

Microsoft Robotics Soccer Challenge:

Movement Optimization of a Quadruped Robot

João Oliveira

Master's Thesis submitted to Faculdade de Engenharia da Universidade do Porto in partial fulfillment of the requirements for the degree of Master's in Electrical and Computers Engineering at, supervised by Professor Doutor Luís Paulo Reis and by Professor Doutor Nuno Lau.

(The President of the Juri, Professor Doutor João Carlos Pascoal de Faria)

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

March 2008

Microsoft Robotics Soccer Challenge:

Movement Optimization of a Quadruped Robot

João Oliveira



Universidade do Porto

Faculdade de Engenharia

FEUP

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

March 2008

Abstract

Robotics is a field expanding tremendously in the last years. Continuous improvements in technology mean that what, only a few years ago, seemed impossible is now being actively studied and developed. Legged robotics, in particular, has been the focus of research due to its possible applications, and despite continuing to present many problems, the progress has been remarkable.

The Robot World Cup Initiative (RoboCup) attempts to use a soccer game as a basis for a wide range of Artificial Intelligence and Robotics research, in several different leagues using distinct platforms, from two-legged robotic simulation to wheeled robots in a real environment. Among those leagues is the Microsoft Robotics Soccer Challenge, based on a simulation of four-legged robots. This thesis mainly focuses on the optimization of the movements of these simulated robots.

The movements of a legged robot are complex, and trying to optimize them analytically may be impossible or clearly not very efficient. So the work concentrated on studying some metaheuristics and finding their application in this particular problem of movement optimization of a quadruped robot.

Metaheuristics were applied to a limited set of robot movements, and the results show that they are effective in improving the performance of the robot movements. However they also made evident some of the weak parts of this approach.

From the results and from the characteristics of the robot and the simulation used, some conclusions were taken as to how to improve the effectiveness and efficiency of this approach, and this dissertation proposes changes to the methods used to overcome some of the limitations that the basic approach presents.

Resumo

A Robótica é um campo que se tem expandido tremendamente nos últimos anos. Melhoramentos contínuos na tecnologia significam que, o há poucos anos apenas pareceria impossível, está agora a ser activamente estudado e desenvolvido. O estudo dos robôs com pernas, em particular, tem sido um dos focos de pesquisa devido às suas possíveis aplicações, e apesar de continuar a apresentar vários problemas, o progresso tem sido notável.

A Robot World Cup Initiative (RoboCup) procura usar o jogo de futebol como base para uma pesquisa alargada em Inteligência Artificial e Robótica, em várias ligas diferentes que usam plataformas distintas, desde simulação de robôs bípedes até robôs com rodas num ambiente real. Entre estas encontra-se o Microsoft Robotics Soccer Challenge, baseado na simulação de robôs quadrúpedes. Esta tese foca-se principalmente na optimização dos movimentos destes robôs simulados.

Os movimentos de um robô com pernas são complexos, e tentar optimizá-los analiticamente pode ser impossível ou claramente não muito eficiente. Portanto o trabalho foi direccionado ao estudo de algumas metaheurísticas e a sua aplicação neste problema particular de optimização de movimentos de um robô quadrúpede.

Metaheurísticas foram aplicadas a um conjunto limitado de movimentos do robô, e os resultados mostram que são eficazes no aumento do desempenho dos movimentos do robô. No entanto esses resultados também tornam evidentes alguns dos pontos fracos desta abordagem.

A partir dos resultados e das características do robô e da simulação usada, foram tiradas algumas conclusões sobre como melhorar a eficácia e a eficiência desta abordagem, e esta dissertação propões alterações aos métodos usados para ultrapassar algumas das limitações que esta abordagem apresenta.

To my family

Acknowledgements

During the making of this work, many people contributed to make it possible for this dissertation to reach an end.

Firstly, I would like to thank my supervisors, Prof. Luís Paulo Reis and Prof. Nuno Lau, for the proposal of the theme for this dissertation.

I want to thank specially thanks Prof. Nuno Lau for reviewing the work being done, and Prof. Luís Paulo Reis, for always being ready to help, for making my objectives clearer and for keeping the work in the right direction.

I also couldn't forget everyone that kept motivating me and helped me get this work done.

Contents

1. Introduction	12
1.1 Objectives	13
1.2 Thesis Structure.....	13
2. Legged Robotics	15
2.1 Robot.....	15
2.2 Robotics	15
2.3 Legged Locomotion	16
2.4 Legged Robotics	16
2.5 Humanoids	17
2.6 Four-legged robots	18
2.6.1 Sony AIBO	18
2.6.2 RobuDOG.....	19
2.7 Chapter Summary	21
3. RoboCup – Robotic Soccer	22
3.1 RoboCup Soccer	22
3.1.1 Simulation League.....	22
3.1.2 Small Size League	23
3.1.3 Middle Size League.....	23
3.1.4 Standard Platform League	24
3.1.5 Humanoid League	25
3.2 RoboCup Rescue.....	26
3.2.1 Rescue Simulation League	26
3.2.2 Virtual Rescue League	26
3.2.3 Real Rescue Robot League.....	27
3.3 RoboCup Junior	27
3.3.1 Soccer Challenge	27
3.3.2 Dance Challenge.....	27

3.3.3	Rescue Challenge	28
3.4	RoboCup@Home	28
3.5	Chapter Summary	29
4.	<i>Microsoft Robotics Studio</i>	31
4.1	Microsoft Visual Programming Language	31
4.2	Decentralized Software Services (DSS)	32
4.3	PhysX Middleware Physics Engine	36
4.4	Chapter Summary	37
5.	MRSC and the Robudog	38
5.1	RobuDOG movements	39
5.2	Chapter Summary	40
6.	Optimization and Metaheuristics	41
6.1	Hill-Climbing	41
6.2	Tabu Search	43
6.3	Simulated Annealing	44
6.4	Genetic Algorithms	45
6.5	Chapter Summary	47
7.	Optimization of Robudog Movements	49
7.1	Simulation	51
7.2	Parameters	51
7.3	Evaluation	52
7.4	Hill-Climbing	53
7.5	Simplified Tabu Search	53
7.6	Simulated Annealing	54
7.7	Summary	54
8.	Results	55
8.1	Hill-Climbing	56
8.2	Simplified Tabu Search	57

8.3 Simulated Annealing.....	59
8.4 Results Analysis.....	60
9. Conclusions	63
References	65

List of Figures

FIGURE 1 GENERAL ELECTRIC WALKING TRUCK (1968) WAS AN EARLY ATTEMPT AT LEGGED LOCOMOTION IN MACHINES	17
FIGURE 2 THE HONDA ASIMO HUMANOID ROBOT	18
FIGURE 3 SONY AIBO ROBOT.....	19
FIGURE 4 ROBUDOG ROBOT	20
FIGURE 5 SMALL SIZE ROBOTS IN THE SOCCER FIELD.....	23
FIGURE 6 MIDDLE SIZE LEAGUE	24
FIGURE 7 ALDEBARAN NAO ROBOT	24
FIGURE 8 HUMANOID LEAGUE.....	25
FIGURE 9 SAMPLE OF THE MICROSOFT VISUAL PROGRAMMING LANGUAGE.....	32
FIGURE 10 DSS COMPONENTS	33
FIGURE 11 MESSAGE SENDING IN DSS	35
FIGURE 12 VISUALIZATION OF THE SOCCER SIMULATOR WITH THE ROBUDOGS	38
FIGURE 13 BASIC OPERATIONS OF GENETIC ALGORITHMS.....	46
FIGURE 14 FLOWCHART OF THE OPTIMIZATION PROCESS.....	50
FIGURE 15 VARIATION OF THE RESULT WITH THE HILL-CLIMBING ALGORITHM	56
FIGURE 16 VARIATION OF THE RESULT WITH THE HILL-CLIMBING ALGORITHM	56
FIGURE 17 EVOLUTION OF THE RESULT WITH THE HILL-CLIMBING ALGORITHM.....	57
FIGURE 18 VARIATION OF THE RESULT WITH THE SIMPLIFIED TABU SEARCH ALGORITHM.....	58
FIGURE 19 VARIATION OF THE RESULT WITH THE SIMPLIFIED TABU SEARCH ALGORITHM.....	58
FIGURE 20 VARIATION OF THE RESULT WITH THE SIMULATED ANNEALING ALGORITHM	59
FIGURE 21 VARIATION OF THE RESULT WITH THE SIMULATED ANNEALING ALGORITHM	59

List of Tables

TABLE 1 SOME CHARACTERISTICS OF THE ROBUDOG	20
TABLE 2 COMPARISON OF THE RESULTS OF EACH OPTIMIZATION TEST.....	61
TABLE 3 COMPARISON BETWEEN OPTIMIZED SOLUTIONS	61

Chapter 1

1. Introduction

Robotics is a field expanding tremendously in the last years. Continuous improvements in technology mean that what only a few years ago seemed impossible is now being actively studied and developed. The field is so vast that many of the developments in other areas can be incorporated in robotics.

The Robot World Cup Initiative (RoboCup) attempts to use a soccer game as a basis for a wide range of Artificial Intelligence and Robotics research, by providing a standard problem where a wide range of technologies can be integrated and examined [Kitano, 1997]. This provides researchers with a wide range of research topics, such as design of autonomous agents, multi-agent collaboration, strategy and real-time reasoning.

Robocup provides a wide range of platforms to study, from two-legged robotic simulation to wheeled robots in a real environment. Among those is the Microsoft Robotics Soccer Challenge, based on a simulation of four-legged robots. This thesis will mainly focus on the optimization of the movements of these simulated robots.

The use of legged-robots to play soccer cooperatively requires that these robots possess a complete, diverse and effective set of basic behaviours. Among these behaviours is for example the capacity to walk, kick the ball, stand, turn, etc. It is through the use of these basic behaviours that the robot can be made capable of more complex actions that ultimately will make it possible for them to participate in a soccer game.

The definition of the behaviours is a complex task. With a wide range of different movements, and each movement being affected by a great amount of parameters, it is not possible to model and study these movements analytically. Thus the only way to analyze the movements and optimize them will be to make use of a simulation. This will make possible to understand how the different variables affect the behaviour of the robot, and also to know how they can be changed to improve its performance.

Since it is not practical or possible to analyze every set of parameters to discover how the best performance is achieved, a different method will have to be used. A great number

of different methods for simulation optimization have been developed to help solve this problem. These methods can be used to find the optimum parameters for the simulation, or at least to improve these parameters.

Simulation optimization is a field that was studied extensively and where a lot of work has been made but that is still very active. Over the years, several optimization algorithms have been developed, such as Hill-Climbing, Tabu Search and Simulated Annealing, Genetic Algorithms, etc.

These algorithms can be used to study the specific needs of the movement optimization of the simulated robot. There are some constraints, such as the great number of parameters to study and the time it takes to perform all the simulations. The focus of this work will be how to use these algorithms to improve the behaviour of the simulated robot.

1.1 Objectives

The main objective of this thesis is to improve the movement of a four-legged robot, using a set of optimization algorithms.

Also the thesis will focus on trying to understand which optimization algorithms can be used in the optimization of the movement of a four-legged robot, so that they can be used in other robots or for the optimization of other movements not studied here.

1.2 Thesis Structure

This thesis will be structured in the following chapters: Chapter 2 will provide an overview on legged robotics, studying examples of both two-legged and four-legged robots. Chapter 3 will present the RoboCup Initiative. Chapter 4 will present the Microsoft Robotics Studio, which was used to run the simulation. Chapter 5 will present the Microsoft Robotic Soccer Challenge and the Robudog, the robot that will be used in the simulation. Chapter 6 will provide an overview of the field of Optimization and Metaheuristics. Chapter 7 will present the work that was done on the optimization of the Robudog. Chapter 8 will present the conclusions and pointers to future research.

Chapter 2

2. Legged Robotics

2.1 Robot

The definition of what is a robot is not an absolute one, and different definitions may be considered for different purposes. According to Webster dictionary, a robot is an automatic device that performs functions normally ascribed to humans or a machine in the form of a human. A definition used by the Robot Institute of America (1979) is: “A robot is a reprogrammable multi-functional manipulator designed to move materials, parts, tools, or specialized devices, through variable programmed motions for the performance of a variety of tasks” [Fu, 1987]. This definition may be a little outdated. Robot technology evolved greatly since this definition was proposed, and now robots can do almost anything. Robot can mean a physical robot, or a virtual software agent. A more modern definition of robot may be “a physical agent that generates intelligent connection between perception and action”.

The word "robot" was popularized in 1921 by Czech writer Karel Capek, but people have envisioned creating robotlike devices for thousands of years. In Greek and Roman mythology, the gods of metalwork built mechanical servants made from gold [Gates, 2007].

