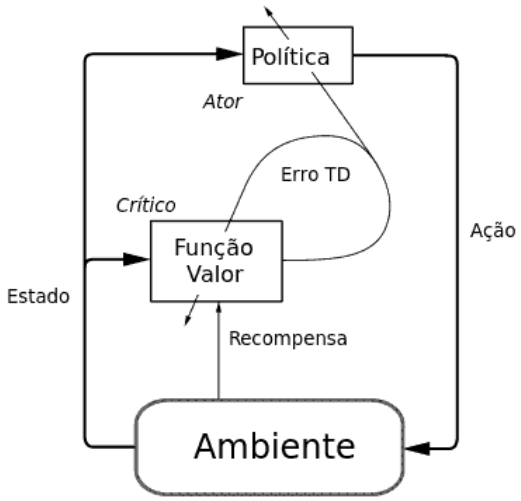


Figura 2.10: Arquitetura do método Ator-Crítico.

Normalmente, o crítico é uma função estado valor $V(s)$. Após cada ação, o crítico avalia o novo estado para determinar se foi melhor ou pior do que o esperado. Essa avaliação é dada pelo erro TD:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t). \tag{2.55}$$

Esse erro pode ser usado para avaliar a ação, a_t, usada no estado s_t. Se o erro TD é positivo, ele sugere que uma tendência para escolher a_t deve ser fortalecida, e se o erro TD é negativo, sugere que essa tendência deve ser enfraquecida. Supondo uma política gerada pelo método Gibbs softmax:

$$\pi_t(s,a) = Pr\{a_t = a | s_t = s\} = \frac{e^{p(s,a)}}{\sum_b e^{p(s,b)}}, \tag{2.56}$$

onde $p(s,a)$ são valores no tempo t dos parâmetros da política do ator. Então a ideia descrita acima de fortalecer ou enfraquecer tendencias pode ser dada pela equação abaixo.

$$p(s_t,a_t) \leftarrow p(s_t,a_t) + \beta\delta_t, \tag{2.57}$$

onde β é um parâmetro positivo de tamanho de passo. Uma outra forma de atualização possível pode ser dada pela equação abaixo.

$$p\left(s_t, a_t\right) \leftarrow p\left(s_t, a_t\right) + \beta\delta_t\left[1 - \pi_t\left(s_t, a_t\right)\right]. \tag{2.58}$$

Como dito por Sutton [51], duas vantagens do método ator-crítico são:

- Ele requer esforço computacional mínimo para escolher ações. Por exemplo, considerando o caso de um número infinito de ações, ações continuas por exemplo, os outros métodos que aprendem somente os valores das ações devem procurar no conjunto infinito uma ação para ser escolhida. Se a política é explicita, esse esforço computacional extensivo é evitado.
- Ele pode aprender uma política estocástica explicita, ou seja, ele pode aprender probabilidades ótimas de escolher varias ações. Essa característica se torna útil em casos competitivos e não Markovianos.

2.5.6 Rastro de elegibilidade

O uso do rastro de elegibilidade se dá no método TD(λ), onde λ se refere ao uso dos rastros de elegibilidade. Ele é uma ponte entre os métodos MC e TD, unindo os dois métodos, onde em um extremo temos o método de MC e no outro o método de TD. No meio temos métodos que normalmente são melhores que os dois extremos. Ele também pode ser visto como um histórico temporário da ocorrência de eventos, marcando estados elegíveis para as mudanças de aprendizagem. Dessa forma quando um erro TD ocorre, somente os estados ou ações elegíveis são creditados ou culpados pelo erro.

No método de MC, a estimativa $V_t\left(s_t\right)$ de $V^{\pi}\left(s_t\right)$ é atualizada em direção do retorno completo:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{T-t-1} r_T. \tag{2.59}$$

Podemos dizer que o método de MC realiza um *backup* de cada estado baseado na sequencia de recompensas. Chamemos então essa quantidade de alvo de *backup*. Podemos dizer que o método TD realiza o *backup* de um passo:

$$R_t^{(1)} = r_{t+1} + \gamma V_t\left(s_{t+1}\right). \tag{2.60}$$

Dessa forma podemos ter o *backup* de n-passos, chamado de retorno de n-passos, dado por:

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + ... + \gamma^{n-1} r_{t+n} + \gamma^n V_t\left(s_{t+n}\right). \tag{2.61}$$

Resultando na atualização de $V_t\left(s_t\right)$:

$$V_t\left(s_t\right) \leftarrow V_t\left(s_t\right) + \alpha\left[R_t^{(n)} - V_t\left(s_t\right)\right]. \tag{2.62}$$

O método TD(λ) utiliza backup dado pela média de n-passos chamada retorno-λ, onde $0 \leq \lambda \leq 1$, dado por:

$$R_t^{(\lambda)} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}. \tag{2.63}$$

Note que após um estado terminal ser atingido, todos os retornos de n-passos subsequentes são iguais a R_t, pois a recompensa de um estado terminal é zero. Dessa forma, separando obtemos:

$$R_t^{(\lambda)} = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t. \tag{2.64}$$

Note que se $\lambda = 1$ obtemos $R_t^{(\lambda)} = R_t$, que é o método de MC. E se $\lambda = 0$, então é o mesmo que o método TD.

Assim obtemos a atualização de $V_t(s_t)$:

$$V_t(s_t) \leftarrow V_t(s_t) + \alpha \left[R_t^{(\lambda)} - V_t(s_t) \right]. \tag{2.65}$$

Esse método de se utilizar o retorno-λ, é chamada de vista para frente de TD(λ) e é não causal, logo não é possível implementá-la. A forma que ela é implementável é chamada de vista para trás de TD(λ). Ela utiliza uma variável de memória associada a cada estada, chamada de rastro de elegibilidade. O rastro de elegibilidade para um estado s num tempo t é denotado por $e_t(s) \in \Re^+$. A cada passo, todos os rastros de elegibilidade decaem exponencialmente a uma taxa de $\lambda\gamma$, e os rastros de elegibilidade do estado visitado nesse passo são incrementado de 1, ou seja:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{se } s \neq s_t \\ \gamma\lambda e_{t-1}(s) + 1 & \text{se } s = s_t \end{cases}. \tag{2.66}$$

Nesse caso, nos referimos a λ como parâmetro de decaimento de rastro. Esse tipo de rastro de elegibilidade é chamado de rastros acumulados, outro tipo de rastro é o chamado rastro de substituição dado por:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{se } s \neq s_t \\ 1 & \text{se } s = s_t \end{cases}. \tag{2.67}$$

Os rastros registram os estados que foram recentemente visitados, onde o termo "recente" é definido em termos de $\lambda\gamma$. O rastro é um indicativo do grau que um estado é elegível para ser submetido por mudanças pela aprendizagem. Como a avaliação é dada em termos do erro TD, δ_t, obtemos que a atualização de $V_t(s)$ é dada por:

$$V_t(s) \leftarrow V_t(s) + \alpha\delta_t e_t(s), \ \forall s \in S. \tag{2.68}$$

Algoritmo 2.13 TD(λ).

Inicializa, $V(s)$ arbitrário e $e(s) = 0, \forall s$
Repete (para cada episodio):
  Inicializa s
  Repete (para cada passo do episódio)
    $a =$ ação dada por π para s
    Toma ação a, observa r, s'
    $\delta = r + \gamma V(s') - V(s)$
    $e(s) = e(s) + 1$
    Para todo s:
      $V(s) = V(s) + \alpha\delta e(s)$
      $e(s) = \gamma\lambda e(s)$
    $s = s'$
  até s seja terminal

E o algoritmo de atualização pode ser visto no Algoritmo 2.13.

Sutton [51] demonstra que as duas formas, a vista para frente e a vista para trás são equivalentes.

Para o método SARSA(λ) a atualização é dada por:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a), \ \forall s, a, \tag{2.69}$$

onde

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t), \tag{2.70}$$

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{se } s = s_t \text{ e } a = a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{caso contrário} \end{cases} \forall s, a. \tag{2.71}$$

Já no caso do Q-Learning uma das formas de se utilizar o rastro de elegibilidade é utilizá-los como no SARSA(λ), exceto quando uma ação de exploração é tomada, uma ação que não maximiza $Q_t(s,a)$, nesse caso os rastros de elegibilidade são zerados. Isso resulta em:

$$e_t(s,a) = \mathcal{I}_{ss_t}\mathcal{I}_{aa_t} + \begin{cases} \gamma\lambda e_{t-1}(s,a) & \text{se } Q_{t-1}(s_t,a_t) = \max_a Q_{t-1}(s_t,a), \\ 0 & \text{caso contrário} \end{cases}, \tag{2.72}$$

$$\mathcal{I}_{xy} = \begin{cases} 1 & \text{se } x = y \\ 0 & \text{se } x \neq y \end{cases}, \tag{2.73}$$

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a), \ \forall s, a, \tag{2.74}$$

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t). \tag{2.75}$$

E no método ator-crítico com rastros de elegibilidade a atualização do crítico pode ser dada pelas Equações 2.66 e 2.68. Já no caso em que a atualização do ator sem os rastros seja dada pela equação 2.57 então a atualização se torna:

$$p_{t+1}(s,a) = p_t(s,a) + \alpha\delta_t e_t(s,a),\ \forall s,a, \tag{2.76}$$

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{se } s = s_t \text{ e } a = a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{caso contrário} \end{cases} \forall s,a. \tag{2.77}$$

Caso a atualização original seja dada pela equação 2.58 então a atualização se torna:

$$p_{t+1}(s,a) = p_t(s,a) + \alpha\delta_t e_t(s,a),\ \forall s,a, \tag{2.78}$$

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 - \pi_t(s_t,a_t) & \text{se } s = s_t \text{ e } a = a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{caso contrário} \end{cases} \forall s,a. \tag{2.79}$$

2.5.7 Aprendizado por reforço para estados contínuos

Nos métodos descritos acima, foi assumido que os estados, ou par estado-ação, têm um valor tabelado a ser atualizado. Em muitos casos reais, utilizar esse método não é possível, pois para o caso em que há muitos estados, ou par estado-ação, o custo de armazenamento das tabelas pode ser muito elevado em termos da memória computacional requerida, e pode tornar a convergência dos algoritmos extremamente lenta.

Para isso são utilizados aproximadores de funções para as funções V e Q, como redes neurais, redes de base radial, logica fuzzy, etc. Dessa forma, a função V_θ pode ser descrita em termos de $\overline{\theta}_t$, onde $\overline{\theta}_t = (\theta_t(1), \theta_t(2), ..., \theta_t(n))^T$ são os parâmetros ajustáveis do aproximador. Para atualizar os valores dos parâmetros podemos utilizar o método do gradiente descendente, que pode utilizar o erro TD por exemplo para atualizar $\overline{\theta}_t$. O método TD(λ) utilizando o método do gradiente descendente pode ser visto abaixo:

$$\overline{\theta}_{t+1} = \overline{\theta}_t + \alpha\delta_t\overline{e}_t, \tag{2.80}$$

$$\delta_t = r_{t+1} + \gamma V_{\theta_t}(s_{t+1}) - V_{\theta_t}(s_t), \tag{2.81}$$

$$\overline{e}_t = \gamma\lambda\overline{e}_{t-1} + \nabla_{\overline{\theta}_t} V_{\theta_t}(s_t). \tag{2.82}$$

Um caso especial de se utilizar o método do gradiente descendente em aproximadores de funções é quando a função V_t é uma função linear do vetor de parâmetros $\overline{\theta}$. Correspondendo a todo estado s, há um vetor de características $\overline{\phi_s} = (\phi_s(1), \phi_s(2), ..., \phi_s(n))^T$, com o mesmo número de componente de $\overline{\theta}_t$. Dessa forma a função V pode ser aproximada por:

$$V_{\theta_t}(s) = \overline{\theta}_t^T\overline{\phi_s} = \sum_{i=1}^{n} \theta_t(i)\,\phi_t(i). \tag{2.83}$$

Esse caso de aproximação é chamada de linear nos parâmetros. E o gradiente nesse caso é simplesmente:

$$\nabla_{\overline{\theta}_t} V_{\theta_t}(s) = \overline{\phi_s}. \tag{2.84}$$

Uma forma de se utilizar o vetor de características $\overline{\phi_s}$ é o chamado *tile coding*, em que o espaço do estado é divido em partições e cada partição é chamada de *tile*, e se o valor de s está na partição então o valor dessa característica é 1 caso contrário 0. Uma forma de se particionar um espaço de duas dimensões por exemplo é uma grade uniforme como da Figura 2.11.

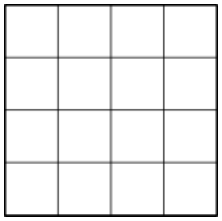


Figura 2.11: Aproximador de função usando *tile coding*.

Outra forma é utilizar uma função de base radial. Uma possível base é uma função gaussiana que depende da distancia do estado s com o centro da característica c_i e tamanho relativo a característica σ_i:

$$\phi_s(i) = \exp\left(-\frac{\|s - c_i\|^2}{2\sigma_i^2}\right). \tag{2.85}$$

2.5.8 Aprendizado por reforço para ações contínuas (Algoritmo da política gradiente)

Para ações contínuas também pode ser utilizado um aproximador de funções como nos estados contínuos. Para isso podemos descrever a política $\pi_\psi(s, a)$ em termos de $\overline{\psi}_t$, onde $\overline{\psi}_t = (\psi_t(1), \psi_t(2), ..., \psi_t(n))^T$. Como mostrado por Hasselt [20], podemos atualizar ψ da seguinte forma:

$$\overline{\psi}_{t+1} = \overline{\psi}_t + \beta_t(s_t) \nabla_\psi V^\pi(s_t). \tag{2.86}$$

Caso $\nabla_\psi V^\pi(s_t)$ não seja conhecido, é necessário obter uma estimativa para ela. Para isso vamos definir o conceito de trajetória, uma trajetória $\mathscr{S}$ como uma sequencia de estados e ações:

$$\mathscr{S} = \{s_0, a_0, s_1, a_1, ...\}.$$

Assim a probabilidade de que uma dada trajetória ocorra é igual a probabilidade que a sequencia correspondente de estados e ações ocorram com a política dada:

$$\begin{aligned} P(\mathscr{S}|\, s, \psi) &= P(s_0 = s)\, P(a_0|s_0, \psi)\, P(s_1|s_0, a_0)\, P(a_1|s_1, \psi) .... \\ &= P(s_0 = s) \prod_{t=0}^{\infty} \pi_\psi(s_t, a_t)\, \mathcal{P}^{a_t}_{s_t s_{t+1}}. \end{aligned} \tag{2.87}$$

Podemos expressar V^{π} como uma integral sobre todos as possíveis trajetórias para uma dada política e as recompensas esperadas:

$$V^{\pi}(s) = \int_{\mathscr{S}} P(\mathscr{S}|\, s, \psi)\, E\left\{ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| \mathscr{S} \right\} d\mathscr{S}. \tag{2.88}$$

Dessa forma o gradiente pode ser expresso por:

$$\begin{aligned}
\nabla_{\psi} V^{\pi}(s) &= \int_{\mathscr{S}} \nabla_{\psi} P(\mathscr{S}|\, s, \psi)\, E\left\{ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| \mathscr{S} \right\} d\mathscr{S} \\
&= \int_{\mathscr{S}} P(\mathscr{S}|\, s, \psi)\, \nabla_{\psi} \log P(\mathscr{S}|\, s, \psi)\, E\left\{ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| \mathscr{S} \right\} d\mathscr{S} \\
&= E\left\{ \nabla_{\psi} \log P(\mathscr{S}|\, s, \psi)\, E\left\{ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| \mathscr{S} \right\} \middle| s, \psi \right\},
\end{aligned} \tag{2.89}$$

onde a propriedade $\nabla_x f(x) = f(x)\, \nabla_x \log f(x)$ foi usada. O produto da Equação 2.87 implica que o logaritmo da equação 2.89 é dado por uma soma de termos e assim:

$$\begin{aligned}
\nabla_{\psi} \log P(\mathscr{S}|\, s, \psi) &= \nabla_{\psi} \left(\log P(s_0 = s) + \sum_{t=0}^{\infty} \log \pi_{\psi}(s_t, a_t) + \sum_{t=0}^{\infty} \log \mathcal{P}^{a_t}_{s_t s_{t+1}} \right) \\
&= \sum_{t=0}^{\infty} \nabla_{\psi} \log \pi_{\psi}(s_t, a_t).
\end{aligned} \tag{2.90}$$

Dessa forma, não precisamos do modelo de transição. Mas isso só vale se a política é estocástica, pois caso ela seja determinística precisamos do gradiente $\nabla_{\psi} \mathcal{P}^{a_t}_{s_t s_{t+1}} = \nabla_a \mathcal{P}^{a}_{s_t s_{t+1}} \nabla_{\psi} \pi_{\psi}(s, a)$, o que só está disponível se as probabilidades de transições forem conhecidas. O que não é um problema, pois normalmente a política é estocástica para garantir exploração.

Se tivermos tarefas periódicas a soma da Equação 2.90 se torna finita e assim usando o método de MC obtemos:

$$\nabla_{\psi} V^{\pi}(s_t) = E\left\{ R_t(s_t) \left(\sum_{j=t}^{T_k - 1} \nabla_{\psi} \log \pi_{\psi}(s, a) \right) \right\}. \tag{2.91}$$

Como dito por Hasselt [20], a variância de R_t pode ser alta implicando em estimações ruidosas do gradiente. Podemos adicionar também uma base $b(s_t)$ para minimizar a variância da seguinte forma:

$$\overline{\psi}_{t+1} = \overline{\psi}_t + \beta_t(s_t) \left(R_k(s_t) - b(s_t) \right) \sum_{j=t}^{T_k - 1} \nabla_{\psi} \log \pi_{\psi}(s, a), \tag{2.92}$$

porque a base $b(s_t)$ não adiciona nenhuma viés a estimação do gradiente pois:

$$\int_{\mathscr{S}} \nabla_{\psi} P(\mathscr{S}|\, s, \psi)\, b(s_t)\, d\mathscr{S} = b(s)\, \nabla_{\psi} \int_{\mathscr{S}} P(\mathscr{S}|\, s, \psi)\, d\mathscr{S} = b(s)\, \nabla_{\psi} 1 = 0 \tag{2.93}$$

Dessa forma podemos escolher $b(s) = V^{\pi}(s_t)$ e assim podemos utilizar o método TD, que irá resultar na estimativa não enviesada $\delta_t \nabla_{\psi} \log \pi_{\psi}(s,a)$ para o gradiente [52]. Logo, também podemos usar o método TD(λ) da seguinte forma:

$$\overline{\psi}_{t+1} = \overline{\psi}_t + \beta \delta_t \overline{e}_t, \tag{2.94}$$

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t), \tag{2.95}$$

$$\overline{e}_t = \gamma \lambda \overline{e}_{t-1} + \nabla_{\psi} \log \pi_{\psi}(s,a). \tag{2.96}$$

Para o caso de uma política gaussiana com média $\overline{\mu} \in \Re^{D_A}$ e a matriz $D_A \times D_A$ de covariância Σ, obtemos:

$$\pi(s,a,\{\overline{\mu},\Sigma\}) = \frac{1}{\sqrt{2\pi \det \Sigma}} \exp\left(-\frac{1}{2}(\overline{a}-\overline{\mu})^T \Sigma^{-1}(\overline{a}-\overline{\mu})\right), \tag{2.97}$$

$$\nabla_{\mu} \log \pi(s,a,\{\overline{\mu},\Sigma\}) = (\overline{a}-\overline{\mu})^T \Sigma^{-1}, \tag{2.98}$$

$$\nabla_{\Sigma} \log \pi(s,a,\{\overline{\mu},\Sigma\}) = \frac{1}{2}\left(\Sigma^{-1}(\overline{a}-\overline{\mu})(\overline{a}-\overline{\mu})^T \Sigma^{-1} - \Sigma^{-1}\right), \tag{2.99}$$

onde as ações $\overline{a} \in A$ são um vetor do mesmo tamanho de $\overline{\mu}$. Se $\overline{\psi} \in \Psi \subseteq \Re^{D_u}$ é o vetor de parâmetro que determina o local dependente do estado da média $\overline{\mu_{\psi}}(s)$, então

$$\nabla_{\psi} \log \pi_{\psi}(s,a) = J_{\psi}^T(\mu_{\psi}(s)) \nabla_{\mu} \log \pi_{\psi}(s,a,\{\overline{\mu},\Sigma\}),$$

onde $J_{\psi}^T(\mu(s,\psi))$ é a matriz $D_A \times D_{\Psi}$ Jacobiana com as derivadas parciais de cada elemento de $\overline{\mu_{\psi}}(s)$ em relação a $\overline{\psi}$.

