



JESIMAR DA SILVA ARANTES

**UTILIZANDO ROBÓTICA EVOLUTIVA
PARA O DESENVOLVIMENTO DA
MORFOLOGIA DE ROBÔS**

LAVRAS – MG

2014

JESIMAR DA SILVA ARANTES

**UTILIZANDO ROBÓTICA EVOLUTIVA PARA O DESENVOLVIMENTO
DA MORFOLOGIA DE ROBÔS**

Monografia apresentada ao Colegiado do Curso de
Ciência da Computação, para a obtenção do título
de Bacharel em Ciência da Computação.

Orientador

Prof. Dr. Tales Heimfarth

Co-Orientador

Prof. Dr. Raphael W. de

Bettio

LAVRAS – MG

2014

JESIMAR DA SILVA ARANTES

**UTILIZANDO ROBÓTICA EVOLUTIVA PARA O DESENVOLVIMENTO
DA MORFOLOGIA DE ROBÔS**

Monografia apresentada ao Colegiado do Curso de
Ciência da Computação, para a obtenção do título
de Bacharel em Ciência da Computação.

APROVADA em 3 de Fevereiro de 2014.



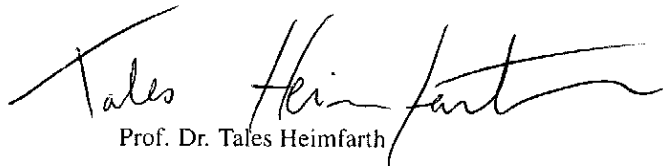
Prof. Dr. Luiz Henrique Correia

UFLA



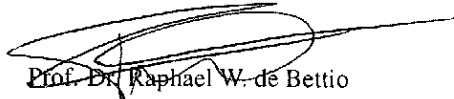
Prof. Dr. Neumar Costa Malheiros

UFLA



Prof. Dr. Tales Heimfarth

(Orientador)



Prof. Dr. Raphael W. de Bettio

(Co-Orientador)

LAVRAS – MG

2014

À minha família!

*Meu pai Jésus Ferreira Arantes e minha mãe Sirley Ramos Arantes
que estiveram sempre presentes, me apoiando, me incentivando e me
ajudando nos momentos em que mais precisei.*

*Meu irmão Márcio da Silva Arantes pela amizade e ajuda,
e por saber que sempre terei com quem contar.*

*A minha tia Sirlaine Maria da Silva Freitas que me
mostrou a importância dos estudos.*

DEDICO

AGRADECIMENTOS

À Universidade Federal de Lavras (UFLA) e ao Departamento de Ciência da Computação (DCC), por me proporcionar um ambiente de estudo adequado à minha formação.

Aos professores da UFLA em especial aos professores do Departamento de Ciência da Computação, pelos ensinamentos transmitidos.

Aos professores Tales Heimfarth e Raphael Winckler de Bettio pela orientação, paciência, amizade e seus ensinamentos que foram de grande relevância para a realização deste trabalho e meu crescimento profissional. E a todos os integrantes do GRUBi que contribuíram e auxiliaram nas minhas pesquisas.

Aos meus colegas de apartamento do Alojamento Estudantil da UFLA que me fizeram crescer durante a minha jornada em Lavras. Aos colegas de turma que estiveram sempre presentes durante os meus estudos.

Agradeço sobretudo aos meus pais, Jésus Ferreira Arantes e Sirley Ramos Arantes, pelo amor e carinho. Ao meu irmão Márcio da Silva Arantes pela ajuda constante e amizade. E a minha namorada Vívian Vilela França por fazer parte deste momento em minha vida.

Muito obrigado por tudo!!!

EPIGRAFE

Se você tem uma maçã e eu tenho uma maçã e nós trocamos as maçãs, então você e eu ainda teremos uma maçã. Mas se você tem uma ideia e eu tenho uma ideia e nós trocamos essas ideias, então cada um de nós terá duas ideias.

George Bernard Shaw

Se não puder se destacar pelo talento, vença pelo esforço.

Dave Weinbaum

O número de bugs no código tendem ao infinito, já os acertos tendem a zero.

Jesimar da Silva Arantes

SUMÁRIO

1	INTRODUÇÃO	1
1.1	Contextualização	1
1.2	Motivação	2
1.3	Objetivos	2
1.3.1	Objetivo Geral	3
1.3.2	Objetivos Específicos	3
1.4	Estrutura do Trabalho	4
2	REFERENCIAL TEÓRICO	5
2.1	Algoritmo Genético	5
2.1.1	Terminologia	6
2.2	Sistema de Controle de Robôs	8
2.2.1	Arquitetura de Subsunção	8
2.3	Robótica Evolutiva	9
2.4	GrubiBots	10
2.5	Biblioteca ODE	12
2.6	Estado da Arte	14
3	METODOLOGIA	17
3.1	Classificação do Trabalho	17
3.2	Ferramentas Utilizadas	18

3.2.1	Plataforma de Robótica	19
3.2.2	Linguagem Utilizada	19
3.2.3	Biblioteca de Física	19
3.2.4	Armazenamento dos Indivíduos	20
3.3	Conjunto de Peças dos Robôs	20
3.4	Morfologia do Robô	20
3.4.1	Chassi	21
3.4.2	Processador	23
3.4.3	Roda Sem Motor	23
3.4.4	Roda Com Motor	24
3.4.5	Sensor de Distância	24
3.5	Pontos de Conexão do Robô	25
3.6	Montagem da Morfologia do Robô	25
3.6.1	Operadores de Crossover	27
3.6.1.1	Crossover de Processador	27
3.6.1.2	Crossover de Roda Sem Motor	27
3.6.1.3	Crossover de Roda Com Motor	27
3.6.1.4	Crossover de Sensor	27
3.6.2	Operadores de Mutação	28
3.6.2.1	Adicionar Peça	28
3.6.2.2	Remover Peça	29

3.6.2.3 Trocar Peças	29
3.7 Controle do Robô	30
3.8 Algoritmo Genético	30
3.9 Cenários Simulados e Função de Fitness	32
3.10 Configurações e Parâmetros Gerais Utilizados	35
3.11 Configurações dos Computadores Utilizados	36
4 RESULTADOS E DISCUSSÕES	38
5 CONCLUSÃO E TRABALHOS FUTUROS	44
REFERÊNCIAS	46

LISTA DE FIGURAS

2.1	Exemplos de cenários simulados na ferramenta ODE4J.	13
2.2	Características da ODE4J.	13
2.3	Trabalho de Karl Sims criaturas evoluídas. Fonte: (SIMS, 1994b). . .	15
2.4	Trabalho de Karl Sims criaturas evoluídas. Fonte: (SIMS, 1994a). . .	15
2.5	Trabalho de Peter Krcah criaturas evoluídas. Fonte: (KRCAH, 2007). .	16
3.1	Classificação dos tipos de pesquisa. Fonte: (JUNG, 2004)	18
3.2	Conjunto de peças modeladas.	21
3.3	Conjunto de peças do robô.	22
3.4	Pontos de conexão do Chassi.	25
3.5	Etapas envolvidas na montagem da morfologia do robô.	26
3.6	Mapeamento de setores do sensor no chassi.	28
3.7	Problemas básicos de deslocamento do robô.	32
4.1	Resultados obtidos nos experimentos para <i>Problema</i> ₁	39
4.2	Resultados para novas configurações no <i>Problema</i> ₁	39
4.3	Resultados obtidos nos experimentos para <i>Problema</i> ₂	41
4.4	Resultados gerais para a morfologia dos robôs.	41
4.5	Resultados gerais para a morfologia dos robôs.	42
4.6	Resultados gerais para a morfologia dos robôs.	42
4.7	Trajetória descrita pelo robô no <i>Problema</i> ₂	43

LISTA DE TABELAS

3.1	Quantidade de peças utilizadas na montagem do robô.	35
3.2	Pesos das peças utilizadas na montagem do robô.	36
3.3	Pesos utilizados nas simulações efetuadas.	36
3.4	Configurações do computador utilizado nos testes.	37
3.5	Configurações do <i>cluster</i> utilizado	37
4.1	Configurações utilizadas nas simulações do <i>Problema</i> ₁	38
4.2	Configurações utilizadas nas simulações do <i>Problema</i> ₂	40

RESUMO

A montagem da morfologia de robôs é uma tarefa complexa de ser desenvolvida. Isto ocorre principalmente no que tange a determinação de quantas e quais tipos de peças o robô deverá possuir para resolver uma determinada tarefa. Um outro fator que aumenta a dificuldade para projetar o hardware de um robô é saber onde deverão ficar localizadas as peças do robô. Em cada um dos problemas a serem resolvidos, um robô diferente deveria ser projetado para minimizar os recursos gastos. Pensando nisto, este trabalho apresenta um módulo de robótica evolutiva. Este módulo apresenta um banco de dados de peças implementadas que poderão constituir o robô. As peças programadas são baseadas em peças reais de robôs. O programa desenvolvido encontrará de forma automática uma configuração para o robô. Para isto, o programador define o cenário e uma função de avaliação que um algoritmo genético utilizará para encontrar uma configuração para a morfologia do robô. Este trabalho lida com dois problemas distintos. Um dos problemas é o deslocamento linear do robô, já o outro é o atendimento de marcadores colocados em um ambiente com obstáculos. O primeiro problema foi resolvido de forma ótima, já para o segundo soluções razoáveis foram encontradas.

Palavras-Chave: Evolução de Robôs; Algoritmos Genéticos; Programação Genética; Robótica.

ABSTRACT

The assembly of the morphology of robots is a complex task to be developed. This occurs mainly when it comes to determining how many and what types of parts the robot will have to solve a given task. Another factor that increases the difficulty to design the hardware of a robot is to know where the parts of the robot should be located. In each of the problems to be solved, a different robot should be designed to minimize the resources spent. Thinking about it, this work presents a module of evolutionary robotics. This module provides a database of parts that can be used to assemble a robot. They are based on real robot parts. The developed system is able to find the optimal configuration for a robot. For this purpose the programmer sets the scene and a fitness function which uses genetic algorithm to find a setting for the morphology of the robot. This work deals with two different problems. One of the problems is the linear displacement of the robot, and the other is the service of targets placed in an environment with obstacles. The first problem was solved optimally, while reasonable solutions were found for the second one.

Keywords: Evolution of Robots; Genetic Algorithms; Genetic Programming; Robotics.

1 INTRODUÇÃO

Este trabalho apresenta o desenvolvimento um sistema de Robótica Evolutiva (RE) para a evolução da morfologia de robôs. Nesse sistema as peças do robô foram modeladas com base em algumas peças para construção de robôs. A parte responsável pelos controles do robô (comportamentos) foram definidas de forma genérica baseadas na arquitetura de subsunção. A evolução dos robôs se dará a partir de regras definidas em algoritmos genéticos a partir de uma função de avaliação (*fitness*).

1.1 Contextualização

A utilização de robôs está presente em praticamente todas as tarefas humanas, desde tarefas simples como tirar o pó da casa e cortar a grama do jardim, até tarefas extremamente complexas, como auxiliar em acidentes nucleares, explorar vulcões e até permitir a exploração outros planetas (BATURONE, 2001). Em suma, estas máquinas estão mudando a maneira como as pessoas vivem e trabalham e, segundo as palavras do mesmo autor, estão expandindo as fronteiras da experiência humana.

A Construção de robôs eficientes para resolução de problemas, até mesmo os mais simples, necessitam de um extenso número de cálculos e projeções tanto em hardware quanto em software. Os robôs atualmente são projetados e contruídos por projetistas de robôs.