2.2 Robotics

Robotics technology is likely to become the highest technology field of the 21st century. Developments in fields such as microprocessors, vision and artificial intelligence will be used to fulfill the needs of a competitive worldwide manufacturing industry. Robotics technology is finding applications in many other fields as well, such as medicine and health care, space exploration, and transportation [U. Texas, 2008].

2.3 Legged Locomotion

Legged locomotion has been used by animals for millions of years. Wheeled locomotion, in contrast, only started being used a few thousand years ago. However, wheeled locomotion dominates when it comes to artificial machines. So we should understand why legged locomotion is not more widely used for man-made vehicles.

There is nothing inherently inferior, from a mechanical point of view, in legged locomotion [Song, 1989]. Animals use legged locomotion at speeds varying approximately from zero up to 100 kilometers per hour. They can walk or run with no difficulty on rough terrain, unlike wheeled vehicles. The range of weights of animals using legged locomotion goes from almost zero up to around the 7000 kilograms of the African Elephant. The United States Army has estimated that approximately half of the Earth's terrain is inaccessible to wheeled or tracked vehicles [Anon, 1967]. Animals and humans with legged locomotion can manage the other half with little difficulty.

One of the reasons that make legs provide better mobility in rough terrain is that they can use isolated footholds that optimize support and traction, whereas a wheel requires a continuous path of support. As a consequence, a legged system can choose among the best footholds in the reachable terrain; a wheel must negotiate the worst terrain. A ladder illustrates this point: rungs provide footholds that enable the ascent of legged systems, but the spaces between the rungs prohibit the ascent of wheeled systems [Raibert, 1986].

Despite all of these advantages, legged locomotion is still not actively used. The main reason is the full coordination that legged locomotion requires. Despite all the research that is being done on this subject, legged locomotion being routinely used may still be some distance away.

2.4 Legged Robotics

Applying legged locomotion to human-built machines has proven a complex task. However the problems are being solved through research, and there are plenty of functional walking robots.

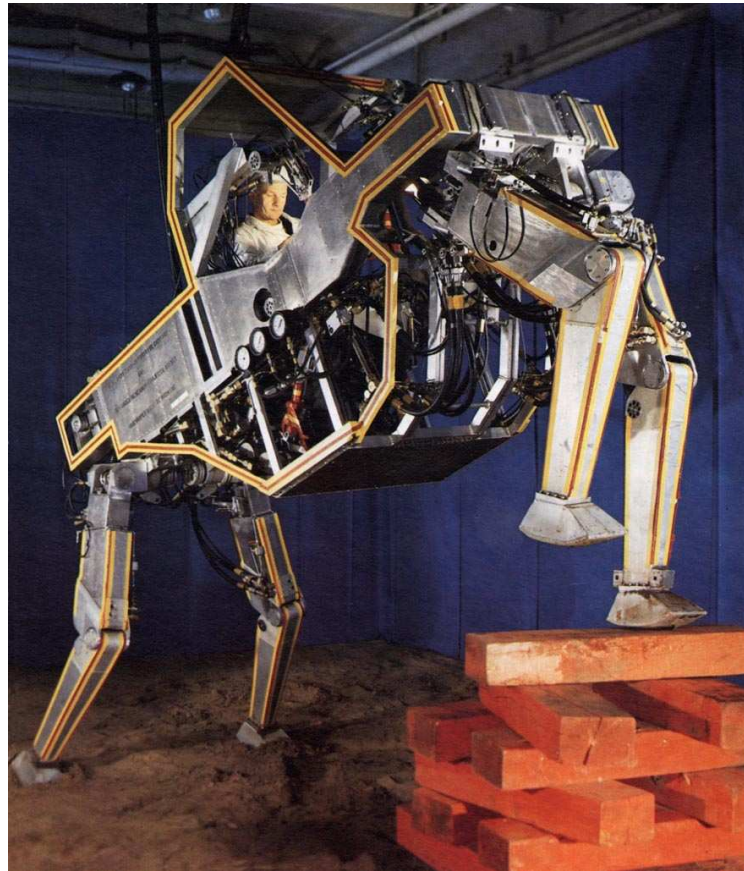


Figure 1 General Electric Walking Truck (1968) was an early attempt at legged locomotion in machines

Nowadays there are robots with varied number of legs, although the problem of control remains. The desing of a legged robot (may it be software and/or hardware) is often inspired by animal locomotion. The main reason for this is the difficulty to cope with the many degrees of freedom of a legged robot even in a relatively simple robot design [Duffert, 2005]. Biped robots present the toughest problem, since they are inherently unstable.

2.5 Humanoids

Humanoid robots are those robots that resemble humans. Leonardo da Vinci's 1495 sketch of a mechanical knight, which could sit up and move its arms and legs, is considered to be the first plan for a humanoid robot [Gates, 2008].

Humanoid robots have an appearance that is based on the human body. Figure 2 shows the Asimo humanoid robot developed by Honda [Gnsin, 2005].



Figure 2 The Honda ASIMO humanoid robot

Building a humanlike robot is an extremely complex engineering task that requires a combination of mechanical engineering, electrical engineering, computer architecture, real-time control, and software engineering. There are some problems when trying to build a robot that looks and behaves like a human that are not easily solved. Biped locomotion is inherently unstable, and maintaining balance is a complex task vision. There are other characteristics that the humans possess and are very difficult to implement on a robot level, like social behaviour, emotions and adaptability.

2.6 Four-legged robots

2.6.1 Sony AIBO

AIBO (Artificial Intelligence roBOt, homonymous with "companion" in Japanese) was one of several types of robotic pets designed and manufactured by Sony; there have been several different models since their introduction in 1999. Able to walk, "see" its environment by using a camera, and recognize spoken commands, they are considered to be autonomous robots, since they are able to learn and mature based on external stimuli from their owner or environment, or from other AIBOs.



Figure 3 Sony AIBO Robot

The Aibo resembles a dog, and it has been used in the Robocup Four-Legged League Soccer Simulation. Sony has discontinued its production. So, in 2008 it will make its last appearance at RoboCup, as the Four-Legged League is being replaced by the Standard platform League that will use humanoid robots.

2.6.2 RobuDOG

RobuDOG is a 4-legged robot with major innovations such as abduction of front legs to kick the ball with accuracy and better control its trajectory, auto-adaptive feet for increased traction on the floor and a powerful programming and 3D simulation environment based on the Robosoft's robuBOX and Microsoft Robotics Studio.



Figure 4 RobuDOG Robot

Table 1 shows some of the characteristics of the robuDOG robot:

Platform	dimensions	33cm x 25cm x 34cm
	weight	3 kg
	17 joints (4 for each front leg, 3 for rear legs and 3 for the head)	
robuBOX	Geode LX800 @ 500 MHz	
	512 Mb + 2 Gb SSD	
	Wifi 802-11 b/g	
	Ethernet RJ-45	
	1 x USB	
Sensors and Equipments	2D gyro	
	2 infrared sensors	
	CMUcam3 : intelligent programmable color camera	
	floor contact sensor on each foot	
	Stereo micro and loudspeaker	
	li-po battery pack (4800 mAh)	

Table 1 Some characteristics of the robuDOG

2.7 Chapter Summary

This chapter described the concepts of robot and robotics and some of the more modern legged robots available in the market. A simulated version of the RobuDog robot, briefly described in this section will be used as the base of the movement optimization approach followed in this thesis.

Chapter 3

3. RoboCup – Robotic Soccer

RoboCup is an international robotics project that started in 1997. It aims to promote intelligent robotics and artificial intelligence research, by providing a standard problem where a wide range of technologies can be integrated and examined. The official goal of the RoboCup project is as follows:

“By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent World Cup.”[RoboCup, 2008]

In order for a robot to play soccer reasonably well, there are a number of technologies that must be integrated, and a number of technical breakthroughs must be accomplished. These include design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor fusion. [Kitano, 1997]

Currently Robocup has four major types of competitions, which are RoboCup Soccer, RoboCup Rescue, RoboCup Junior and Robocup@Home, each with a number of leagues that focus on different aspects of this challenge.

3.1 RoboCup Soccer

3.1.1 Simulation League

There are no actual robots in the RoboCup Soccer Simulation League, all the action occurs within a computer simulation. Each simulated robot player may have its own play strategy and characteristic and the simulated teams actually consist of a collection of programmes or agents.

There are several different divisions within the Simulation League: 2D, 3D, 3D Development, Coach Competition and Mixed Reality [RoboCup, 2008].

In the main league, the 2D simulation competition, two teams of 11 virtual robots play in a virtual field that is 105m x 68m. Each robot is controlled by a different process or computer.

The Simulation league is based on the “soccerserver” simulation system that sends perceptions and receives actions from the agents, while performing the physics simulation of the game.

3.1.2 Small Size League

Small Size robot soccer (or F180 as it is also known), focuses on the problem of intelligent multi-agent cooperation and control in a highly dynamic environment with a hybrid centralized/distributed system.

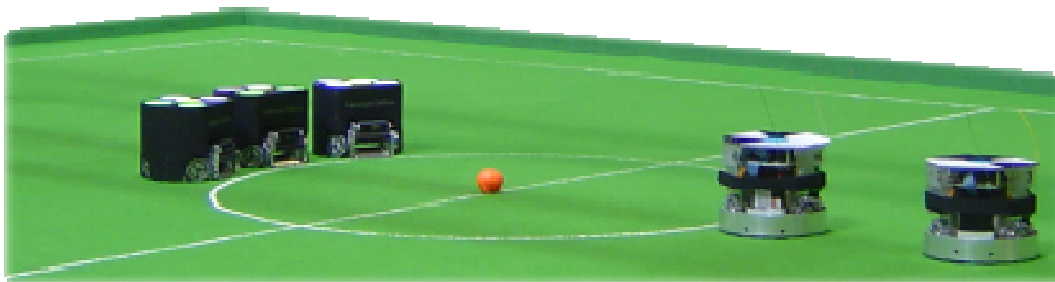


Figure 5 Small Size Robots in the Soccer field

A Small Size robot soccer game takes place between two teams of five robots each. These robots use wheels as a mean for locomotion, and each robot must fit within a 180mm diameter circle and they can be no higher than 15mm (unless they use on-board vision). They play soccer on a green carpeted field, 4.9 meters long by 3.4 meters wide.

There are two types of robots when it comes to vision and positioning. Some use global vision (the most common variety), that makes use of an overhead camera and off-field computer to identify and track the robots as they move around the field. Robots that use local vision have their sensing on the robot itself.

Building a successful team requires the design, implementation and integration of many hardware and software components in a complete team [Small-Size, 2008].

3.1.3 Middle Size League

In the Middle Size League two teams of 4 or 5 robots play a game of soccer on a large field. All sensors must be on-board of the robot. Relevant objects are distinguished by colors. Communication among robots is supported on wireless communications.



Figure 6 Middle Size League

The robots must fit inside a 50 cm x 50 cm square and its height must be between 40 cm and 80 cm. [MSL, 2007]

3.1.4 Standard Platform League

In the Standard Platform League, all teams use identical robots. The league has recently transitioned from the Sony AIBO four-legged robot and now uses the humanoid Aldebaran Nao (figure 7).



Figure 7 Aldebaran Nao Robot

By using identical robots, the teams concentrate on software development only, while still using state-of-the-art robots. The robots operate fully autonomously, not using any external control, neither by humans nor by computers.

3.1.5 Humanoid League

In the Humanoid League, autonomous robots with a human-like body plan and human-like senses play soccer against each other. In addition to soccer games, penalty kick competitions and technical challenges will take place. The robots are divided into two size classes: KidSize (30-60cm height) and TeenSize (80-130cm height). Dynamic walking, running, and kicking the ball while maintaining balance, visual perception of the ball, other players, and the field, self-localization and team play are among the many research issues investigated in the Humanoid League [Humanoid Soccer, 2008].