3 DESENVOLVIMENTO

3.1 INTRODUÇÃO

Os métodos desenvolvidos nesse trabalho foram utilizados em duas plataformas, a plataforma Bioloid e a plataforma NAO. Inicialmente utilizou-se a plataforma Bioloid, depois o trabalho foi passado para a plataforma NAO, desse forma o método em cada plataforma possui diferenças. Inicialmente será descrito o método para cada plataforma, e depois as plataformas.

3.2 GERANDO A MARCHA COM OS OSCILADORES

3.2.1 Gerando a marcha no Bioloid

Inicialmente usamos quatro osciladores de Matsuoka no total. Dois osciladores em cada perna, onde um oscilador é utilizado para prover o ângulo de arfagem do quadril e outro o ângulo de arfagem do joelho. Para o joelho apenas valores menores ou iguais a zero foram permitidos. Os ângulos do tornozelo foram utilizados de forma que os pés fiquem paralelos em relação ao chão considerando um ângulo de arfagem e rolagem do corpo do robô próximo de zeros graus. As conexões dos osciladores podem ser vistos na Figura 3.1 e o modelo do robô na Figura 3.2. O joelho esquerdo e o joelho direito são conectados com o intuito de obter um acoplamento mais forte entre as pernas direita e esquerda, para assim obter uma adaptação mais rápida quando ocorrerem pertubações no sinal de uma perna.

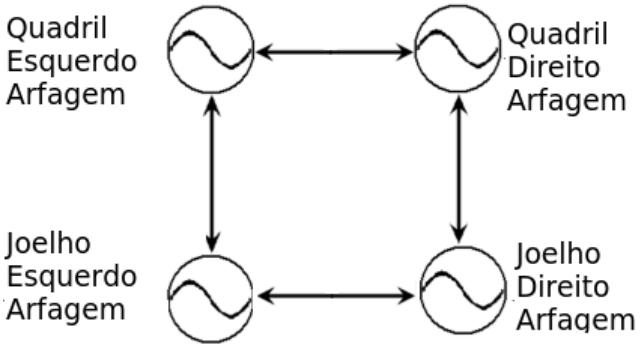


Figura 3.1: Conexões dos osciladores no Bioloid.

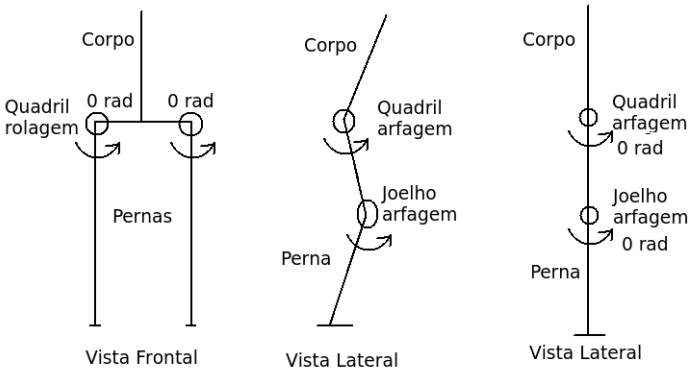


Figura 3.2: Modelo de pernas do robô usado.

Tabela 3.1: Valor dos parâmetros fixos do oscilador de Matsuoka usados no Bioloid.

Parâmetros	Valor
τ_1(paras todos osciladores)	0,03355
τ_2(paras todos osciladores)	0,33553
β(paras todos osciladores)	1,5
g(paras todos osciladores)	1,5
$w_{lh-rh} = w_{rh-lh}$	0,2
$w_{lk-rk} = w_{rk-lk}$	0,2

Tabela 3.2: Bordas Superior e Inferior dos parâmetros.

Parâmetros	Valor máximo	Valor mínimo
$w_{lh-lk} = w_{rh-rk}$	-0,05	0,05
$w_{lk-lh} = w_{rk-rh}$	-0,50	0,50
$c_{lh} = c_{rh}$	0,30	0,75
$c_{lk} = c_{rk}$	0,01	0,50

Para otimizar o movimento utilizamos o AG e o PSO nos osciladores. Alguns parâmetros do oscilador foram mantidos constantes os parâmetros τ_1, τ_2, β, g e as conexões $w_{lh-rh} = w_{rh-lh}$ e $w_{lk-rk} = w_{rk-lk}$, onde os subíndices lh, rh, lk, rk significam *left hip* (quadril esquerdo), *right hip* (quadril direito), *left knee* (joelho esquerdo), right knee (joelho direito), respectivamente, e o subíndice $i - j$ significa de i para j. Então no total foram otimizados seis parâmetros, $c_{lh} = c_{rh}$, $c_{lk} = c_{rk}$, $w_{lh-lk} = w_{rh-rk}$ e $w_{lk-lh} = w_{rk-rh}$. Nós também limitamos o espaço de busca de cada parâmetro. O valor dos parâmetros mantidos fixos podem ser vistos na Tabela 3.1. As bordas superior e inferior do espaço de busca estão na Tabela 3.2. Os parâmetros fixos foram escolhidos de acordo com as restrições obtidas por Huang [22] e descritas em 2.2.1. O valor dos parâmetros usados na limitação do espaço de busca foram inspiradas em testes feitos anteriormente onde os parâmetros eram obtidos por tentativa e erro [13].

Inicialmente foi utilizada uma função de custo que considerava apenas a distância percorrida pelo robô e se ele caiu ou não. Mas foi observado que algumas marchas eram instáveis e poderiam fazer com que o robô caia após o tempo utilizado e outras que eram ineficientes. Então foram adicionadas outras condições a função de custo resultando na seguinte função de custo:

$$f = \begin{cases} -1000, & \text{se } |\phi| > \frac{\pi}{4} \text{ ou } |\psi| > \frac{\pi}{4} \\ -500, & \text{se } \triangle_p < 0 \\ -100, & \text{se } \sigma_\phi > \sigma_{\phi_{max}} \\ -80, & \text{se } \sigma_h > \sigma_{h_{max}} \\ -70, & \text{se } \triangle_h > \triangle_{h_{max}} \\ -50, & \text{se } \sigma_\psi > \sigma_{\psi_{max}} \\ [100 \triangle_p K_p \left(\sigma_{\phi_{max}} - \sigma_\phi\right) \left(\sigma_{h_{max}} - \sigma_h\right) & \\ \left(\triangle_{h_{max}} - \triangle_h\right) \left(\sigma_{\psi_{max}} - \sigma_\psi\right)]. & \text{caso contrário,} \end{cases}, \tag{3.1}$$

onde ϕ é o ângulo de arfagem do copo, ψ é o ângulo de rolagem, h é a altura do centro de massa do corpo do robô em relação ao chão. $\triangle_p = p_f - p_i$, onde p_f é a posição final do robô e p_i é a posição inicial. σ_ϕ é o desvio padrão de ϕ. σ_h é o desvio padrão de h. σ_ψ é o desvio padrão de ψ. $\triangle_h = h_f - h_i$, onde h_f é a altura final do centro de massa do corpo do robô em relação ao chão e h_i a altura inicial. E o subíndice max significa o máximo permitido.

Ou seja, se o robô cai a função de custo é -1000. Se o robô se move para trás f=-500. Se $\sigma_\phi > \sigma_{\phi_{max}}$, i.e. se a variação em torno do eixo y é maior do que o permitido, f=-100. Se $\sigma_h > \sigma_{h_{max}}$, se a variação do corpo do robô é maior do que o permitido, f=-80. Se $\triangle_h > \triangle_{h_{max}}$, i.e. a diferença entre a altura final e inicial maior do que o permitido f=-70. Se $\sigma_\psi > \sigma_{\psi_{max}}$, i.e. tem uma variação em torno do eixo x maior do que o permitido f=-40. E se nada disso ocorre a função de custo é dada pela expressão final da Equação 3.1. Essa equação foi utilizada com o fim de se obter uma marcha eficiente, reduzindo movimentos desnecessários e/ou instáveis. O valor dos parâmetros da equação estão na Tabela 3.3.

Tabela 3.3: Valores da função de custo utilizada no Bioloid.

Parâmetro	Valor
$\sigma_{\phi_{max}}$(rad)	0,2618
$\sigma_{h_{max}}$(m)	0,01
$\sigma_{\psi_{max}}$(rad)	0,1745
$\triangle_{h_{max}}$(m)	0,05

Nós utilizamos uma inicialização randômica uniforme, i.e., cada partícula ou cromossomo foi escolhido dentro de cada intervalo usando uma variável aleatória de distribuição uniforme.

Tabela 3.4: Parâmetros do PSO utilizados no Bioloid.

Parâmetro	Valor
Número de partículas	100
c_1	2,05
c_2	2,05
φ	4,1

Tabela 3.5: Parâmetros do AG utilizados no Bioloid.

Parâmetro	Valor
Número de cromossomos	100
Taxa de cruzamento	0,8
Probabilidade de Mutação (por gene)	0,1
Desvio padrão da distribuição Gaussiana	0,3

Para o PSO utilizamos a topologia *lbest* com restrição. Se uma partícula estiver fora do espaço de busca, nós não atualizamos a função de custo dessa posição, deixando que as equações de atualização de velocidade e posição levem a partícula de volta ao espaço de busca. Os parâmetros do PSO estão na tabela 3.4.

Para o AG utilizamos a seleção dada pela regra da roleta, o cruzamento convexo e mutação Gaussiana. O desvio padrão da mutação foi multiplicada pelo tamanho do espaço de busca. Se um cromossomo estivesse fora do espaço de busca nós o limitamos ao espaço de busca, tendo valor igual a borda. Os parâmetros do AG estão na Tabela 3.5.

O AG descrito acima também foi utilizado com mais osciladores. Foram utilizados dois osciladores a mais, um para o ângulo de rolagem de cada perna. Além disso, para observar o desempenho, ângulos positivos do joelho foram permitidos, mas foram limitados pelos limites físicos do robô.

3.2.2 Gerando a marcha no NAO

No NAO foram usados dois tipos de osciladores, o oscilador de Matsuoka e o oscilador de Kuramoto e foram utilizados o GA e o PSO para se obterem os parâmetros os osciladores. Além disso, no oscilador de Kuramoto também foi utilizada uma forma de acoplamento que realiza o acoplamento entre a dinâmica do oscilador e a dinâmica do movimento do robô utilizando-se o ponto de momento nulo, ZMP. Ele foi utilizado no trabalho de Morimoto [40–42] e será explicado mais a frente.

3.2.2.1 Utilizando o oscilador de Matsuoka no NAO

Foram utilizados seis osciladores de Matsuoka no total, três em cada perna, um para o ângulo de rolagem do quadril, outro para o ângulo de arfagem do quadril e mais um para o ângulo do joelho. Os ângulos do tornozelo foram utilizados de forma a ficarem perpendiculares em relação ao corpo do robô, para podermos obter um contato perpendicular com o chão. Os ângulos obtidos pelos osciladores são limitados pelos valores extremos de cada junta do robô. O modelo do robô usado foi o mesmo modelo do Bioloid na Figura 3.2. As conexões dos osciladores podem ser vistas na Figura 3.3. Todos os osciladores da perna direita estão conectadas com os da perna esquerda com o intuito de obter um acoplamento mais forte entre as pernas direita e esquerda, para assim obter uma adaptação mais rápida quando ocorrerem pertubações no sinal de uma perna assim como no Bioloid.

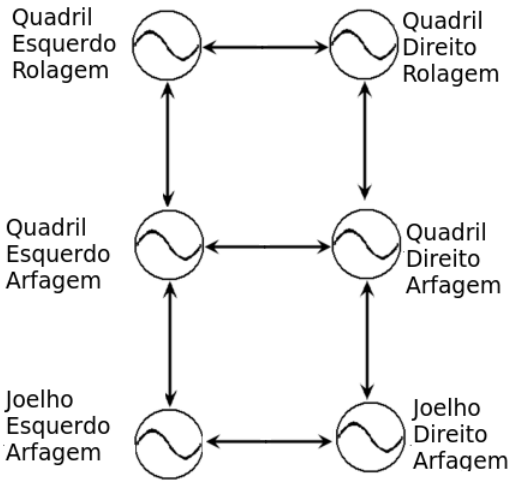


Figura 3.3: Conexões dos osciladores de Matsuoka no NAO.

Também foram utilizados o AG e o PSO para otimizar os seguintes parâmetros: $c_{lh_{roll}} = c_{rh_{roll}}$, $c_{lh_{pitch}} = c_{rh_{pitch}}$, $c_{lk} = c_{rk}$, $w_{lh_{roll}-lh_{pitch}} = w_{rh_{roll}-rh_{pitch}}$, $w_{lh_{pitch}-lh_{roll}} = w_{rh_{pitch}-rh_{roll}}$, $w_{lh_{pitch}-lk} = w_{rh_{pitch}-rk}$, $w_{lk-lh_{pitch}} = w_{rk-rh_{pitch}}$, $off_{hip_{pitch}}$ e off_{knee}, onde os subíndices lh_{roll}, rh_{roll}, lh_{pitch}, rh_{pitch}, lk, rk significam *left hip roll* (do inglês quadril esquerdo rolagem), *right hip roll* (do inglês quadril direito rolagem), *left hip pitch* (do inglês quadril esquerdo arfagem), *right hip pitch* (do inglês quadril direito arfagem), *left knee* (do inglês joelho esquerdo), right knee (do inglês joelho direito), respectivamente, e o subíndice $i - j$ significa de i para j. off é um offset dado a saída do oscilador. Então no total foram otimizados 9 parâmetros, já os parâmetros τ_1, τ_2, β, g e as conexões $w_{lh_{roll}-rh_{roll}} = w_{rh_{roll}-lh_{roll}}$, $w_{lh_{pitch}-rh_{pitch}} = w_{rh_{pitch}-lh_{pitch}}$ e $w_{lk-rk} = w_{rk-lk}$ foram mantidos constantes. O valor dos parâmetros mantidos fixos podem ser vistos na Tabela 3.6. Para otimizar as conexões entre os osciladores foi utilizado um parâmetro para cada conexão a ser utilizada, w_{aux1}, w_{aux2}, w_{aux3} ,w_{aux4} , no lugar delas de modo que

$$w_{lh_{roll}-lh_{pitch}} = w_{rh_{roll}-rh_{pitch}} = \frac{w_{aux_1} c_{lh_{roll}}}{c_{lh_{roll}} + c_{lh_{pitch}}}, \tag{3.2}$$

$$w_{lh_{pitch}-lh_{roll}} = w_{rh_{pitch}-rh_{roll}} = \frac{w_{aux_2} c_{lh_{pitch}}}{c_{lh_{roll}} + c_{lh_{pitch}}}, \tag{3.3}$$

$$w_{lh_{pitch}-lk} = w_{rh_{pitch}-rk} = \frac{w_{aux_3} c_{lh_{pitch}}}{c_{lk} + c_{lh_{pitch}}}, \tag{3.4}$$

$$w_{lk-lh_{pitch}} = w_{rk-rh_{pitch}} = \frac{w_{aux_4} c_{lk}}{c_{lk} + c_{lh_{pitch}}}. \tag{3.5}$$

Perceba que isso é uma forma de normalizar o valor a ser usado no algoritmo de otimização em relação à outra constate que está sendo otimizada, que é a constante responsável pela amplitude da saída de cada oscilador. As bordas superior e inferior do espaço de busca estão na Tabela 3.7.

Tabela 3.6: Valor dos parâmetros fixos do oscilador de Matsuoka usados no NAO.

Parâmetros	Valor
τ_1(paras todos osciladores)	0,03183
τ_2(paras todos osciladores)	0,31831
β(paras todos osciladores)	1,5
g(paras todos osciladores)	1,5
$w_{lh_{roll}-rh_{roll}} = w_{rh_{roll}-lh_{roll}}$	0,2
$w_{lh_{pitch}-rh_{pitch}} = w_{rh_{pitch}-lh_{pitch}}$	0,2
$w_{lk-rk} = w_{rk-lk}$	0,2

Tabela 3.7: Bordas Superior e Inferior dos parâmetros do oscilador de Matsuoka no NAO.

Parâmetros	Valor máximo	Valor mínimo
$c_{lh_{roll}} = c_{rh_{roll}}$	0,3	0,001
$c_{lh_{pitch}} = c_{rh_{pitch}}$	0,5	0,1
$c_{lk} = c_{rk}$	0,5	0,1
w_{aux_1}	2	-2
w_{aux_2}	2	-2
w_{aux_3}	2	-2
w_{aux_4}	2	-2
$off_{hip_{pitch}}$	0	-25 graus
off_{knee}	50 graus	0

Nós fizemos a seguinte função de custo:

$$f = \begin{cases} -5000 + t, & \text{se } |\phi| > \frac{\pi}{4} \text{ ou } |\psi| > \frac{\pi}{4} \\ -1000, & \text{se } \triangle_p < 0 \\ -900 + (\sigma_{\phi_{max}} - \sigma_\phi), & \text{se } \sigma_\phi > \sigma_{\phi_{max}} \\ -800 + (\sigma_{h_{max}} - \sigma_h), & \text{se } \sigma_h > \sigma_{h_{max}} \\ -700 + (\triangle_{h_{max}} - \triangle_h), & \text{se } \triangle_h > \triangle_{h_{max}} \\ -600 + (\sigma_{\psi_{max}} - \sigma_\psi), & \text{se } \sigma_\psi > \sigma_{\psi_{max}} \\ -400 + \triangle_p K_p (\sigma_{\phi_{max}} - \sigma_\phi)(\sigma_{h_{max}} - \sigma_h)(\triangle_{h_{max}} - \triangle_h)(\sigma_{\psi_{max}} - \sigma_\psi) & \text{se } \mu_{pl} > \mu_{pl_{max}} \\ \triangle_p K_p & \text{caso contrário, .} \end{cases} , \tag{3.6}$$

onde ϕ é o ângulo de arfagem do copo, ψ é o ângulo de rolagem, h é a altura do centro de massa do corpo do robô em relação ao chão. $\triangle_p = p_f - p_i$, onde p_f é a posição final do robô e p_i é a posição inicial. σ_ϕ é o desvio padrão de ϕ. σ_h é o desvio padrão de h. σ_ψ é o desvio padrão de ψ. $\triangle_h = h_f - h_i$, onde h_f é a altura final do centro de massa do corpo do robô em relação ao chão e h_i a altura inicial. μ_{pl} é a média do deslocamento lateral do robô. E o subíndice max significa o máximo permitido.

Note que a função de custo foi modificada em relação a função de custo usada no Bioloid. Isso ocorreu porque essa função de custo possui um desempenho melhor que no caso anterior. Inicialmente foi utilizada a mesma função de custo do caso do Bioloid, mas como dito anteriormente a função foi alterada para se ter um desempenho melhor.

