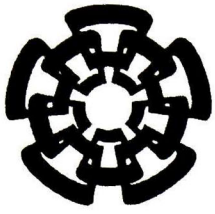
XX(147081.1)



Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

Animación del Brazo de un Avatar Usando Planeación Dinámica

Tesis que presenta:
Angel Rafael Rodríguez Moreno

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco, Marzo de 2008.

Animación del Brazo de un Avatar Usando Planeación Dinámica

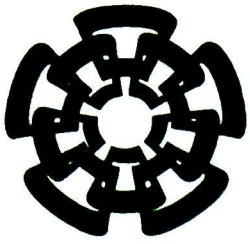
Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Angel Rafael Rodríguez Moreno
Ingeniero en Sistemas Computacionales
Universidad Autónoma de Aguascalientes 1999-2004

Becario de CONACYT, expediente no. 199447

Director de Tesis
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Marzo de 2008.



Centro de Investigación y de Estudios Avanzados

del I.P.N.

Unidad Guadalajara

Avatar's Arm Animation Using Dynamic Planning

A thesis presented by:
Angel Rafael Rodríguez Moreno

to obtain the degree of:
Master in Science

in the subject of:
Electrical Engineering

Thesis Advisors:
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco, March 2008.

Avatar's Arm Animation Using Dynamic Planning

Master of Science Thesis
In Electrical Engineering

By·
Angel Rafael Rodríguez Moreno
Engineer in Computer Science
Universidad Autónoma de Aguascalientes
1998-2002

Scholarship granted by CONACYT, No. 199447

Thesis Advisors:
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, March, 2008.

- **ANIMACIÓN DEL BRAZO DE UN AVATAR USANDO PLANEACION DINÁMICA**

La planeación dinámica es importante en sistemas complejos donde es necesario alcanzar metas independientemente de la dinamicidad del ambiente. Los agentes juegan un papel importante en la planeación dinámica debido a que cada agente toma en cuenta el comportamiento del sistema en el cual evoluciona. Existen algunos ejemplos de aplicaciones basadas en agentes con ambientes dinámicos, tales como: *Planeación aplicada a Múltiples Robots, Mapas en Ambientes Cambiantes, Planeación de Corredores con Obstáculos, etc.* En esta clase de aplicaciones no se puede tener una lista estática de eventos, es decir, no hay certeza tanto del comportamiento de los agentes como del ambiente. Sin embargo, los agentes siguen una *agenda* que contiene una sucesión de metas programadas. Por lo tanto, es necesario proveer a los agentes con capacidades especiales, tales como: sensado en tiempo real del ambiente y re-planeación de las metas iniciales. Estas capacidades que se acaban de mencionar caracterizan a la *Planeación Dinámica.*

El objetivo principal de esta tesis es *proponer* un Algoritmo de Planeación Dinámica apropiados para animar el brazo de un avatar. Siguiendo la estrategia de *divide y vencerás*, se ha dividido el objetivo principal en tres sub-objetivos:

1. Proveer un vector compuesto por *propuestas detalladas de acciones ordenadas* (planes).
2. El agente debe reaccionar a cambios inesperados del ambiente.
3. El agente debe crear sus planes basándose en el conocimiento generado por tareas previas.
4. El Algoritmo de Planeación Dinámica tiene que interactuar con un Algoritmo de Aprendizaje con el propósito de obtener información (conocimiento) requerido para crear cada plan.

- **AVATAR'S ARM ANIMATION USING DYNAMIC PLANNING**

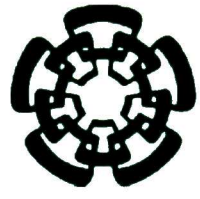
Dynamic Planning is important in complex systems where it is important to achieve goals independently of environment's behavior. The agents play an important role in dynamic planning because every agent is provided with a sensing mechanism of the environment where he evolves. There are several agent-based applications with dynamic environments, such as: *Planning with multiple robots, Creating Roadmaps in Changing Environments, Planning Corridors amidst Obstacles, etc.* In this sort of applications it is impossible to use a static scheduling, mainly because there is not a definition of the agent or environment's behavior. However, every agent follows an agenda that specifies an order for the goals to be achieved. Thus, it is necessary to provide an agent with special skills, such as: real time sensing of the environment and re-planning the initial goals.

The objective of this thesis is *proposing* a useful Dynamic Planning Algorithm which animates the avatar's arm in 3D. Following a *divide and conquer* strategy, we have divided the main objective in three sub-objectives:

1. Providing a vector composed by *detailed proposals of ordered actions* (plans).
2. The agent must react to unexpected changes of the environment.
3. The agent must create its plans based on the knowledge generated by previous tasks.
4. The Dynamic Planning Algorithm has to interact with a Learning Algorithm in order to obtain the information (knowledge) needed to create every plan.