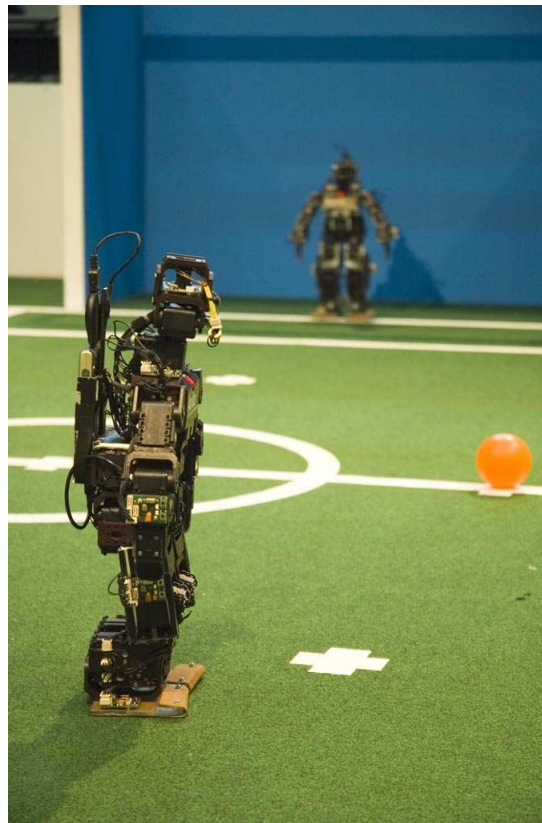


Figure 8 Humanoid League

3.2 RoboCup Rescue

Disaster rescue is one of the most serious social issues, which involves very large numbers of heterogeneous agents in the hostile environment. The intention of the RoboCupRescue project is to promote research and development in this socially significant domain at various levels involving multi-agent team work coordination, physical robotic agents for search and rescue, information infrastructures, personal digital assistants, a standard simulator and decision support systems, evaluation benchmarks for rescue strategies and robotic systems that are all integrated into a comprehensive systems in future [Rescue, 2008].

RoboCup Rescue currently has three leagues: Simulation League, Virtual Rescue and Robotic League.

3.2.1 Rescue Simulation League

RoboCupRescue Simulation's main purpose is to provide emergency decision support by integration of disaster information, prediction, planning, and human interface.

A generic urban disaster simulation environment is constructed on network computers. Heterogeneous intelligent agents such as fire fighters, commanders, victims, volunteers, etc. conduct search and rescue activities in this virtual disaster world. Real-world interfaces such as helicopter image synchronizes the virtuality and the reality by sensing data. Mission-critical human interfaces such as PDA support disaster managers, disaster relief brigades, residents and volunteers to decide their action to minimize the disaster damage.

This problem introduces researchers advanced and interdisciplinary research themes. As AI/robotics research, for example, behavior strategy (e.g. multi-agent planning, realtime/anytime planning, heterogeneity of agents, robust planning, mixed-initiative planning) is a challenging problem. For disaster researchers, RoboCupRescue works as a standard basis in order to develop practical comprehensive simulators adding necessary disaster modules.

The challenge of RoboCup Rescue SimulationLeague is to promote competitive international research cooperation by providing opportunities to researchers to evaluate their research results [Rescue, 2008].

3.2.2 Virtual Rescue League

The goal of the virtual rescue league is, on an urban search and rescue (US&R) virtual environment, to increase awareness of the challenges involved in search and rescue

applications, providing objective evaluation of robotic implementations in representative environments, and promote collaboration between researchers.

A realistic simulator of real robotic rescue environments enable robots to demonstrate their capabilities in mapping, cooperation, mobility, sensory perception, planning, and practical operator interfaces, while searching for simulated victims in unstructured environments. The first virtual rescue competitions provided direct comparison of robotic approaches and objective performance evaluation in a more flexible environment, but at the same time realistic than the real rescue robot league.

3.2.3 Real Rescue Robot League

The goal of annual RoboCupRescue Robot League competitions is to increase awareness of the challenges involved in urban search and rescue (US&R) applications, provide objective evaluation of robotic implementations in representative environments, and promote collaboration between researchers [NIST, 2008]. A real rescue scenario with victims spread around a building is used as the basis of the competition.

3.3 RoboCup Junior

RoboCup Junior introduces the aims and goals of the RoboCup project to the secondary school age level. RoboCup Junior started in 1998 with a demonstration held at the RoboCup international competition in Paris, France. In 2000, the first international RoboCup Junior competition was held in Melbourne, Australia.

3.3.1 Soccer Challenge

In the RoboCup Junior Soccer Challenge, 1-on-1 or 2-on-2 teams of autonomous mobile robots play in a dynamic environment, tracking a special light-emitting ball in an enclosed, landmarked field. For the 1-on-1 League the robot must fit inside an upright 18cm diameter cylinder, while for the 2-on2 League the robot must fit inside an upright 18cm diameter cylinder [RoboCup Junior, 2008].

3.3.2 Dance Challenge

A team creates both a robot and a dance composition. The aim of the competition is to create a two minute dance performance choreographed to music; with particular attention going to construction and programming. Team members can join in to dance alongside the robot. A panel of judges decides the winner based on a number of

different criteria. The dance competition is most popular amongst younger students, mostly of primary school ages. The robots range in size from 10cm tall to 2/3 meters tall with intricate mechanical details.

3.3.3 Rescue Challenge

In the RoboCup Junior Rescue Challenge, small autonomous robots must follow a black line on the floor. Along that line there are several “victims” (that can be identified by being made of either green tape or aluminium foil [RoboCup China, 2008] that should be located by the robot. Besides the “victims”, the route may have ramps, obstacles or gaps on the black line. The final score is based on the total time, as well as the ability to locate victims, manage obstacles and gaps in the black line, and on penalties suffered along the route [Robotica UA, 2008].

3.4 RoboCup@Home

RoboCup@Home focuses on real-world applications and human-machine interaction with autonomous robots. The aim is to foster the development of useful robotic applications that can assist humans in everyday life.

The competition of RoboCup@Home consists of tests which the robots have to solve. The tests will change over the years to become more advanced and function as an overall quality measurement in desired areas. The tests should:

- include human machine interaction;
- be socially relevant;
- be application directed/oriented;
- be scientifically challenging;
- be easy to set up and low in costs;
- be simple and have self-explaining rules;
- be interesting to watch;
- take a small amount of time;

The ultimate scenario where the RoboCup@Home aims to work on is the real world itself. To build up the required technologies gradually a basic home environment is provided as a general scenario. In the first years it will consist of a living room and a kitchen but it is also starting to involve other areas of daily life, such as a garden/park area, a shop, a street or other public places.

3.5 Chapter Summary

This chapter presented the RoboCup international initiative with emphasis on the several leagues that compose the main competition. The work developed used a new league that started in the 2007 competition held in Atlanta, USA: the Microsoft Robotics Soccer Challenges. The simulator environment used in this league: Microsoft Robotics Studio and the league proposed to RoboCup researchers: Microsoft Robotics Soccer Challenge, will be explained in the next chapters.

Capítulo 4

4. Microsoft Robotics Studio

The Microsoft Robotics Studio is a Windows-based environment for robot control and simulation [MSRS, 2008]. It is an integrated environment to develop robotics applications. It may work with real robots and with simulated environments.

Microsoft Robotics Studio includes a visual programming tool, Microsoft Visual Programming Language, for creating and debugging robot applications, web-based and windows-based interfaces, 3D simulation, a lightweight services-oriented runtime (DSS), easy access to a robot's sensors and actuators via a .NET-based concurrent library implementation, and support for a number of languages including C# and Visual Basic .NET, JScript, and IronPython [MSRS, 2008].

Microsoft Robotics Studio ships with a simulator that uses the PhysX physics engine by AGEIA.

4.1 Microsoft Visual Programming Language

Microsoft Visual Programming Language, or MVPL, is a visual programming and dataflow programming language developed by Microsoft for the Microsoft Robotics Studio. It aims at providing a simple and fast way to develop robotic applications and simulations.

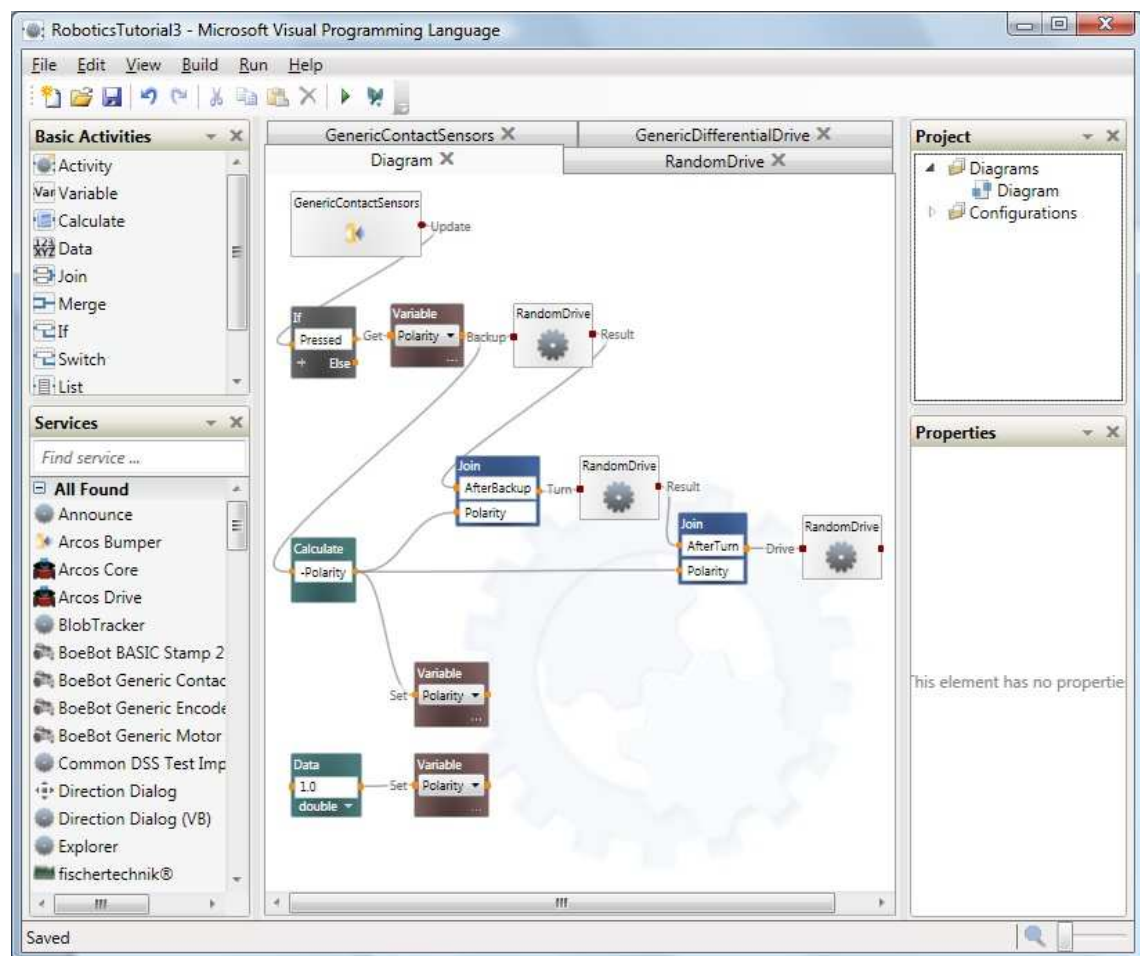


Figure 9 Sample of the Microsoft Visual Programming Language

4.2 Decentralized Software Services (DSS)

Decentralized Software Services (DSS) is a lightweight, service oriented runtime that combines key aspects of Web-based architecture with pieces from Web Services to provide a flexible application model for writing robotics applications. A primary goal of DSS is to promote simplicity, interoperability, and loose coupling [MSRS, 2008].

DSS is built using Concurrency and Coordination Runtime (CCR) and uses HTTP and DSSP as the foundation for interacting with services. CCR is a lightweight runtime that provides a message oriented programming model enabling concurrency and complex coordination of messages. DSSP is a lightweight SOAP-based protocol that extends the current Web architecture by enabling manipulation of structured state and for an event model driven by changes to the structured state. DSSP is used for manipulating and subscribing to services and complimenting HTTP in providing a lightweight, state-driven application model.

Services are the basic building block for writing applications using Microsoft Robotics Studio and are a key component of the DSS application model [DSS, 2008].

Services can be used to represent anything, like for example:

1. Hardware components such as sensors and actuators.
2. Software components such as User Interface, storage, directory services.
3. Aggregations: Sensor fusion, mash-ups.

Services are executed within the context of a DSS Node. A DSS node is a hosting environment that provides support for services to be created and managed until they are deleted or the DSS node is stopped. Services are network enabled and can communicate with each other in a uniform manner regardless of whether they are executed within the same DSS node or across the network [DSS, 2008].