Como no caso do Bioloid, essa função de custo foi utilizada com o intuito de se obter uma marcha eficiente.

As características do AG e do PSO foram as mesmas usadas no Bioloid, somente a taxa de cruzamento do AG utilizada que foi diferente, sendo seu valor 0,9. Essa mudança se deve porque com uma taxa de cruzamento de 0,8 havia uma maior número de soluções que não tinham seus parâmetros mudados pelos algoritmo, fazendo com que a convergência do mesmo fosse mais lenta. O valor dos parâmetros da equação estão na Tabela 3.8.

Também foi utilizado o NSGA-II utilizando dois objetivos, o primeiro deles dado pela função de custo anterior dada pela Equação 3.6 e a outra dada pela equação abaixo:

$$f_2 = \begin{cases} -5000 + t, & \text{se } |\phi| > \frac{\pi}{4} \text{ ou } |\psi| > \frac{\pi}{4} \\ -1000, & \text{se } \triangle_p < 0 \\ -900 + (\sigma_{\phi_{max}} - \sigma_\phi), & \text{se } \sigma_\phi > \sigma_{\phi_{max}} \\ -800 + (\sigma_{h_{max}} - \sigma_h), & \text{se } \sigma_h > \sigma_{h_{max}} \\ -700 + (\triangle_{h_{max}} - \triangle_h), & \text{se } \triangle_h > \triangle_{h_{max}} \\ -600 + (\sigma_{\psi_{max}} - \sigma_\psi), & \text{se } \sigma_\psi > \sigma_{\psi_{max}} \\ -400 + \triangle_p K_p (\sigma_{\phi_{max}} - \sigma_\phi)(\sigma_{h_{max}} - \sigma_h)(\triangle_{h_{max}} - \triangle_h)(\sigma_{\psi_{max}} - \sigma_\psi) & \text{se } u_{pl} > u_{pl_{max}} \\ 1000 - W_{total} & \text{caso contrário, .} \end{cases} , \tag{3.7}$$

onde W_{total} é o trabalho total realizado pelos motores da perna durante a marcha. Para calculá-lo foi calculado o trabalho de cada motor da perna em cada instante de tempo dado pelo período de amostragem dt. Para isso temos que no instante de tempo, t, o trabalho do motor é dado pelo torque multiplicado pelo deslocamento angular nesse momento. Foi feito um somatório de todos os momentos t durante a marcha e assim obteve-se W_{total}.

O valor dos parâmetros utilizados para o NSGA-II são os mesmos da Tabela 3.8.

Tabela 3.8: Valores da função de custo utilizada no NAO com oscilador de Matsuoka.

Parâmetro	Valor
$\sigma_{\phi_{max}}$(graus)	2,5
$\sigma_{h_{max}}$(m)	0,1
$\sigma_{\psi_{max}}$(graus)	5,0
$\triangle_{h_{max}}$(m)	0,5
K_p	10000
$\mu_{pl_{max}}$(m)	0,05

3.2.2.2 Utilizando o oscilador de Kuramoto no NAO

Como foi dito na Subseção 2.2.2 a saída do oscilador foi uma Série de Fourier truncada. Utilizamos 2 harmônicos para a Série de Fourier e foram utilizados 6 osciladores também, como no caso de Matsuoka. Os ângulos obtidos pelos osciladores são limitados pelos valores extremos de cada junta do robô. O modelo de pernas do robô usado foi o mesmo adotado para o Bioloid que está presente Figura 3.2, pois esses graus de liberdade utilizados da perna estão presentes nas duas plataformas. As equações do oscilador podem ser vistas nas Equações 2.13, 2.25, 2.26 e 2.27, mas serão repetidas abaixo por praticidade do texto:

$$\dot{\phi_i} = 2\pi f + \sum_{j \neq i} k_{ij} \text{sen}\left(\phi_j - \phi_i - \varphi_{ij}\right), \; i = 1, 2, \tag{3.8}$$

$$y_{ji} = o_{ji} + a_{ji} \sum_{k=1}^{3} c_{kji} \cos\left(\phi_i - \theta_{kji}\right),\ j = 1, 2, 3, \tag{3.9}$$

$$\dot{a}_{ji} = \alpha \left(A_{ji} - a_{ji}\right), \tag{3.10}$$

$$\dot{o}_{ji} = \alpha \left(O_{ji} - o_{ji}\right), \tag{3.11}$$

onde a fase dada por $i = 1$ corresponde a fase da perna direita e a fase dada por $i = 2$ corresponde a fase da perna esquerda. As saídas dadas por $j = 1, 2, 3$ são as das juntas quadril rolagem, quadril arfagem e joelho arfagem respectivamente.

Os parâmetros que foram mantidos constastes foram α, k_{ij}, φ_{ij} e A_{ji} $\forall i = 1, 2$ e $\forall j = 1, 2, 3$. Os valores de cada um deles podem ser vistos na Tabela 3.9.

Tabela 3.9: Valor dos parâmetros fixos do oscilador de fase usados no NAO.

Parâmetros	Valor
f (Hz)	1
α	10
$k_{1,1} = k_{2,2}$	0
$k_{1,2} = k_{2,1}$	1
$\varphi_{1,1} = \varphi_{2,2}$(rad)	0
$\varphi_{1,2} = -\varphi_{2,1}$(rad)	π
$A_{ji}, \forall i = 1, 2$ e $\forall j = 1, 2, 3$	1

Tabela 3.10: Bordas Superior e Inferior dos parâmetros do oscilador de fase no NAO.

Parâmetros	Valor máximo	Valor mínimo	Parâmetros	Valor máximo	Valor mínimo
$c_{111} = c_{112}$(graus)	6,5	3	$c_{111} = c_{112}$(graus)	55	15
$c_{211} = c_{212}$(graus)	0	0	$c_{211} = c_{212}$(graus)	30	0
$c_{311} = c_{312}$(graus)	0	0	$c_{311} = c_{312}$(graus)	15	0
$c_{111} = c_{112}$(graus)	25	15	$O_{11} = O_{12}$(graus)	0	0
$c_{211} = c_{212}$(graus)	15	0	$O_{21} = O_{22}$(graus)	0	-35
$c_{311} = c_{312}$(graus)	7,5	0	$O_{31} = O_{31}$(graus)	50	0
$\theta_{211}, \theta_{212}$, $\theta_{311}, \theta_{312}$	0	0	Todos os outros θ_{kij}(graus)	180	-180

Os parâmetros otimizados pelo AG foram os parâmetros da Série de Fourier truncada. Como utilizamos 2 harmônicos e a fundamental, foram 7 parâmetros por oscilador, totalizando 21 parâmetros, pois tivemos três osciladores por perna, sendo que os parâmetros de uma perna são iguais aos da outra. Os valores máximos e mínimos das bordas do espaço de busca podem ser vistos na Tabela 3.10.

Nós utilizamos a mesma função de custo do caso do oscilador de Matsuoka dada pela Equação 3.6. Nesse caso foi utilizado apenas o AG, dado que com os dois métodos é possível de se obter os parâmetros, mas para se utilizar de menos tempo das simulações optou-se por utilizar somente um deles. E as características do AG utilizadas são dadas pela Tabela 3.11. Também foi utilizado o NSGA-II da mesma forma que no caso anterior com os parâmetros da função de custo iguais aos ao do AG.

Tabela 3.11: Valores da função de custo utilizada no NAO com oscilador de Kuramoto.

Parâmetro	Valor
$\sigma_{\phi_{max}}$(graus)	5,0
$\sigma_{h_{max}}$(m)	0,1
$\sigma_{\psi_{max}}$(graus)	7,5
$\triangle_{h_{max}}$(m)	0,5
K_p	10000
$\mu_{pl_{max}}$(m)	0,05

3.2.2.3 Utilizando o oscilador de Kuramoto com acoplamento com a dinâmica do movimento no NAO

Com o objetivo de sincronizar os padrões rítmicos gerados pelo CPG e a dinâmica da marcha do robô, Morimoto [40–42] utilizou um acoplamento entre os dois utilizando um oscilador de Kuramoto. O modelo dinâmico utilizado do robô foi o modelo simplificado de um pêndulo invertido onde o topo do pêndulo corresponde ao centro de massa (CdM) do robô e a base do pêndulo ao centro de pressão (CdP) do robô [21, 25, 26] conforme visto na Figura 3.4.

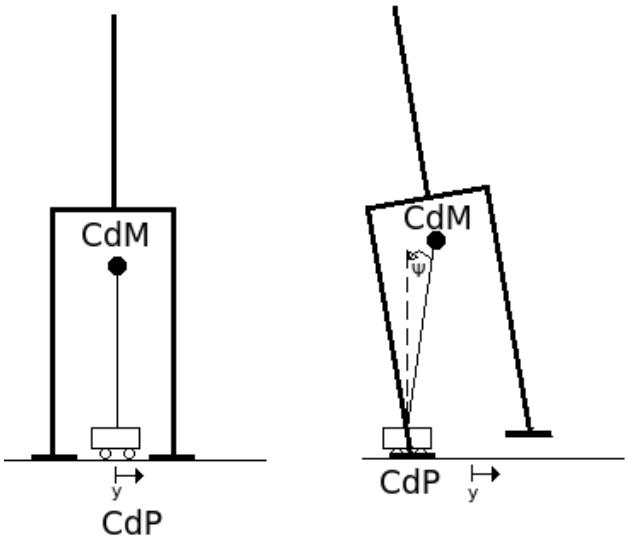


Figura 3.4: Modelo de um pêndulo invertido para um robô bípede.

O modelo de um pêndulo invertido possui quatro variáveis de estado $\overline{x} = \left(y, \dot{y}, \psi, \dot{\psi}\right)$. Para detectar a fase da dinâmica do movimento foi utilizado a posição do centro de pressão y e sua velocidade $\dot{y}$ da seguinte forma:

$$\phi_r = -\arctan\left(\frac{\dot{y}}{y}\right). \tag{3.12}$$

E o acoplamento com o oscilador se deu da seguinte forma:

$$\dot{\phi}_{osc} = 2\pi f_c + K_c \text{sen}\left(\phi_r - \phi_{osc}\right), \tag{3.13}$$

onde ϕ_{osc} é a fase do oscilador, f_c é a frequência do oscilador, ϕ_r é fase da dinâmica do movimento do robô e K_c é a constante de acoplamento. Essa fase é usada para uma das pernas, para outra perna é usada essa fase com um atraso de 180 graus.

Para calcular o CdP foram utilizados os sensores de pressão localizados nos pés do robô. Para simplificar o calculo do CdP foi utilizada a seguinte equação:

$$y = \frac{y_p^e F_z^e + y_p^d F_z^d}{F_z^e + F_z^d}, \tag{3.14}$$

onde y_p^e e y_p^d são as posições laterais dos pés esquerdo e direito respectivamente, e F_z^e e F_z^d são as forças de reação do chão nos pés esquerdo e direito respectivamente. Foi suposto que os pés são colocados simetricamente, ou seja, $y_p^e = -y_p^d$. Como o CdP só foi utilizado para o calculo da fase do robô ϕ_r, a escala para a posição dos pés y_p^e e y_p^d pode ser arbitraria. Foi escolhido $y_p^d = -y_p^e = 1,0$m. Desse modo, com o calculo simplificado do CdP não foi necessário o modelo cinemático direto para o calculo do CdP.

Também foi utilizado um filtro passa-baixa em F_z^e e F_z^d para diminuir o ruído de alta frequência dos sensores de pressão na sola dos pés. Foi utilizado um filtro digital Butterworth de primeira ordem com frequência de corte de 4 Hz dado pela equação:

$$a(1)y_{filtro}(k) = b(1)x_{filtro}(k) + b(2)x_{filtro}(k-1) + a(2)y_{filtro}(k-1), \tag{3.15}$$

onde y_{filtro} é a saída do filtro e x_{filtro} a entrada do filtro, e o valor das constantes $a(1)$, $a(2)$, $b(1)$ e $b(2)$ para um período de amostragem de 40ms podem ser vistos na Tabela 3.12.

Tabela 3.12: Valores das constantes do filtro Butterworth.

Constante	Valor
$a(1)$	$1,0$
$a(2)$	$-0,290526856731916$
$b(1)$	$0,354736571634042$
$b(2)$	$0,354736571634042$

Dessa forma, a estrutura do método utilizado pode ser vista no diagrama de blocos da Figura 3.5.

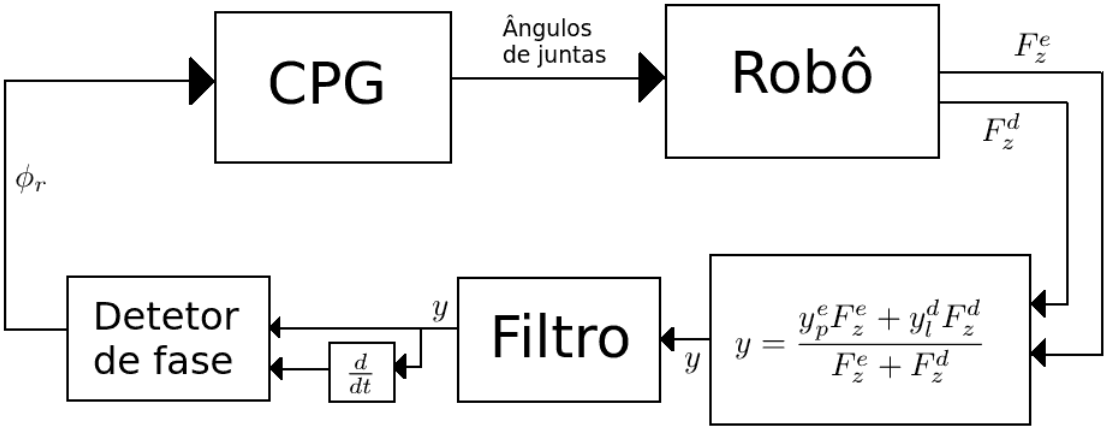


Figura 3.5: Diagrama de blocos do oscilador de Kuramoto com acoplamento com a dinâmica do movimento no NAO.

Para a saída do oscilador foram usados dois tipos de saída, uma sendo uma Série de Fourier truncada igual a dada pelas Equações 2.25, 2.26 e 2.27 e outra sendo senoides simples. Para a senoide simples foram utilizadas as mesmas equações mas a Série de Fourier truncada não possui harmônicos. Alguns parâmetros do oscilador foram mantidos constantes e são os mesmos do caso anterior e seus valores são dados pela Tabela 3.9.

Os parâmetros da Série de Fourier foram otimizados pelo AG, no primeiro caso sendo 21 parâmetros, e no segundo caso, a saída sendo uma senoide, 9 parâmetros. Os valores máximos e mínimos das bordas do espaço de busca do primeiro caso podem ser vistos na Tabela 3.13. Os do segundo caso podem ser vistos na Tabela 3.14.

Tabela 3.13: Bordas Superior e Inferior dos parâmetros do oscilador de fase no NAO com acoplamento.

Parâmetros	Valor máximo	Valor mínimo
$c_{111} = c_{112}$(graus)	6,5	3
$c_{211} = c_{212}$(graus)	0	0
$c_{311} = c_{312}$(graus)	0	0
$c_{111} = c_{112}$(graus)	25	15
$c_{211} = c_{212}$(graus)	15	0
$c_{311} = c_{312}$(graus)	15	0
$\theta_{111}, \theta_{112}, \theta_{211}, \theta_{212}, \theta_{311}, \theta_{312}$	0	0

Parâmetros	Valor máximo	Valor mínimo
$c_{111} = c_{112}$(graus)	55	15
$c_{211} = c_{212}$(graus)	25	0
$c_{311} = c_{312}$(graus)	25	0
$O_{11} = O_{12}$(graus)	0	0
$O_{21} = O_{22}$(graus)	-10	-20
$O_{31} = O_{31}$(graus)	40	10
Todos os outros θ_{kij}(graus)	180	-180

Tabela 3.14: Bordas Superior e Inferior dos parâmetros do oscilador de fase no NAO com acoplamento e a saída sendo uma senoide.

Parâmetros	Valor máximo	Valor mínimo
$c_{111} = c_{112}$(graus)	6,5	3
$c_{111} = c_{112}$(graus)	25	15
$c_{111} = c_{112}$(graus)	55	15
$\theta_{111} = \theta_{112}$	0	0

Parâmetros	Valor máximo	Valor mínimo
$O_{11} = O_{12}$(graus)	0	0
$O_{21} = O_{22}$(graus)	-10	-20
$O_{31} = O_{31}$(graus)	40	10
Todos os outros θ_{kij}(graus)	180	-180

Nós utilizamos a mesma função de custo do caso do oscilador de Matsuoka dada pela Equação 3.6. Nesse caso foi utilizado apenas o AG, dado que com os dois métodos é possível de se obter os parâmetros, mas para se utilizar de menos tempo das simulações optou-se por utilizar somente um deles. As características do AG utilizadas são dadas pela Tabela 3.15. Também foi utilizado o NSGA-II da mesma forma que no caso anterior com os parâmetros da função de custo iguais aos ao do AG.

Tabela 3.15: Valores da função de custo utilizada no NAO com oscilador de Kuramoto acoplado.

Parâmetro	Valor
$\sigma_{\phi_{max}}$(graus)	2,5
$\sigma_{h_{max}}$(m)	0,1
$\sigma_{\psi_{max}}$(graus)	5,0
$\triangle_{h_{max}}$(m)	0,5
K_p	10000
$\mu_{pl_{max}}$(m)	0,05

3.2.3 Treinando um controlador de postura usando aprendizagem por reforço

Também foi utilizado a aprendizagem por reforço para treinar um controlador de postura para o NAO na marcha gerada pelo oscilador de Kuramoto com acoplamento com a dinâmica do movimento no NAO. Para isso foi considerado que o ambiente é composto pelo robô mais a CPG que gera seu movimento. A ação, $\overline{a} = (a_{qa}, a_{ja})^T$, do agente é somar um valor aos ângulos de arfagem do quadril, a_{qa}, e arfagem do joelho, a_{ja}, de forma a corrigir a postura do robô, minimizando a velocidade de arfagem do corpo do robô, ω_{ϕ_c}. O estado utilizado, $\overline{s} = (\phi_{osc}, \omega_{\phi_c})^T$, foram a fase do oscilador, ϕ_{osc}, e a a velocidade angular do ângulo de

arfagem do corpo, ω_{ϕ_c}. A recompensa foi dada pela seguinte expressão:

$$r_t = \begin{cases} -1 & \text{se } |\phi_c| < \frac{\pi}{4} \text{ ou } |\psi_c| < \frac{\pi}{4}, \\ 0 & \text{se } |\omega_{\phi_c}| > \omega_{\phi_{cmax}}, \\ 1 - \frac{|\omega_{\phi_c}|}{\omega_{\phi_{cmax}}} & \text{caso contrário.} \end{cases}, \tag{3.16}$$

onde ψ_c é o ângulo de rolagem do corpo do robô, ϕ_c é o ângulo de arfagem do corpo do robô e $\omega_{\phi_{cmax}}$ é o valor máximo da velocidade angular do ângulo de arfagem do corpo permitida. Foi utilizado o método ator crítico como mostrado na Figura 3.6. O diagrama de blocos do método pode ser observador na Figura 3.7.