CINVESTAV

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara



Avatar's Arm Animation Using Dynamic Planning

MASTER IN SCIENCES THESIS IN ELECTRICAL ENGINEERING

PRESENTED BY:

Angel Rafael Rodríguez Moreno

TO OBTAIN THE DEGREE OF:

Master in Sciences

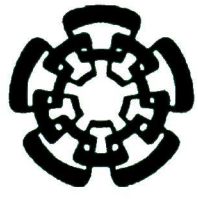
IN THE SPECIALTY OF:

Electrical Engineering

Guadalajara, Jalisco. Winter of 2008

CINVESTAV

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara



Avatar's Arm Animation Using Dynamic Planning

MASTER IN SCIENCES THESIS IN ELECTRICAL ENGINEERING

PRESENTED BY:
Angel Rafael Rodríguez Moreno

TO OBTAIN THE DEGREE OF:
Master in Sciences

IN THE SUBJECT OF:
Electrical Engineering

THESIS ADVISOR
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco. February of 2008

Master in Sciences Thesis in Electrical Engineering

Presented by:

Angel Rafael Rodríguez Moreno

to obtain the degree of:

Master in Sciences

in the specialty of:

Electrical Engineering

Dr. Félix Francisco Ramos Corchado Thesis Advisor	Dr. Luis Ernesto López Mellado Sinodal
Dr. Mario Angel Siller González Pico Sinodal	Dr. Ricardo Vilalta López Sinodal

21 of February of 2008

Acknowledgments

To my Son ... who represents my immortality.

To my Wife ... who represents my inspiration.

To my Mother ... who represents my faith.

To my Father ... who represents my persistence.

To my Thesis Advisor ... who was my guide in this part of my education.

Preface

Dynamic Planning is important in complex systems where it is important to achieve goals independently of environment's behavior. The agents play an important role in dynamic planning because every agent is provided with a sensing mechanism of the environment where he evolves. There are several agent-based applications with dynamic environments, such as: **Planning with multiple robots**, **Creating Roadmaps in Changing Environments**, **Planning Corridors amidst Obstacles**, etc. In this sort of applications it is impossible to use a static scheduling, mainly because there is not a definition of the agent or environment's behavior. However, every agent follows an agenda that specifies an order for the goals to be achieved. Thus, it is necessary to provide an agent with special skills, such as: real time sensing of the environment and re-planning the initial goals.

The objective of this thesis is proposing a useful Dynamic Planning Algorithm which animates the avatar's arm in 3D. Following a *divide and conquer* strategy, we have divided the main objective in three sub-objectives:

- Providing a vector composed by detailed proposals of ordered actions (plans).
- The agent must react to unexpected changes of the environment.
- The agent must create its plans based on the knowledge generated by previous tasks.
- The Dynamic Planning Algorithm has to interact with a Learning Algorithm in order to obtain the information (knowledge) needed to create every plan.

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Objectives

Following the previous definitions, we now propose a set of goals to accomplish in this work.

Following the previous definitions, we now propose a set of goals to accomplish in this work.

1.2 Problem Definition

Currently, the Multi-Agent systems are used in dynamic applications. Some examples of dynamic applications are: Multi-Robot planning [26], Roadmaps in Changing environments [13], Navigation In Computer Games [4], etc. In this kind of applications there is not an exact definition of task execution time, i.e. there is not certainty about the environment, objects and agents' behavior. In addition these applications are developed with autonomous agents. These agents follow an agenda, which contains an ordered list of programmed goals.

Based in the nature these applications, it is necessary to provide the agents with particular

capabilities, such as: **sensing** the environment, **acting** on this environment and **changing** it later. All these capabilities constitute what we define as *Dynamic Planning*.

1.2.1 Dynamic Planning

Here we present our two definitions of the *Dynamic Planning* concept:

Definition 1.1. *Process of setting goals, developing strategies, and outlining tasks and schedules to accomplish the goals in uncertain conditions.*

Definition 1.2. *Method used by an agent to establish the future actions to perform and their execution time in a changing environment.*

Following the previous definitions, we now propose a set of goals to accomplish in this work.

1.3 GOALS

The main objective in this thesis is proposing a Dynamic Planning Algorithm which animates the avatar's arm in 3D. In order to accomplish such objective, we have decomposed it in three sub-objectives or goals:

1. We must provide a vector composed by detailed proposals of ordered actions (*plans*) in order to complete the agent's task.
2. The agent must react to unexpected changes of environment.
3. The agent must create its plans based on previous knowledge.
4. The Dynamic Planning Algorithm has to interact with a Learning Algorithm in order to obtain the information (*knowledge*) needed to create every plan.

In the next section we describe the proposed solutions to accomplish the previous goals.

1.4 Proposal

In order to understand the problem that we are treating, we have split the problem in two sub-problems, presented as follows:

- **Multi-objective Problem:** There are several performance criteria which are usually in conflict with each other.
- **Dynamic Environment Problem:** Environment changes constantly and different decisions must be evaluated in different stages.

Thus, to solve such sub-problems and fulfill the goals proposed earlier, we propose the following solutions:

- **Design a Planning Algorithm based in stochastic methods:** In this case we have decided to employ a Genetic Algorithm, particularly the **MOEA** (*Multi-Objective Evolutive Algorithms*).
- **Use a Sense Algorithm to evaluate the environment:** We have chosen the *Collision-Detection Algorithms* due to it is the one of the most used *Sense Algorithms* and it fulfills our Study Case requirements.
- **Design a Knowledge Representation for the Planning Algorithm:** In order to prevent the usage of *Translation Routines* [1] the information's format provided by the Learning Algorithm is equal to format used in Planning Algorithm.