A fim de facilitar a construção e o projeto de robôs, podemos utilizar técnicas de RE. Esta técnica surgiu da utilização de técnicas de Computação Evolutiva para sintetizar automaticamente controladores embarcados para robôs, com o propósito de treiná-los para desenvolver tarefas específicas. Os Algoritmos Genéticos (AG)

têm sido empregados com sucesso no desenvolvimento automatizado de controladores para robôs, em experimentos realizados em simulação, onde a população de robôs não existe fisicamente, mas é representada matematicamente em software (CARVALHO, 2002).

Este trabalho tem como objetivo principal utilizar as técnicas de RE para criar robôs autônomos aptos a resolverem determinadas tarefas. O foco do trabalho é projetar e desenvolver um módulo de robótica para evolução da morfologia de robôs. O presente módulo evolui também os controles dos robôs (comportamentos), mas neste trabalho não dará foco a essa etapa.

1.2 Motivação

Um fator motivador são os potenciais ganhos que a presente pesquisa poderá trazer ao mundo acadêmico. Como exemplo de ganhos tem-se o fato do robô ser construído de forma autônoma pelo programa desenvolvido. Assim, economiza-se em mão de obra especializada na projeção e desenvolvimento do robô. Caso o algoritmo genético consiga efetuar a evolução do robô para resolver uma determinada tarefa, isso contribuirá significativamente para a área de montagens de robôs.

1.3 Objetivos

Os objetivos presentes neste trabalho serão apresentados a seguir.

1.3.1 Objetivo Geral

Esse trabalho tem como objetivo geral a construção de um módulo de robótica evolutiva que utilizará algoritmos genéticos para evoluir a morfologia de robôs em um ambiente virtual.

1.3.2 Objetivos Específicos

1. **Construir o módulo de evolução de robôs na plataforma GrubiBots:**

Nesta etapa será construído um módulo de simulação para evolução de robôs virtuais para a plataforma *GrubiBots*. A plataforma *GrubiBots* é uma plataforma desenvolvida para controle de robôs tanto reais quanto simulados (virtuais).

2. **Modelar as peças do robô:** Um conjunto de peças como chassi, processador, sensor de distância, rodas com e sem motor serão modeladas em software para serem utilizadas na plataforma de simulação. Ao final desta tarefa um banco de peças para o robô estará pronto e poderá ser utilizado na sua construção.

3. **Definir a função objetivo:** A definição da função objetivo do robô é uma etapa que determinará algumas características morfológicas do robô e como será o seu controle. Essa função de avaliação se baseará na solução de dois problemas que o robô resolverá.

4. **Modelar e definir o cenário:** Esta tarefa definirá o ambiente do robô. Considera-se um cenário diferente para cada tipo de função de avaliação.

5. **Desenvolver o algoritmo genético:** Um algoritmo genético será desenvolvido para evoluir o robô com base nas peças e comportamentos criados, na função objetivo e no cenário estabelecido para o problema.

6. **Definir os parâmetros do algoritmo genético:** A etapa de definição dos melhores parâmetros do algoritmo genético para cada um dos problemas a serem solucionados é uma etapa importante para a evolução.

1.4 Estrutura do Trabalho

Este trabalho está organizado da seguinte forma: O Capítulo 2 apresenta o referencial teórico levantado. O Capítulo 3 mostra a metodologia de trabalho utilizada para a construção do módulo de robótica, como foram feitas as modelagens das peças, dos ambientes e a evolução do robô onde foram feitas as simulações. O Capítulo 4 apresenta os resultados e discussões do presente trabalho. No Capítulo 5 são apresentados as conclusões e trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico sobre algoritmos genéticos, sistema de controle de robôs, robótica evolutiva, além de apresentar brevemente o estado da arte em robótica evolutiva. Aqui estão presentes os principais conceitos técnicos utilizados no desenvolvimento deste trabalho.

2.1 Algoritmo Genético

Os Algoritmos Genéticos (AGs) são algoritmos de computação evolutiva inicialmente proposto por Holland (1975). Esses algoritmos seguem o princípio da seleção natural. O conceito de seleção natural foi proposto por Charles Darwin em 1859 em seu livro “A Origem das Espécies”. A ideia principal consiste em que as espécies com características mais adaptadas ao meio onde vivem tendem a sobreviver. Já as espécies pouco adaptadas tendem a ser extintas.

A ideia proposta por Darwin era que o meio ambiente seleciona naturalmente os seres que vivem neles. De forma resumida, os seres que são adaptados ao meio em que vivem sobrevivem. Já os seres que não se adaptam morrem e são extintos. Esse conceito de seleção natural leva em conta o conceito de hereditariedade, que é a capacidade dos seres vivos de transmitirem características genéticas aos seus descendentes.

Um Algoritmo Genético inicialmente define uma população de indivíduos. Cada indivíduo representa uma solução do problema em questão. Essa população tem seus indivíduos inicializados de forma totalmente aleatória. Em seguida, executa-se um laço (estrutura de repetição) onde se evolui os indivíduos durante um certo número de gerações. Primeiramente é executada uma rotina de seleção, que seleciona dois indivíduos da população para se realizar o cruzamento ou

crossover. Um novo indivíduo é gerado nessa recombinação, em seguida, ele é processado por uma rotina que realiza mutações em sua codificação. Esse procedimento de seleção, cruzamento e mutação é executado até que a população da próxima geração seja totalmente gerada.

2.1.1 Terminologia

A seguir serão apresentados alguns termos sobre a área de AG que são utilizados neste trabalho:

População: Conjunto de indivíduos que representam várias soluções para o problema. Em geral, os indivíduos da população melhoram ao longo das gerações.

Indivíduo: Um simples membro da população. Um indivíduo é uma solução para o problema proposto. Cada indivíduo possui uma aptidão associada a sua qualidade como solução do problema.

Função de Fitness: Também chamada de função de avaliação ou ainda função objetivo. Tem por objetivo avaliar as soluções representadas pelos indivíduos, associando a cada um deles uma nota de acordo com sua qualidade. Esta função leva em conta características do problema para calcular a aptidão do indivíduo, dessa forma são necessárias diferentes funções de *fitness* para problemas distintos.

Seleção: A técnica de seleção consiste na forma de selecionar os indivíduos da população para se reproduzirem. Existem vários métodos de seleção encontrados na literatura entre eles tem-se a Roleta, Torneio e Torneio-q (BENTO; KAGAN, 2008).

Torneio: É o operador de seleção mais simples de ser implementado. Seu funcionamento consiste no sorteio aleatório de dois indivíduos da população, em seguida, pega-se o melhor indivíduo entre os selecionados (BENTO; KAGAN, 2008).

Torneio- q : É uma variante do método de seleção Torneio. Seu funcionamento consiste no sorteio aleatório de q indivíduos da população, em seguida, pega-se o melhor indivíduo entre os selecionados (BENTO; KAGAN, 2008).

Crossover: Também conhecido como cruzamento ou ainda recombinação. São escolhidos dois indivíduos para se reproduzirem. A recombinação é um processo que imita o processo biológico de reprodução sexuada: os descendentes recebem em seu código genético parte do código genético do pai_1 e parte do código do pai_2 . Esta recombinação garante que os melhores indivíduos sejam capazes de trocar entre si as informações que os levam a ser mais aptos a sobreviver, e assim, gerar descendentes ainda mais aptos (GOLDBERG, 1989).

Mutação: É um operador que tem como objetivo permitir maior variabilidade genética na população, impedindo que a busca fique estagnada em um mínimo local. Em geral o valor da taxa de mutação é baixo para a grande maioria dos problemas (GOLDBERG, 1989).

Gerações: Número de vezes que criará (gerará) um nova população. Este fator é associado a hereditariedade do algoritmo genético. A cada nova geração as soluções tendem a melhorar.

Elitismo: A ideia do elitismo é manter as melhores soluções na geração seguinte. Assim quando cria-se uma nova população por cruzamento e mutação, tem-se uma grande chance de perder os melhores cromossomos. Elitismo é o nome da técnica que primeiro copia os melhores indivíduos para a nova população. O restante da população é construída de forma tradicional, através de cruzamento e mutação. O elitismo pode aumentar rapidamente o desempenho do AG, porque previne a perda da melhor solução já encontrada (BENTO; KAGAN, 2008).

2.2 Sistema de Controle de Robôs

A tarefa do sistema de controle é fazer com que todo o sistema alcance um determinado estado. O sistema de controle obtêm informações do ambiente através dos seus sensores e transmite comandos para os atuadores dos robôs. Uma das principais técnicas de controles de robôs é a Arquitetura de Subsunção e esta será utilizada neste trabalho.

Vale ressaltar que o sistema de controle dos robôs não será aprofundado neste trabalho, o enfoque deste trabalho é a evolução do hardware (morfologia) do robô e não do software (controle).

2.2.1 Arquitetura de Subsunção

A arquitetura de subsunção foi proposta por Brooks (1986). Nesta arquitetura, os comportamentos são definidos através de leituras dos sensores e envio de comandos para os atuadores. Os comportamentos são conectados uns aos outros formando uma rede organizada em camadas dispostas hierarquicamente. Cada camada é responsável por uma atividade do robô. Nas camadas mais altas estão os comportamentos mais abstratos e que levam o robô a efetuar um objetivo. Nas camadas mais baixas ficam os comportamentos responsáveis por ações básicas do robô.

Os comportamentos em cada uma das camadas funcionam de forma concorrente e independente. Estes comportamentos são coordenados de forma competitiva, sendo que os comportamentos em camadas mais baixas têm prioridade em relação aos comportamentos em camadas superiores. A coordenação é feita por meio de dois mecanismos principais chamados inibição e supressão. Na arquitetura de subsunção a hierarquia entre os comportamentos é definida de forma bastante específica (BROOKS, 1986).

2.3 Robótica Evolutiva

Segundo Paredis (1996) a área de Robótica Evolutiva surgiu da utilização de técnicas de Computação Evolutiva para sintetizar automaticamente controladores embarcados para equipes de robôs, com o propósito de treiná-los para desenvolver tarefas específicas. Algoritmos Genéticos têm sido empregados com sucesso no desenvolvimento automatizado de controladores para robôs, em experimentos realizados em simulação, onde a população de robôs não existe fisicamente, mas é representada matematicamente em software (DICK, 1990).

Alguns pesquisadores empregam simuladores a fim de desenvolver um controlador adequado, e depois transferi-lo para um robô físico (SMITH, 1998). Outros conduzem sua pesquisa de maneira que o algoritmo genético é executado em simulação, mas cada indivíduo da população é transferido para um robô real a fim de ser avaliado individualmente.

Esses experimentos ainda não podem ser considerados verdadeiros sistemas evolutivos, porque os indivíduos da população de robôs não podem interagir uns com os outros, e muitos dos imprevistos de um sistema físico em que todos os indivíduos existem e interagem em tempo real não estão presentes (JAKOBI, 1997). Finalmente, somente uns poucos experimentos publicados por Floreano D. e Nolfi (2001) fazem uso de dois ou mais robôs reais, mas ainda assim conectados por cabos a um computador externo, onde o algoritmo genético estava presente.

O propósito dos experimentos descritos acima era o desenvolvimento de um controlador “ótimo” para um só robô. Uma vez que o algoritmo genético encontra uma solução aceitável, ela é replicada para todos os robôs produzidos depois (FLOREANO; MONDADA, 1998). O principal problema desse método é que o controlador, uma vez evoluído para uma determinada aplicação, é copiado para o robô e não pode mais ser modificado. Se algum fator for alterado no ambiente de

trabalho ou não for previsto pelo projetista, o robô não tem mais condições de se adaptar. Diferentemente desses trabalhos, um verdadeiro sistema evolutivo deve ser capaz de produzir não somente um robô treinado para uma tarefa específica, mas um processo evolutivo contínuo, em que a população de robôs trabalhe em conjunto, e seus controladores sejam continuamente ajustados pelo sistema para se adaptarem às modificações do ambiente de trabalho.