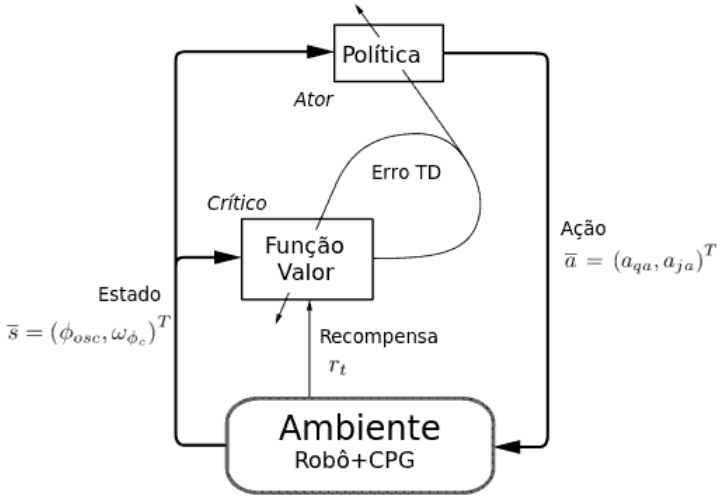


Figura 3.6: Estrutura do método ator crítico para corrigir postura do robô.

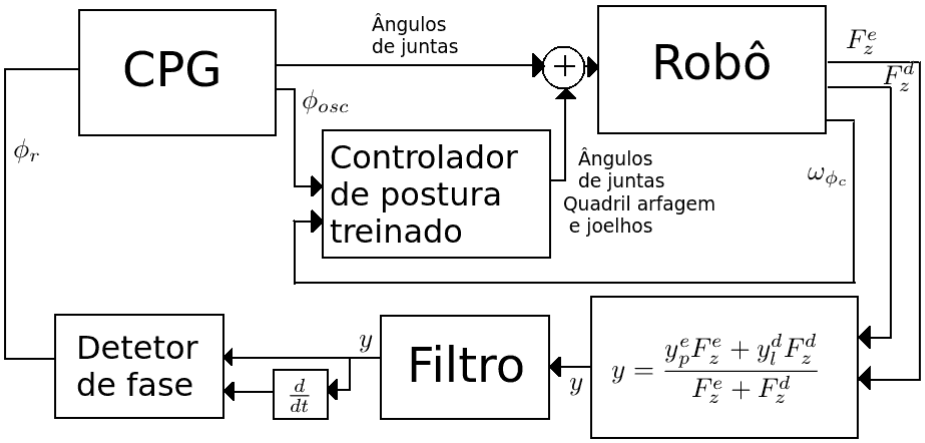


Figura 3.7: Diagrama de blocos do o oscilador de Kuramoto com acoplamento e do controle de postura.

Como os estados e as ações são contínuas, a função valor $V_\theta(\overline{s})$ foi parametrizada pelo vetor de parâmetros $\overline{\theta}_t = (\theta_t(1), \theta_t(2), ..., \theta_t(n))^T$. Foi utilizada a aproximação linear nos parâmetros com vetor de características, $\overline{\phi_s} = (\phi_s(1), \phi_s(2), ..., \phi_s(n))^T$, obtido pelo *tile coding*, de forma que $V_\theta(\overline{s}) = \overline{\theta}_t^T \overline{\phi_s}$. A política utilizada foi um política gaussiana $\pi(\overline{s}, \overline{a}, \{\overline{\mu_\psi}, \Sigma\})$ onde a média é parametrizada pelo vetor $\overline{\psi}_t = (\psi_t(1), \psi_t(2), ..., \psi_t(n))^T$ e é dada como $\overline{\mu_\psi(\overline{s})} = \overline{\psi}_t^T \overline{\phi_s}$. Dessa forma,

$$\nabla_\psi \log \pi_\psi(s,a) = J_\psi^T(\mu_\psi(s)) \nabla_\mu \log \pi_\psi(s,a,\{\overline{\mu},\Sigma\}) = \overline{\phi_s}(\overline{a}-\overline{\mu})^T \Sigma^{-1}.$$

Foi utilizado o método TD(λ) com gradiente para atualização da função valor V_θ e o método da política gradiente com TD(λ) para atualizar a política, resultando nas seguintes equações para atualização:

$$\overline{\theta}_{t+1} = \overline{\theta}_t + \alpha\delta_t \overline{e_{vt}}, \tag{3.17}$$

$$\delta_t = r_{t+1} + \gamma V_{\theta t}(s_{t+1}) - V_{\theta t}(s_t), \tag{3.18}$$

$$\overline{e_{vt}} = \gamma\lambda \overline{e_{vt-1}} + \overline{\phi_s}, \tag{3.19}$$

$$\overline{\psi}_{t+1} = \overline{\psi}_t + \beta\delta_t \overline{e_{\pi t}}, \tag{3.20}$$

$$\overline{e_{\pi t}} = \gamma\lambda \overline{e_\pi}_{t-1} + \overline{\phi_s}(\overline{a}-\overline{\mu})^T \Sigma^{-1}. \tag{3.21}$$

Tabela 3.16: Parâmetros usados no método de aprendizagem por reforço com a saída do oscilador sendo uma senoide.

Parâmetro	Valor
$t_{transiente}$	0,5s
Número de divisões para fase, ϕ_{osc}, no *tile coding*	24
Número de divisões para velocidade angular, ω_{ϕ_c}, no *tile coding*	12
ϕ_{osc} mínimo e máximo usados no *tile coding*	0 e 2π rad
ω_{ϕ_c} mínimo e máximo usados no *tile coding*	-100 e 100 graus/s
γ	0,9

Parâmetro	Valor
λ	0,9
α	0,5
β	0,005
Σ	$\begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{bmatrix}$ graus
$\overline{a}_{max}$	$(10,10)^T$ graus
$\overline{a}_{min}$	$(-10,-10)^T$ graus

Esse método foi aplicado em duas marchas diferentes, uma onde a saída do oscilador é dada apenas por uma senoide e uma outra marcha dada por uma Série de Fourier, que tinha desempenho melhor que a senoide. A marcha do robô ocorreu durante 5 segundos e após 5 segundos a posição e a marcha eram reiniciadas. Isso foi chamado de um episódio. Nos dois casos foi esperado um tempo de transiente $t_{transiente}$ para o controlador e o treinamento

se iniciarem em cada episódio. Isso foi feito pois o transiente tem uma dinâmica diferente do que o restante da marcha. A ação também foi limitada por um valor máximo, $\overline{a}_{max}$ e mínimo, $\overline{a}_{min}$. Os parâmetros usados no método de aprendizagem por reforço com a saída do oscilador sendo uma senoide podem ser vistos na Tabela 3.16. Para o caso da Série de Fourier na Tabela 3.17.

Tabela 3.17: Parâmetros usados no método de aprendizagem por reforço com a saída do oscilador sendo uma Série de Fourier.

Parâmetro	Valor
$\overline{a}_{max}$	$(3, 3)^T$ graus
$\overline{a}_{min}$	$(-3, -3)^T$graus
Restante dos parâmetros.	Os mesmos da Tabela 3.16.

3.3 PLATAFORMAS UTILIZADAS

3.3.1 Plataforma Bioloid

Inicialmente foi utilizada a plataforma Bioloid da Robotis, que pode ser vista na Figura 3.8. Ele possui 16 graus de liberdade e foi adicionada a plataforma uma IMU composta por um girômetro de 2 eixos e um acelerômetro de 3 eixos. Também foram adicionados sensores de pressão no pé e um sistema embarcado (gumstix) [8].

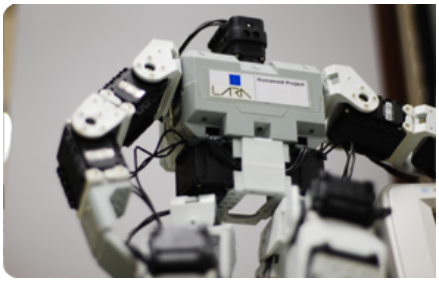


Figura 3.8: Plataforma Bioloid.

Para simulação do robô foi utilizado o simulador *Bioloid Control: Physics Sim* encontrado em `http://bioloidcontrol.sourceforge.net/index.php?category=11`[1]. O robô em ambiente simulado pode ser visto na Figura 3.9. Esse simulador utiliza a biblioteca *ODE, open dynamics engine*, para simular o *bioloid* e sua interação física com o ambiente. A biblioteca ODE é uma biblioteca de código aberto para simular a dinâmica de corpos rígidos. Não se pode realizar testes no robô real, pois o mesmo estava com problemas de *hardware* que não foram solucionados há tempo.

[1]Ultima vez acessado dia 14/06/12

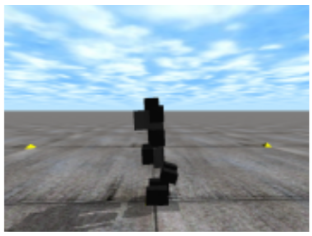


Figura 3.9: Simulador da plataforma Bioloid.

3.3.2 Plataforma NAO

Também testes foram realizados na plataforma NAO. Ela é produzida pela empresa Aldebaran e pode ser visto na Figura 3.10. Atualmente está na versão 4.0, possui IMU, sensor de pressão nos pés, sistema embarcado (CPU), microfones, auto falante, câmera, infravermelho, sonares, sensores de contato. Há uma versão do NAO com 21 graus de liberdade e outra com 25 graus de liberdade. No simulador utilizado, a versão do NAO possui 21 graus de liberdade, mas os graus de liberdade a mais da versão com 25 graus de liberdade possui 4 graus de liberdade a mais nos punhos e mãos do robô, não afetando o método utilizado.

Figura 3.10: Plataforma NAO.

O ambiente de simulação para o robô é o simulador Webots, que possui a versão "Webots for NAO" e a versão gratuita dele permite a simulação do robô NAO no contexto da robocup, que é uma competição de robótico que utiliza a plataforma para serem jogadores de futebol. O robô no ambiente de simulação pode ser visto na Figura 3.11. O Webots também utiliza a biblioteca *ODE, open dynamics engine* e foi utilizada a versão 7.4.0 do simulador.

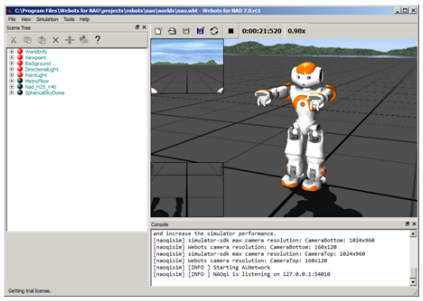


Figura 3.11: Simulação da plataforma NAO.

4 RESULTADOS EXPERIMENTAIS

4.1 RESULTADOS NO BIOLOID

Os métodos descritos na Subseção 3.2.1 foram utilizados no Bioloid por meio do simulador "Bioloid Control: Physics Sim"[1]. Os osciladores de Matsuoka foram iniciados com um sinal impulso. Em cada simulação o robô se moveu por 20 segundos e o passo de tempo foi de 0,05 segundos. A Figura 4.1 mostra a evolução do melhor resultado que obtivemos com cada algoritmo.

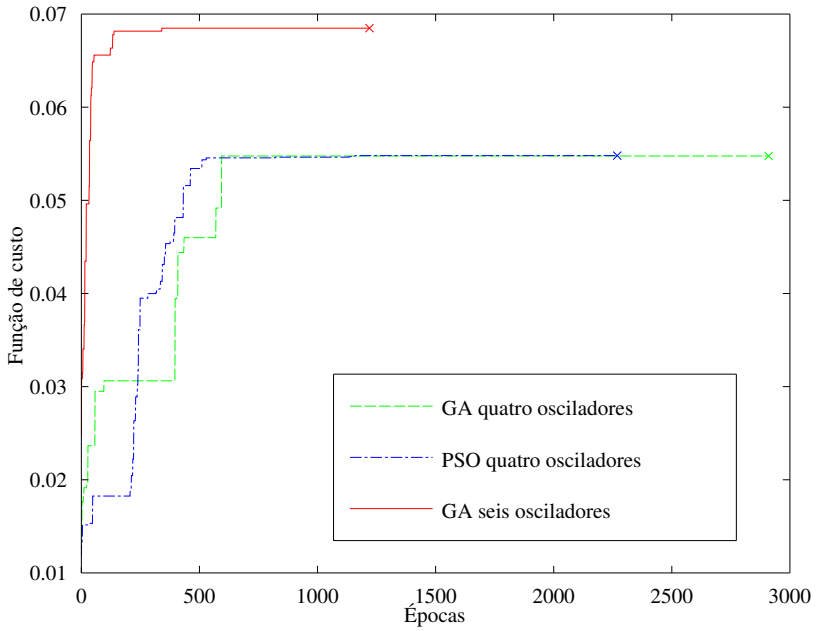


Figura 4.1: Melhor evolução da função de custo no Bioloid (o x é o momento em que paramos a busca).

Cada método obteve um ótimo local após algumas épocas, e após muitas épocas o método não conseguiu encontrar parâmetros melhores, por isso nós paramos a simulação. O AG e o PSO acharam ótimos locais com quase o mesmo valor num número de épocas próximo. Devido à natureza do PSO em que as partículas se movem no espaço de busca obtiveram-se muitos aperfeiçoamentos até o ótimo local, diferentemente do AG que obtém os parâmetros através da combinação de parâmetros de parentes selecionados. Também é possível notar que ao se utilizarem seis osciladores ao invés de quatro osciladores os resultados foram melhores. A causa disso será explicada mais a frente.

[1]Disponível em `http://bioloidcontrol.sourceforge.net/index.php?category=11`. Última vez acessado: 24/08/12

Os melhores resultados obtidos com o PSO estão na Tabela 4.1. A Figura 4.2 mostra a saída dos osciladores e a Figura 4.3 mostra a posição do corpo do robô durante o movimento com os parâmetros otimizado com o PSO. Uma marcha estável foi obtida dessa forma que pode ser vista na Figura 4.8.

Os melhores resultados obtidos com o AG estão na Tabela 4.2. A Figura 4.4 mostra a saída dos osciladores e a Figura 4.5 mostra a posição do corpo do robô durante o movimento com os parâmetros otimizado com o AG. Como no outro método foi possível obter uma marcha estável que pode ser vista na Figura 4.9. Sendo assim possível obter os parâmetros com os dois algoritmos, embora eles tenham suas diferenças.

Os melhores resultados obtidos ao se utilizar mais osciladores estão na Tabela 4.3. A Figura 4.6 mostra a saída dos osciladores e a Figura 4.7 mostra o movimento do corpo do robô. Foi possível obter uma marcha eficiente em que o robô se locomoveu mais que nos outros casos. Isso ocorreu porque ao se utilizar o ângulo de rolagem do quadril o robô pôde utilizar seu balanço lateral de forma com que ele contribua na marcha.

Tabela 4.1: Resultados para o Bioloid usando o algoritmo PSO.

Parâmetros	Valor
$w_{lh-lk} = w_{rh-rk}$	0,04326
$w_{lk-lh} = w_{rk-rh}$	-0,26601
$c_{lh} = c_{rh}$	0,46759
$c_{lk} = c_{rk}$	0,20217
função de custo	0,05480
épocas	1218
$\triangle_p$ (m)	0,39756
σ_ϕ (rad)	0,00610
σ_h (m)	0,00165
σ_ψ (rad)	0,01480
$\triangle_h$ (m)	0,00955

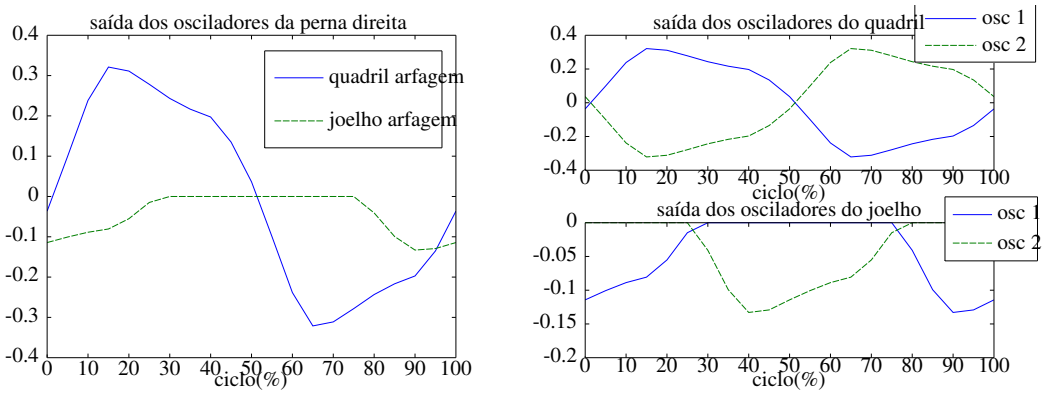


Figura 4.2: Saída dos osciladores para o Bioloid usando o PSO.

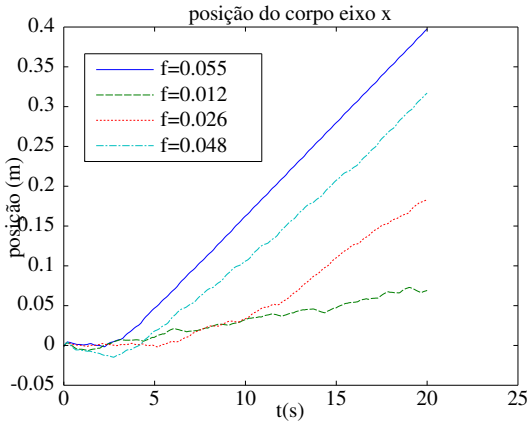


Figura 4.3: Posição do corpo do Bioloid usando o PSO.

Tabela 4.2: Resultados para o Bioloid usando o algoritmo AG.

Parâmetro	Valor
$w_{lh-lk} = w_{rh-rk}$	0,03993
$w_{lk-lh} = w_{rk-rh}$	-0,27091
$c_{lh} = c_{rh}$	0,30000
$c_{lk} = c_{rk}$	0,13712
função de custo	0,05476
épocas	594
$\triangle_p$ (m)	0,31366
σ_ϕ (rad)	0,00475
σ_h (m)	0,00080
σ_ψ (rad)	0,00218
$\triangle_h$ (m)	0,00716

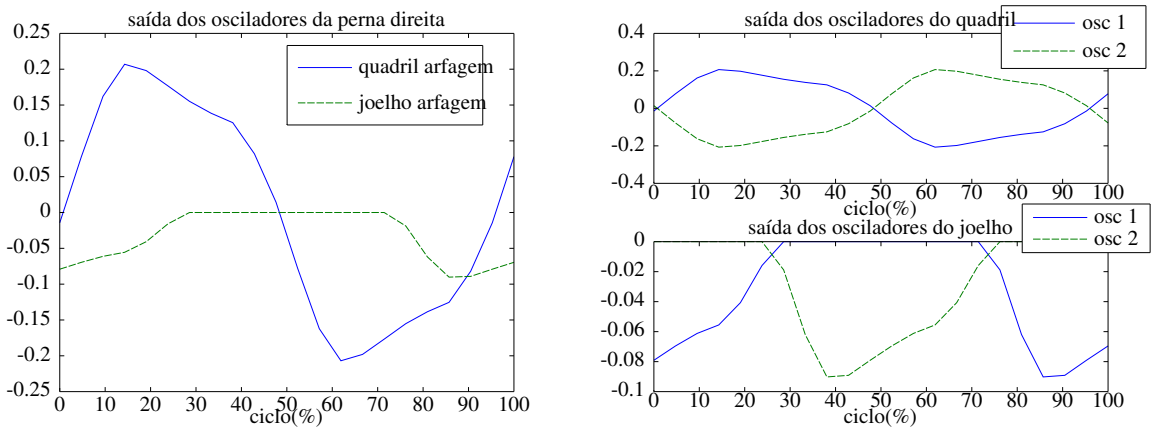


Figura 4.4: Saída dos osciladores no Bioloid para o AG.