[1] Translation Routine: Interpretation process executed by two or more algorithms in order to use the information that they interchange

Chapter 2

STATE OF ART

2.1 Objectives

2.2 Introduction

The Multi-Agent Environments present many situations where a Dynamic Planning Routine is required. Some of those situations are related with a Motion task, for instance: when an agent is changing its position, when an avatar moves one of his extremities, when the agent's environment changes its structural configuration or even when an agent tries to avoid a collision with other (s) agent(s).

Some approaches have treated Dynamic Planning problem as a *Path Planning* [4, 14, 15] or *Motion Planning* [21, 25, 13, 7] one. At the same time, other approaches have complemented its Path or Motion Planning's solutions with some *Collision-Detection Algorithms* [18, 11, 17, 23].

This chapter presents four methods that some approaches have used in order to solve the Path or Motion Planning problem: **PRM** (*Probabilistic Roadmap Method*), **Voronoi Diagrams**, **Evolutionary Algorithms** and **Simulated Annealing**.

After we have presented these methods, we explain the Collision-Detection problem and also we mention some approaches that have treated this subject.

The objective of presenting different approaches about Collision-Detection algorithms is to mention what is the most accurate technique for our implementation, in order to focus on the Planning algorithm and leave the Collision-Detection module as a complementary feature of the model.

2.3 PRM (*Probabilistics Roadmap Method*)

To create the motion of entities in a virtual environment, we should plan the path of entities between locations in the virtual world. There are several techniques to plan such paths, but one popular *Path Planning* technique in robotics is PRM [4]. This method builds a roadmap of possible motions of the robot through the environment. When a particular path planning query must be solved, a path is retrieved from this roadmap using a simple and fast graph search.

In the following paragraphs we present several approaches that have used this technique in order to solve Path or Motion Planning problems.

2.3.1 Constraint-Based Motion Planning

Here there is an interaction between the PRM method and a constraint-based planning [21] which simulates robot deformation and make appropriate path adjustments and corrections at the moment of computing a collision-free path.

In a constrained-based Planning technique, the planning algorithm is used to compute the intermediate states that link two points and satisfy the constraints imposed on the dynamical system. Motion planning is not treated as a purely geometric problem, but also incorporates

mechanical and physical properties of the robots and obstacles.

2.3.2 High Quality Navigation

In the **RTS** (*Real Time Strategy*) and **FPS/TPS** (*First/Third Person Shooters*) there are a large number of entities what means that the environment is really complex. PRM is very suitable for complicated environments, but it has the disadvantage that it produces low quality paths consisting of straight line segments and this method consume a lot of time, during the game play.

Nevertheless, a novel path planning [4], built on the PRM method, constructs a roadmap of possible motions but guarantees short paths that have enough clearance from the obstacles and leads to natural looking motions. Every path can be retrieved almost instantaneously, without post-processing.

2.3.3 Motion Planning in Dynamic Environments

Some methods build a roadmap in a pre-processing phase [25]. The roadmap is built for the static part of the scene without the dynamic obstacles and without the additional dimension for time. This can be done using a standard PRM method.

The method searches for a near-time-optimal trajectory between a start and a goal configuration in the roadmap, without collisions with the dynamic obstacles. A two-level approach is used to find a trajectory. On the local level, trajectories on single edges of the roadmap are found in an implicit grid in *state-time* space, using a *depth-first search* strategy. On the global level, the local trajectories are coordinated to find a near-time-optimal global trajectory in the entire roadmap.

2.3.4 Motion Planning in Changing Environments

A Changing Environment implies that every entity will face "occasionally" several obstacles, which makes him to change their placement. It's assumed that the potential obstacle placement is known. The knowledge of potential placements is employed to create roadmaps that incorporate the possible changes of the environment.

The method [13] covers the topology of the free space independent of the placements of the moving obstacles, and also produces roadmaps for changing environments that are robust (any placement of the moving obstacles, the road contains a path between any pair of query configurations). The approach follows the PRM paradigm, with the constraint of PRM implementation adds edges between nodes that were already in the same connected component of the roadmap.

2.4 Generalized Voronoi Diagrams

GVDs (*Generalized Voronoi diagrams*) have long been used as a basis for motion planning algorithms [7]. The GVD represents the *c*onnectivity of a space but has a dimension lower by one, and (in the three dimensions) it is composed of *s*urfaces of maximal clearance.

2.4.1 Constraint-based Motion Planning Using Voronoi Diagrams

The algorithmic framework presented in [7] is based on constrained dynamics used for physically-based modeling. The motion planning problem is reformulated as a dynamical system simulation, where constraints are enforced by virtual forces imposed on the system. Each robot is treated as a rigid body, or a collection of rigid bodies, and is moved subject to all types of constraint forces (some inherent to the planning scenario). The global geometric analysis from the generalized *Voronoi Diagram* is used to define constraints of static and

moving obstacles.