Este trabalho implementará um módulo de robótica evolutiva em um ambiente virtual. Porém, o módulo proposto foi projetado de forma a permitir uma possível implementação real.

2.4 GrubiBots

A plataforma computacional *GrubiBots* foi projetada e desenvolvida pelo GRUBI/UFLA - Grupo de Redes Ubíquas da UFLA. Essa plataforma tem por objetivo ser utilizada no desenvolvimento e teste de algoritmos voltados a programação de robôs móveis autônomos baseados em sensores.

Os robôs reais que podem ser controlados pela plataforma *GrubiBots* são:

LEGO MindStorms NXT: Trata-se de uma linha de robótica da LEGO voltado para a educação (LEGO, 2013). A CPU do robô é um microcontrolador ARM7, onde podem ser conectados até quatro sensores e três atuadores. Possui comunicação por *bluetooth*.

RoboCore: É um robô da linha de robótica de baixo custo e fácil aprendizado (ROBOCORE, 2013). Seu microcontrolador é um Arduino. Este robô é voltado para a montagem de basicamente dois tipos de robôs: Robô Seguidor de Linha e Robô Sumô, mas nada impede seu uso para desenvolver outros tipos de robôs.

Parrot Ar.Drone: Trata-se um quadricóptero comercial vendido pela companhia Parrot SA (SEYDOUX, 2014). Este *drone* possui diversas características tecnológicas, como a capacidade de ser programado e controlado remotamente, capacidade de gravar e transmitir remotamente vídeos, é capaz de ser utilizado em ambientes fechados e também possui a capacidade de voo estável e capacidade de planar.

Grubi Sumo Robot: É um robô móvel construído no laboratório GRUBI/UFLA. Trata-se de um robô construído baseado na plataforma *Stomper Sumo Kit* (LYNXMOTION, 2014). Este robô pode ser controlado a distância utilizando-se tecnologia de comunicação *bluetooth* e um protocolo próprio também desenvolvido pelo GRUBI, ele também é capaz de movimentar objetos de pequeno porte.

O *GrubiBots* possui as características tecnológicas/arquitetônicas abaixo relacionadas.

Implementado na linguagem Java: A linguagem Java possui como vantagem ser multiplataforma, essa característica permite flexibilidade no que tange o *deployment* dos módulos do *GrubiBots*, ou seja, os módulos poderão ser executados em computadores com diferentes arquiteturas e sistemas operacionais. Uma outra vantagem da linguagem é sua ampla divulgação e aceitação na comunidade acadêmica nacional e internacional.

Capacidades de Simulação: Em geral as ferramentas de simulação incorporam a capacidade de simular leis da física de forma a tornar a simulação a mais realística possível, essa característica tem por objetivo garantir que os resultados dos algoritmos executados na versão simulada e na versão real sejam similares.

Extensibilidade: O modelo arquitetônico deverá garantir que novos tipos de robôs baseados em sensores possam ser incluídos na plataforma da maneira mais simples possível, essa característica poderá contribuir com a continuidade do desenvolvimento e difusão da ferramenta, já que novos robôs poderão ser incorporados ao sistema.

Modelo Arquitetônico Distribuído: É imperativo que a simulação e a visualização da mesma sejam implementados em softwares separados. Essa separação de conceitos tem por objetivo permitir que a simulação seja executada em tempo menor do que o tempo real em casos onde não exista a necessidade de acompanhamento visual do desenrolar da mesma. Outra característica importante no que tange o modelo arquitetônico é que o mesmo permita a distribuição da execução, de forma que durante a utilização de robôs reais possam ser controlados por diferentes computadores aumentando assim a capacidade de processamento do sistema.

Subsunção: A arquitetura de subsunção é o modelo de programação de robôs mais utilizado pela comunidade acadêmica, pois apresenta melhor resultados com menor necessidade de capacidade de processamento que as arquiteturas tradicionais, dessa forma a plataforma *GrubiBots* deverá permitir a implementação de algoritmos baseados em subsunção.

2.5 Biblioteca ODE

A ODE - *Open Dynamics Engine* é uma biblioteca de alto desempenho e é especialmente indicada para simulação de objetos móveis em ambientes dinâmicos (ZÄSCHKE, 2014). A ODE4J é a implementação da ODE para Java. A biblioteca ODE é útil para criar veículos simulados e robôs, como ilustrado na Figura 2.1. Esta biblioteca é um motor de física e é uma ferramenta *Open Source*.

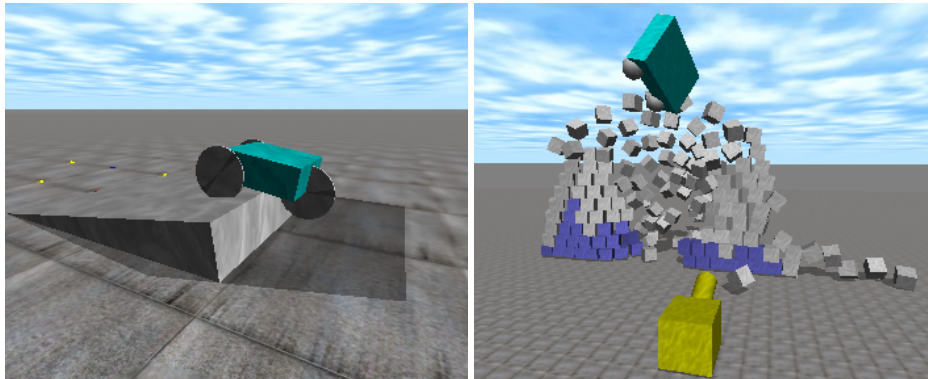


Figura 2.1: Exemplos de cenários simulados na ferramenta ODE4J.

A Figura 2.2 apresenta algumas das características da ODE. Seus dois principais componentes são o motor de dinâmicas de corpos rígidos e o motor de detecção de colisão. Atualmente ela é utilizada em aplicações como: jogos de computador, ferramentas de criação 3D e ferramentas de simulação. Ela foi concebida originalmente em C mas há implementações de ODE para Python (PyODE), Ruby (Ruby-ODE), Java (ODE4J), entre outras. Esta biblioteca é muito usada para aplicações de simulação de robótica. Como exemplo, tem-se cenários como a locomoção de robôs móveis (ENGINE, 2013).

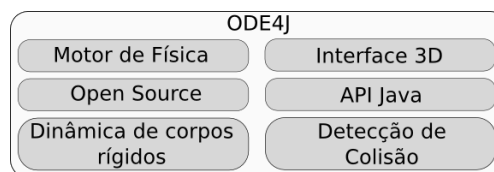


Figura 2.2: Características da ODE4J.

A ODE possui um sistema de detecção de colisões nativo, que suporta as seguintes primitivas de colisão: *sphere* (esfera), *box* (caixa), *capped cylinder* (cilindro com as extremidades arredondadas) e *plane* (plano).

Uma simulação ODE típica ocorre da seguinte forma (WOLFF; NORDIN, 2003):

1. Criação do mundo dinâmico;
2. Criação dos corpos rígidos no mundo dinâmico;
3. Ajuste do estado (posição e inclinação) dos corpos rígidos;
4. Criação das articulações no mundo dinâmico;
5. Conexão das articulações aos corpos rígidos;
6. Ajuste dos parâmetros de todas as articulações;
7. Criação do mundo colisivo e dos objetos geométricos neste mundo;
8. Criação de um grupo de articulações para armazenar juntas do tipo contato;
9. Repetir:
 - Aplicação de forças aos corpos conforme a necessidade;
 - Ajuste dos parâmetros das articulações conforme a necessidade;
 - Execução da rotina de detecção de colisões;
 - Criação de juntas do tipo contato para todos os pontos de colisão;
 - Execução de um passo da simulação;
 - Remoção de todas as juntas do tipo contato;
10. Destruição do mundo dinâmico e do mundo colisivo.

2.6 Estado da Arte

A seguir será descrito o estado da arte que encontra-se os trabalhos de evolução da morfologia de robôs.

Um dos grandes pioneiros da área de robótica evolutiva foi Karl Sims, que no ano de 1994 publicou os artigos (SIMS, 1994b), (SIMS, 1994a), onde utilizando

um ambiente tridimensional baseado em física efetuou a evolução de criaturas virtuais. Nestas criaturas, o sistema de controle neural foi evoluído em conjunto com a morfologia, ou seja, este trabalho evoluiu o hardware e software do robô. A Figura 2.3 mostra algumas criaturas virtuais, após a evolução, obtida no trabalho de Karl Sims evoluídas com base nas funções objetivo: caminhar (a função de *fitness* utilizada é a distância percorrida), nadar, pular e seguir. Já a Figura 2.4 mostra uma criatura já evoluída para a função de avaliação competir (criatura que consegue retirar o bloco cinza da outra consegue maior *fitness*). As configurações utilizadas neste trabalho foram: tamanho da população 300, número de gerações 100, taxa de *crossover* de 30% (mais de um corte), taxa de *crossover* 30% (um único corte) e taxa de reprodução assexuada de 40% (copia um dos pais e aplica mutação). Toda a modelagem das criaturas é feita baseados em paralelepípedos de tamanhos variados e pontos de junções aplicados a qualquer região da criatura.

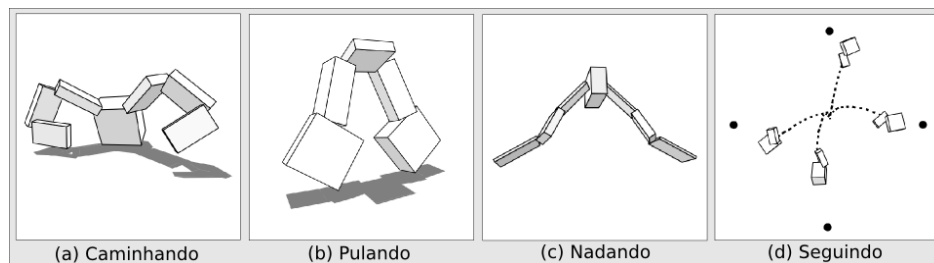


Figura 2.3: Trabalho de Karl Sims criaturas evoluídas. Fonte: (SIMS, 1994b).

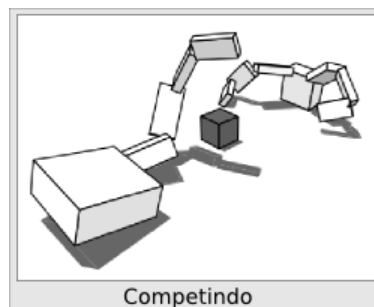


Figura 2.4: Trabalho de Karl Sims criaturas evoluídas. Fonte: (SIMS, 1994a).

O que difere o trabalho proposto com o de Karl Sims é a forma do objeto a ser evoluído. Este autor faz a evolução de criaturas virtuais que foram modeladas através de paralelepípedos, com múltiplos pontos de junção. Já este trabalho faz a evolução de robôs virtuais, que foram modeladas através de um conjunto de peças prontas (chassi, processador, roda sem motor, roda com motor e sensor), o qual o algoritmo genético irá determinar a melhor forma para a morfologia (hardware) e controle (software) da criatura.

Uma outra diferença considerável são as funções objetivos utilizadas nesta evolução. Karl Sims utilizou como função *fitness* andar, nadar, pular e seguir. Já este trabalho utiliza como função objetivo um custo associado ao hardware, ao número de colisões, ao deslocamento para o robô e ainda ao número de marcadores atendidos pelo robô.