The DSS service model has been designed to facilitate reuse of services by making them easy to use and compose with each other while enforcing a loose coupling between them. Figure 10 represents the DSS components:

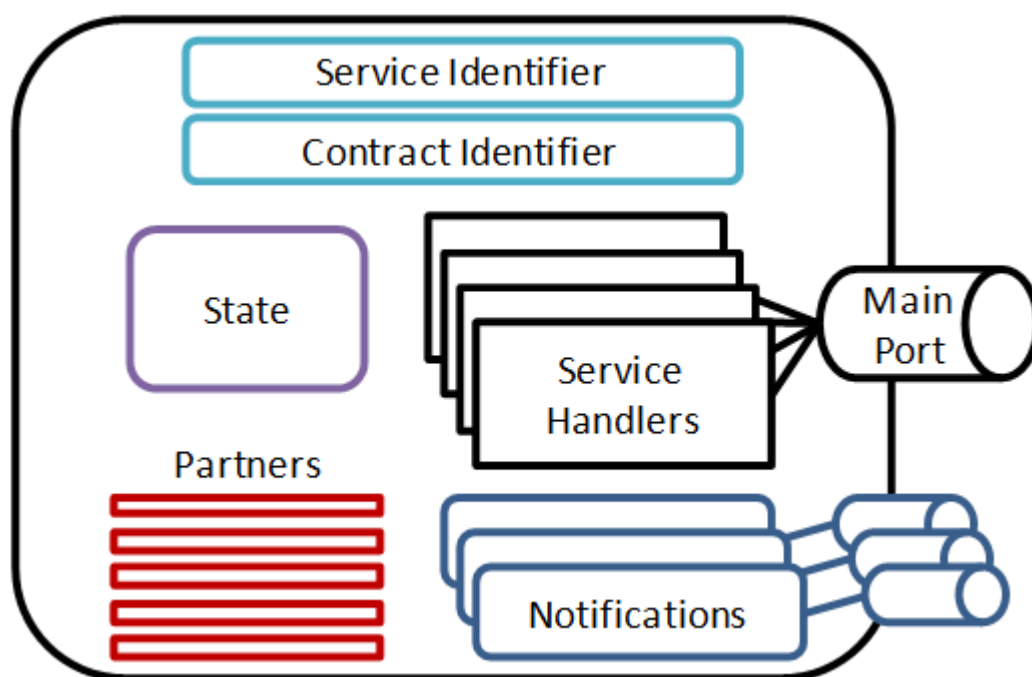


Figure 10 DSS Components

Service Identifier

When a service instance is created within a DSS Node, it is dynamically assigned a URI by the Constructor Service. This service identifier refers to a particular instance of a

service identifier running on a particular DSS node. The service identifier is what enables other services to communicate with that service as well as accessing DSS Services through a Web Browser [DSS, 2008].

Contract Identifier

A contract is a condensed description of a service implementation that describes its behavior so that other services can compose and reuse services with a given contract.

A contract identifier is a URI that uniquely identifies the contract of a service.

Service State

The service state is a representation of a service at a point in time. It can be considered a document that describes the current content of a service. Examples of state documents include [DSS, 2008]:

1. The state of a service representing a motor may consist of rotations per minute, temperature, oil pressure, and fuel consumption.
2. A service representing a keyboard may contain information about which keys have been pressed.

Service Partners

A critical part of the DSS application model is to enable services to compose with each other to provide higher level functions. This means that in order for services to take advantage of each other, they must be able to discover and establish communication with each other in an efficient and predictable way. However, because services are loosely coupled, a service does not know whether other services that it depends on are available or even where these other services are. To address this issue, the DSS application model provides the notion of partner services [DSS, 2008].

Partner services are other services that a service interacts with and possibly depends on in order to function properly. By declaring a set of other services as partners, a service can indicate to the runtime that it wants to be connected with these services as part of the creation process of the service itself. For example, a service can declare that a partner is required in order for the service to run and if that partner can not be found then the service does not need to start at all.

Main Port

The main port is a CCR port where messages from other services arrive. Because service implementations do not link against each other directly, a service can only talk to another service by sending a message to its main port.

Service Handlers

For each of the DSSP operations defined on the main port, service handlers need to be registered to handle incoming messages arriving on that port.

A service can send messages to other services. There are two ways a service can send messages:

1. Unsolicited in the form of a request message sent to another service.
2. Solicited in the form of an event notification sent to a subscriber as a result of a state change within the service generating the event notification.

In both cases, messages are sent through a service forwarder which represents the main port of the remote service. When a message is sent through the service forwarder, it will get forwarded down through the runtime until it reaches a transport that will route the message to the transport of the other service. Here the message will get forwarded back up through the runtime until it reaches the main port of the receiving service. This is represented in figure 11:

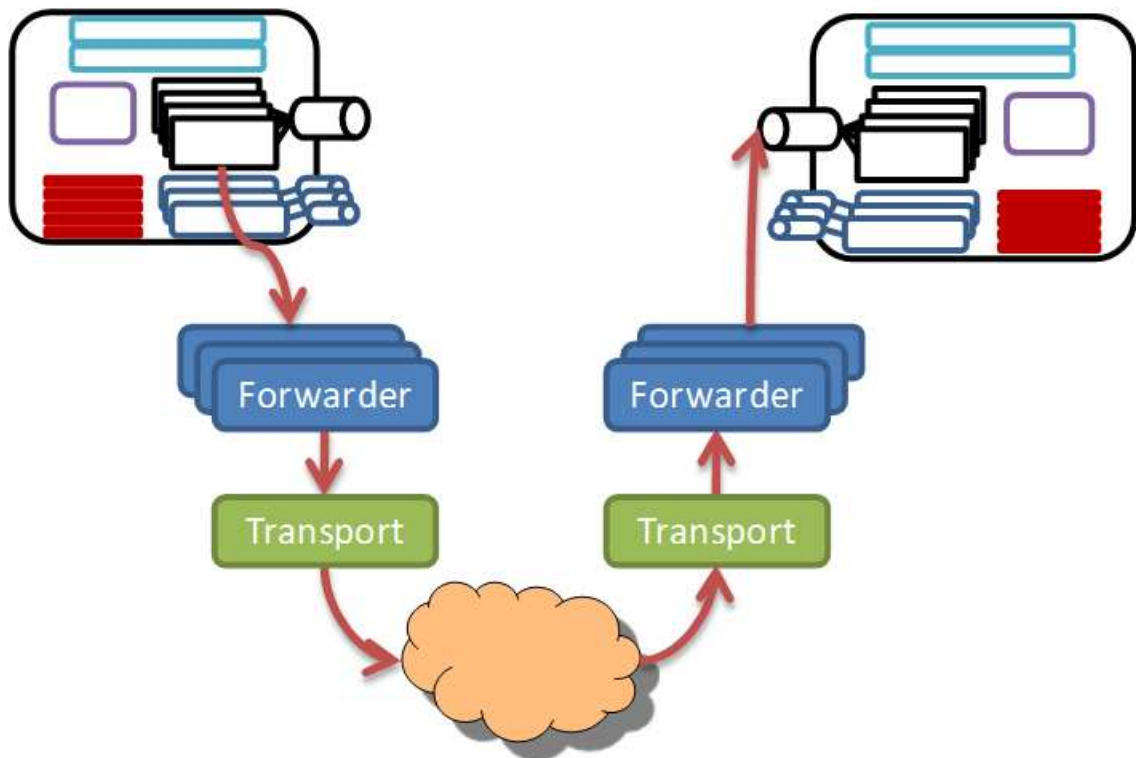


Figure 11 Message Sending in DSS

Event Notifications

A common pattern used in DSS services is that of subscribing to other services. A service generates event notifications as a result of changes to its own state. For each subscription that a service has established with other services, the service will receive event notifications on separate CCR ports. By using different ports for each subscription, it is possible to differentiate event notifications and determine which subscription each event notification is associated with. Furthermore, because event notifications arrive on ports, it is possible to orchestrate event notification messages using the full spectrum of CCR primitives.

4.3 PhysX Middleware Physics Engine

PhysX is a middleware physics engine developed by AGEIA, and it is used as the physics engine for the simulations in Microsoft Robotics Studio [MSRS, 2008]. It was developed mainly with the aim of providing a physics framework for computer games, but its characteristics make it useful as a physics engine in simulators.

It has several important characteristics: complex rigid body object physics system, advanced character control, volumetric fluid creation and simulation, cloth and clothing authoring and playback, soft Bodies and volumetric Force Field Simulation [Ageia, 2008].

The rigid body dynamics component enables to simulate objects with a high degree of realism. It makes use of physics concepts such as reference frames, position, velocity, acceleration, momentum, forces, rotational motion, energy, friction, impulse, collisions, constraints in order to be a construction kit with which it's possible to build many types of mechanical devices.

- Collision primitives (sphere, box, capsule, plane, heightfield, convex shape, triangular mesh);
- Various joints types (spherical, revolute, prismatic, cylinders, fixed, distance, pulley, 6DOF);
- Advanced ragdoll creation and editing;
- Materials and friction model;
- Continuous Collision Detection support;
- Collision groups + Collision Filtering;

The PhysX software may be used in conjunction with a PhyX physics processing unit (PPU). A PPU is a dedicated microprocessor designed to handle the calculations of physics from a physics engine. This way the CPU is relieved from an important amount of processing, making the simulation faster.

4.4 Chapter Summary

This chapter presented Microsoft Robotics Studio and some of its characteristics. Microsoft Robotics Studio was used as the basis for the implementation of the robotic movement optimization framework developed in the context of this thesis.

Chapter 5

5. MRSC and the Robudog

Microsoft started the Microsoft Robotic Soccer Challenge in 2007 with the aim of promoting the use of Microsoft Robotics Studio with the help of the promotion potential of the RoboCup international initiative, since RoboCup 2007 was held in Atlanta, USA.

The competition used a simple simulation of a soccer match using 2 vs 2 games between RoboDOG simulated robots. The robuDOG robot was developed mainly with its use in RoboCup in mind.

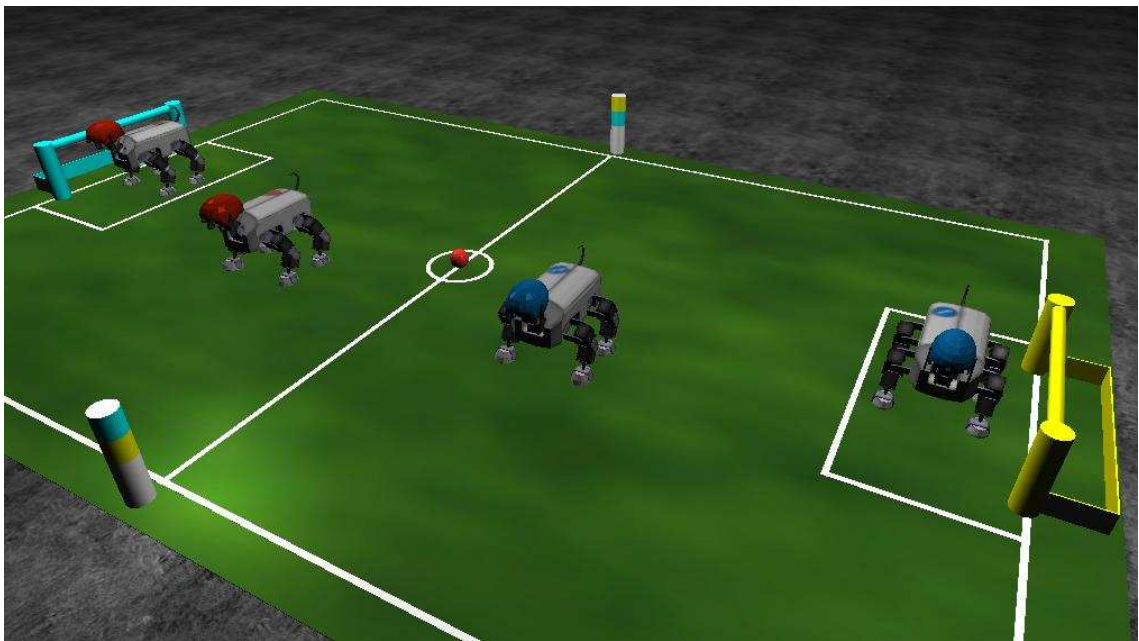


Figure 12 Visualization of the Soccer simulator with the robuDOGS

Although being a very simplified soccer environment, the complexity of the robot used, the quadruped RobuDOG turned this league into a huge challenge.

5.1 RobuDOG movements

A robuDOG movement is defined as a combination of leg movements for each of the robot's four legs. A leg movement is defined by the movement of its joints.

There are 3 joints in each leg, the joint order in the definition file is from the top of the leg to the bottom of the leg [robuBOX, 2008]:

- *Shoulder*: moving forward and backward;
- *Shoulder*: moving left and right;
- *Knee*: moving forward and backward;

The front legs have an additional joint in the ankle.