The solution of the motion planning problem is the collection of configurations of the dynamical system that satisfy all geometric and mechanical constraints.

2.5 Evolutionary Algorithms

Single-objective EAs, in particular **GAs** (*Genetic Algorithms*), **EP** (*Evolutionary Programming*) and **ES** (*Evolution Strategies*) have been shown to find whether the solution is optimal or not, i.e. "satisfies" the user.

The goal is to search the associated *objective/fitness* function landscape (phenotype space) through exploration and exploitation for the *optimal* solution. Such activity is controlled through the use of biologically inspired **mating**, **mutation** and **selection** operators.

Specific evolutionary algorithm development involves the encoding of the independent variables (genotype) and the structuring of specific parametric mating, mutation, and selection operators. These operators manipulate each genotype individual appropriately as the search proceeds through the phenotype landscape.

2.5.1 Single-objective EAs : Genetic Algorithms

In order to apply the Genetic Algorithms as a method to solve Robot Motion Planning [12], it is necessary to focus on reducing task's time; in this case, the task consists on moving a six-degree-of-freedom arm among moving obstacles.

To solve a path planning problem is necessary by using GA it's necessary to solve the *inverse kinematic* problem.

The method used in the paper to solve such problem is the following:

1. **Coding the problem:** The search space is discretized.

2. **Generating an initial population:** A random set of n "individuals"

3. **Operating a selection:** A fitness function (f) is applied to each member of the population and the "individuals" are ranked accordingly to their associated value of f.

   > *" The closer a configuration brings the extremity of the arm to the desired goal without collision, the better its rank will be ... "*

4. **Creating couples and combining individuals:** A set of n couple is generated. This couple is obtained by randomly picking an element of the population with a probability proportional to their rank. The *cross-over* operation is used to produce two new individuals.

5. **Termination conditions:** There are two termination conditions:

   (a) The absolute minimum is obtained for one element of the population and a solution is found.

   (b) The population stabilized and the algorithm is stuck in a local minimum. If neither condition is true the step No. 3 is applied to the new population.

They also propose a simple path planner where they consider a discretized subset of all the possible paths (with or without collision) starting from the initial configuration.

2.5.2 MOEA

The Multiobjective Optimization Problem (MOP)

MOP [2] is defined as a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions

form a mathematical description of performance criteria which are usually in conflict with each other.

Optimum Solution

In order to standardize the meaning of "optimum solution" the most commonly accepted term is *Pareto optimum* [2]. A vector $\vec{x}^*$ is *Pareto optimal* if there is not feasible vector of decision variables $\vec{x} \in$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion.

Use of EA's

Evolutionary algorithms seem particularly suitable to solve multi-objective optimization problems, because they deal simultaneously with a set of possible solutions. This allows us to find several members of the Pareto optimal set in a single run of the algorithm.

Classification of Evolutionary Multi-Objective Optimization approaches [1]

▷ *First Generation Pareto-based Approaches:*

1. Used to solve the problems with Schaffer's VEGA[2]
2. Requires a ranking procedure and a technique to maintain diversity in the population (otherwise, the GA will tend to converge to a single solution).

MOGA (*Multi-Objective Genetic Algorithm*).- Consist on a scheme in which the rank of certain individual corresponds to the number of individuals in the current population by which it is dominated. It uses fitness sharing and mating restrictions.

[1] We focus on Pareto-Based approaches

[2] *Vector Evaluated Genetic Algorithm* (**VEGA**): It uses subpopulations that optimize each objective separately

NSGA (*Non-dominated Sorting Genetic Algorithm*).- Based on several layers of classifications of the individuals. Non-dominated individuals get a certain dummy fitness value and then are removed from the population. The process is repeated until the entire population has been classified.

NPGA (*Niched-Pareto Genetic Algorithm*).- It uses a tournament selection scheme based on Pareto dominance. Two individuals randomly chosen are compared against a subset from the entire population (10% of the population). When both competitors are either dominated or non-dominated the result of the tournament is decided through fitness sharing in the objective domain (equivalent class sharing).

NPGA 2.- It uses Pareto ranking but keeps tournament selection (solving ties through fitness sharing as in the original NPGA). Niche counts in the NPGA 2 are calculated using individuals in the partially filled next generation, rather than using the current generation (continuously updated fitness sharing).

▷ *Second Generation Techniques*

1. Emphasizes EFFICENCY.
2. External populations become common and clever ways are devised to generate solutions that are both non-dominated and uniformly distributed.

PAES (*Pareto Archived Evolution Strategy*).- It uses a (1+1) evolution strategy together with an external archive that records all the non-dominated vectors previously found. It uses and adaptive grid to maintain diversity.

PESA (*Pareto Envelope-based Selection Algorithm*).- It uses a small internal population and a larger external population. It uses the same hyper-grid division of phenotype space adopted by PAES, but its selection mechanism is based on the crowding measure used by the hyper-grid. This crowding measure is used to decide what solutions to introduce into the external population.