Um reestudo sobre o trabalho de Karl Sims foi elaborado por Peter Krcah em 2007 (KRCAH, 2007). Esse estudo mostra como seria a evolução de criaturas virtuais após 13 anos da pesquisa inicial. O estudo manteve os mesmos parâmetros e técnicas utilizados por Karl Sims para efetuar a comparação com o cenário antigo. Este trabalho foi feito utilizando a biblioteca de física ODE - *Open Dynamics Engine* na linguagem Java. A Figura 2.5 mostra algumas criaturas já evoluídas através deste trabalho. A função de objetivo utilizada por Peter Krcah são as mesmas de Karl Sims.

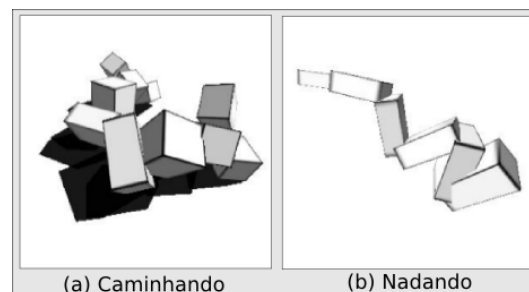


Figura 2.5: Trabalho de Peter Krcah criaturas evoluídas. Fonte: (KRCAH, 2007).

3 METODOLOGIA

Esta seção apresenta as características metodológicas do trabalho, como a classificação do tipo de pesquisa, ferramentas utilizadas, modelagem do robô, modelagem dos operadores de *crossover* e mutação, algoritmo genético, cenários simulados e quais as funções de *fitness* utilizadas.

3.1 Classificação do Trabalho

A metodologia utilizada neste trabalho pode ser classificada de várias maneiras. Segundo Jung (2004), pode-se classificar a pesquisa científica em quatro tipos: conforme a natureza, os objetivos, os procedimentos e o local da pesquisa. A Figura 3.1 apresenta um diagrama desta classificação.

O presente trabalho apresenta as descrições do tipo de pesquisa, baseada em (JUNG, 2004) e (MARCONI; LAKATOS, 2003). As classificações do trabalho conforme tal modelo são apresentadas a seguir:

1. **Quanto à natureza:** Aplicada ou tecnológica. O presente trabalho lida com a criação de um módulo de robôs que evoluem ao longo das gerações.
2. **Quanto aos objetivos:** Exploratória. A proposta é a construção de um software que seja capaz de evoluir robôs. Sendo assim é considerada exploratória já que explora este segmento.
3. **Quanto aos procedimentos:** Experimental. O trabalho visa o desenvolvimento de um módulo de robôs onde serão feitos experimentos de forma a evoluir tais robôs.

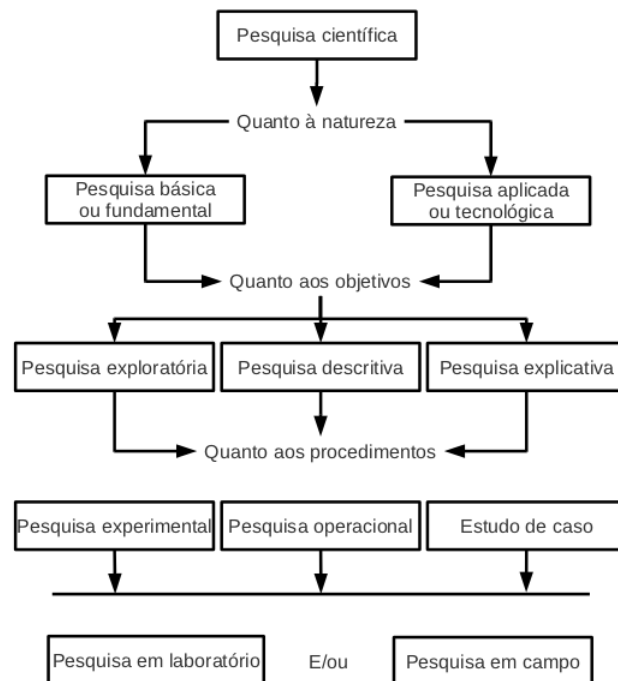


Figura 3.1: Classificação dos tipos de pesquisa. Fonte: (JUNG, 2004)

4. **Local da Pesquisa:** Em laboratório. Todo o desenvolvimento do programa se dará utilizando o computador e softwares de programação e será realizado no laboratório.

3.2 Ferramentas Utilizadas

A seguir, são descritos as principais ferramentas utilizadas no desenvolvimento deste trabalho.

3.2.1 Plataforma de Robótica

O módulo de robótica a ser projetado e implementado funcionará como uma extensão da plataforma *GrubiBots*. As principais características desta plataforma são encontradas na Seção 2.4

3.2.2 Linguagem Utilizada

A linguagem de desenvolvimento do módulo de robótica evolutiva foi a linguagem Java, já que a plataforma *GrubiBots* esta desenvolvida nesta linguagem. O Java foi escolhida também por apresentar as seguintes características. Java é uma linguagem distribuída através da licença GNU - *General Public License* e suporta os principais conceitos de orientação a objetos. Favorece extensibilidade e reusabilidade de código. É altamente portátil, aplicações funcionam da mesma maneira em qualquer ambiente completamente especificado, ou seja, é independente de plataforma. Suporta aplicações concorrentes fazendo uso de *multithreads*. Possui uma grande quantidade de recursos disponíveis através de suas APIs - *Application Programming Interface* (DEITEL, 2010).

3.2.3 Biblioteca de Física

A construção do módulo vai utilizar a biblioteca ODE que foi especialmente desenvolvida para simulação da dinâmica de corpos rígidos. A justificativa para a utilização desta biblioteca de física é que a plataforma *GrubiBots* utiliza ela como motor de física. As principais características desta biblioteca são encontradas na Seção 2.5.

3.2.4 Armazenamento dos Indivíduos

O formato utilizado para salvar cada indivíduo da população de robôs é o JSON - *JavaScript Object Notation*. Este formato foi escolhido por ser um mais leve que o formato XML, que é uma linguagem de marcação. É importante salvar cada indivíduo da população como sendo uma forma de *backup* dos indivíduos e para futura análise das soluções geradas.

O JSON possui a característica de que é fácil de ler e escrever para humanos e para computadores é fácil de interpretar e gerar. Este formato foi baseado em um subconjunto da linguagem de programação JavaScript. JSON é em formato texto e completamente independente de linguagem, pois usa convenções que são comuns à linguagem C e familiares, incluindo C++, Java, JavaScript, Perl, Python entre outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados (JSON, 2013).

3.3 Conjunto de Peças dos Robôs

A Figura 3.2 apresenta o conjunto de peças que serviram de base para a modelagem das peças do módulo. Todas as peças modeladas nesta implementação são baseadas neste conjunto de peças aqui mostrado. Nesta figura vê-se o processador do robô (Arduino), o sensor de distância (ultrasom), o motor (servo motor de rotação contínua) e as rodas (roda para servo motor).

3.4 Morfologia do Robô

Um robô é constituído por um conjunto de dispositivos eletromecânicos capazes de realizar trabalhos de maneira autônoma ou pré-programada.

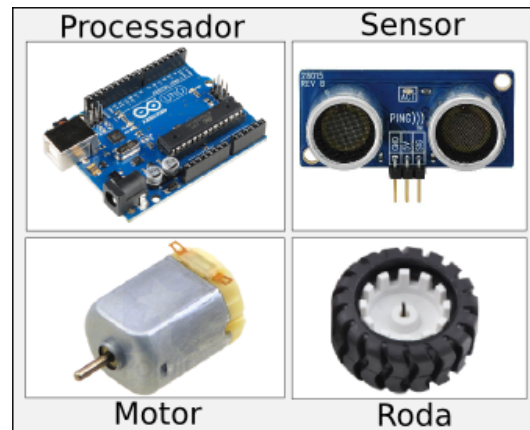


Figura 3.2: Conjunto de peças modeladas.

Neste trabalho o robô foi modelado de maneira simplificada. O termo indivíduo será utilizado neste trabalho como sinônimo da palavra robô. Os elementos que compõem um indivíduo são: *Id*, cor, posição (x, y, z) , ângulo em relação ao sistema de coordenadas, DNA e o conjunto de peças que o formam. A seguir, será explicado melhor como são as peças constituintes do robô.

Um robô é constituído por um conjunto de peças. Neste trabalho foram definidas as seguintes peças que constituirão o indivíduo: chassi, processador, roda sem motor, roda com motor e sensor de distância. Futuramente outras peças poderão ser facilmente incorporadas no módulo de robótica.

A Figura 3.3 mostra o conjunto de peças implementadas no módulo de robótica. Este conjunto de peças serão representados neste trabalho por *P*.

3.4.1 Chassi

O Chassi é uma das peças que representa o corpo do robô. Esta peça é uma plataforma de encaixe para todas as outras peças do indivíduo. Visando se conseguir uma simplificação, no momento de acoplar as peças, o Chassi foi subdividido em células. O formato desta peça é de um paralelepípedo e possui dimensões de

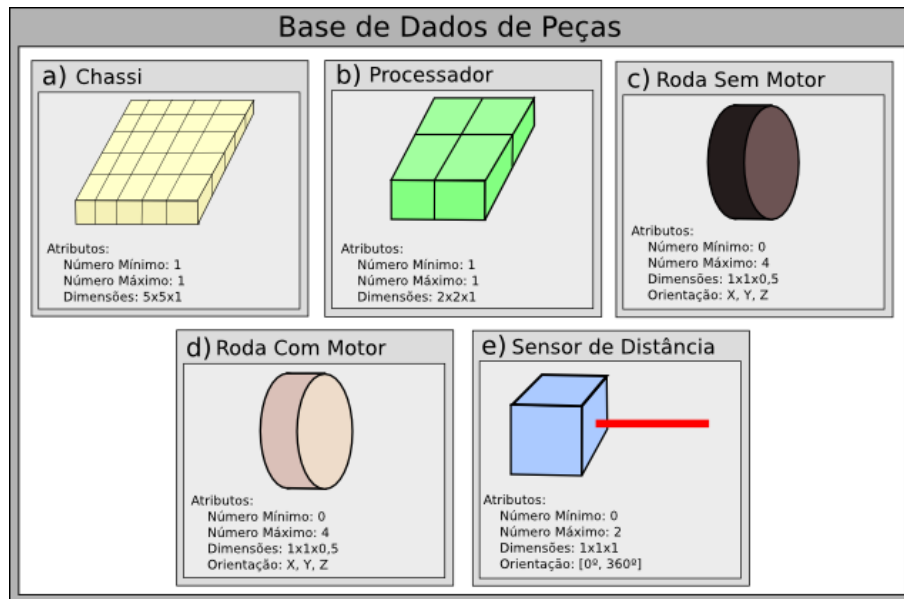


Figura 3.3: Conjunto de peças do robô.

$5 \times 5 \times 1$. O Chassi possui $5 \times 5 = 25$ células no topo e $4 \times 5 = 20$ células nas laterais.

Este componente possui várias células que encontram-se no topo e dos lados. As células do Chassi são as unidades mínimas para acoplagem das peças. Uma célula é um quadrado com dimensões 1×1 . Sendo assim as células possuem pontos de junção onde as outras peças poderão ser encaixadas.

A Figura 3.3-a) mostra a morfologia do Chassi. Seus atributos básicos são: comprimento (5), largura (5), altura (1), cor e massa. Foi definido que um robô deverá possuir somente um Chassi. Esta peça será representada neste trabalho por *Pchs*.