The gaits and movements of the robuDOG are defined in two xml files: dogmvt0.xml and legmvt0.xml. dogmvt0.xml contains the combination of leg movements, while legmvt0.xml contains the combination of the joints' movement for each leg movement. Here is an example of the definition of a leg movement in the legmvt0.xml:

```
<RobuLegMouvement>
  <name>LegStand</name>
  <Mouvements>
    <SerialManipulatorControlRequest>
      <Controls
xmlns="http://www.robosoft.fr/2006/11/serialmanipulator.html">
        <JointControl>
          <Mode>Position</Mode>
          <Target>0.55</Target>
          <Reached>>false</Reached>
        </JointControl>
        <JointControl>
          <Mode>Position</Mode>
          <Target>0</Target>
          <Reached>>false</Reached>
        </JointControl>
        <JointControl>
          <Mode>Position</Mode>
          <Target>0.96</Target>
          <Reached>>false</Reached>
        </JointControl>
      </Controls>
    </SerialManipulatorControlRequest>
  </Mouvements>
</RobuLegMouvement>
```

Each action on a joint defines a position target to reach for the joint (in radian). The joint has 3 parameters : `<Mode>` (which must always have the value: Position), `<Target>` (which contains the angular target to reach in radians), and `<Reached>` (which must always have the value: false) [Robudog Simulator, 2008].

The movement of the robuDOG is defined in the dogmvt0.xml file, based on the definition of leg movements in the legmvt0.xml file. Here is an example of the definition of the “Stand” movement:

```
<DogMouvement>
  <Name>Stand</Name>
  <motion>
    <SingleMouvement>
      <mouv>
        <string>ArmStand</string>
        <string>ArmStand</string>
        <string>LegStand</string>
        <string>LegStand</string>
      </mouv>
    </SingleMouvement>
  </motion>
</DogMouvement>
```

Each leg movement is defined by a `<string>` which refers to the name of the leg movement as defined in the LegMvt0.xml file. The order of the leg in the `<mouv>` list is:

- Front Right;
- Front Left;
- Rear Right;
- Rear Left;

5.2 Chapter Summary

This chapter presented the Microsoft Robotics Soccer Challenge and the model of the simulated RobuDOG robot used as the basis for this challenge. Due to its complexity, it is very difficult to create, analitically, optimized movements for the RobuDOG robot and thus the use of optimization techniques may be the answer to create these movements. The next chapters will discuss optimization methodologies and its application in the creation of appropriate optimized RobuDOG movements.

Chapter 6

6. Optimization and Metaheuristics

Learning by induction is a search process, where the search for the correct rule, relationship, or statement is steered by the examples that are encountered. Numerical learning systems can be viewed in the same light. An initial model is set up, and its parameters are progressively refined in the light of experience. The goal is invariably to determine the maximum or minimum value of some function of one or more variables. This is the process of optimization. [Hopgood, 2001]

Metaheuristics, in their original definition, are solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space [Glover, 2003].

Metaheuristics have developed dramatically since their inception in the early 1980's. They have had widespread success in attacking a variety of practical and difficult optimization problems. These families of approaches include but are not limited to greedy random adaptive search procedure, genetic algorithms, problem-space search, neural networks, simulated annealing, tabu search, threshold algorithms, and their hybrids. They incorporate concepts based on biological evolution, intelligent problem solving, mathematical and physical sciences, nervous systems and statistical mechanics [Osman, 1996]. This chapter presents a study of some of these metaheuristics.

6.1 Hill-Climbing

Hill Climbing (HC), in its basic form, is the simpler of the optimization methods for discrete variable optimization problems. From a starting point in the search space, a step is taken in a random direction. If the solution of the new point is better than the previous position, it is accepted as the new solution. The process is repeated until the algorithm

no longer finds a better solution. At this point, this solution is assumed to be the optimum.

Hill-Climbing Algorithm:

1. *Generate an initial solution.*
2. *Loop until a stop criterion is met or there no untested neighbour solutions:*
 - (a) *Test a neighbour solution to the current one that hasn't been tested yet*
 - (b) *If the new solution is better than the current solution make it the current solution*
3. *Return the current solution*

Some of the well-known weaknesses of the Hill-Climbing algorithm are [Russell, 2002]:

Local Maximum - A local maximum is a peak that is higher than its neighbour solutions but is not the highest value of the function. The Hill-Climbing algorithm will stop at these points.

Plateau - A plateau is an area where the objective function is essentially at. Hill-Climbing will search erratically in these kinds of areas.

Ridges - A ridge is an area with a point, where even without it being a local maximum, all available moves will make the solution worse. Ridges depend on the method chosen to calculate neighbour solutions.

One way of guarding against the trap of detecting a local optimum is to repeat the process many times with different starting points [Hopgood, 2001]. This method works well when there are only a few local maximums but in more realistic problems it will take an exponential amount of time to find the best solution to the problem [Russell, 2002].

Steepest gradient ascent is a refinement of Hill-Climbing that can speed the convergence toward a maximum solution [Hopgood, 2001]. From a given starting point, a trial solution, the direction of steepest descent is determined. A point lying a small distance along this direction is then taken as the new trial solution. The process is repeated until it is no longer possible to descend, at which point it is assumed that the optimum has been reached.

6.2 Tabu Search

Tabu Search (TS) is another optimization method created to solve the local maximum problems revealed by the Hill Climbing algorithm. The basic principle of TS is to pursue Local Search whenever it encounters a local optimum by allowing non-improving moves; cycling back to previously visited solutions is prevented by the use of memories, called tabu lists, that record the recent history of the search. [Glover, 2003]

The main idea behind the TS algorithm is to use the search history in order to impose restrictions, and additions, on the neighbourhood of the solution currently being analyzed.

Tabus are one of the distinctive elements of Tabu Search. Tabus are used to prevent cycling when moving away from local optima through non-improving moves. The key realization here is that when this situation occurs, something needs to be done to prevent the search from tracing back its steps to where it came from. This is achieved by declaring tabu (disallowing) moves that reverse the effect of recent moves. In this way, we can go away from local optima in search of more attractive regions.

There are several ways to keep tabu lists, but the most commonly used tabus involve recording the last few transformations performed on the current solution and prohibiting reverse transformations.

An algorithm of Tabu Search is presented below: [Glover, 2003]

Notation

- S , the current solution,
- S^* , the best-known solution,
- f^* , value of the objective function of S^*
- $N(S)$, the neighborhood of S ,
- $\tilde{N}(S)$, the “admissible” subset of $N(S)$ (i. e., non-tabu or allowed by aspiration).

Initialization

Choose (construct) an initial solution S_0 .

Set $S = S_0, f^* = f(S_0), S^* = S_0, T = \{\}$.

Search

While termination criterion not satisfied do

Select S in $\arg \max [f(S')]; S' \in \tilde{N}(S)$

if $f(S) > f^*$, then set $f^* = f(S), S^* = S$

record tabu for the current move in T (delete oldest entry if necessary);

endwhile.

6.3 Simulated Annealing

Simulated annealing [Kirkpatrick, 1983] owes its name to its similarity to the problem of atoms rearranging themselves in a cooling metal. In the cooling metal, atoms move to form a near-perfect crystal lattice, even though they may have to overcome a localized energy barrier called the activation energy, E_a , in order to do so. The atomic rearrangements within the crystal are probabilistic. The probability P of an atom jumping into a neighboring site is given by:

$$P = e^{\frac{-E_a}{kT}}$$

where k is Boltzmann's constant and T is temperature. At high temperatures, the probability approaches 1, while at $T = 0$ the probability is 0.

The Simulated Annealing algorithm tries to solve the Local Maximum problem described in 6.1. This is accomplished by allowing the search to sometimes accept worst solutions with a certain probability that diminishes with the temperature of the system [Restivo, 2006]. In this way, if the new solution is better than the previous, it is accepted, just as with the Hill-Climbing algorithm. If the new solution would decrease the objective function (Δf), the probability of accepting that solution at a certain temperature, would be given by the following formula [Kirkpatrick, 1983]:

$$p(\Delta f, t) = e^{\frac{\Delta f}{T}}, \Delta f \leq 0$$

It is clear from the formula that downhill transitions are possible. Here, Δf is the variation in the objective function that would result from the step and is analogous to the activation energy in the atomic system.

The temperature T determines the stability of a trial solution. If T is high, new solutions will be accepted easily, even if they reduce the value of the objective function. If T is high, the probability of a new solution being accepted is low, so the algorithm will behave similarly to the Hill-Climbing algorithm, moving to a local or global maximum. It's necessary to decide what the initial temperature is, and how it should be reduced. The slower the temperature is decreased, the greater the chance of finding an optimal solution. In fact, running the simulation an infinite number of times is needed to be sure the optimal solution to the problem has been found [Aarts, 1989].

Most times a reasonably good cooling schedule can be achieved by using an initial temperature (T_0), a constant temperature decrement (α) and a fixed number of iterations at each temperature. These kind of cooling schedules are called fixed schedules. The problem with these schedules is that it is often impractical to calculate the ideal values for T_0 and α . Another approach is to use a scheduling that can automatically adapt to the problem at hand. These are called self-adaptive schedules [Restivo, 2006].

A commonly used simple cooling schedule is [Hopgood, 2001]:

$$T_{t+1} = \alpha T_t$$

where T_t is the temperature at step number t and α is a constant close to, but below, 1. While T is high, the optimization routine is free to accept many varied solutions, but as it drops, this freedom diminishes.

6.4 Genetic Algorithms

Genetic Algorithms are based on the evolutionary aspects of biology, using concepts such as chromosomes, alleles and mutations that are brought from the biological sciences. The basic principle is that from an initial population of several solutions, the solutions will mix and mutate to form new solutions, with each solution being evaluated for its fitness.

There is no clear definition of “genetic algorithm” accepted by all that differentiates Genetic Algorithms from other evolutionary computation methods. It can be said however, that most methods called “Genetic Algorithms” have at least the following elements in common [Mitchell, 1996]: populations of chromosomes, selection according to fitness, crossover to produce new offspring and random mutation of new offspring.

The chromosomes in a Genetic Algorithm population are normally bit strings. So each locus on the chromosome has two possible alleles: 0 and 1.

Figure 13 represents these basic concepts of selection, mutation and recombination [Restivo, 2006]:

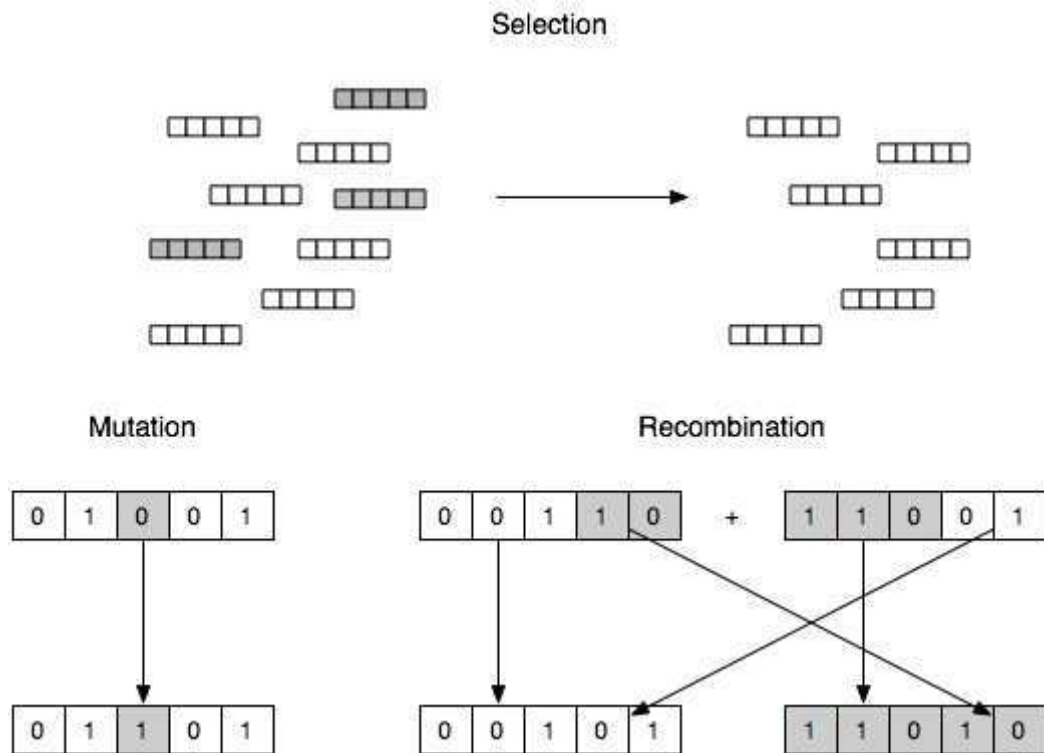


Figure 13 Basic operations of Genetic Algorithms

Each chromosome can be considered a point in the search space of candidate solutions. The Genetic Algorithm processes populations of chromosomes, replacing one population with another.