PESA II.- Revised version of PESA which region-based selection is adopted. In region-based adopted selection, the unit of selection is a hyperbox rather than an individual. The procedure consists of selecting a hyperbox and then randomly selecting an individual within such hyperbox.

SPEA (*Strength Pareto Evolutionary Algorithm*).- It uses an external archive containing non-dominated solutions that have been already found. A clustering technique called *average linkage method* is used to keep diversity.

SPEA 2.- It incorporates a fine-grained fitness assignment strategy taking into account (for each individual) the number of individuals that dominate it and the number of individuals by which its is dominated. It uses a nearest neighbor density estimation technique (efficient search). And It has and enhanced archive truncation method that guarantees the preservation of boundary solutions.

NSGA-II.- More efficient (computationally speaking) than NSGA. Uses elitism and a crowded comparison operator that keeps diversity without specifying any additional parameters.

MOMGA (*Multi-Objective Messy Genetic Algorithm*).- Attempt to extend the messy GA to solve multi-objective optimization problems. It has three phases:

1. *Initialization Phase*.- Production of building blocks of certain specified size using partially enumerative initialization.
2. *Primordial Phase*.- Performs tournament selection on the population and reduces the population size if necessary.
3. *Juxtapositional Phase*.- Messy GA proceeds by building up the population trough the use of the cut and splice recombination operator.

MOMGA-II.- As MOMGA, MOMGA-II this method has three phases:

1. *Initialization Phase.*- It uses probabilistic complete initialization which creates a controlled number of building block clones of a specified size.
2. *Block Filtering Phase.*- Reduces the number of building blocks through a filtering process and stores the best building blocks found.
3. *Juxtapositional Phase.*- Same as MOMGA.

Micro Genetic Algorithm.- In this method a random population is taken, then each individual is dived in two parts Replaceable and Non-replaceable sections. The Non-replaceable section will remains during the process of generation creation. The replaceable section will be modified (even combined) in the Selection, Cross-over and mutation processes. Then, by elitism method, a new generation is created, and the Nominal Convergence criteria are applied to the new population, in case of negative response, the process will begin with the new population. In case of affirmative response, the population will pass through a filter and then "store" the characteristics of the new population in the external memory, to create the new population memory.

2.5.3 Steady State Multi-Objective GAs

There are some challenges faced in the application of GAs to engineering design domains, for instance:

- The search space can be very complex with many constraints and the feasible (physically realizable) region in the search space can be very small.
- Determining the quality (*fitness*) of each point may involve the use of a simulator or an analysis code which takes a non-negligible amount of time. This simulation time can range from a fraction of a second to several days in some cases. Therefore it is impossible to be cavalier with the number of objective evaluations in an optimization.

In order to propose a solution to such problems, some approaches have used the **ssGAs** (*Steady State GAs*)[3] in conjunction with MOEAs. In [3] two methods are proposed, the first one called the **OEGADO** (*Objective Exchange Genetic Algorithm for Design Optimization*) several single objective GAs run concurrently. Each GA optimizes one of the objectives. At certain intervals these GAs exchange information about their respective objectives with each other. In the second method called the **OSGADO** (*Objective Switching Genetic Algorithm for Design Optimization*) a single GA runs multiple objectives in a sequence switching at certain intervals between objectives.

2.6 Simulated Annealing

The **SA** (*Simulated Annealing*) [24] process consists of first "melting" the system being optimized at a high effective temperature, then lowering the temperature by slow stages until the system "freezes" and no further changes occur. At each temperature, the simulation must proceed long enough for the system to reach a *steady state*. The sequence of temperatures and the number of re-arrangements of the $\{x_i\}$ attempted to reach equilibrium at each temperature can be considered an *annealing schedule*.

2.6.1 Metropolis procedure

Metropolis, et al., [20] in the earliest days of scientific computing, introduced a simple algorithm that can be used to provide an efficient simulation of a collection of atoms in equilibrium at a given temperature. In each step of this algorithm, an atom is given a small random displacement and the resulting change, ΔE, in the energy of the system is computed. If $\Delta E = 0$, the displacement is accepted and the displaced-atom's configuration would be the

[3] In the ssGAs, the worst individual is discarded every generation and substituted by a chromosome which is product of the genetic operators' execution.

starting point of the next step. The case $\Delta E > 0$ is treated probabilistically: the probability of the configuration is accepted is $P(\Delta E) = exp^{(\Delta E/k_b T)}$ [4] Random number uniformly distributed in the interval (0,1) are a convenient means of implementing the random part of the algorithm. One such number is selected and compared with $P(\Delta E)$. If it is less than $P(\Delta E)$ the new configuration is retained; if not, the original configuration is used to start the next step. By repeating the basic step many times, one simulates the thermal motion of atoms in thermal contact with a heat bath at temperature T. This choice of $P(\Delta E)$ has the consequence that the system evolves into a Boltzmann distribution [5]

Annealing, as implemented by the Metropolis procedure, differs from iterative improvement in that the procedure need not get stuck since transitions out of a local optimum are always possible at non-zero temperature. A second and more important feature is that a sort of *adaptative divide-and-conquer* occurs. Gross features of the eventual state of the system appear at higher temperatures; fine details develop at lower temperatures.