3.4.2 Processador

O `Processador` é o local onde ficará armazenado as decisões de controle do indivíduo (comportamentos). Esta peça ocupará as células de um quadrado de dimensões 2×2 .

A Figura 3.3-*b*) apresenta a morfologia do `Processador`. Entre seus atributos básicos tem-se: comprimento (2), largura (2), altura (1), cor e massa. Este componente só pode ser colocado sobre a parte do topo do `Chassi`. Cada robô deverá ter um único `Processador`. A representação utilizada neste trabalho para o processador é p_{cpu} .

3.4.3 Roda Sem Motor

A `Roda Sem Motor` representa uma das peças modeladas, esta peça auxilia no deslocamento do robô, diminuindo o atrito com o chão. Este componente ocupará uma célula por completo e parcialmente as células vizinhas.

A Figura 3.3-*c*) mostra a morfologia da `Roda Sem Motor`. Os seus atributos básicos são: raio, largura, cor, massa e orientação. Esta peça pode ser acoplada tanto sobre o topo quanto nas laterais do `Chassi`. Cada robô poderá ter de no mínimo 0 e no máximo 4 `Roda Sem Motor`. Esta peça será representada neste trabalho por p_{rsm} .

A `Roda Sem Motor` possuirá uma determinada orientação de posicionamento e se adequará automaticamente de acordo com o lugar escolhido para ser posicionada. Esta orientação formará um ângulo perpendicular entre o eixo de rotação e a superfície o qual foi acoplada.

3.4.4 Roda Com Motor

A Roda Com Motor faz com que o robô consiga se deslocar no ambiente desde que esta roda esteja ligada. Para ligar ou desligar a roda um programa de controle transmitirá um comando para a mesma. Este componente ocupará uma célula por completo e parcialmente as células vizinhas.

A Figura 3.3-d) mostra a morfologia da Roda Com Motor. Seus atributos básicos são: raio, largura, cor, massa, orientação e velocidade. Sua orientação pode variar entre girar a roda para frente ou trás. Este componente pode ser acoplada tanto sobre o topo quanto nas laterais do Chassi. Cada indivíduo poderá ter de no mínimo 0 e no máximo 4 Roda Com Motor. A representação para essa peça neste trabalho é dada por p_{rcm} .

3.4.5 Sensor de Distância

O Sensor de Distância é uma peça que consegue extrair informações do ambiente visando localizar obstáculos e conseqüentemente calcular a distância que o robô se encontra deste obstáculo. Este componente ocupará uma célula de dimensões de 1×1 .

A Figura 3.3-e) mostra a morfologia do Sensor de Distância. Os seus atributos básicos são: comprimento (1), largura (1), altura (1), cor, massa, orientação e alcance do sensor. Sua orientação pode ser: para cima, baixo, esquerda ou direita. O sensor de distância foi definido como sendo possível sua acoplagem somente no topo do chassi. Cada robô poderá ter no mínimo 0 e no máximo 2 sensores. Esta peça será representada neste trabalho por p_{sdd} .

3.5 Pontos de Conexão do Robô

Uma etapa anterior à montagem do robô é definir no simulador onde estão os pontos que poderão ser acopladas as peças do indivíduo. Neste trabalho conforme citado anteriormente o Chassi é constituído de um conjunto de células. Em cada uma das células do robô foi definido um ponto de junção, ou seja, local onde é possível acoplar as peças no robô.

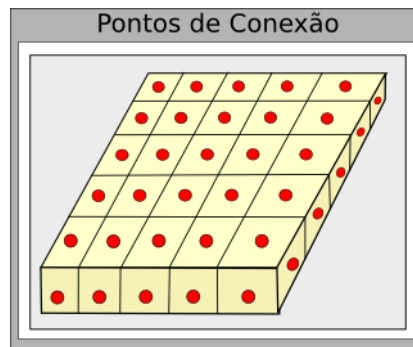


Figura 3.4: Pontos de conexão do Chassi.

A Figura 3.4 ilustra onde estão localizados cada ponto de junção do robô (pontos vermelhos). Sendo assim, as peças poderão somente ser acopladas nestes pontos. Não é permitido ao AG colocar uma peça que não seja em um destes pontos de conexão. Para fins didáticos subdividiu-se o Chassi em topo e laterais. Na parte inferior (abaixo) do Chassi não é possível o acoplamento de peças do indivíduo.

3.6 Montagem da Morfologia do Robô

A montagem do robô efetuada pelo algoritmo genético funcionará da seguinte forma. Existe uma conjunto de peças $P = \{p_{chs}, p_{cpu}, p_{rsm}, p_{rcm}, p_{sdd}\}$, a partir deste conjunto se selecionará algumas peças que constituirão o indivíduo. Em se-

guida estas peças seleccionadas serão acopladas em alguma(s) célula(s) do Chassi do robô, conforme Figura 3.5.

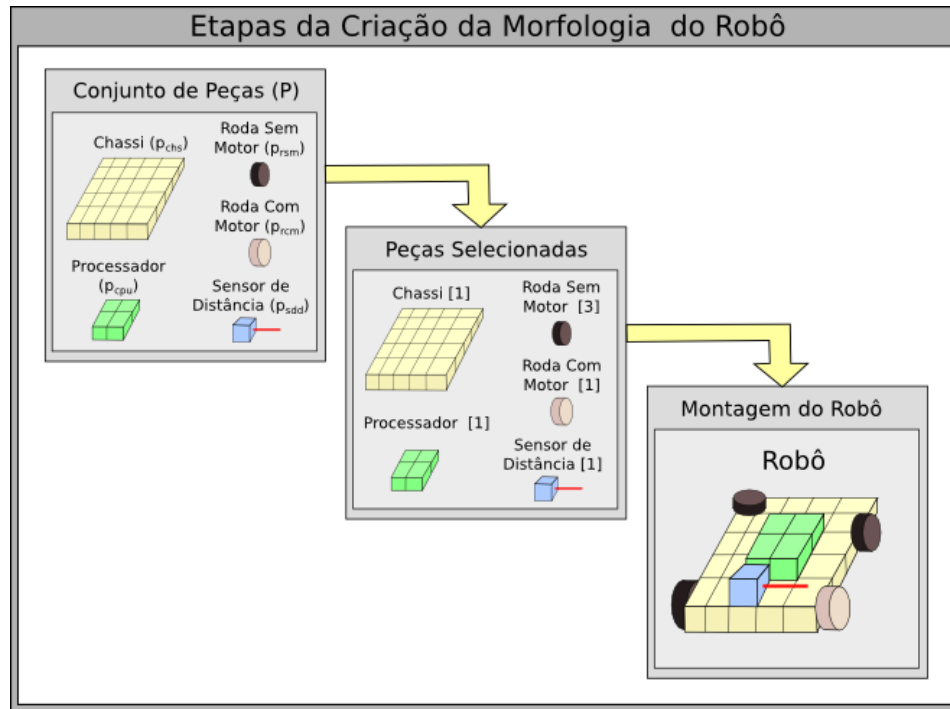


Figura 3.5: Etapas envolvidas na montagem da morfologia do robô.

A evolução se dará após gerado toda a população de robôs em uma dada geração. Após a completa montagem dos robôs, os mesmos serão avaliados pelo AG e pontuado com algum valor de acordo com seu desempenho na resolução da tarefa e dos recursos gastos. Serão seleccionados os melhores robôs para que sejam combinados, gerando assim um filho. Este processo tende a garantir que a cada geração os robôs melhorem a sua pontuação. Assim ao final de um certo tempo espera-se um robô razoável na resolução dos problemas a serem resolvidos. Tais problemas serão descritos posteriormente.

3.6.1 Operadores de Crossover

O operador de recombinação faz com que os descendentes recebam em seu código genético parte do código genético do pai_1 e parte do pai_2 . Em seguida, são descritos todos os operadores de *crossover* implementados.

3.6.1.1 Crossover de Processador

A operação de recombinação de processador para constituir o filho é definida da seguinte forma: primeiramente será selecionado um dos pais e em seguida pegará o processador do pai selecionado para constituir o filho.

3.6.1.2 Crossover de Roda Sem Motor

O operador de *crossover* para a seleção das rodas sem motor para construção do filho se fará da seguinte forma: primeiramente é criada uma lista contendo o conjunto de rodas sem motor de ambos os pais. A seguir, itera-se pela lista e seleciona-se uma roda de um dos pais com uma probabilidade de 50%.

3.6.1.3 Crossover de Roda Com Motor

O operador de *crossover* de Roda Com Motor é análogo ao operador de *crossover* de Roda Sem Motor.

3.6.1.4 Crossover de Sensor

O operador de recombinação dos sensores para compor o filho foi definido da seguinte forma: Primeiramente foi feito um mapeamento do topo do Chassi do robô, conforme a Figura 3.6, a qual existem quatro setores e os sensores somente

podem ser acoplados sobre estes setores. A implementação do cruzamento garante que sejam selecionados sensores de ambos os pais a partir dos setores definidos nesta figura.

Estes setores foram definidos para que não exista a sobreposição de um sensor atuando sobre uma região de outro sensor. Assim, por exemplo, o sensor S_1 possui um ângulo que varia dentro de um subconjunto de $[90^\circ, 180^\circ]$, o ângulo do sensor $S_2 \in [180^\circ, 270^\circ]$, o ângulo do sensor $S_3 \in [270^\circ, 360^\circ]$, o ângulo do sensor $S_4 \in [0^\circ, 90^\circ]$. É importante lembrar que todos os ângulos dos sensores são um subconjunto dos intervalos dados, variando de 30 em 30 graus.

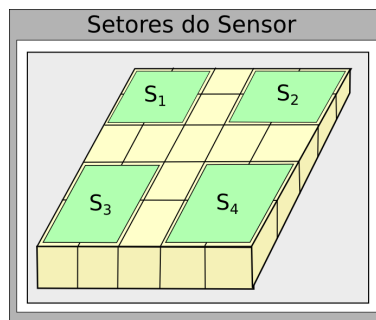


Figura 3.6: Mapeamento de setores do sensor no chassi.

3.6.2 Operadores de Mutação

O operador de mutação tem como objetivo permitir maior variabilidade genética na população, impedindo que a busca fique estagnada em um mínimo local. Adiante, são descritos todos os operadores de mutação implementados.

3.6.2.1 Adicionar Peça

Este operador é responsável por adicionar uma peça ao robô. As peças que podem ser adicionadas dependem do problema a ser resolvido. No primeiro problema

as peças são: Roda Sem Motor e Roda Com Motor. Já no segundo problema as peças são: Roda Sem Motor, Roda Com Motor e Sensor de Distância.

A adição da nova peça se dará da seguinte forma: será dada uma probabilidade igual para cada peça. Em seguida, o algoritmo selecionará uma posição onde a peça possa se acoplar, e por fim colocará naquela posição. A chance de adicionar uma nova peça é de 25% das vezes que ocorrer a mutação.

3.6.2.2 Remover Peça

O operador *remover peça* é responsável por remover uma peça ao robô. As peças que podem ser removidas dependem do problema a ser resolvido. No primeiro problema estas peças são: Roda Sem Motor e Roda Com Motor. Já o segundo problema as peças são: Roda Sem Motor, Roda Com Motor e Sensor de Distância.

A remoção de uma determinada peça se dará da seguinte forma: será selecionada uma peça aleatória do conjunto de peças. E em seguida, será removida a peça selecionada. A chance de remoção de uma peça é de 25% das vezes que ocorrer a mutação.

3.6.2.3 Trocar Peças

Um último operador implementado é o *trocas de peças* que é responsável por efetuar trocas de peças do robô. A chance de ocorrer a troca de uma peça é de 50% das vezes que ocorrer a mutação.