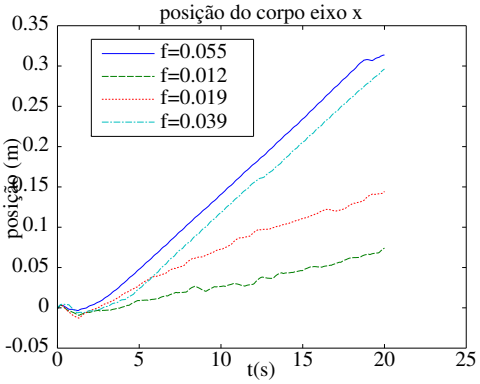


Figura 4.5: Posição do corpo do Bioloid para o AG.

Tabela 4.3: Parâmetros obtidos no Bioloid ao se utilizar 6 osciladores de Matsuoka.

Parâmetros	Valor
$w_{lh_y-lh_x} = w_{rh_y-rh_x}$	-0,07891
$w_{lh_x-lh_y} = w_{rh_x-rh_y}$	0,55319
$w_{lh_x-lk} = w_{rh_x-rk}$	-1,18586
$w_{lk-lh_x} = w_{rk-rh_x}$	-0,28354
$c_{lh_y} = c_{rh_y}$	0,14617
$c_{lh_x} = c_{rh_x}$	0,31017
$c_{lk} = c_{rk}$	0,21294
função de custo	0,06968
épocas	1047
$\triangle_p$ (m)	1,19509
σ_ϕ (rad)	0,03261
σ_h (m)	0,00284
σ_ψ (rad)	0,07086
$\triangle_h$ (m)	0,01574

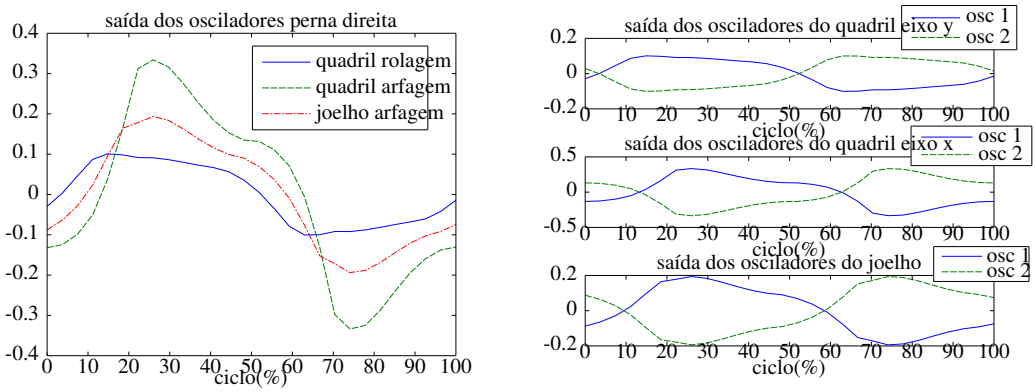


Figura 4.6: Saída dos osciladores no Bioloid ao se utilizar 6 osciladores de Matsuoka.

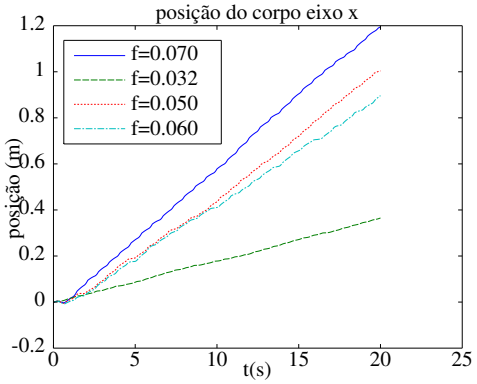


Figura 4.7: Posição do corpo do Bioloid ao se utilizar 6 osciladores de Matsuoka.

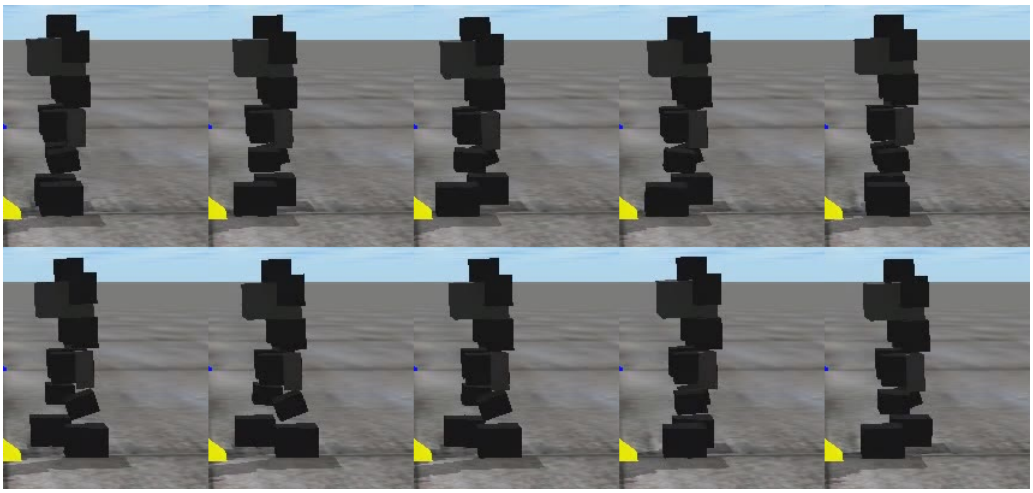


Figura 4.8: Marcha do Bioloid obtida pelo PSO.

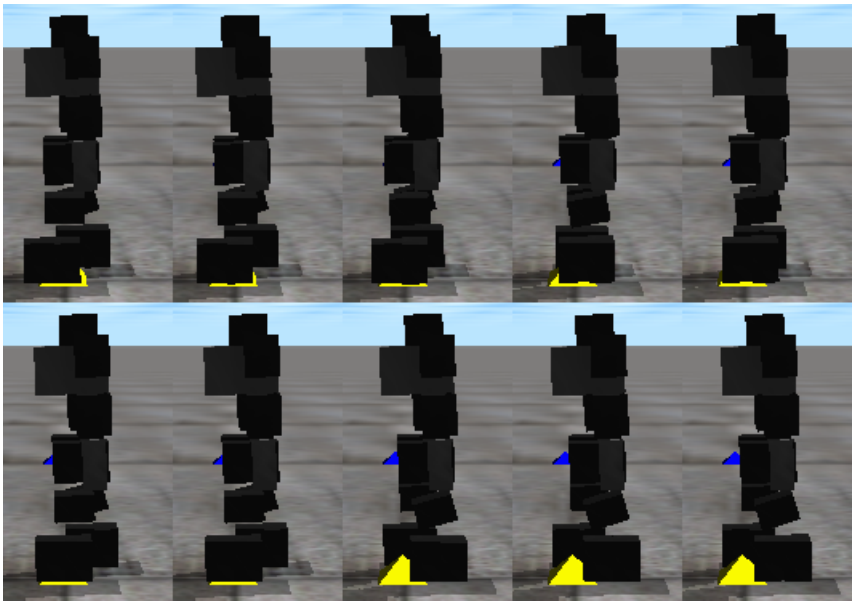


Figura 4.9: Marcha do Bioloid obtida pelo AG.

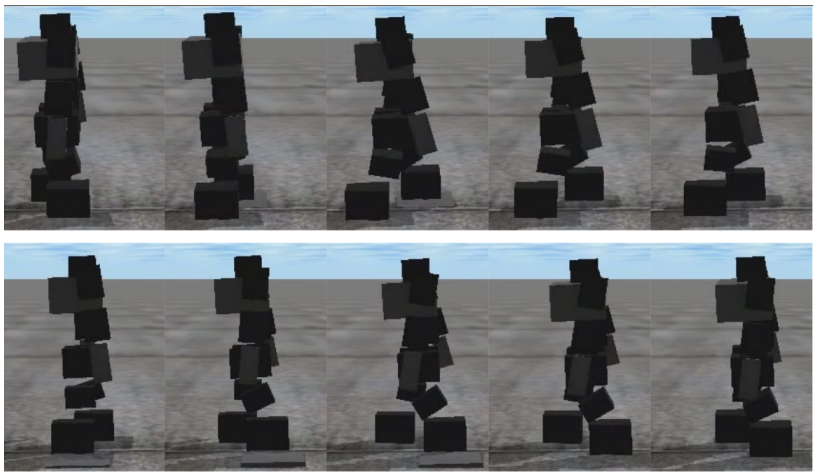


Figura 4.10: Marcha obtida no Bioloid ao se utilizar 6 osciladores de Matsuoka.

4.2 RESULTADOS NO NAO

Os métodos descritos na Subseção 3.2.2 foram utilizados no NAO utilizando o simulador Webots. Foi utilizado o passo de tempo de 0,04 segundos e em cada simulação o robô se moveu por 5 segundos.

4.2.1 Utilizando o oscilador de Matsuoka no NAO

Como no caso do Bioloid, os osciladores de Matsuoka foram iniciados com um sinal impulso. Na Figura 4.11 é possível ver a evolução da função de custo para um resultado que obtivemos usando o AG e o PSO. É possível ver que nesse caso foi possível obter um valor da função de custo próximo para os dois algoritmos, assim como no caso do Bioloid, mostrando que é possível utilizar diferentes plataformas para os métodos.

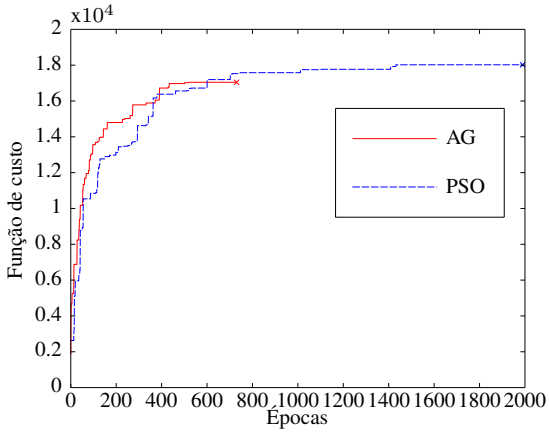


Figura 4.11: Melhor evolução da função de custo no NAO usando o oscilador de Matsuoka (o x é o momento em que paramos a busca).

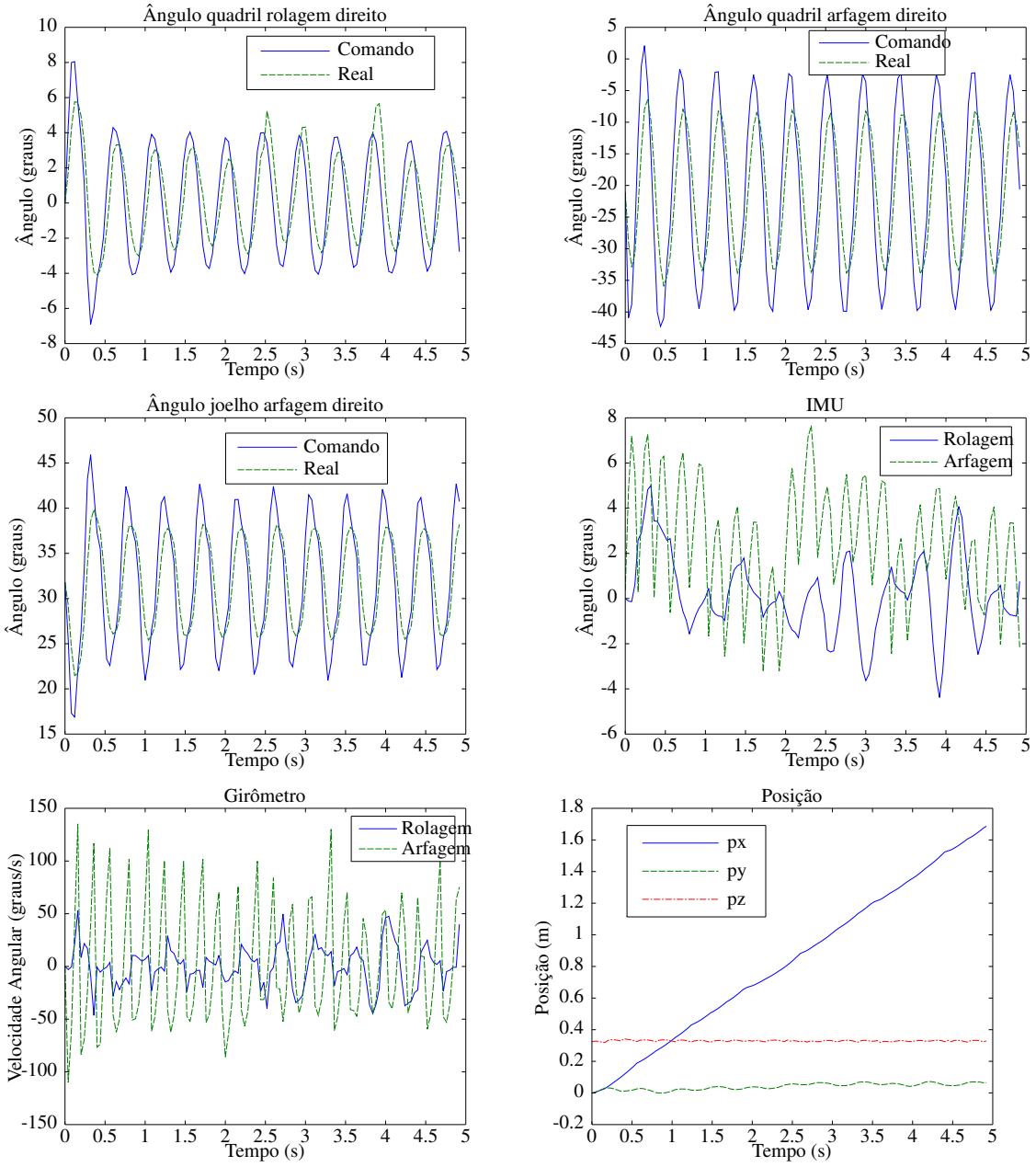


Figura 4.12: Resultados usando o oscilador de Matsuoka no NAO com parâmetros otimizados pelo AG.

Na Figura 4.12 é possível ver o ângulo de rolagem e de arfagem do quadril direito, o ângulo de arfagem do joelho direito, os dados da IMU (ângulo de arfagem e rolagem do corpo e suas respectivas velocidades) e a posição do robô para uma das marchas obtidas pelo AG. Já na Figura 4.13 o mesmo para o caso do PSO. Também é possível ver as respectivas marchas obtidas na Figura 4.14 para o AG e na Figura 4.15 para o PSO. Como visto nas Figuras, nos dois casos foi possível obter marchas estáveis, embora os sinais da IMU não tenham uma forma periódica, devido aos osciladores estarem em malha aberta,i.e., os osciladores não estão utilizando nenhum *feedback*. Ainda assim o ângulo da IMU está menor do que 10 graus, o que é um valor pequeno visto não haver nenhum controle para a postura e o AG só

considerar um limite para o desvio padrão do mesmo.

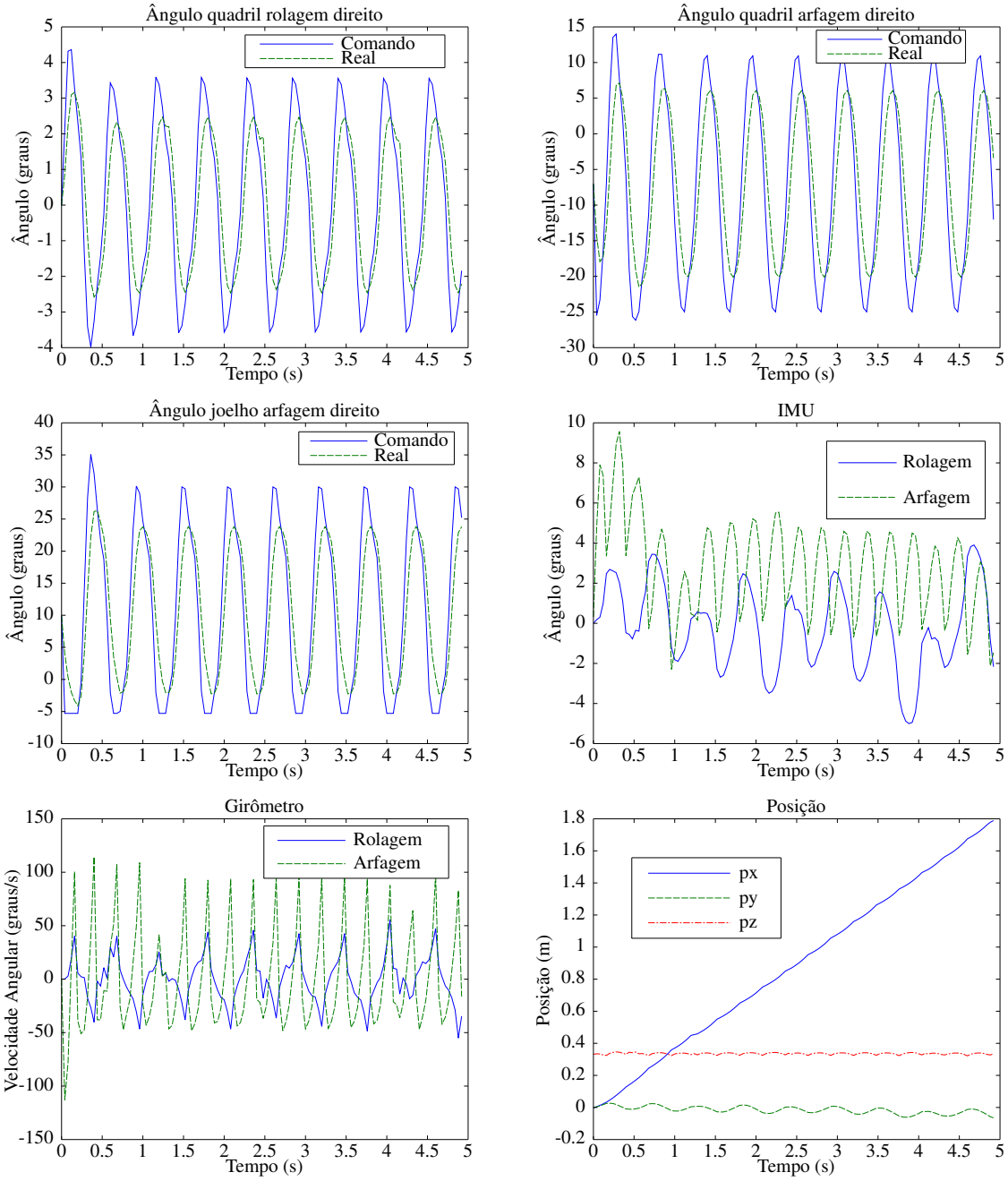


Figura 4.13: Resultados usando o oscilador de Matsuoka no NAO com parâmetros otimizados pelo PSO.

Como com ambos os métodos, AG e PSO, foi possível obter marchas estáveis, para diminuir o tempo de simulação, foi escolhido apenas o AG para otimizar os parâmetros dos outros osciladores.

Na Figura 4.17 é possível ver as três primeiras frentes de Pareto na geração final obtidas usando o NSGA-II. Lembrando que a função de custo 1 nesse caso está sendo a distância em metros percorrida multiplicado por 10000 e a função de custo 2 é dada por 1000 subtraído do

trabalho total em Joules. Como pode ser visto, foram obtidos parâmetros em que em geral quanto maior a distância percorrida maior foi o trabalho realizado. Dessa forma, é possível realizar um *trade-off* entre o trabalho realizado pelo robô e a distância percorrida.