2.7 Collision-Detection Methods

Some approaches [18] propose a collision phase separated into three parts: *collision detection*, *contact area determination* and *collision response* (which is application dependent).

2.7.1 Collision Detection between Multiple Moving Objects

Temporal and geometric coherence

Temporal and **Geometric coherence** is the property of the state of the application does not change significantly between successive time steps of simulation frames. The objects

[4] Where k_b is the Boltzmann constant

[5] Boltzmann distribution: Kinetic energy in a substance

move only slightly from frame to frame. This slight movement of the objects translates into geometric coherence, since their spatial relationship does not change much between frames.

The algorithm uses a *Sweep-and-Prune*[11] technique to eliminate testing object pairs that are far apart, and later we show that the technique can be extended to eliminate testing features that are far apart between two colliding objects.

2.7.2 Exact Collision Detection

Collision Detection between Convex Polytopes

The algorithm proposed in [17] computes the convex hull of all objects as part of pre-processing. It classifies the feature of convex hulls into red and green features. The red features correspond to the features of the original model and the green features are introduced by the convex hull computation.

1. **Voronoi Regions:** Each convex *polytope* is pre-processed into a modified boundary representation. The polytope data structure has fields for its features (faces, edges, and vertices) and corresponding Voronoi regions. A Voronoi region associated with a feature is a set of point closer to that feature than any other. This regions form a partition of the space outside the polytope, and they form the generalized Voronoi Diagram of the polytope.

2. **Closest Feature Tests:** The method for finding closest feature pair is based on Voronoi regions. First, it starts with a candidate pair of features, one from each polytope, and check whether the closest points lie on these features. If either feature fails the test, we step to a neighboring feature of one of both candidates, and try again.

3. **Penetration Detection for Convex Polytopes:** The key to detecting penetrations lies in partitioning the interior as well as the exterior of the convex polytope. Some times

is used an approximation [18] labeled as pseudo-internal Voronoi region. It is calculated by first computing the centroid of each convex polytope (the weighted average of all vertices). Then a hyperplane from each edge is extended towards the centroid. The extended hyperplane tapers to a point, forming a pyramid-type one over each face. Each of the faces of the given polytope is now used as a constraint plane. If a candidate feature fails the constraint imposed by the face, the algorithm *stepping* "enters" inside of the polytope. If at any time we find one point on a feature of one polytope is contained within the pseudo internal Voronoi region, it corresponds to a penetration.

4. **Feature Classification:** The algorithm proposed in 3 returns all pairs of overlapping features between the convex hulls.
   - *Red-Red feature overlap*: actual collision between the original models.
   - *Red-Green feature overlap*: This may or may not correspond to a collision.
   - *Green-Green feature overlap*: Same as Red-Green feature overlap.

Exact Contact Determination

The algorithm computes a hierarchical representation using **OBBs** (*Oriented Bounding Boxes*). An OBB is a rectangular bounding box at an arbitrary orientation in 3D space. The resulting hierarchical structure is referred to as an *OBBTree* [23].

1. **Building an OBBTree:** The tree construction has two components: the first one consists of the *placement* of a tight fitting OBB around a collection of polygons, and the second one would be *grouping* of nested OBB's into a tree hierarchy. We triangulate all polygons composed of more than three edges. The OBB computation algorithm makes use of mean (μ) and the covariance matrix **C**. The eigenvectors of a symmetric matrix, such as C, are mutually orthogonal. After normalizing them, they are used as a basis. We find the extremal vertices along each axis of this basis and size the bounding box,

oriented with the basis vectors in order to bound those extremal vertices.

2.8 Discussion

After we have analyzed the previously mentioned methods, we can obtain several conclusions:

- PRM is very suitable for complicated environments, but it has the disadvantage that it produces low quality paths consisting of straight line segments and this method consume a lot of time, during the game play.
- No good and practical algorithms are known for computing the Voronoi diagrams or large environments. In the worst case the complexity of Voronoi diagrams is $O(n^2)$ [7], where n is the number of polygons in the environment. It is hard to accurately compute an arrangement of curves and surfaces in polygonal environment using fixed precision arithmetic.
- Simulated Annealing may become trapped by any local minima, which does not allow moving up or down, or take a long time to find a reasonable solution, which makes the method unpreferrable sometimes. For these reasons, many SA implementations have been done as part of a hybrid method, [1, 10, 16, 9, 19]
- The ssGAs may perform better than generational GAs because they better retain the feasible points found in their populations and may have higher selection pressure which is desirable when evaluations are very expensive. With good diversity maintenance, ssGAs have done very well in several realistic domains [22].
- The Multi-Objective approach let us consider constraints and objectives of the task to be performed. Thus, a Planning algorithm can be designed with this orientation, besides a Collision-Detection technique which complements the Dynamic Planning.

- The Collision Detection Algorithms represent an important module in a Planning task, but we will use a simple OBB-Tree technique, which is already provided by the Java-3D library. The main reason of using this resource is we focus on evaluating the Planning Algorithm instead of detailing a Collision Detection Algorithm, which is not the main objective in this work.