For each new population, each string in the population has its objective function evaluated. Based on this evaluation, strings that have higher fitness (i. e., represent better solutions) can be identified, and these are given more opportunity to breed [Glover].

The mutation operator is used to introduce innovation in the current population allowing the algorithms to explore areas of the search space that are not being explored at the moment. The selection operators are those that make the method reach better results with each new generation, selecting the solutions with an higher fitness value over the ones with a lower one. The recombination operator allows some information exchange between current solutions by introducing a new solution into the population from the merge of two previous selected solutions [Restivo, 2006].

Next is presented a basic algorithm of a generic Genetic Algorithm [Pierreval, 2000]:

1. Start with the generation counter equal to zero.
2. Initialize a population of individuals (either randomly or by means of a heuristic function).

3. *Evaluate fitness of all initial individuals in population.*
4. *Increase the generation counter.*
5. *Select a subset of the population for children reproduction (selection).*
6. *Recombine selected parents (recombination).*
7. *Perturb the mated population stochastically (mutation).*
8. *Evaluate the mated population's fitness (evaluation).*
9. *Test for termination criterion (number of generations, fitness, etc.) and stop or go to step 4.*

The main problem with GA is how to code each solution into meaningful bit strings. Its advantages are that mutation and recombination operators are easily implemented as bit flips (mutations) and string cuts followed by concatenations (recombination) [Restivo, 2006].

6.5 Chapter Summary

This chapter presented a brief overview of some of the most known meta-heuristics and their algorithms and main characteristics. The next chapter will show how these meta-heuristics may be used to optimize some of the RobuDOG movements.

Chapter 7

7. Optimization of Robudog Movements

A defined set of robuDOG movements already exists. However we want to understand if they can be improved, and understand how they can be improved so that we end up with a better set of movements and understanding on how to make new movements more efficient.

To do that it is necessary to understand how the meta-heuristics that we studied can be applied in solving this problem. With them we will attempt to perform the optimization of some robudog movements, understand how these algorithms perform in this specific scenario and what the problems with each of the methods used are.

Furthermore, we'll study the results to understand how the optimization methods can be used to further improve the performance of the robot, and to understand what changes should be adopted to make this optimization faster and more reliable.

From the optimization algorithms studied in the last chapter, Hill-Climbing and Simulated Annealing are going to be used. Tabu Search requires that several or all of the elements in the neighbourhood of the present solution are tested, which makes that algorithm not very useful in this particular case, given the time it takes to test each solution. Instead an adaptaion of Hill-Climbing to include some extra characteristics was used. It will be called in this dissertation Simplified Tabu Search and it will be presented in this chapter.

A program was made which implements each optimization algorithm. The simulation environment was changed so that it becomes possible to move the robot and set its position and orientation. The simulation environment was also changed so that it's possible to query for the current position and orientation of the robot, without need for an auto-localization algorithm. A log file is created with all the relevant information in each iteration, so that the results can be better studied and understood.

Several different functions had to be created, specifically to read and parse the LegMvt0.xml, as well as to create new solutions by changing this LegMvt0.xml file. An evaluation function was applied, and ways to read the values needed to calculate it was developed.

Being a program that is going to run for a long time, and performing the movement simulation hundreds of times at once, it is important that it is robust so it can be run for the amount of time needed to obtain strong results. For this, the main program monitors the state of the simulation program, in order to recover from errors that may happen and recover from them without losing more than a single iteration.

Figure 14 represents the basic flowchart representing the optimization process:

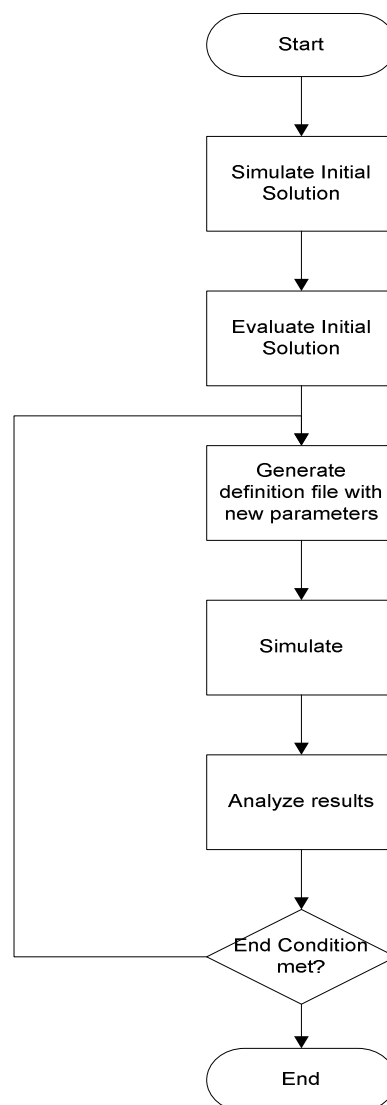


Figure 14 Flowchart of the optimization process

7.1 Simulation

The first step is to set up the simulator so that it produces useful results to be studied. It has to be capable of making the robot perform the requested movements, and it has to be able to find the position and orientation of the robot and pass this information to the optimization program.

The simulator gives no clear output for a given set of parameters, because it is affected by a significant amount of noise. This way, it can be difficult to evaluate and compare each solution, because the results may not accurately portrait the response of the robudog. For example, a solution may be accepted as a better solution because the output was better, but this could be the result of the noise, and it does not accurately reflect the effect of a new solution being tested. This problem was reduced by evaluating several times each solution, thus reducing the effect of noise in the results.

The biggest problem in setting up the simulation was the time that each iteration took to complete. The simulator has to restart to make any changes in the defined movements, with the result of each iteration taking more than 30 seconds. This will make it harder to make the required amount of testing, especially for the more complex movements. This means that a reasonable task for now would be to test only one of the movements, and leave the foundations for future work with other movements.

The long time to perform each iteration also means that the complete Tabu Search algorithm and the Genetic Algorithm will not be used here. The testing of several solutions in each iteration would make the Tabu Search and the Genetic algorithms very slow in this particular case. A faster version of Hill-Climbing with some Tabu Search Characteristics was used instead of Tabu Search.

7.2 Parameters

The main robot movements to optimize were the forward movement, backward movement, and the Turn Right and Turn Left movements. As it was already seen in chapter 5, the movements are defined in two xml files, Legmvt0.xml and Dogmvt0.xml. For each specific robot movement (“Forward”, “Turn Left”, etc.), Dogmvt0.xml defines the sequence of leg movements that should be performed. For each leg movement, Legmvt0.xml defines the sequence of joint movements that should be performed.

The values in the Legmvt0.xml will be the parameters to change in the optimization process. Each of these values in the Legmvt0.xml file represents the target angle of the joint in radians for a specific leg movement. Therefore, by changing these angles the movement will be changed and can then be simulated for evaluation

The different movements have different amounts of parameters that can be changed. It is desired for now, and because of the computation constraints described in 7.1, that the number of parameters to change is not too big, so that the optimization can be done in a reasonable amount of time. Forward movement has 238 parameters that can be changed, with each of these parameters taking a wide range of values. With the time it takes for each iteration of the algorithms, this makes it unpractical to reasonably and consistently use these meta-heuristic algorithms on the forward movement. The work concentrated on the Rotation movements (“Turn Left” and “Turn Right”).

A solution in this case can be the group of defined parameters that are present in the LegMvt0.xml file for a specific movement, as was presented earlier. With the solution space having a potentially infinite number of solutions, being the target angle a continuous variable, here must be a way to define which of the combinations to consider in the search, and how to define the smallest possible change the solution can have. For a given solution, its neighbours were considered to be all the solutions that could be found by changing a single parameter by a minimum value. This value has to be chosen, and a value of 0,05 radians was considered for most cases. This value seems reasonable, it represents a tradeoff between computation time and precision. Besides, considering smaller values would make it more difficult to evaluate the differences between neighbour solutions, especially because of the noise that is present in the simulation.

7.3 Evaluation

For the Turn Right and the Turn Left movements, what we want to improve is the rotation of the robot. It also shouldn't move much from its initial position, as we want to keep it in the same place. The evaluation function should reflect this, so it penalizes translation of the robot, and prizes rotation. Being θ the orientation of the robot, and p its position, the following objective function to maximize was used:

$$f(S) = \Delta\theta - 100\Delta p$$

The position was weighed so that it had a meaningful influence in the result, since the units are different. These constants were chosen after observing the results for different evaluation functions. With this evaluation function, the translation movement has enough importance, without taking too much importance in relation to the rotation of the robot, which is going to be maximized.

1. Write new solution to LegMvt0.xml file
2. Simulate

(a) Test new solution several times

(b) Return Position and orientation for each test

3. Calculate Evaluation Function for new solution

To make the evaluation, After each change in the movement parameters, the simulation is run with the new movement file. After the robot tests a series of “Trun Left” and “Turn Right” movements, the simulator returns the initial and final position and orientation of the robudog for each test. With these values, the main program can find the average $\Delta\theta$ and the average Δp , from which it can calculate the evaluation function for the tested movement.

7.4 Hill-Climbing

The Hill-Climbing Algorithm was applied to this problem. Having the shortcomings that were discussed in (7.3), it has proved useful in this problem, since we’re not starting from a random solution, but from a solution that is already based in how a quadruped robot should be moving. This can mean that the initial solution is already in the neighbourhood of an optimal solution, possibly reducing the risk of finding a local optimum that is still far from the global optimum.

7.5 Simplified Tabu Search

A variation of the Hill Climbing Algorithm was used to speed up the improvement in the objective function. Instead of using random variation of the movement parameters, this algorithm will favour changes that we have reason to believe have a higher probability of improving the objective function. This is achieved by maintaining a table with the information if the objective function improved or not after the last change in each parameter. The algorithm is presented next:

1. Start from an initial solution.

2. Loop until a stop criterion is:

(a) Randomly choose one of the following:

- Test a random neighbour solution to the current one that hasn’t been tested yet*
- Test a neighbour solution achived by changing one of the parameters that made the solution improve*

(b) If the new solution is better than the current solution make it the current solution

3. Return the current solution

This algorithm doesn't solve the biggest drawback of the Hill-Climbing Algorithm, which is the danger of finding local optima instead of the global one, but it attempts at improving its efficiency by reaching the same final solution in fewer iterations. This is important especially in problems that take a lot of computation time, which is the case here.

7.6 Simulated Annealing

The previous algorithms don't address the problem raised by the possibility of all the tested solutions being in a region that doesn't contain the global maximum. The Simulated Annealing algorithm might help finding solutions not considered by the previous algorithms. Applying this algorithm to the specific problem may take a lot of time to compute.

The Simulated Annealing algorithm requires us to choose an initial temperature T and the way temperature is reduced. It was decided to use the most common way to simulate the variation of temperature [Hopgood, 2001]:

$$T_{t+1} = \alpha T_t$$

The values of T and α were chosen to make the probability of accepting a non-improving solution close to 1 in the first iterations and close to 0 in the last iterations. In the last iterations the algorithm should behave like the Hill-Climbing algorithm, finding the local best solution, which should also be the global optimum.

7.7 Summary

For this initial study on this problem, we chose the movement of rotation of the robot. This was done due to the relatively low set of different parameters that can be changed, while at the same time having a quite straightforward and objective way to make the evaluation. This is important because this way the limitations and weaknesses of this approach will become clearer, enabling us to better find ways to improve the methods.

Chapter 8

8. Results

Based on the methods developed in the previous chapter, some tests were conducted to understand the way the optimization develops with the different approaches used.

In each of these experiments, the optimization algorithm was used varying some parameters, such as the number of iterations, the change step (the variation between a given parameter and its neighbour), and in the case of the Simulated Annealing algorithm, the temperature and the α coefficient. There are no strict rules or guidelines for defining these values in this specific case, so these values were adjusted based on previous experiences.

The parameters that form the search space are the target angle of each of the joints of the robuDOG robot. Although in theory these variables are continual, they are going to be considered discrete variables for these experiences, with a fixed change interval defined for each experience.

From an initial solution that was already performing the movement to optimize, different tests were done, using different algorithms.

It was also tested the possibility of creating the movement from zero, starting from an initial solution where the robot just stands still, and following the evolution of the movement towards the desired movement. This was only done with the Simulated Annealing algorithm, because this is the only one of the algorithms tested that is capable of moving between regions even when not improving the solution, thus having a better chance of finding a good region in the search space.

Next in this chapter the results are presented and possible explanations for them are discussed.