Seu funcionamento será da seguinte forma: primeiramente sorteará uma peça a ser trocada por outra peça. Caso esta peça seja um Sensor de Distância a troca se dará na orientação do sensor. Já no caso da peça ser uma Roda Sem Motor ou ainda Roda Com Motor, será verificado se a peça encontra-se nas late-

rais ou no topo do chassi. Caso esteja nas laterais então trocará o tipo de roda, por exemplo, Roda Sem Motor virará Roda Com Motor, já a peça Roda Com Motor virará Roda Sem Motor. Caso esteja no topo do Chassi então a peça poderá se transformar em um Sensor de Distância também.

3.7 Controle do Robô

Este trabalho visa encontrar a morfologia ideal para o robô, embora em alguns dos problemas abordados esteja-se evoluindo tanto a morfologia quanto o controle do indivíduo. A etapa que dá ênfase no controle (software) do robô será abstraída neste trabalho, mas será abordada em outro trabalho feito pelo grupo de pesquisa. Esta etapa responsável pelo controle dos comportamentos do indivíduo fará uso de Arquitetura de Subsunção.

3.8 Algoritmo Genético

Neste trabalho foi implementado o algoritmo genético clássico. A seguir, tem-se o pseudocódigo do AG, mostrado pelo Algoritmo 1.

Na linha 2 até 6 tem-se um laço de repetição para que irá gerar toda a população inicial de robôs. Todos estes robôs serão inicializados de forma aleatória (linha 3), em seguida serão simulados (linha 4), e por fim serão avaliados pela função de *fitness* (linha 5).

A seguir, na linha 7 um contador de gerações de robôs é inicializada. O laço da linha 8 repetirá até que termine o tempo ou o total de gerações da simulação. Em seguida (linha 9) tem-se a estrutura de repetição para que irá percorrer até o fim da população. Neste laço um valor aleatório é sorteado (linha 10). Verifica-se em seguida se irá ocorrer *crossover* (linha 11) caso ocorra, dois indivíduos são

Algoritmo 1: Algoritmo Genético.

Entrada: População: *pop*, Inteiro: *taxaMutação*, *taxaCrossover*, *totalGerações*
Saída: Melhores Indivíduos da População

```

1  início
2  para  $i \leftarrow 0$  até número de indivíduos faça
3      inicializa o indivíduo  $ind(i)$ ;
4      simulação do indivíduo  $ind(i)$ ;
5      avaliação do indivíduo  $ind(i)$ ;
6  fim para
7  geraçãoAtual  $\leftarrow 0$ ;
8  enquanto (( $tempoAtual < tempoTotal$ ) and ( $geraçãoAtual < totalGerações$ ))
9      faça
10         para  $i \leftarrow 0$  até número de indivíduos faça
11              $rnd \leftarrow valorAleatório()$ ;
12             se ( $rnd < taxaCrossover$ ) então
13                 ( $ind1, ind2$ )  $\leftarrow$  seleciona dois pais;
14                  $novInd \leftarrow crossover(ind1, ind2)$ ;
15             fim se
16             senão
17                  $novInd \leftarrow$  seleciona um pai;
18             fim se
19              $rnd \leftarrow valorAleatório()$ ;
20             se ( $rnd < taxaMutação$ ) então
21                  $novInd \leftarrow$  mutação( $novInd$ );
22             fim se
23             simulação do indivíduo  $novInd$ ;
24             avaliação do indivíduo  $novInd$ ;
25             insere  $novInd$  na população  $pop$ ;
26         fim para
27         geraçãoAtual  $\leftarrow$  geraçãoAtual + 1;
28     fim enquanto
29 fim
  
```

sorteados pelo método do torneio (linha 12) e por fim é aplicado o operador *crossover* gerando um novo indivíduo (linha 13). Caso não ocorra *crossover* então um indivíduo apenas é sorteado pelo método do torneio gerando um novo indivíduo (linha 16). Uma nova verificação é feita para ver se irá ocorrer mutação (linha 19), se a operação lógica for verdadeira a operação mutação irá ocorrer no indivíduo novo. Em seguida, o novo indivíduo criado é simulado (linha 22) e em seguida é avaliado o seu *fitness* (linha 23). Para finalizar este laço de repetição o indivíduo é inserido na população original. Na linha 26 a geração atual é incrementada.

Após encerrada a simulação do Algoritmo 1 espera-se uma população de indivíduos mais apta a resolver o problema dado proposto para o mesmo.

3.9 Cenários Simulados e Função de Fitness

Esta seção mostra alguns problemas e quais foram os cenários utilizados nestes problemas. Em cada um dos cenários propostos uma função objetivo foi determinada e o problema foi modelado como um problema de minimização. O algoritmo utilizará este objetivo para atualizar o *fitness* de cada indivíduo.

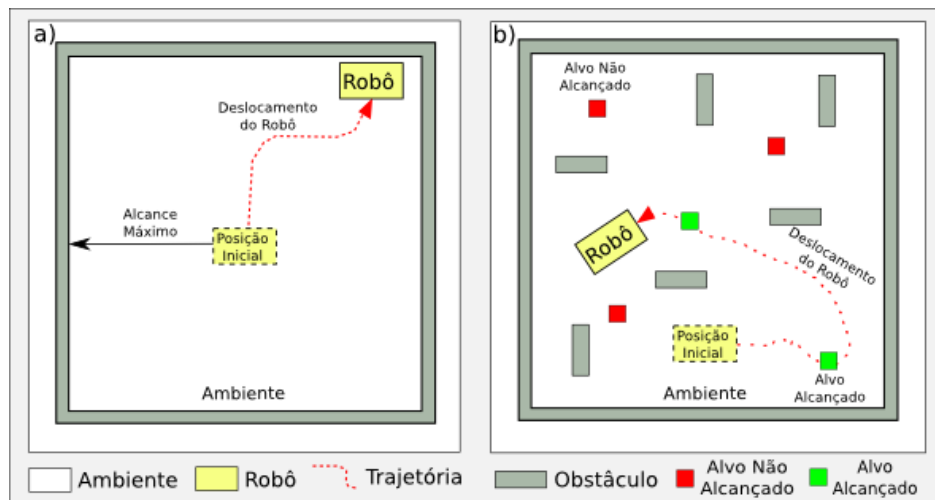


Figura 3.7: Problemas básicos de deslocamento do robô.

A Figura 3.7 mostra dois problemas básicos. O primeiro problema, apresentado na Figura 3.7-a), está descrito o problema da locomoção do robô neste ambiente. Este problema será chamado de *Problema₁* e sua definição é dada a seguir:

Definição 1. *Problema₁*: Problema da locomoção do robô no ambiente livre de obstáculos. Nesse problema, o robô está localizado no centro de uma determinada região quadrada com comprimento e largura $200ud$, onde ud significa Unidade de Distância. O robô que conseguir o maior deslocamento nesta região terá maior probabilidade de passar suas características para a próxima geração. Assim para

resolver este problema vai-se utilizar as seguintes peças: Chassi, Processador, Roda Sem Motor e Roda Com Motor. Neste problema, não é necessário o uso de Sensor de Distância, já que o ambiente é livre de obstáculos.

Um outro problema a ser resolvido é o de deslocamento em um ambiente com obstáculos e captura de marcadores encontrados no cenário, veja Figura 3.7-b). Este segundo problema será chamado de *Problema₂* e sua definição encontra-se a seguir:

Definição 2. *Problema₂*: Problema de locomoção do robô no ambiente com obstáculos. Nesse problema, o robô está localizado abaixo do centro de uma região quadrada com comprimento e largura $200ud$. Sobre este cenário foram espalhados cinco objetos marcadores que o robô terá que passar sobre os mesmos. O *fitness* do robô melhora quanto maior a quantidade de marcadores alcançados. Um fator que penaliza o valor dado a avaliação do robô são as colisões contra os obstáculos no ambiente. Neste problema é necessário o uso de todas as peças modeladas que são: Chassi, Processador, Roda Sem Motor, Roda Com Motor e Sensor de Distância.

A função de avaliação para o *Problema₁* foi definida por partes e é dada por:

$$\text{Min } F(\dots) = \underbrace{w_{hw} \cdot \sum_{p \in P} (w_p \cdot q_p)}_{\text{custo do hardware}} + \underbrace{w_{sw} \cdot (C_{lado}/2 - R_{dest})}_{\text{custo do software}} \quad (3.1)$$

A função objetivo para o *Problema₂* foi definida por partes e é dada por:

$$\text{Min } F(\dots) = \underbrace{w_{hw} \cdot \sum_{p \in P} (w_p \cdot q_p)}_{\text{custo do hardware}} + \underbrace{w_{mark} \cdot (q_{markT} - q_{markC})}_{\text{custo dos marcadores}} + \underbrace{w_{col} \cdot q_{col}}_{\text{custo colisões}} \quad (3.2)$$

Em ambas as funções objetivo o custo associado ao hardware é dado por:

$$\sum_{p \in P} (w_p \cdot q_p) = \underbrace{w_{chs} \cdot q_{chs}}_{\text{custo chs}} + \underbrace{w_{cpu} \cdot q_{cpu}}_{\text{custo cpu}} + \underbrace{w_{rsm} \cdot q_{rsm}}_{\text{custo rsm}} + \underbrace{w_{rcm} \cdot q_{rcm}}_{\text{custo rcm}} + \underbrace{w_{sdd} \cdot q_{sdd}}_{\text{custo sdd}} \quad (3.3)$$

Parâmetros:

w_{hw} Peso do Hardware - Representa o peso dado ao hardware do robô.

w_{sw} Peso do Software - Representa o peso dado ao software do robô.

w_{mark} Peso dos Marcadores - Representa o peso dado aos marcadores colocados sobre o ambiente.

w_{col} Peso das Colisões - Representa o peso dado as colisões do robô contra os obstáculos.

P Conjunto de Peças do Robô - Representa o conjunto de peças utilizadas na construção do robô.

w_p Peso da Peça p - Representa o peso atribuído a peça p .

q_p Quantidade da Peça p - Representa a quantidade da peça p .

chs Peça Chassi - Representa a peça Chassi.

cpu Peça Processador - Representa a peça Processador.

rsm Peça Roda Sem Motor - Representa a peça Roda Sem Motor.

rcm Peça Roda Com Motor - Representa a peça Roda Com Motor.

sdd Peça Sensor de Distância - Representa a peça Sensor de Distância.

C_{lado} Lado do Cenário - Representa o comprimento do lado do cenário.

R_{desl} Deslocamento do Robô - Representa o deslocamento do robô no cenário.

q_{markT} Quantidade de Marcadores Totais - Representa a quantidade de marcadores totais espalhados pelo ambiente.

q_{markC} Quantidade de Marcadores Capturados - Representa a quantidade de marcadores atendidos pelo robô no ambiente.

q_{col} Quantidade de Colisões - Representa a quantidade de colisões do robô com os obstáculos do ambiente.

3.10 Configurações e Parâmetros Gerais Utilizados

Esta seção apresenta as principais configurações utilizadas nas simulações da evolução do robô. A Tabela 3.1 mostra a quantidade de cada tipo de peças utilizada na montagem do robô. Os intervalos para a quantidade mínima e máxima de peças foram definidos empiricamente, tendo como base a quantidade de células presentes no Chassi.

Atributos	Mínimo	Máximo
Quantidade de Chassi - q_{chs}	1	1
Quantidade de Processador q_{cpu}	1	1
Quantidade de Roda Sem Motor - q_{rsm}	0	4
Quantidade de Roda Com Motor - q_{rcm}	0	4
Quantidade de Sensor de Distância - q_{sdd} (<i>Problema₁</i>)	0	0
Quantidade de Sensor de Distância - q_{sdd} (<i>Problema₂</i>)	0	2

Tabela 3.1: Quantidade de peças utilizadas na montagem do robô.