In the following chapter we will detail the charateristics of our proposal. We will present how we combine the Multi-Objective algorithm with a Learning algorithm in order to obtain a near-optimal solution to the Avatar's Arm model.

Chapter 3

PROPOSAL

3.1 Introduction

In this chapter we will detail the techniques and definitions that we made and use in order to combine a Multi-Objective Evolutive Algorithm (MOEA) and one kind of Learning Algorithm.

We define how the information is shared between this two Algorithms and they interpretated it in order to generate near-optimal solutions. In addition, we present a brief *Interaction Architecture* between the MOEA and Learning Algorithm.

Finally, we describe how we design the *Avatar's Model* and its main features.

3.2 Representation of the population

3.2.1 Individual's configuration

The population's representation is one of the most important parameters that we have to take into account when design a Genetic Algorithm.

In this work, we have defined a vector of **integer-valued** alleles as the content for each individual. The possible values for each alles are in the rank of 0..3, where every value represent a movement to reach a pre-defined goal (Fig 3.2.1).

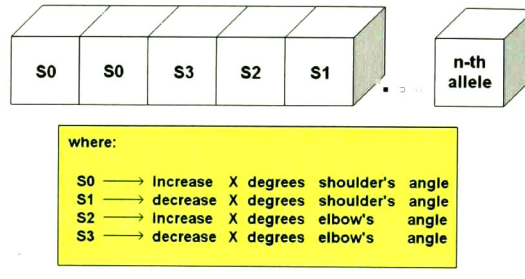


Figure 3.1: Chromosome with its alleles' specification.

We define formally an *individual* as follows:

Definition 3.1. *Let be S a population's individual and x a chromosome's allele, such as S=* $\{x|x\}$*, and* $|S| = n$*, where n is a integer value defined for the user (See section Chromosome's Length).*

and, the *population* is formaly defined as follows:

Definition 3.2. *Let be P the population to analyze such as P=* $\{I|I \in S\}$

3.2.2 Chromosome's Translation

The *Translation Process* of a chromosome is divided in two tasks: **reading** and **evaluation**. The *reading task* consist on going through the chromosome and get the allele in the current position. The *evaluation task* will assign an arbitrary value to every allele. The evaluation task makes the assignation according to a rule or directive (*Interpretation rule*) that we have included as a population's characteristic.

The algorithm implemented in this work uses the following Interpretation rule:

- **0** → *Increase X degrees shoulder's angle.*
- **1** → *Decrease X degrees shoulder's angle.*
- **2** → *Increase X degress elbow's angle.*
- **3** → *Decrease X degrees elbow's angle.*

where X is an integer-valued parameter that accomplishes with the following value restriction X∈ {0..180}.

3.2.3 Chromosome's Length

The individual that we consider in this work consist on a chromosome of length **n**, where **n** is a **problem-dependent** parameter that represents the number of alleles that will be evaluated by a "Evaluation Function"

We define n as problem-dependent because it depends of three main variables, such as: *problem's complexity* [1] *algorithm's complexity* and *individual's precision.*

The relation between n and each of these variables varies from a *directly-proportional* relation to *indirectly-proportional* one.

Based on the previous asumption, we propose the following assumptions:

- If the problem's complexity is *low* or *medium* [2] then we should assign a *low* value to n.
- If the problem's complexity is *high* [3] then we should assign a *high* value to n.
- If the algorithm's complexity is *low* then we could assign a *high* value to n.
- If the algorithm's complexity is *high* then we should assign a *low* to n.
- If we need a *high* individual's precision then we should assign a *high* value to n.

[1] We refer to *temporal complexity*
[2] Logaritmic or polinomial
[3] Exponential

- If we need a *low* individual's precision then we should assign a *low* value to n.

In this work, we are dealing with a high complexity problem (NP-Problem) but the algorithms (mainly NSGA) that we have chosen have a high complexity too. In addition, the individual´s precision that we need must be as higher as possible.

In order to deal with the previous scenario, we have assigned the value of 100 to the parameter n. This decision causes an increment of the algorithm's complexity, but we are able to obtain a near-optimal solution without decreasing the solution's precision.

3.3 Evaluation of the population

3.3.1 Evaluation Function

In the previous section we had defined the main characteristics of every chromosome. One of such characteristics was the **interpretation** of the chromosome's alleles. Such interpretation is useful when we discriminate "well-behaved" from "non-well-behaved individuals" in the population.

In order to clarify the concept of *well-behaved* and *non-well-behaved*, we present the following definitions:

Definition 3.3. *A* **well-behaved** *individual is a chromosome which calculated value is greater that the rest of the chromosomes in the population.*

Definition 3.4. *A* **non-well-behaved individual** *is a chromosome which calculated value is lower than the rest of the chromosome in the population.*

Now that we have defined the well-behaved and non-well-behaved individuals, we will proceed to define the process to assign every individual to one group.

In order to discriminate individuals in the population, we have to define an **Evaluation Function**.