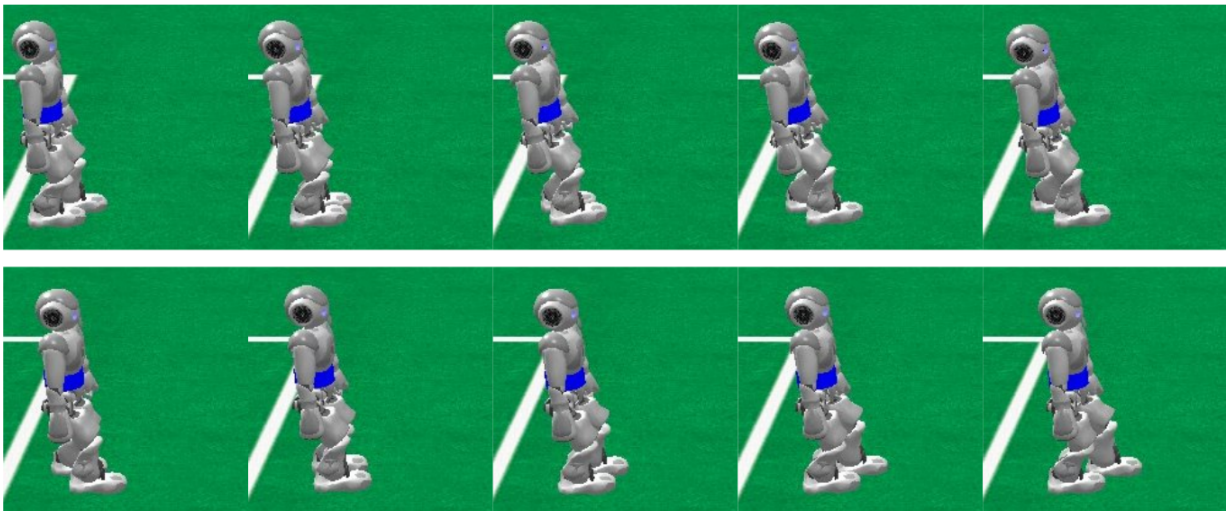


Figura 4.14: Marcha do NAO usando o oscilador de Matsuoka com parâmetros otimizados pelo AG.

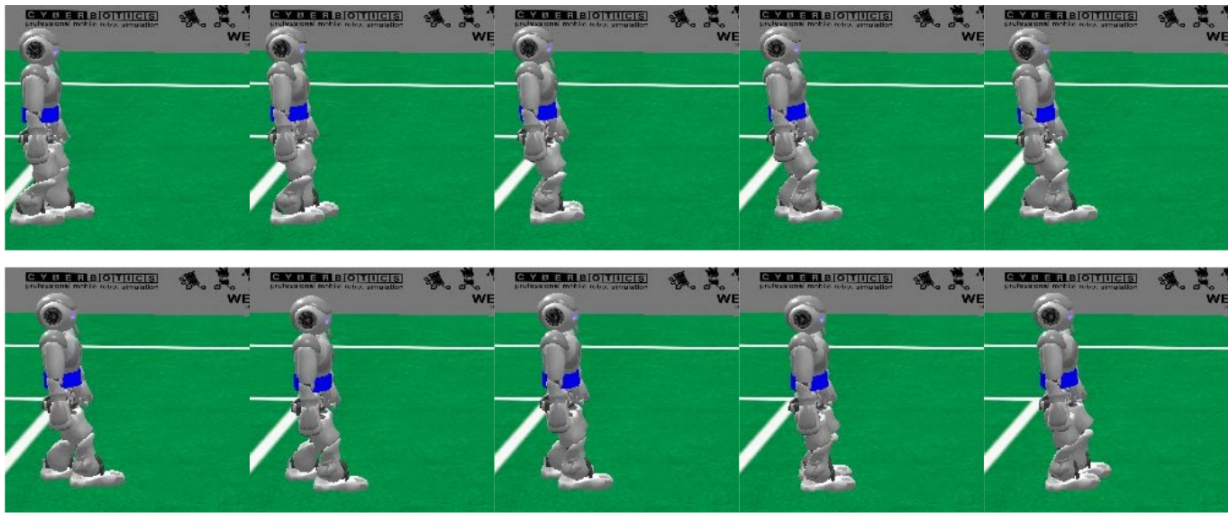


Figura 4.15: Marcha do NAO usando o oscilador de Matsuoka com parâmetros otimizados pelo PSO.

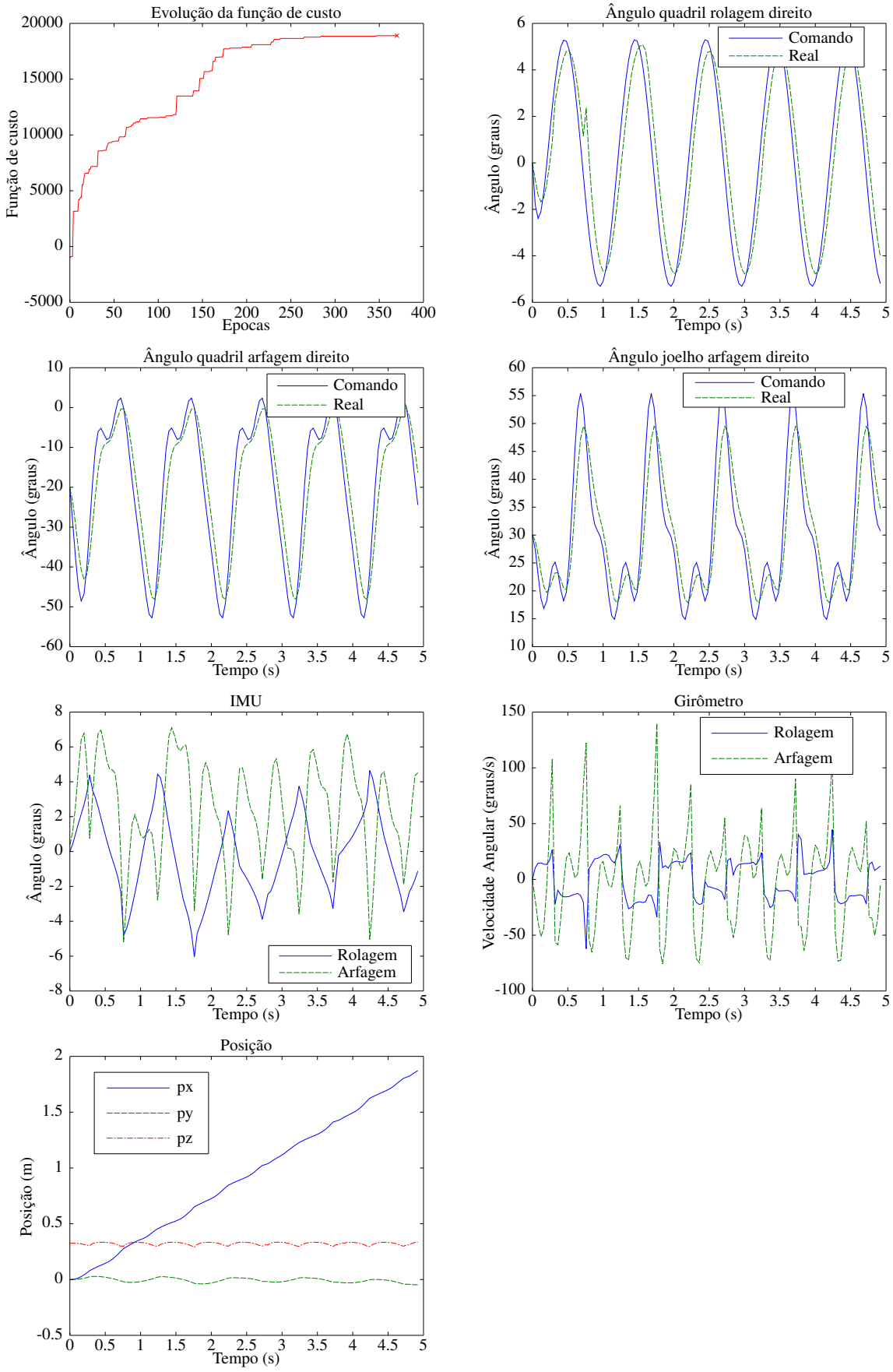


Figura 4.16: Resultados usando o oscilador de Kuramoto com acoplamento no NAO com parâmetros otimizados pelo AG.

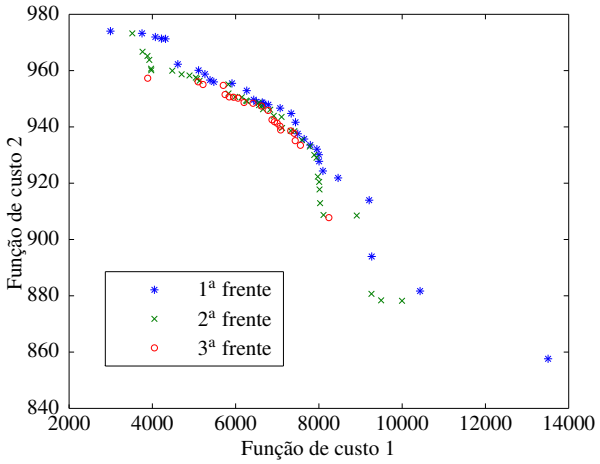


Figura 4.17: As 3 primeiras frentes de Pareto na geração final obtidas usando o NSGA-II com o oscilador de Matsuoka.

4.2.2 Utilizando o oscilador de Kuramoto no NAO

Na Figura 4.16 é possível ver a evolução da função de custo, o ângulo de rolagem e de arfagem do quadril direito, o ângulo de arfagem do joelho direito, os dados da IMU (ângulo de arfagem e rolagem do corpo e suas respectivas velocidades) e a posição do robô para uma das marchas obtidas pelo AG. E na Figura 4.18 é possível ver a marcha obtida. Mesmo com um número de parâmetros maior do que no caso passado, pode-se ver que foi possível obter parâmetros para realizar uma marcha no robô. Novamente pode ser visto que os sinais da IMU não têm uma forma periódica, mas estão menores do que 6 graus aproximadamente, o que é um valor pequeno, visto não haver nenhum controle especifico para postura e o AG só considerar um limite para o desvio padrão do mesmo. Uma nota que deve ser feita é que devido a forma que os osciladores foram usados, da saídas serem uma serie de Fourier, eles deram outras possibilidades de formas de onda, o que em certas ocasiões provocou o AG não conseguir sair de mínimos locais não desejáveis, como por exemplo andar para trás, que acarreta num valor de -1000 para função de custo. Nesses casos, o AG era reinicializado, visto que sua inicializa possui um grau de randomização.

E na Figura 4.19 é possível ver as três primeiras frentes de Pareto na geração final obtidas usando o NSGA-II. Lembrando que a função de custo 1 nesse caso está sendo a distância em metros percorrida multiplicado por 10000 e a função de custo 2 é dada por 1000 subtraído do trabalho total em Joules. Novamente, foram obtidos parâmetros em que em geral quanto maior a distância percorrida maior foi o trabalho realizado. Dessa forma, é possível realizar um *trade-off* entre o trabalho realizado pelo robô e a distância percorrida.

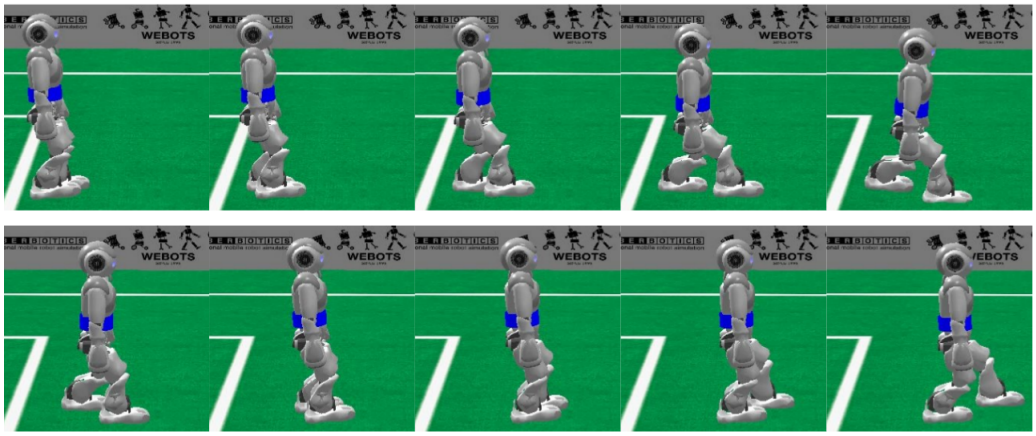


Figura 4.18: Marcha do NAO usando o oscilador de Kuramoto com parâmetros otimizados pelo AG.

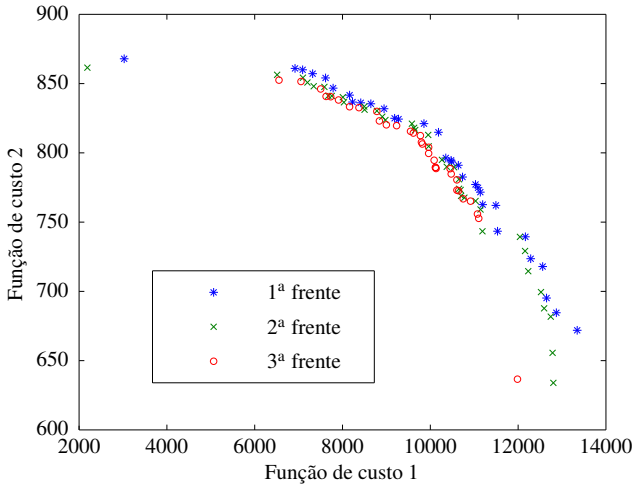


Figura 4.19: As 3 primeiras frentes de Pareto na geração final obtidas usando o NSGA-II com o oscilador de Kuramoto.

4.2.3 Utilizando o oscilador de Kuramoto com acoplamento com a dinâmica do movimento no NAO

Na Figura 4.21 é possível ver a evolução da função de custo, o ângulo de rolagem e de arfagem do quadril direito, o ângulo de arfagem do joelho direito, os dados da IMU (ângulo de arfagem e rolagem do corpo e suas respectivas velocidades), a posição do robô e a posição y_{CdP} do centro de pressão para uma das marchas obtidas pelo AG. E na Figura 4.20 a marcha obtida. É possível ver na evolução da função de custo que esse método obteve parâmetros com um valor da função de custo melhores que os casos anteriores e que ele obteve um valor próximo do valor que os outros osciladores obtiveram em um número de épocas menor que nos outros casos. Além disso, a possibilidade de outras formas de onda por conta dela ser gerada por uma Série de Fourier não influenciando tanto no AG convergir em mínimos locais indesejados como no caso anterior. Essas duas características podem ser explicadas pelo fato

de o acoplamento entre o oscilador e a dinâmica do movimento servir como um “catalisador” para o método. Novamente pode ser visto que os sinais da IMU não têm uma forma periódica, embora depois do transiente de inicialização eles possuam uma forma “quase” periódica e com valor menores do que no outro caso. Dessa forma, foi possível obter melhorias na marcha por conta do acoplamento do oscilador com a dinâmica do movimento. Mas uma desvantagem do acoplamento é “forçar” o oscilador a ter uma frequência fixa dada pela dinâmica do robô. Embora ao se diminuir a constante de acoplamento a frequência da forma de onda será mais próxima a frequência natural do oscilador ao custo de se ter um acoplamento menor.

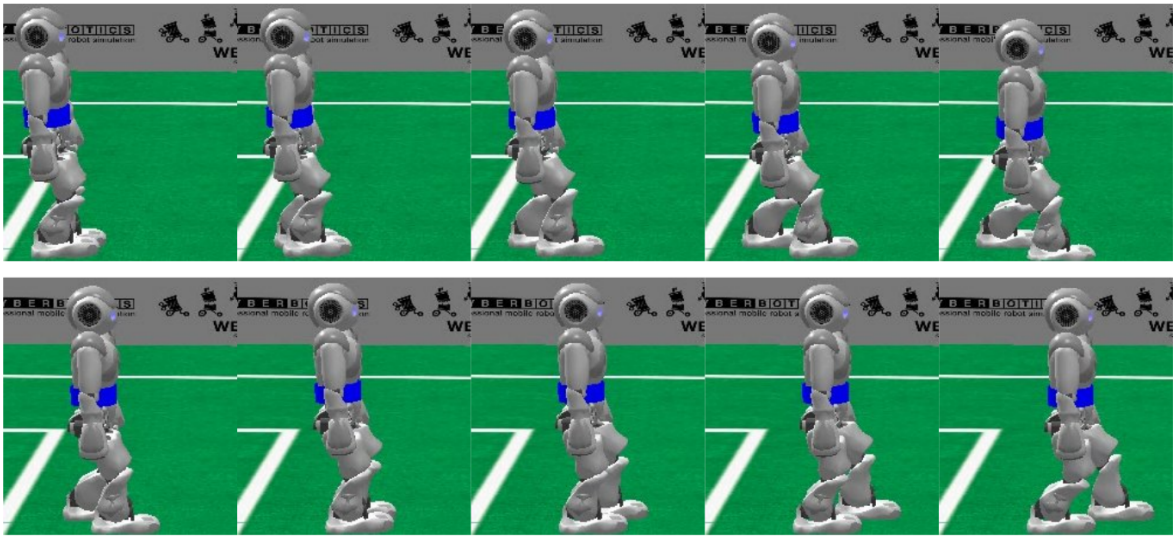


Figura 4.20: Marcha do NAO usando o oscilador de Kuramoto com acoplamento com parâmetros otimizados pelo AG.

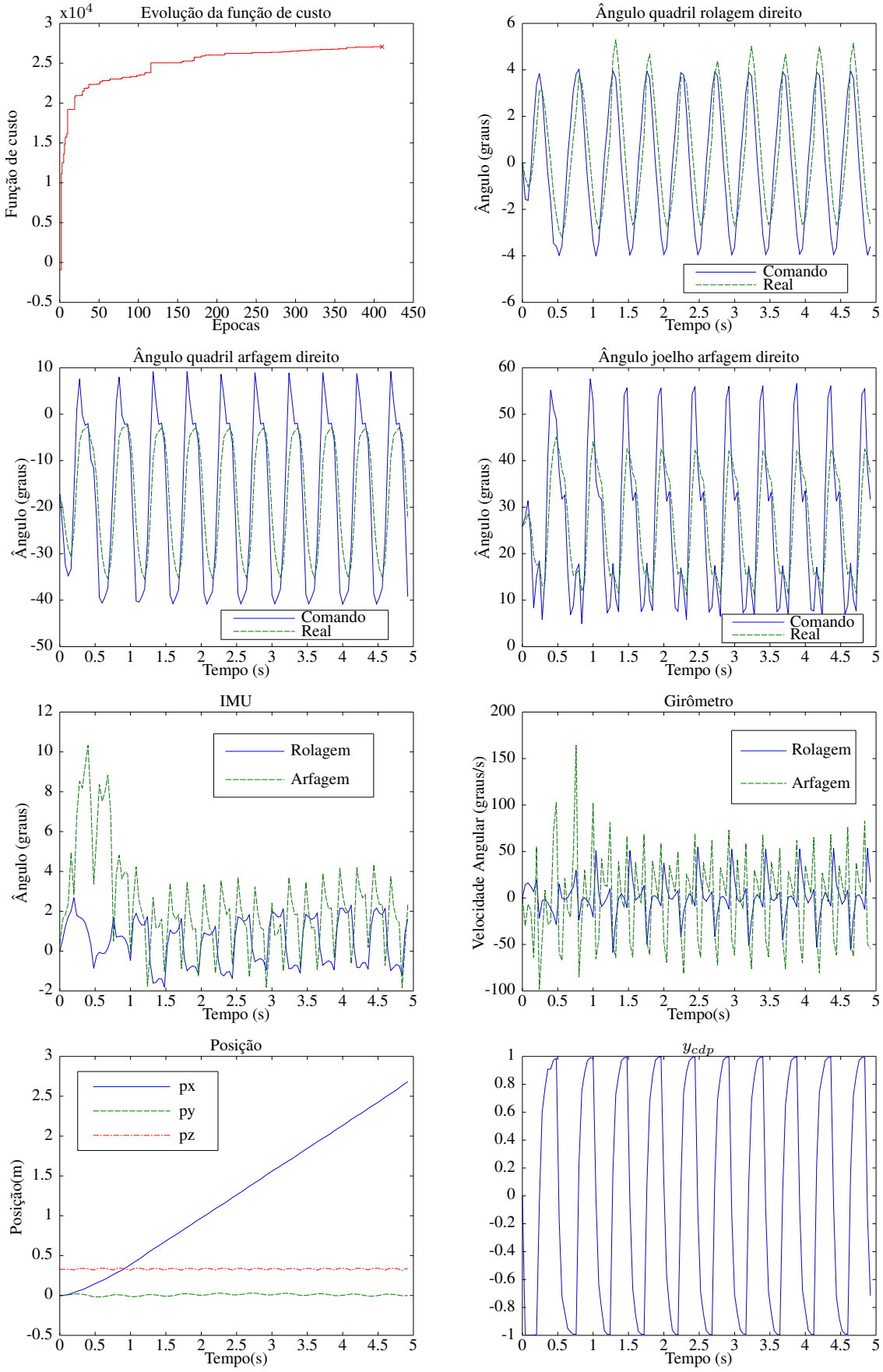


Figura 4.21: Resultados usando o oscilador de Kuramoto com acoplamento no NAO com parâmetros otimizados pelo AG.