A Tabela 3.2 apresentada mostra as configurações utilizadas para os pesos de cada uma das peças do robô. Analisando esta tabela pode-se observar que há um maior peso para o Chassi, seguido pelo Processador, Roda Com Motor, Sensor de Distância e por fim Roda Sem Motor. Os valores dados aos pesos das peças foram definidos empiricamente.

A Tabela 3.3 mostra os pesos de cada uma das partes da simulação. O peso do hardware é o menor, seguido pelo peso das colisões e por fim empatados tem-

Atributos	Valor
Peso do Chassi - w_{chs}	6
Peso do Processador - w_{cpu}	4
Peso da Roda Sem Motor - w_{rsm}	1,2
Peso da Roda Com Motor - w_{rcm}	2,4
Peso do Sensor de Distância - w_{sdd}	1,5

Tabela 3.2: Pesos das peças utilizadas na montagem do robô.

se os pesos do software e dos marcadores. No primeiro problema o algoritmo genético tentará minimizar primeiro o software encontrado para o robô, seguindo pela minimização da quantidade de recursos gasto no hardware do robô. Já para o segundo problema o AG tentará minimizar primeiro o custo associado aos marcadores, seguindo pela minimização da quantidade de colisões dada pelo robô e por fim minimizará a quantidade de recursos gasto no hardware do robô. Os valores dados aos pesos utilizados em cada uma das parte das simulações foram definidos empiricamente.

Atributos	Valor
Peso do Hardware - w_{hw}	1
Peso do Software - w_{sw}	200
Peso dos Marcadores - w_{mark}	200
Peso das Colisões - w_{col}	10

Tabela 3.3: Pesos utilizados nas simulações efetuadas.

Os cenários utilizados nos experimentos contém quadrado com $200ud$ de lado. O primeiro cenário não possui obstáculos, já o segundo contém um total de seis obstáculos.

3.11 Configurações dos Computadores Utilizados

O algoritmo para evolução dos robôs que foi implementado neste trabalho é paralelo e distribuído. O fato de ser paralelo significa que cada parte seja exe-

cutado em uma *thread* independente e por ser distribuído significa que diferentes computadores podem executar um indivíduo representado um solução diferente.

Um único computador foi utilizado nos experimentos realizados para o *Problema₁*. As configurações básicas para este computador encontra-se na Tabela 3.4. Como este experimento foi executado somente em um computador não se pode extrair vantagens do fato dele ser distribuído. Entretanto nestes primeiros experimentos se extraiu vantagens do paralelismo do programa.

Configuração do Computador	
Processador	Intel(R) Core(TM) i3-2100
Frequência do Processador	3.1 GHz
Arquitetura do Processador	64 bits
Memória RAM	8 GB
Capacidade do HD	1 TB
Sistema Operacional (SO)	Ubuntu 12.04
Arquitetura do SO	32 bits

Tabela 3.4: Configurações do computador utilizado nos testes.

O outro problema que foi resolvido (*Problema₂*) utilizou um *cluster* para obter os resultados. O *cluster* apresenta um conjunto com cinco computadores. As configurações básicas dos computadores utilizados nos testes em *cluster* podem ser encontrados na Tabela 3.5.

Configuração do Cluster	
Processador	Intel(R) Core(TM) 2 Quad CPU Q8400
Frequência do Processador	2.66GHz
Arquitetura do Processador	64 bits
L2 Cache	4 MB
Memória RAM	4 GB
Capacidade do HD	320 GB
Sistema Operacional (SO)	Ubuntu 13.04
Arquitetura do SO	32 bits

Tabela 3.5: Configurações do *cluster* utilizado

4 RESULTADOS E DISCUSSÕES

A seguir são apresentados os resultados obtidos nos experimentos executados e discussões sobre os mesmos.

As configurações utilizadas no *Problema₁* são apresentadas na Tabela 4.1. Pode-se perceber que este primeiro problema não foi executado no *cluster*. O número de *steps*, que é uma maneira de efetuar a contagem do tempo de simulação, é de 600. O algoritmo genético foi utilizado com elitismo e torneio com tamanho 3. A etapa de controle do robô utilizando subsunção utilizou uma *trigger* que foi a *True Trigger*, já a parte de comandos utilizados foram *Start Command* e *Stop Command*, neste trabalho não serão apresentados maiores detalhes sobre esta etapa de controle.

Parâmetro	Valor
Executado no Cluster	Não
Número de Steps (Tempo)	600
Tamanho da População	100
Número de Gerações	50
Tamanho do Torneio	3
Elitismo	Sim
Triggers Utilizadas	True
Comandos Utilizados	Start Command e Stop Command

Tabela 4.1: Configurações utilizadas nas simulações do *Problema₁*.

Os resultados para o *Problema₁* são mostrados a seguir na Figura 4.1. Foram utilizadas as taxas de *crossover* de 75% e de mutação de 85% na configuração destes primeiros testes. Nos experimentos foram executadas três simulações para avaliar a evolução da morfologia dos robôs. Essa figura mostra a evolução do *fitness* nas três simulações executadas e o *fitness* médio obtido.

Analisando o gráfico da simulação 1 da Figura 4.1 vê-se que ocorreu uma estabilização do *fitness* (convergência) do robô após a geração 1. Já o gráfico da simulação 2 vê-se a convergência do *fitness* após a geração 6. E por fim o gráfico

da simulação 3 o indivíduo converge após também a geração 1. O tempo total gasto por cada uma das três simulações são: *1h 45min 27seg*, *1h 44min 6seg* e *1h 44min 33seg*.

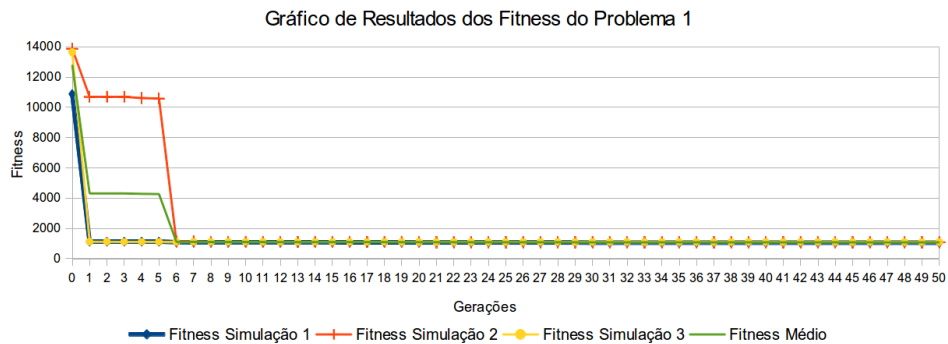


Figura 4.1: Resultados obtidos nos experimentos para *Problema₁*.

Novas simulações para este primeiro problema (*Problema₁*) foram executadas com taxas de *crossover* de 80% e a taxa de mutação de 10%. A Figura 4.2 mostra os melhores *fitness* em função das gerações para as três simulações executadas e o *fitness* médio. O tempo total gasto por cada uma das três simulações são: *1h 44min 14seg*, *1h 44min 50seg* e *1h 44min 46seg*.

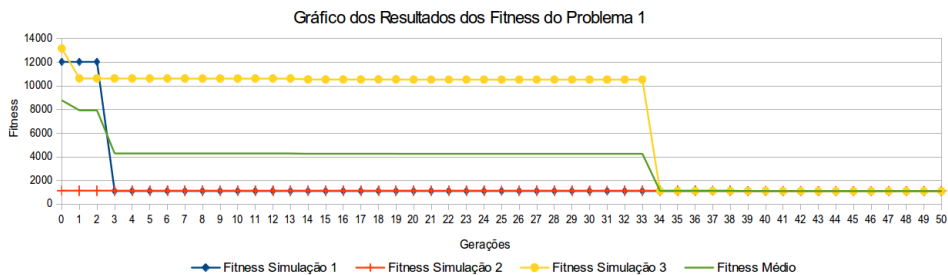


Figura 4.2: Resultados para novas configurações no *Problema₁*.

O melhor *fitness* obtido para o primeiro problema com taxas de *crossover* de 75% e a taxa de mutação de 85% foi de 1048,92. O *fitness* médio para as três execuções com estas configurações foi de 1077,65. Já o melhor *fitness* para o primeiro problema com taxas de *crossover* de 80% e a taxa de mutação de 10%

foi de 1087,81. O *fitness* médio para as três execuções com estas configurações foi de 1099,09. Sendo assim as configurações de recombinação de 75% e mutação de 85% foram ligeiramente melhores que as de 80% e 10%.

As configurações utilizadas no *Problema₂* são apresentadas na Tabela 4.2. Já este segundo problema foi executado utilizando o *cluster*. O número de *steps* utilizados são de 6000. A etapa de controle do robô utilizando subsunção utilizou *triggers* que foram True, False, Random, Sensor e Logic Trigger, já a parte de comandos utilizados foram Start Command e Stop Command.

Parâmetro	Valor
Executado no Cluster	Sim
Número de Steps (Tempo)	6000
Tamanho da População	100
Número de Gerações	50
Tamanho do Torneio	3
Elitismo	Sim
Triggers Utilizadas	True, False, Random, Sensor e Logic
Comandos Utilizados	Start Command e Stop Command

Tabela 4.2: Configurações utilizadas nas simulações do *Problema₂*.

Alguns experimentos foram executados para o *Problema₂*. As taxas de *crossover* e mutação utilizadas foram de 75% e 85%. Foram utilizadas estas taxas devido aos melhores resultados obtidos por estas taxas para o *Problema₁*. Este segundo cenário faz o uso sensores de distância. O alcance destes sensores variam de 200ud a 30ud. O tempo total gasto em cada uma das duas simulações são: 7h 19min 27seg e 7h 16min 58seg.

Os resultados da morfologia dos robôs para os experimentos realizados podem ser visualizados nas Figuras 4.4 e 4.5. Analisando estas figuras vê-se o formato de alguns dos robôs, a qual geração eles pertencem e qual o *fitness* do mesmo. Ao analisar a trajetória descrita pelos robôs finais (robôs da geração 50 do *Problema₁* com configuração 1 e 2) dessa figura observou-se que a trajetória ocorreu em linha reta. O que realmente representa o melhor caminho para o robô, já que em linha

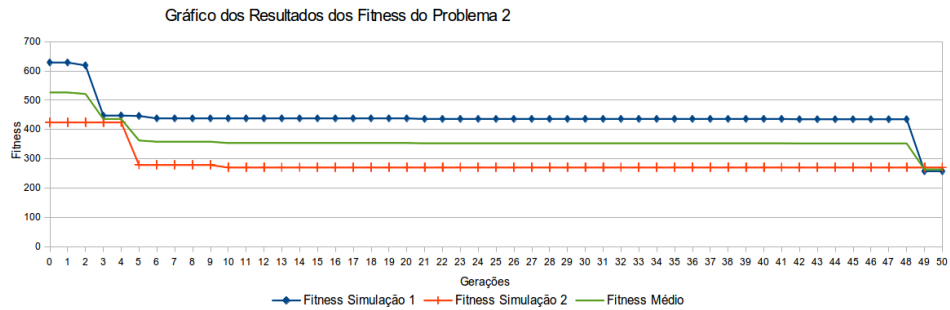


Figura 4.3: Resultados obtidos nos experimentos para *Problema₂*.

reta o robô atinge o maior deslocamento. Pode-se observar para este *Problema₁* que o melhor menor custo de hardware foi obtido já que o robô possui um único chassi, um único processador, uma única roda com motor, e duas rodas sem motores (note que são necessárias 3 rodas para manter o equilíbrio do robô), nenhum sensor foi necessário para esta solução deste problema.