8.1 Hill-Climbing

The Hill-Climbing Algorithm provided a marked improvement over the initial solution. The results are displayed next for two tests using a change step of 0.1, one with 300 iterations (Figure 15) and the other with 800 iterations (Figure 16):

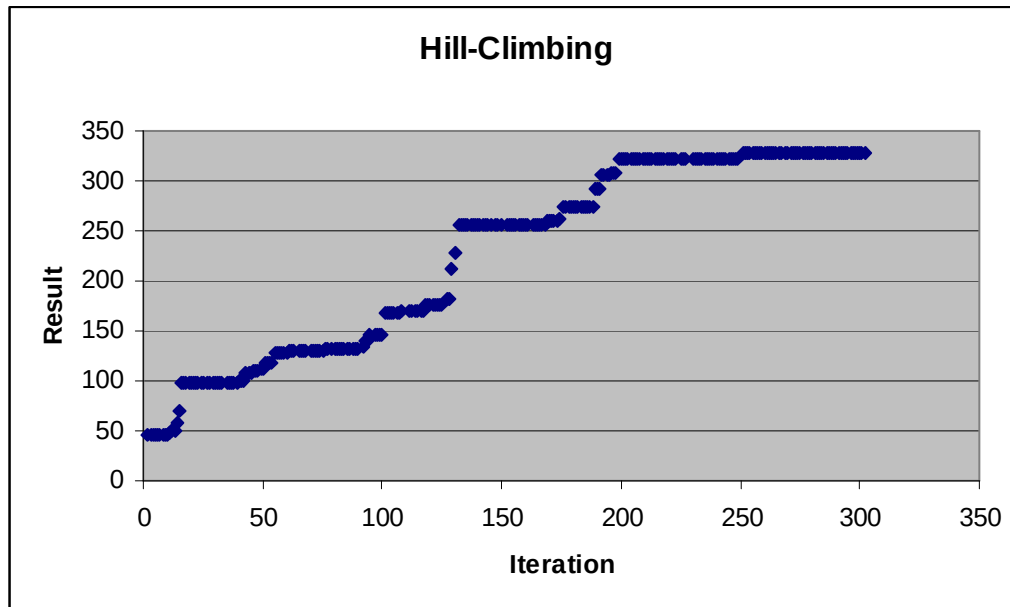


Figure 15 Variation of the result with the Hill-Climbing algorithm



Figure 16 Variation of the result with the Hill-Climbing algorithm

A test with a parameter change of 0.05 was done and the results are displayed in Figure 17:

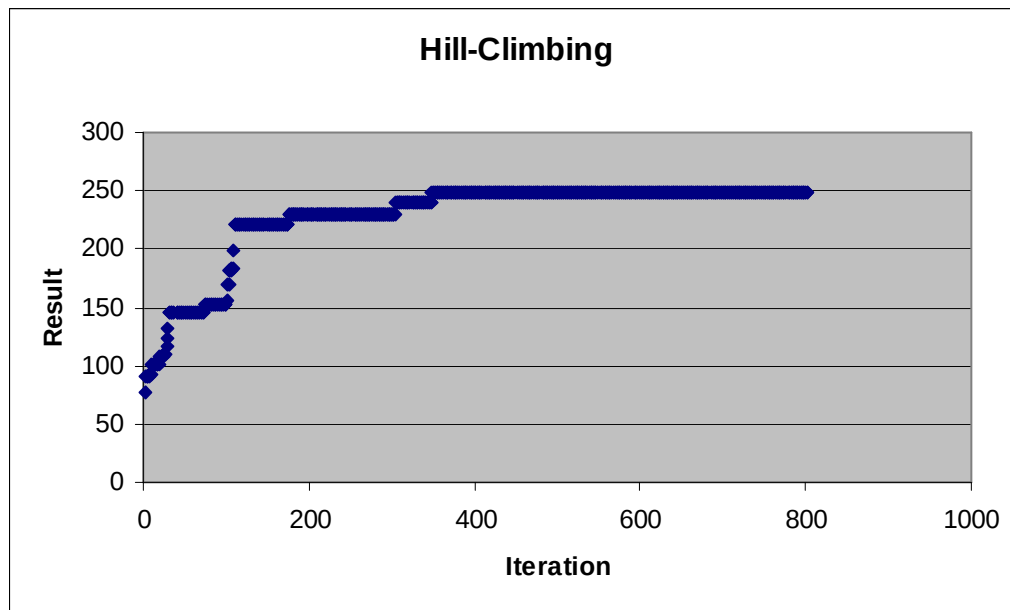


Figure 17 Evolution of the result with the Hill-Climbing algorithm

The performance of the robot was greatly increased, which means that the Hill-Climbing Algorithm was useful in this problem, despite all of its shortcomings. However when the change is only 0.05, the evolution is much slower.

8.2 Simplified Tabu Search

The results provided by the Simplified Tabu Search algorithm using a change step for the solution parameters of 0.05 are shown in the following two graphics. The solution reached by the two experiences using the Simplified Tabu Search algorithm showed an improvement almost ten-fold relatively to the initial solution (which was not a random solution, but the solution provided in the simulator).

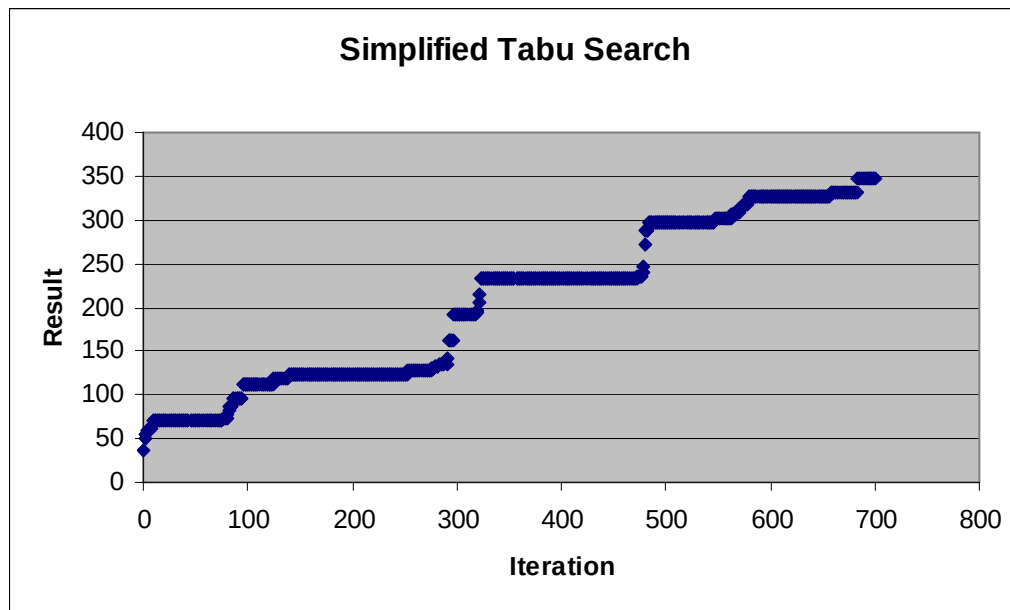


Figure 18 Variation of the result with the Simplified Tabu Search algorithm

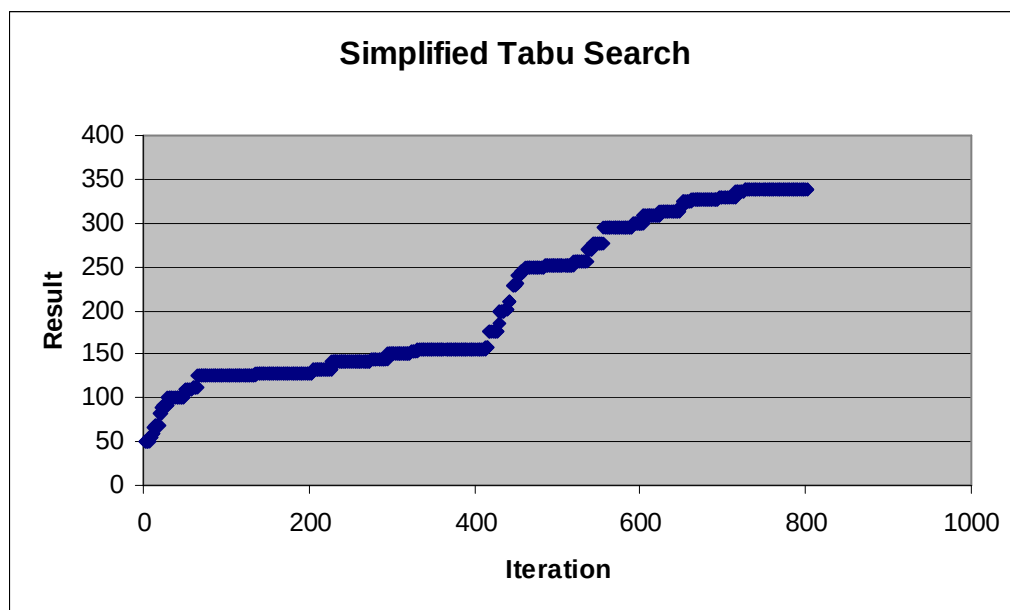


Figure 19 Variation of the result with the Simplified Tabu Search algorithm

It is clear that the improvements are not isolated, but usually improvements come in groups, which indicate that when the algorithm finds a parameter that improves the objective function, it is likely that another change in that parameter in the same direction will also improve the objective function. This has the effect of substantially increasing the speed of improvement of the solution, when compared to the Hill-Climbing using the same minimum change of parameter (0.05).

8.3 Simulated Annealing

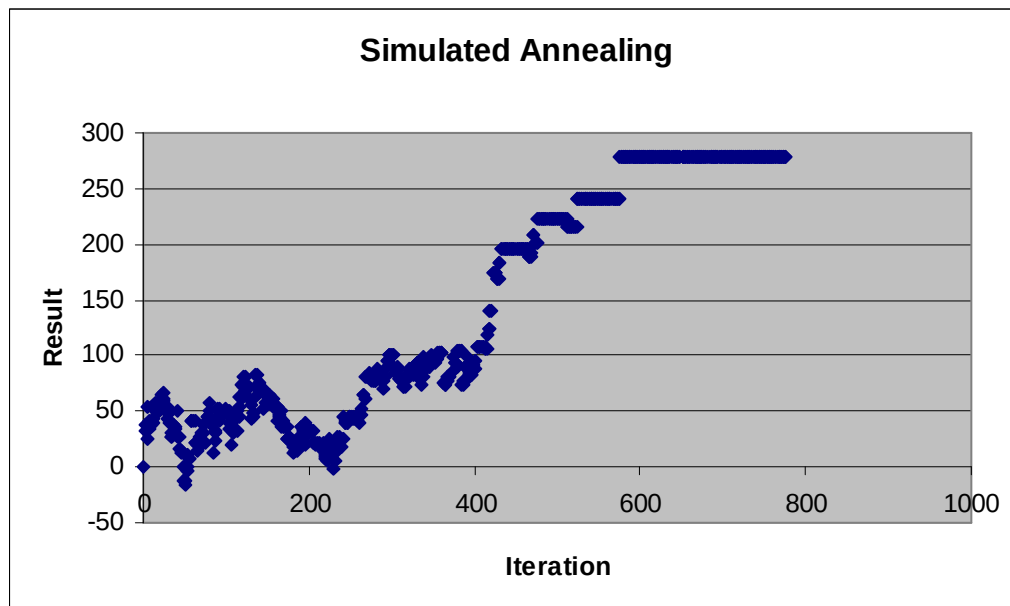


Figure 20 Variation of the result with the Simulated Annealing algorithm

The results from the Simulated Annealing algorithm show the initial volatility of the considered solutions, accepting new solutions very fast, some of them with quite a low performance. However, once the temperature starts to decrease and the algorithm finds a favourable region, the improvements are happen in a clear way.