Na Figura 4.22 é possível ver as três primeiras frentes de Pareto na geração final obtidas usando o NSGA-II. Lembrando que a função de custo 1 nesse caso está sendo a distância em metros percorrida multiplicado por 10000 e a função de custo 2 é dada por 1000 subtraído do trabalho total em Joules. Novamente, foram obtidos parâmetros em que em geral quanto maior a distância percorrida maior foi o trabalho realizado. Dessa forma, é possível realizar um *trade-off* entre o trabalho realizado pelo robô e a distância percorrida.

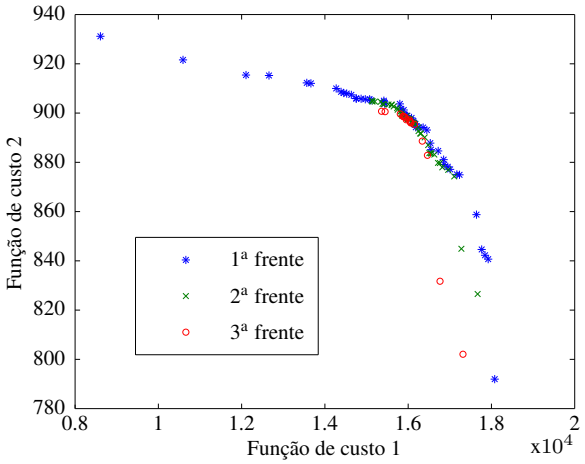


Figura 4.22: As 3 primeiras frentes de Pareto na geração final obtidas usando o NSGA-II com o oscilador de Kuramoto com acoplamento no NAO.

Também foram realizados testes com a saída de um oscilador sendo apenas a fundamental da Série de Fourier, ou seja, uma senoide, um sinal mais simples. Na Figura 4.23 é possível ver a evolução da função de custo, o ângulo de rolagem e de arfagem do quadril direito, o ângulo de arfagem do joelho direito, os dados da IMU (ângulo de arfagem e rolagem do corpo e suas respectivas velocidades), a posição do robô e a posição y_{CdP} do centro de pressão para uma das marchas obtidas pelo AG. Na Figura 4.24 é possível observar a marcha obtida. Como anteriormente, o acoplamento ajudou a encontrar parâmetros com um valor da função de custo nas primeiras iterações melhores do que os outros osciladores encontraram nas suas primeiras iterações. Nesse caso, o robô percorreu uma distância menor, a variação obtida para o sinal da IMU e do girômetro foram maiores que no caso anterior, onde a Série de Fourier é utilizada com o acoplamento. Mas embora isso, foi possível mostrar que com o acoplamento mesmo com um sinal simples de uma senoide foi possível obter uma marcha.

Também foi utilizado o NSGA-II nesse caso, como visto na Figura 4.25, e o mesmo foi obtido que nos casos anteriores, é possível escolher parâmetros de forma a realizar um *trade-off* entre o trabalho realizado pelo robô e a distância percorrida.

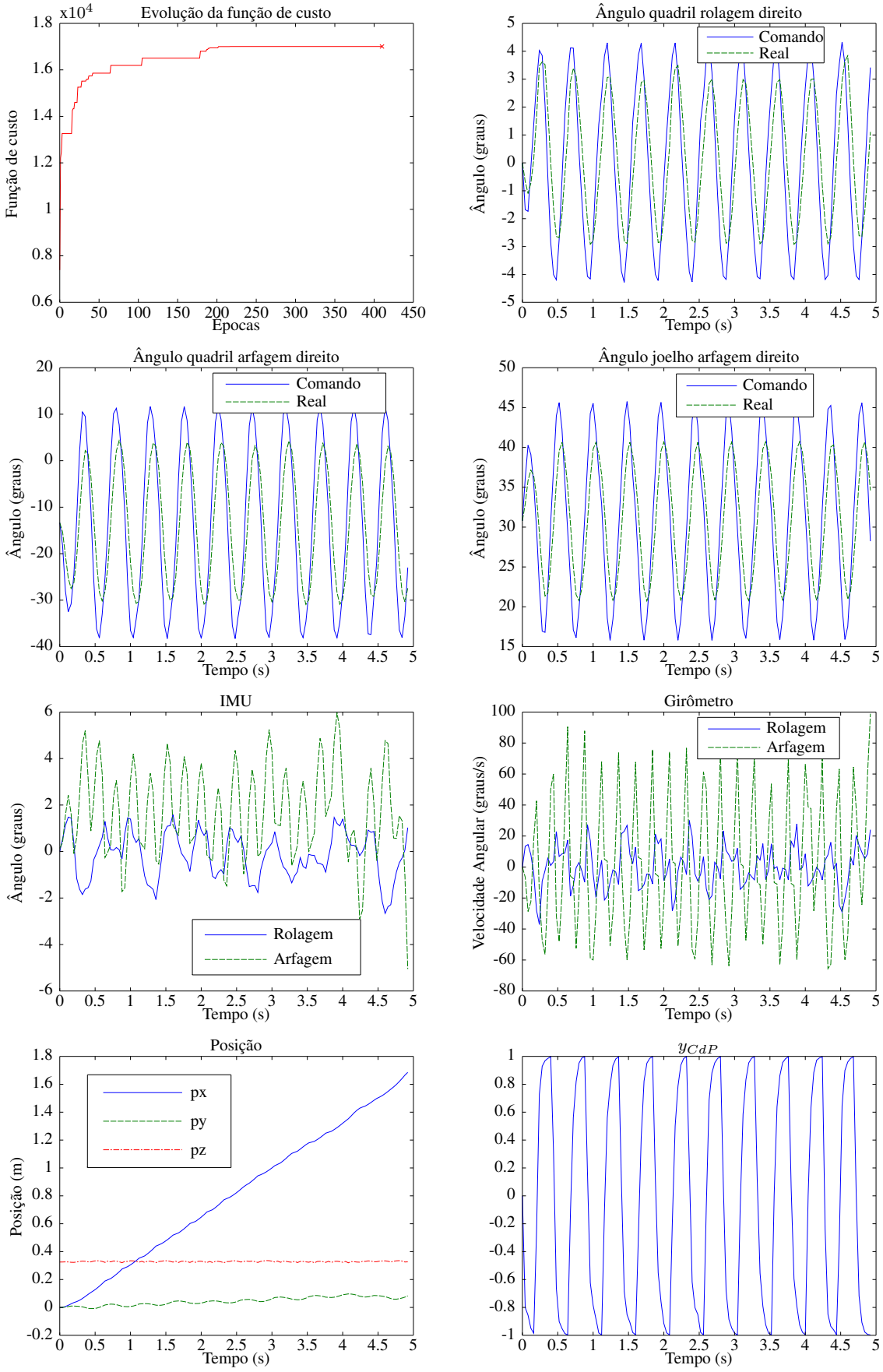


Figura 4.23: Resultados usando o oscilador de Kuramoto com saída sendo a fundamental da Série de Fourier, ou seja, uma senoide, com acoplamento no NAO com parâmetros otimizados pelo AG.

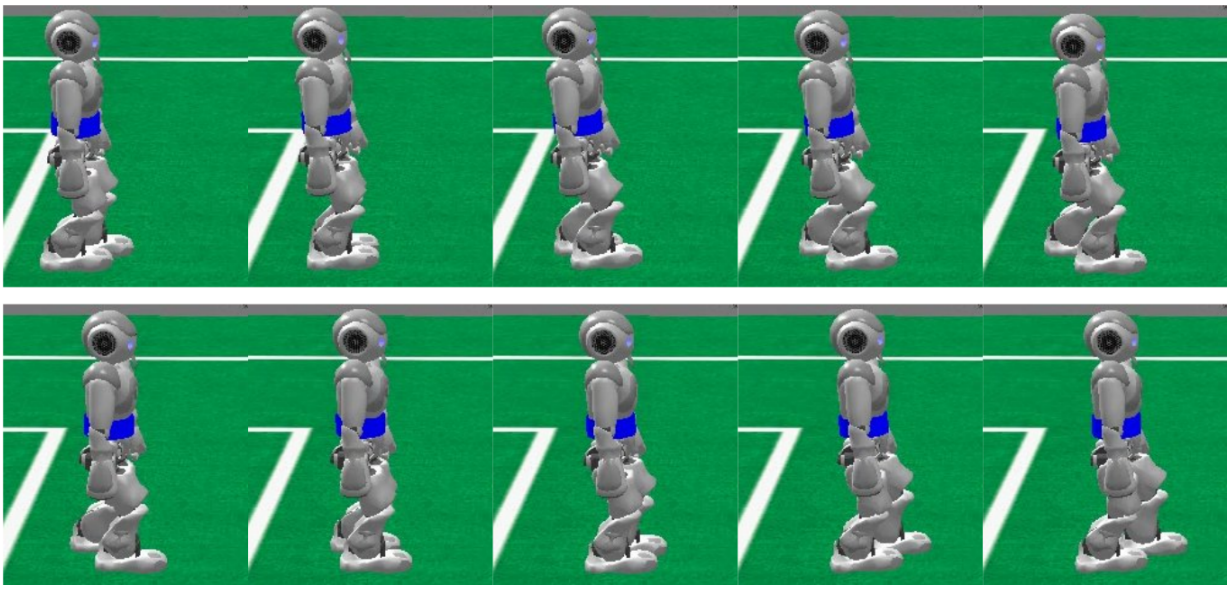


Figura 4.24: Marcha do NAO usando o oscilador de Kuramoto com acoplamento otimizada pelo AG sendo a saída apenas a fundamental da Série de Fourier, ou seja, uma senoide, com parâmetros otimizados pelo AG.

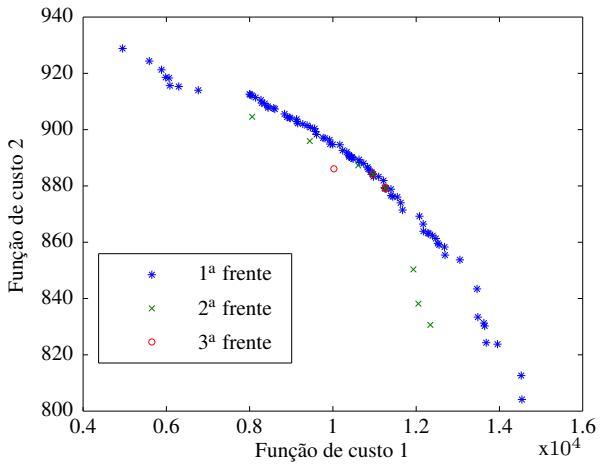


Figura 4.25: As 3 primeiras frentes de Pareto na geração final obtidas usando o NSGA-II com o oscilador de Kuramoto com saída sendo apenas a fundamental da Série de Fourier, ou seja, uma senoide, com acoplamento no NAO.

4.2.4 Treinando um controlador de postura usando aprendizagem reforçada

Os resultados para o treinamento do controle de postura usando uma saída senoidal podem ser vistos na Figura 4.26. Nela podem ser vistos os sinais do girômetro, IMU, ângulo de arfagem do quadril direito, ângulo de arfagem do joelho direito e a posição antes e depois do treinamento. Como pode ser visto nos dados do girômetro e IMU antes e depois do treinamento o treinamento funcionou, uma vez que depois do treinamento a variação da velocidade do ângulo de arfagem do corpo ficou pequena em relação a antes do treinamento

e consequentemente a variação do ângulo de arfagem também ficou pequena. Isso acarretou em mudanças nos ângulos de juntas enviado ao robô como podem ser vistas nos ângulo de arfagem do quadril e joelho direito. Um outro resultado interessante é que embora a velocidade do ângulo de rolagem do corpo não tenha sido usada nem nos estados e nem na recompensa é possível ver que o controle o tornou mais suave e também o tornou em uma "forma periódica" de sinal. Uma outra coisa que o controle acarretou foi que o robô percorreu uma distância menor com o controlador de postura, isso pois ele se tornou mais "cauteloso" no movimento para poder manter a postura.

Os resultados para o treinamento do controle de postura usando uma saída dada pela Série de Fourier podem ser vistos na Figura 4.27. Nela podem ser vistos os sinais do girômetro, IMU, ângulo de arfagem do quadril direito, ângulo de arfagem do joelho direito e a posição antes e depois do treinamento. Como pode ser visto foi possível treinar também para uma marcha com melhor performance. Para isso foi necessário diminuir o módulo do valor de ações máximo e mínimo utilizados, pois grandes valores de ações permitidos acarretaram em mudanças nessa marcha, tornando-a instável em alguns momentos e não sendo possível o treinamento dela nessa situação.

Assim foi possível corrigir a postura da marcha do robô, mas se deve observar aos valores máximo e mínimos da saída que o controlador deve ter para cada tipo de marcha (forma de onda gerada para as juntas).

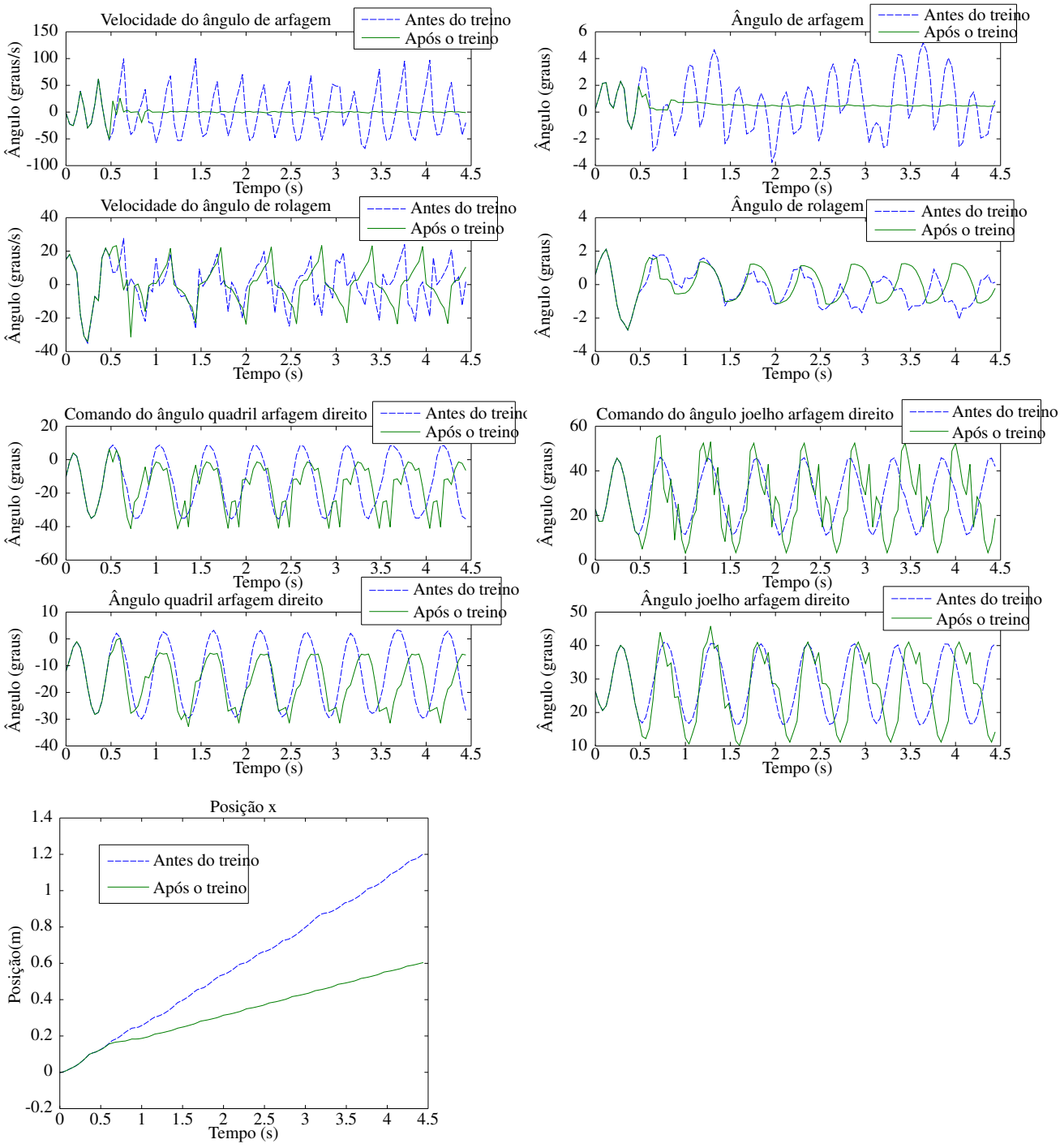


Figura 4.26: Resultados usando o oscilador de Kuramoto com acoplamento no NAO, com a saída apenas a fundamental da Série de Fourier, ou seja, uma senoide, e com um controlador de postura treinado.