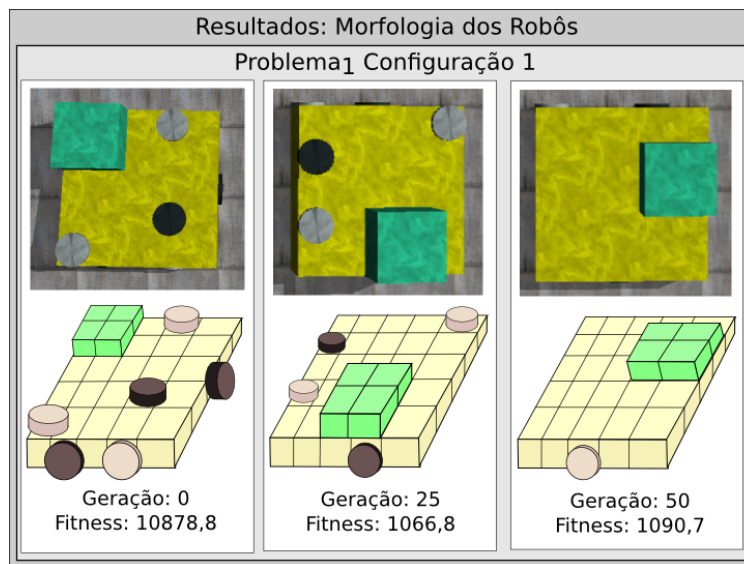


Figura 4.4: Resultados gerais para a morfologia dos robôs.

A Figura 4.6 apresenta os resultados gerais da morfologia para o *Problema₂*. Este robô apresenta em sua estrutura morfológica os seguintes componentes: chassi, processador, rodas com motor, rodas sem motor e sensores.

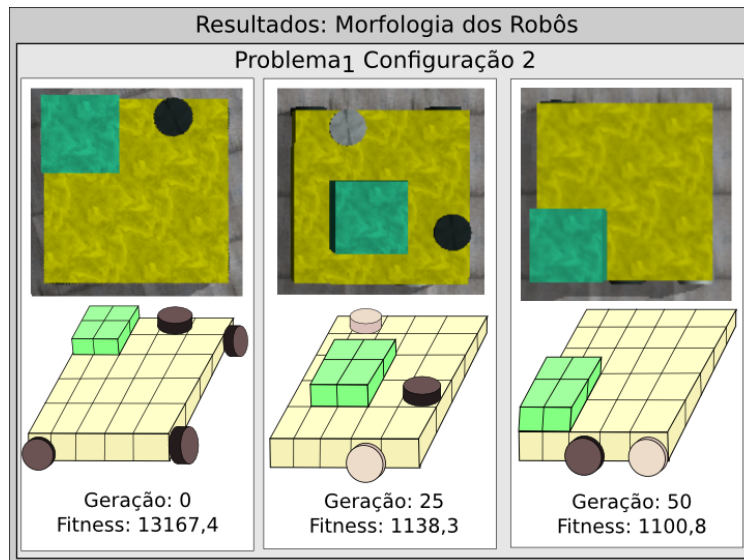


Figura 4.5: Resultados gerais para a morfologia dos robôs.

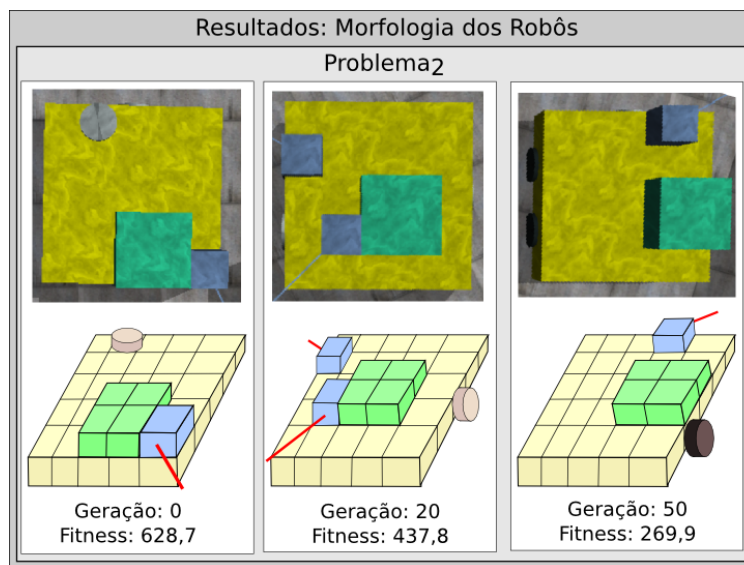


Figura 4.6: Resultados gerais para a morfologia dos robôs.

A Figura 4.7 apresenta a solução encontrada por o robô no *Problema2*. Este robô foi capaz de atender três marcadores de um total de cinco no intervalo de tempo (*steps*) estabelecido. Como pode-se perceber o robô acabou colidindo com alguns obstáculos no cenário o que fez com que o *fitness* deste indivíduo fosse

5 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou um módulo de robótica evolutiva capaz de evoluir a morfologia de um robô, com base em um conjunto de peças, um cenário e uma função objetivo. Este módulo foi implementado dentro da plataforma de robótica *GrubiBots*.

Os resultados obtidos foram satisfatórios para o primeiro cenário onde foi feita a evolução da morfologia e do controle do robô. Foram executados testes com duas configurações diferentes para o primeiro cenário. Os primeiros testes utilizaram: tamanho da população 100, número de gerações 50, taxa de *crossover* 75% e taxa de mutação 85% e obteve-se um *fitness* médio de 1077,65. Nestes testes foi gasto em média *1h 44min 42seg* para terminar a execução. Outros testes foram desenvolvidos com configurações: tamanho da população 100, número de gerações 50, taxa de *crossover* 80% e taxa de mutação 10% e obteve-se um *fitness* médio de 1099,09. Neste teste foi gasto em média *1h 44min 37seg* para finalizar a execução. Assim, com a primeira configuração testada obteve-se um valor de *fitness* ligeiramente melhor, o que comprova que uma alta taxa de mutação para este problema é melhor.

Experimentos para o segundo cenário visando a evolução da morfologia do robô também foram executados. Neste novo cenário foi utilizada as seguintes configurações nos experimentos: tamanho da população 100, número de gerações 50, taxa de *crossover* 75% e de mutação 85%. No total de duas execuções para este cenário foram simuladas. O valor do *fitness* médio para este segundo problema foi de 263,55. O tempo gasto em média nas execuções neste cenário foi de *7h 18min 22seg*. Nas duas execuções para este cenário a quantidade de marcadores atendidas foram de três de um total de cinco. Em ambas as execuções houveram penalização por causa de colisões obtidas com os obstáculos do cenário.

Uma das grandes contribuições deste trabalho é que durante o seu desenvolvimento novas funcionalidades foram incrementadas e melhoradas na plataforma *GrubiBots*. As peças do robô, cenários elaborados, algoritmo genético, entre outras podem funcionar em várias aplicações diferentes, o que viabiliza a reutilização dos códigos produzidos neste trabalho.

Como trabalhos futuros tem-se a possibilidade em se estabelecer novas regras e cenários e evoluí-los para aplicações totalmente diferentes. Como exemplo de aplicações tem-se: trabalhos envolvendo evolução de enxame de robôs, trabalhos com evolução colaborativa, evolução competitiva, evolução cooperativa e competitiva, resolver problemas em cenários mais complexos (cenários dinâmicos), resolver problemas de deslocamento de cargas, entre outros.

REFERÊNCIAS

BATURONE, A. O. *Robotica: Manipuladores y robots moviles*. [S.l.]: marbombo, boixareu editores, 2001.

BENTO, E. P.; KAGAN, N. Algoritmos geneticos e variantes na solução de problemas de configuração de redes de distribuição. *SBA: Controle e Automação Sociedade Brasileira de Automatica*, scielo, v. 19, p. 302 – 315, 09 2008. ISSN 0103-1759. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-17592008000300006&nrm=iso>.

BROOKS, R. A. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, v. 2, n. 10, 1986.

CARVALHO, A. *Robotica Evolutiva*. <http://www2.icmc.usp.br/simoes/web/aulas/biol/>, 2002.

DEITEL, P. J. *Java TM : como programar*. [S.l.]: São Paulo : Pearson Prentice Hall, 2010.

DICK, P. *Blade runner (do androids dream of electric sheep?)*. *Publisher: Ballantine Books, New York, USA*, 1990.

ENGINE, O. D. *Open Dynamics Engine*. Mar 2013. Disponível em: <<http://www.ode.org/ode.html>>.

FLOREANO, D.; MONDADA, F. Hardware solutions for evolutionary robotics. *Publisher: Springer Verlag, Berlin, Germany*, p. 137–151, 1998.

FLOREANO D., M. F.; NOLFI, S. Co-evolution and ontogenetic change in competing robots. *Publisher: MIT Press, Cambridge, MA, USA*, p. 137–151, 2001.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press, 1975.

JAKOBI, N. Evolutionary robotics and the radical envelope of noise hypothesis. *In Adaptive Behavior*, v. 6, p. 131–174, 1997.

JSON. *JSON*. <http://www.json.org/json-pt.html>, 12 2013. Disponível em: <<http://www.json.org/json-pt.html>>.

JUNG, C. F. *Metodologia para pesquisa & desenvolvimento: aplicada a novas tecnologias, produtos e processos*. [S.l.]: Rio de Janeiro - RJ, 2004.

KRCAH, P. Evolving virtual creatures revisited. In: LIPSON, H. (Ed.). *GECCO*. [S.l.]: ACM, 2007. p. 341.

LEGO. *MindStorms*. <http://www.lego.com/en-us/mindstorms/>: [s.n.], 11 2013. Disponível em: <<http://www.lego.com/en-us/mindstorms/>>.

LYNXMOTION. *Sumo Robots*. Janeiro 2014. Disponível em: <<http://www.lynxmotion.com/c-5-sumo-robots.aspx>>.

MARCONI, M. de A.; LAKATOS, E. M. *Fundamentos de metodologia científica*. [S.l.]: São Paulo - SP, 2003.

PAREDIS, J. Coevolutionary computation. *Artificial Life - Publisher: MIT Press/Bradford Books*, v. 2, p. 355–375, 1996.

ROBOCORE. *RoboCore*. <http://www.robocore.net/>: [s.n.], 11 2013. Disponível em: <<http://www.robocore.net/>>.

SEYDOUX, H. *Parrot*. Janeiro 2014. Disponível em: <<http://www.parrot.com-usa/>>.

SIMS, K. Evolving 3d morphology and behavior by competition. In: *Proceedings of Artificial Life IV*. [S.l.]: MIT Press, 1994. p. 28–39.

SIMS, K. Evolving virtual creatures. In: . [S.l.: s.n.], 1994.

SMITH, T. Blurred vision: Simulation-reality transfer of a visually guided robot. *Publisher: Springer Verlag*, p. 123–136, 1998.

WOLFF, K.; NORDIN, P. Evolutionary learning from first principles of biped walking on a simulated humanoid robot. In: ADES, M.; DESCHAIINE, L. M. (Ed.). *Proceedings of the Business and Industry Symposium of the Advanced Simulation Technologies Conference (ASTC'03)*. Orlando, FL, USA: SCS, 2003. p. 31–36.

ZÄSCHKE, T. *ode4j A Java 3D Physics Engine & Library*. Janeiro 2014. Disponível em: <<http://www.ode4j.org/>>.