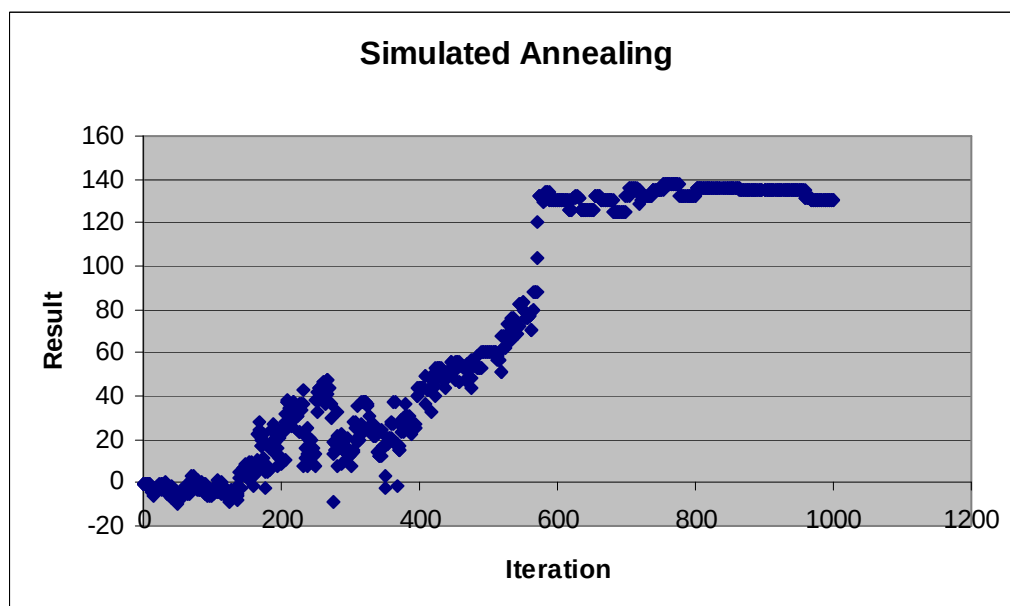


Figure 21 Variation of the result with the Simulated Annealing algorithm

A second test using the Simulated Annealing algorithm was made, this time with the initial solution being the robot performing no movement at all, that is, all initial parameters are set to zero. The results are displayed in figure 21:

8.4 Results Analysis

Although the Simulated Annealing algorithm is supposed to find the optimal solution of the problem, in this situation the results provided are in fact worse than most of the tests done with the other algorithms. This can be explained by the big amount of noise in the simulator, which can produce a different result for the same solution (and often does), and this has an effect on the evolution of the solution. For example, a solution that is not very good may be accepted because in that particular simulation the noise makes the results much better than they are on average. Once a solution like this is accepted, it may be difficult for the algorithm to find again a more favourable region, especially when the temperature is lower, and thus be stuck in a less favourable region.

Another possible explanation, is that we did not start from a random solution. The initial solution was possibly already in the neighbourhood of the optimal solution, so the Simulated Annealing algorithm doesn't help in finding a better region (which is its main advantage) and may actually take the solution away from the more favourable region. It has been proven that the Simulated Annealing algorithm can find the optimal solution, but this is only guaranteed after an infinite number of iterations [Aarts, 1989]. So it is possible that the solution found by the Simulated Annealing algorithm after only 800 iterations is still far from the optimal solution, and can in fact be worse than the solution found by the Hill-Climbing algorithm.

Looking now at the test with the Simulated Annealing algorithm using the robot being completely stopped as a starting point, it evolved in the way of improving the solution. However, the solution reached was not as good as with the other tests. This is expected, due to the fact that the initial solution wasn't favourable. To reach a better solution, a very big number of iterations would be necessary, as well as a possible increase in initial temperature and α coefficient in the Simulated Annealing algorithm. It also demonstrates that it is possible to create new movements from zero, just by defining an objective function that will be maximized by an optimization algorithm. However, choosing a starting point that already is closer to the desired movement may be more efficient. Some analysis of the movements of the robot may be required to do that.

Running again each of the solutions provided by each of the tests, we can compare the final solutions, and compare the results with the results that these solutions had produced in the optimization algorithms.

Test	Result during optimization	Result of the solution test
Hill-Climbing 1	328,24	228,92
Hill-Climbing 2	387,09	312,38
Hill-Climbing 3	248,82	211,04
Tabu Search 1	347,85	283,41
Tabu Search 2	339,22	231,68
Simulated Annealing	278,68	149,16

Table 2 Comparison of the results of each optimization test

The big influence of noise can be seen clearly in this table. When each solution was tested again, the results were clearly worse than their result during the optimization tests. This can show that a solution was accepted during those tests as the best, because it was just a good solution that provided exceptional results during that one test. However, being the best in that test doesn't mean it is the best on average.

All of these solutions found are much better than the initial solution, proving that the algorithm is successful in finding a good solution. Comparing good solutions to find the best one may be hard, because of wide range of results a particular solution may produce.

We can compare the solutions, in order to see if they are very different from each other:

Leg Movement	Joint	Initial Solution	HC 1	HC 2	HC 3	TS 1	TS 2	SA
LegRotate1	1	0	0.1	0.4	0.05	0.05	0	0.2
	2	0.3	0.7	0.8	0.5	0.75	0.65	0.6
	3	0	-0.1	-0.1	0.05	-0.2	-0.15	0.1
LegRotate2	1	0	-0.1	-0.2	-0.05	-0.1	-0.15	-0.15
	2	0.3	0	0.3	0.25	0.35	0.15	0.15
	3	0.8	1	0.8	0.9	1	1	0.9
ArmRotate1	1	0	0.1	0.1	0.1	0	0.15	0.25
	2	0	0.1	0.1	0.05	0.05	0.25	0.05
	3	0.3	0.3	0.5	0.25	0.15	0.3	0.35
	4	0	-0.1	0	-0.05	0.05	0.05	-0.3
ArmRotate2	1	0	0.1	0.1	0	0.1	0.25	-0.5
	2	0	-0.3	-0.3	-0.35	-0.4	-0.25	-0.2
	3	0.3	0.4	0.3	0.35	0.35	0.3	0.4
	4	0.8	0.9	1	0.65	0.85	0.8	0.95

Table 3 Comparison between optimized solutions

We can see that neither of the solutions found are very far from the initial solution. Even the Simulated Annealing algorithm didn't go far to a different region. This can mean that there is not much space to find a good solution, so it would be useful to make the tests with smaller changes. However, with smaller changes, the effect of noise when comparing solutions would become even more important, making these comparisons hard.

Chapter 9

9. Conclusions

This chapter makes a summary of the work described in this dissertation, suggests the areas in which further work is required, and draws the final conclusions.

The current state of legged robotics in general was presented and some of the approaches to legged robots were described. It also presented the robuDOG robot, whose simulated version was used in this work. This robot, although simpler than the Sony AIBO, poses a difficult problem in what regards legged locomotion and walking behaviors.

The RoboCup international initiative was presented with emphasis on the several leagues that compose the main competition.

Microsoft Robotics Studio and some of its characteristics were presented. Microsoft Robotics Studio was used as the basis for the implementation of the robotic movement optimization framework developed in the context of this thesis.

The Microsoft Robotics Soccer Challenge was presented, as well as the model of the simulated RobuDOG robot used as the basis for this challenge.

Some of the most well-known metaheuristics were presented and discussed, so that it could be understood how they would work in this specific problem of optimization of a quadruped robot.

The results obtained from the tests done using the previous methods were presented and analysed.

From the results obtained, we can see that the methods used here were proven to significantly improve the studied robot movements. The Hill-Climbing algorithm did not perform badly, but this can be due to the fact that the experiments started from a favourable, already semi-optimized solution. This can explain why the Simulated Annealing algorithm did not perform better in finding a good solution. However, the Hill-Climbing algorithm is clearly outperformed by the improved algorithm that we

referred to as Simplified Tabu Search. This algorithm can make local search faster when compared to simple Hill-Climbing.

However there are some problems with this approach:

- The simulation time is long, and that makes it impractical to run the optimization algorithms for the large number of iterations that is needed for the algorithms to be truly effective.
- The results from the simulation can vary in a significant range for the same set of movements.

It has been proved that the optimization algorithms work well in this problem. However, to make this optimization more efficient, there are some improvements to make.

The long time of the iteration may be solved by implementing changes to the simulator that would reduce the simulation to the minimum services needed to perform the specific tasks that we're studying. These changes were not yet possible to make during the work done so far. The machine where the simulation is running also has an influence, and performance can be improved by using a more specialized computer, with better capacity for physics and graphics processing.

The wide range of possible results for a single solution needs to be addressed, because it can be an obstacle in applying effectively the optimization methods. This effect can probably be mitigated by making the testing more robust, by making several tests for each solution. This can only be feasible if the simulation time is also addressed, or else the long time to make the testing will make this not practical.

For the optimization of robots, other optimization algorithms may prove useful. A solution not addressed in this work, due to the time needed to perform the corresponding experiments, was the use of Genetic Algorithms. This approach may however prove efficient for solving this kind of problem.

References

- [Aarts, 1989] Aarts E. H. and Korst J., “Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing”, N.Y. John Wiley and Sons, New York, NY, USA, 1989
- [Adams, 2000] Adams B., Breazeal C., Brooks R.A., Scassellati, B., “Humanoid Robots: A New kind of Tool”, Intelligent Systems and Their Applications, IEEE, Volume 15 Issue 4 , 2000
- [Ageia, 2008] <http://www.ageia.com/developers/features.html>, retrieved on 29/02/2008
- [Anon, 1967] Anon, “Logistical Vehicle Off-Road Mobility”, Project TCCO 62-5, U.S. Army Transportation Combat Developments Agency, 1967
- [DSS, 2008] <http://msdn2.microsoft.com/en-us/library/bb483067.aspx>, retrieved on 23/01/2008
- [Duffert, 2005] Duffert U., Hoffman J., “Reliable and Precise Gait Modeling for a Quadruped Robot”, Institut für Informatik, LFG Kunstliche Intelligenz, Berlin, 2005
- [Fu, 1987] Fu K. S., Gonzalez, R. C., Lee C. S. G., “Robotics: Control, Sensing, Vision, and Intelligence”, McGraw-Hill International Editions, 1987
- [Gates, 2007] Gates B., “A Robot in Every Home”, Scientific American, January 2007
- [Glover, 2003] Glover, F. and Kochenberger, G. A., “Handbook of Metaheuristics”, Kluwer Academic Publishers, 2003
- [Gnsin, 2005] http://upload.wikimedia.org/wikipedia/commons/0/05/HONDA_ASIMO.jpg, retrieved on 03/03/2008
- [Hopgood, 2001] Hopgood, A. A., “Intelligent Systems for Engineers and Scientists, CRC Press, 2001
- [Humanoid Soccer, 2008], <http://www.humanoidsoccer.org/>, retrieved on 24/02/2008
- [Kitano, 1997] Kitano H. et al., “RoboCup – The Robot World Cup Initiative”
- [Kirkpatrick, 1983] Kirkpatrick S., Gelatt C. D. and Vecchi M. P., “Optimization by simulated annealing”, Science, Number 4598, 13 May 1983, vol. 220, pages 671-680, 1983.
- [Laarhoven, 1987] Laarhoven P. J. M., Aarts E. H. L., “Simulated Annealing: Theory and Applications”, Springer, 1987

- [Mitchell, 1996] Mitchell M., "An Introduction to Genetic Algorithms", MIT Press, 1996
- [MSL, 2007] MSL Technical Comitee, "Middle Size Robot League Rules and Regulations for 2008", Draft Version 12.2, November 9, 2007
- [MSRS, 2008] <http://www.msdn.com/robotics/>, retrieved on 26/02/2008
- [NIST, 2008] <http://robotarenas.nist.gov/2006/index.htm>, retrieved on 23/02/2008
- [Osman, 1996] Osman I. H., Kelly J. P., "Meta-heuristics: Theory & Applications", Kluwer Academic Publishers, 1996
- [Pierreval, 2000] Pierreval H., Paris J., "Distributed evolutionary algorithms for simulation optimization", IEEE Transactions on Systems, Man and Cybernetics, vol. 30, no. 1, January 2000
- [Raibert, 1986] Raibert Marc H., "Legged Robots that Balance", MIT Press, 1986
- [Rescue, 2008] <http://www.rescuesystem.org/robocuprescue/>, retrieved on 23/02/2008
- [Restivo, 2006] Restivo A., "Dynamic Scenario Simulation Optimization", MSc Thesis, Faculdade de Engenharia da Universidade do Porto, May 2006
- [Robocup, 2008] <http://www.robocup.org/overview/22.html>, retrieved on 22/02/2008
- [RoboCup China, 2008] <http://robocup-cn.org/>, retrieved on 02/03/2008
- [RoboCup Junior, 2008] <http://rcj.sci.brooklyn.cuny.edu/>, retrieved on 24/02/2008
- [Robotica UA, 2008] <http://robotica.ua.pt/robotica2008/>, retrieved on 28/02/2008
- [robuBOX, 2008] <http://www.robubox.com/robosoft/data/Main/MainHome.jsp>, retrieved on 24/01/2008
- [Robudog Simulator, 2008] <http://www.robubox.com/robosoft/data/Robocup/RobudogSimulatorFaq.jsp>, retrieved on 25/01/2008
- [Russell, 2002] Russell S., Norvig P., "Artificial intelligence: A modern Approach", Prentice Hall, 2002
- [Small-Size, 2008] <http://small-size.informatik.uni-bremen.de/>, retrieved on 23/02/2008
- [U. Texas, 2008] http://www.robotics.utexas.edu/rrg/learn_more/low_ed/types, retrieved on 28/02/2008