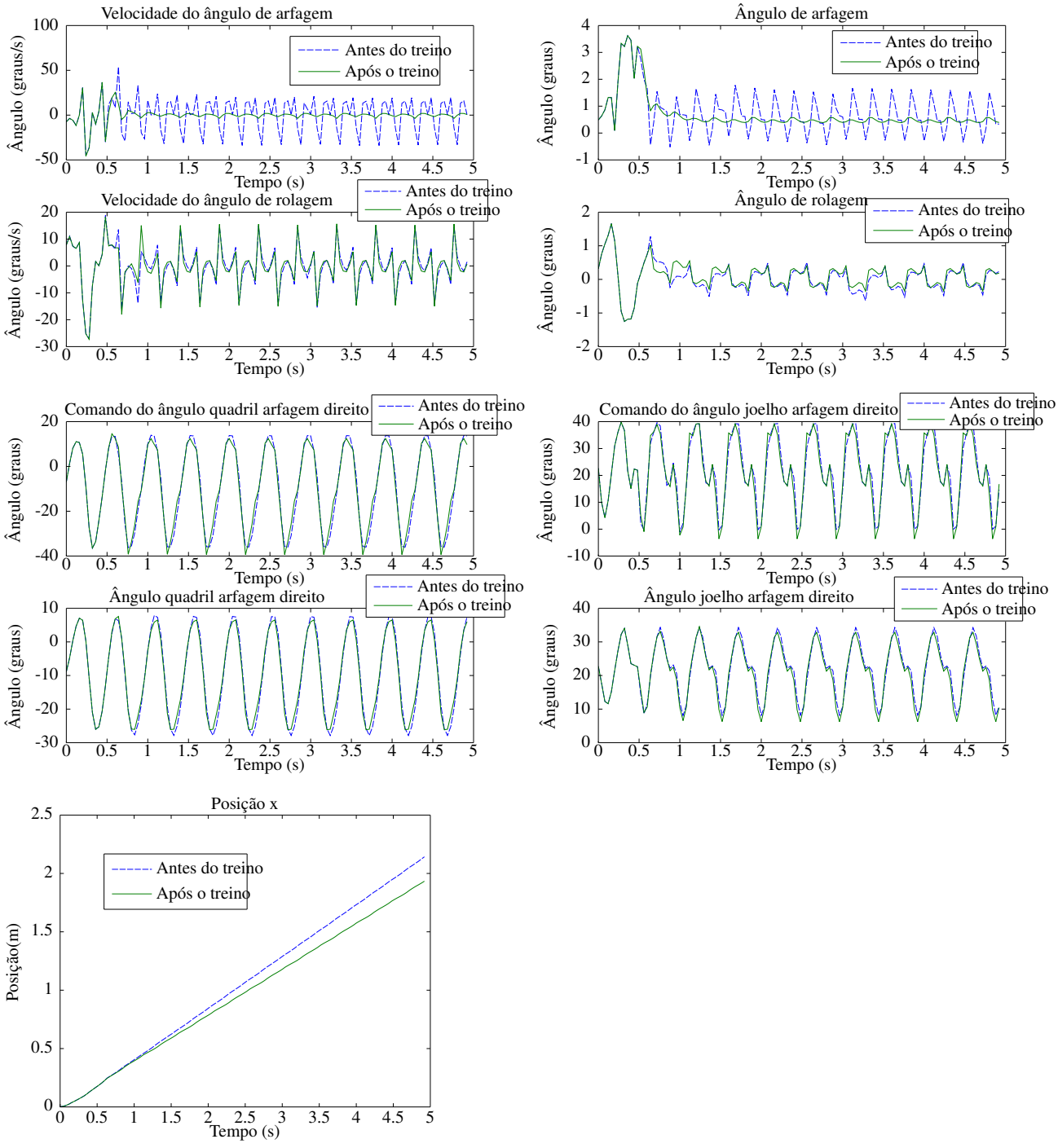


Figura 4.27: Resultados usando o oscilador de Kuramoto com acoplamento no NAO e com um controlador de postura treinado.

5 CONCLUSÃO

Neste trabalho foi implementado um sistema de geração de marcha em robôs bípedes, utilizando técnicas bio-inspiradas. Para isso, foi utilizado o conceito de CPG e com ele osciladores bio-inspirados. Foram utilizadas duas plataformas, a plataforma Bioloid e a plataforma NAO. No Bioloid foi possível gerar uma marcha utilizando osciladores de Matsuoka para gerar os ângulos das juntas das pernas do robô. Foram utilizados de quatro a seis osciladores de Matsuoka. Inicialmente foram usados quatro osciladores, um para o ângulo de arfagem do quadril e outro para o ângulo de arfagem do joelho de cada perna. Depois foram adicionados dois osciladores, um no ângulo de rolagem do quadril de cada perna. Para otimizar os parâmetros do oscilador foram utilizados os algoritmos AG e PSO. Para o NAO foi possível gerar marchas usando osciladores de Matsuoka, Kuramoto e Kuramoto com acoplamento entre a dinâmica do movimento e o oscilador. Foram utilizados seis osciladores, três em cada perna da mesma forma do caso do Bioloid. No oscilador de Kuramoto, a saída do oscilador é dada por uma Série de Fourier truncada. Para o oscilador de Matsuoka no NAO foram utilizados o AG, PSO e NSGA-II para otimizar os parâmetros do oscilador. Para os outros dois osciladores foram utilizados somente o AG e NSGA-II, pois com ambos PSO e AG é possível obter os parâmetros, foi escolhido apenas o AG para diminuir o tempo de simulação. Além disso, para o oscilador de Kuramoto com acoplamento foi treinado um controlador com o algoritmo de aprendizagem reforçada para corrigir a postura do robô durante a marcha.

Para os osciladores utilizados foi implementado um *framework* para obter os parâmetros dos osciladores usando o AG, PSO e NSGA-II. Para o AG e PSO foi proposta uma função de custo para obter uma marcha eficiente que visa determinadas condições como se o robô caiu, ou se o robô balançou o corpo acima de determinado limiar, entre outras. E se a marcha cumpriu as condições a função de custo é dado pela distância percorrida. E para o NSGA-II foi utilizada essa função de custo e outra função de custo que tem como objetivo minimizar o trabalho realizado pelos motores do robô. No Bioloid foi possível obter uma melhor marcha ao se utilizar seis osciladores do que usando quatro osciladores. Isso ocorreu porque ao se utilizar o ângulo de rolagem do quadril o robô pôde utilizar seu balanço lateral de forma com que ele contribua na marcha.

Já no robô NAO observou-se que entre os tipos de osciladores usados, o oscilador de Matsuoka tem como vantagem menos parâmetros e uma entrada sensorial bem definida. O oscilador de Kuramoto utilizado, por outro lado, tem mais parâmetros a serem otimizados, mas em troca permite uma maior variedade de formas de ondas de saída podendo ter diferentes tipos de formas de onda, diferentemente do oscilador de Matsuoka, que possui uma forma de onda definida pela sua dinâmica interna. Por outro lado essa possibilidade de diferentes formas de onda dificultou a procura de parâmetros, o que em certas ocasiões provocou o AG não conseguir sair de mínimos locais não desejáveis, como por exemplo andar para

trás. O acoplamento ajudou o oscilador de Kuramoto a ter resultados melhores e além disso, o acoplamento também ajudou o AG a encontrar parâmetros mais facilmente diminuindo os mínimos locais indesejados encontrados no caso anterior. Mas uma desvantagem do acoplamento é "forçar" o oscilador a ter uma frequência fixa dada pela dinâmica do robô. Embora ao se diminuir a constante de acoplamento a frequência da forma de onda da saída será mais próxima a frequência natural do oscilador ao custo de se ter um acoplamento menor. Além disso, utilizando o acoplamento foi possível obter marchas utilizando apenas a fundamental da Série de Fourier, ou seja, uma senoide que é um sinal mais simples que os dos casos anteriores.

Nos resultados do NSGA-II para todos os modelos de osciladores foram obtidos parâmetro em que em geral quanto maior a distância percorrida maior foi o trabalho realizado. Dessa forma, é possível escolher parâmetros de forma a realizar um *trade-off* entre o trabalho realizado pelo robô e a distância percorrida.

Também foi possível propor um *framework* utilizando aprendizagem por reforço para treinar um controlador para corrigir a postura do robô com a marcha sendo gerado pelo oscilador de Kuramoto com acoplamento. O objetivo do algoritmo foi minimizar a velocidade do ângulo de arfagem do corpo do robô, dessa forma, a variação do ângulo de arfagem também foi minimizada consequentemente. Como resultado, fora a minimização da velocidade do ângulo de arfagem, o robô andou mais "cautelosamente" para poder manter a postura e dessa forma percorreu uma distância menor do que se estivesse sem o controlador. Um outro resultado interessante é que embora a velocidade do ângulo de rolagem do corpo não tenha sido usada nem nos estados e nem na recompensa da aprendizagem por reforço é possível ver que o controle o tornou mais suave e também o tornou em uma "forma periódica" de sinal.

Para trabalhos futuros se sugere utilizar o método no robô real e obter marchas para diferentes situações de solo, como solo inclinado por exemplo. Também utilizar osciladores para gerar outros movimentos fora a marcha do robô, como por exemplo subir uma escada. Outra possibilidade é utilizar os osciladores com outras informações sensoriais. Mais uma possibilidade é comparar e observar a transição de parâmetros obtidos pelo NSGA-II para uma mesma frente de Pareto. Para o *framework* do treinamento do controlador de postura também é possível propor realizar o treinamento no robô real. Nesse caso seria necessário um suporte fixado ao robô que evite a queda dele no chão caso venha a ocorrer. Por fim, mais uma sugestão de trabalho futuro é analisar a adaptação dos osciladores para mudanças de intenções de movimento durante o movimento, como por exemplo alterar a velocidade da marcha durante a marcha.

[1] Juan A. Acebrón, L. L. Bonilla, Conrad J. Pérez Vicente, Félix Ritort, and Renato Spigler. The kuramoto model: A simple paradigm for synchronization phenomena. *Rev. Mod. Phys.*, 77:137–185, Apr 2005.

[2] Mostafa Ajallooeian, Soha Pouya, Alexander Sproewitz, and Auke J. Ijspeert. Central pattern generators augmented with virtual model control for quadruped rough terrain locomotion. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3321–3328, 2013.

[3] S. A. Bailey. *Biomimetic control with a feedback coupled nonlinear oscillator: insect experiments, design tolls, and hexapedal robot adaptation results*. PhD thesis, Stanford University, July 2004.

[4] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 120–127, 2007.

[5] Andreas Daffertshofer Stewart Heitmann Michael Breakspear. Generative models of cortical oscillations: Neurobiological implications of the kuramoto model. *Frontiers in Human Neuroscience*, 4, 2010.

[6] Jonas Buchli and AukeJan Ijspeert. Self-organized adaptive legged locomotion in a compliant quadruped robot. *Autonomous Robots*, 25(4):331–347, 2008.

[7] S. Campbell and DeLiang Wang. Synchronization and desynchronization in a network of locally coupled wilson-cowan oscillators. *Neural Networks, IEEE Transactions on*, 7(3):541–554, May 1996.

[8] F. B. Cavalcanti and G.A Borges. Instrumentação inercial para um robô humanóide. In *Anais do XV Congresso de Iniciacao Cientifca da Universidade de Brasilia*, 2009.

[9] Soon-Jo Chung and Michael Dorothy. Neurobiologically inspired control of engineered flapping flight. *Journal of Guidance, Control and Dynamics*, 33(2):440–453, 2010.

[10] Jorg Conradt and Paulina Varshavskaya. Distributed central pattern generator control for a serpentine robot. In *In ICANN 2003*, 2003.

[11] A. Crespi and A.J. Ijspeert. Online optimization of swimming and crawling in an amphibious snake robot. *Robotics, IEEE Transactions on*, 24(1):75–87, 2008.

[12] Rafael Cortes de Paiva, Alexandre Ricardo Soares Romariz, and Geovany Araujo Borges. Searching cpg parameters for humanoid gait using particle swarm optimization and

genetic algorithm. In *Advanced Robotics (ICAR), 2013 16th International Conference on*, pages 1–6, Nov 2013.

[13] R.C. de Paiva, A.R.S. Romariz, and G.A. Borges. Neural oscillator for gait command of a humanoid robot. In *Nature and Biologically Inspired Computing (NaBIC), 2012 Fourth World Congress on*, pages 118–122, 2012.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, Apr 2002.

[15] MATLAB: Ajuda do MATLAB: seção "Using the Genetic Algorithm": subseção "What is the Genetic Algorithm?". *version 7.13.0.564 (R2011b)*. The MathWorks Inc., Natick, Massachusetts, 2011.

[16] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, pages 39–43, 1995.

[17] Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot. *The International Journal of Robotics Research*, 27(2):213–228, 2008.

[18] A. C. Filho and M. S. Dutra. *Application of hybrid van der Pol-Rayleigh oscillators for modeling of a bipedal robot*. USA: Springer-Verlag, New York, NY, 2009.

[19] Armando C. de Pina Filho, Max S. Dutra, and Luciano S. C. Raptopoulos. Modeling of a bipedal robot using mutually coupled rayleigh oscillators. *Biological Cybernetics*, 92(1):1–7, 2005.

[20] Hado Hasselt. Reinforcement learning in continuous state and action spaces. In Marco Wiering and Martijn Otterlo, editors, *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 207–251. Springer Berlin Heidelberg, 2012.

[21] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1321–1326 vol.2, May 1998.

[22] Weiwei Huang, Chee-Meng Chew, Yu Zheng, and Geok-Soon Hong. Bio-inspired locomotion control with coordination between neural oscillators. *International Journal of Humanoid Robotics*, 6(4):585–608, 2009.

[23] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Preprint of Neural Networks*, 21/4:642–653, 2008.

[24] K. Inoue, Shugen Ma, and Chenghua Jin. Neural oscillator network-based controller for meandering locomotion of snake-like robots. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 5, pages 5064–5069, April 2004.

[25] T. Takenaka K. Hirai, M. Hirose. Dynamical walk of biped locomotion. *International Journal of Robotics Research*, 3(2):60–74, 1984.

[26] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1620–1626 vol.2, 2003.

[27] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji. Automatic locomotion design and experiments for a modular robotic system. *Mechatronics, IEEE/ASME Transactions on*, 10(3):314–325, June 2005.

[28] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, 1995.

[29] Hiroshi Kimura, Seiichi Akiyama, and Kazuaki Sakurama. Realization of dynamic walking and running of the quadruped using neural oscillator. *Autonomous Robots*, 7:247–258, 1999.

[30] Hiroshi Kimura, Yasuhiro Fukuoka, and Avis H. Cohen. Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts. *he International Journal of Robotics Research*, 26(5):475–490, May 2007.

[31] Y.F.H. Kimura and A.H. Cohen. Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *The International Journal of Robotics Research*, 22(3-4):187–202, March-April 2003.

[32] M. Anthony Lewis. Gait adaptation in a quadruped robot. *Autonomous Robots*, February 2002.

[33] Chengju Liu, Qijun Chen, and Danwei Wang. Cpg-inspired workspace trajectory generation and adaptive locomotion control for quadruped robots. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 41(3):867–880, June 2011.

[34] Zhenli Lu, Shugen Ma, Bin Li, and Yuechao Wang. 3d locomotion of a snake-like robot controlled by cyclic inhibitory cpg model. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3897–3902, Oct 2006.

[35] P. Manoonpong, F. Pasemann, and F. Wörgötter. Sensor-driven neural control for omnidirectional locomotion and versatile reactive behaviors of walking machines. *Robotics and Autonomous Systems*, 56:265–288, 2008.

[36] Takamitsu Matsubara, Jun Morimoto, Jun Nakanishi, Masa aki Sato, and Kenji Doya. Learning cpg-based biped locomotion with a policy gradient method. *Robotics and Autonomous Systems*, 54(11):911 – 920, 2006. <ce:title>Planning Under Uncertainty in Robotics</ce:title>.

[37] K. Matsuoka. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, 52(6):367–376, 1985.

[38] K. Matsuoka. Mechanism of frequency and pattern control in the neural rhythm generators. *Biological Cybernetics*, 56(5-6):345–353, 1987.

[39] Christophe Maufroy, Hiroshi Kimura, and Kunikatsu Takase. Towards a general neural controller for quadrupedal locomotion. *Neural Networks*, 21:667–681, 2008.

[40] J. Morimoto, G. Endo, J. Nakanishi, and G. Cheng. A biologically inspired biped locomotion strategy for humanoid robots: Modulation of sinusoidal patterns by a coupled oscillator model. *Robotics, IEEE Transactions on*, 24(1):185–191, 2008.

[41] Jun Morimoto, C.G. Atkeson, G. Endo, and G. Cheng. Improving humanoid locomotive performance with learnt approximated dynamics via gaussian processes for regression. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 4234–4240, 2007.

[42] Jun Morimoto, G. Endo, J. Nakanishi, S. Hyon, G. Cheng, D. Bentivegna, and C.G. Atkeson. Modulation of simple sinusoidal patterns by a coupled oscillator model for biped walking. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1579–1584, 2006.

[43] H. Nagashino, Y. Nomura, and Y. Kinouchi. A neural network model for quadruped gait generation and transitions. *Neurocomputing*, 38-40:1469–1475, 2001.

[44] Kazuki Nakada, Tetsuya Asai, and Yoshihito Amemiya. An analog cmos central pattern generator for interlimb coordination in quadruped locomotion. *IEEE Transactions on Neural Networks*, 14(5), September 2003.

[45] Kazuki Nakada, Tetsuya Asai, and Yoshihito Amemiya. Biologically-inspired locomotion controller for a quadruped walking robot paper: Biologically-inspired locomotion controller for a quadruped walking robot: Analog ic implementation of a cpg-based controller. *Journal of Robotics and Mechatronics*, 16(4):397–403, 2004.

[46] Kazuki Nakada, Tetsuya Asai, and Yoshihito Amemiya. Design of an artificial central pattern generator with feedback controller. *Intelligent Automation and Soft Computing*, 10(2):185–192, 2004.

[47] Ludovic Righetti, Jonas Buchli, and Auke Jan Ijspeert. Dynamic hebbian learning in adaptive frequency oscillators. *Physica D: Nonlinear Phenomena*, 216(2):269 – 281, 2006.

[48] Jae-Kwan Ryu, NakYoung Chong, BumJae You, and HenrikI. Christensen. Locomotion of snake-like robots using adaptive neural oscillators. *Intelligent Service Robotics*, 3(1):1–10, 2010.

[49] Keehong Seo, Soon-Jo Chung, and Jean-JacquesE. Slotine. Cpg-based control of a turtle-like underwater vehicle. *Autonomous Robots*, 28(3):247–269, 2010.

[50] Alexander Sproewitz, Alexandre Tuleu, Massimo Vespignani, Mostafa Ajallooeian, Emilie Badri, and Auke Ijspeert. Towards Dynamic Trot Gait Locomotion—Design, Control and Experiments with Cheetah-cub, a Compliant Quadruped Robot. *International Journal of Robotics Research*, 32(8):932 – 950, 2013.

[51] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[52] Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *In Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.

[53] G. Taga. A model of the neuro-musculo-skeletal system for human locomotion ii. real-time adaptability under various constraints. *Biological Cybernetics*, 73(2):113–121, 1995.

[54] G. Taga, Y. Yamaguchi, and H. Shimizu. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics*, 65(3):147–159, 1991.

[55] Paschalis Veskos and Yiannis Demiris. Experimental comparison of the van der pol and rayleigh nonlinear oscillators for a robotic swinging task. In *AISB'06: Adaptation in Artificial and Biological Systems*, pages 197–202, Jan 2006.

[56] Matthew M. Williamson. Neural control of rhythmic arm movements. *Neural Networks*, 11:1379–1394, 1998.

[57] H. R. Wilson and J. D. Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal*, 12(1):1–24, 1972.

[58] Xiaodong Wu and Shugen Ma. Cpg-based control of serpentine locomotion of a snake-like robot. *Mechatronics*, 20(2):326 – 334, 2010.

[59] J. Yu, M. Tan, J. Chen, and J. Zhang. A survey on cpg-inspired control models and system implementation. *Neural Networks and Learning Systems, IEEE Transactions on*, PP(99):1–1, 2013.

[60] Junzhi Yu, Ming Wang, Min Tan, and Jianwei Zhang. Three-dimensional swimming. *Robotics Automation Magazine, IEEE*, 18(4):47–58, Dec 2011.

[61] Chunlin Zhou and K. H. Low. Kinematic modeling framework for biomimetic undulatory fin motion based on coupled nonlinear oscillators. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 934–939, Oct 2010.

[62] T. Zielinska. Coupled oscillators utilized as gait rhythm generators of a two-legged walking machine. *Biological Cybernetics*, 74(3):263–273, 1996.