Definition 3.5. *An* **Evaluation Function** *is a relation between the integral interpretation of the chromosome and a numeric value.*

The method to assign a numeric value to every individual depends on the nature of the problem that we are analyzing. An Evaluation Function could be as simple as an arithmetic addition, or as complex as a double-integral function.

In this work we used a composed Evaluation Function. We say it is a composed Evaluation Function because we have used three sub-functions: **Degrees-to-Move function**, **Chromosome-Coordinates** and **Euclidean-Distance function**.

The first sub-function is used to calculate the total degrees that shoulder / elbow will move. This sub-function only perform a simple arithmetic addition of the chormosome's alleles.

The second sub-function calculates the effector's coordinates defined in the chromosome using the Pitagoras' Triangle Theorem.

The third sub-function will determine the distance between the goal effector coordinates and the coordnates defined in the chromosome.

Here we present these three sub-functions:

- **Degrees-to-Move Function**

  for-each ith-allele *in* Chromosome

  if ith-allele $\in \{Angles_{Shoulder}\}$

  $\theta_{Shoulder} = \theta_{Shoulder}+$ ith-allele

  if ith-allele $\in \{Angles_{Elbow}\}$

  $\theta_{Elbow} = \theta_{Elbow}+$ ith-allele

- **Chromosome-Coordinates Function**

$$X_{Chromosome} = (l_{Humerus} \bullet cos\,(\theta_{Shoulder})) + (l_{ForeArm} \bullet cos\,(\theta_{Shoulder} + \theta_{Elbow}))$$
$$Y_{Chromosome} = (l_{Humerus} \bullet sin\,(\theta_{Shoulder})) + (l_{ForeArm} \bullet sin\,(\theta_{Shoulder} + \theta_{Elbow}))$$

- **Euclidean Distance Function**

$$distance = \sqrt{(X_{Goal} - X_{Chromosome})^2 + (Y_{Goal} - Y_{Chromosome})^2}$$

In order to adapt this composed function to the characteristics of the problem, we have included the following constraints:

- The value for each individual must be in the rank of 0 and 180 degrees.
- There is a *upper bound* in the number of movements that can be performed. This *upper bound* is defined by the number of alleles contained in the chromosome.

In the following section we present the comparation between Evaluation Function and Fitness Value.

3.3.2 Evaluation Vs Fitness Value

First of all, we define the Fitness Function.

Definition 3.6. *A* **Fitness Function** *is a relation between the value obtained in the Evaluation Function and a numeric value defined by a* **classification criteria**.

In this work, the *classification criteria* that we have used is composed by two sub-criteria: **Ranking** and **Sharing** Procedures (see "Grouping the Individuals" sub-section).

Now that we have defined the concept of *Fitness Function*, we are able to talk about the differences between Fitness and Evaluation Values.

The Evaluation Value is obtained directly from the chromosome's alleles, i.e. it is a value that *depends* completly from every individual and is *unique* among the population.

In contrast, the Fitness Value could be shared by more than one individual, which means that *two or more individuals could be considered as equals in the* **discrimination process** (Ranking and Sharing procedures).

As we can see, the Fitness Function incorporates a "*grouping*" concept, i.e. in order to characterize every individual, we have to assign it a *group.*

In order to define how every individual is assigned to a group, in the following section we talk about "*Grouping the individuals*"

3.3.3 Grouping the Individuals

We define a group as:

- A set of elements that share one or more characteristics.
- A set of individuals that accomplish certain criteria.

Based in this second definition, we have used the criteria of "*domination*" [6], which is defined as follows:

Definition 3.7. *For a problem having more than one objective function (say, f_j, $j = 1,\ldots,M$ and $M > 1$), a solution $x^{(1)}$ is said to* **weakly** *dominate the other solution $x^{(2)}$, if both the following conditions are true.*

- *The solution $\vec{x}^{(1)}$ is no worse (say the operator $\prec$ denotes worse and $\succ$ denotes better) than $\vec{x}^{(2)}$ in all objectives, or $f_j\left(\vec{x}^{(1)}\right)$ not $\prec f_j\left(\vec{x}^{(2)}\right)$ for all $j = 1,2,\ldots,M$ objectives.*

- *The solution* $\vec{x}^{(1)}$ *is strictly better than* $\vec{x}^{(2)}$ *in at least one objective, or* $f_{\bar{j}}\left(\vec{x}^{(1)}\right) \succ f_{\bar{j}}\left(\vec{x}^{(2)}\right)$ *for at least one* $\bar{j} \in \{1, 2, \ldots, M\}$.

In the following, we describe the **Ranking** and **Sharing** procedures, where the concept of "*domination*" [6] will be used to classify the individuals.

Ranking Procedure

Consider a set of N population members, each having M (< 1) objective funcion values. The following procedure [6] can be used to find the non-dominated set of solution:

- **Step 0:** Begin with i=1.
- **Step 1:** For all j=1,...,N and $j \neq i$, compare solutions $\vec{x}^{(i)}$ and $\vec{x}^{(j)}$ for domination using two conditions for all M objectives.
- **Step 2:** If for any j, $\vec{x}^{(i)}$ is dominated by $\vec{x}^{(j)}$, mark $\vec{x}^{(i)}$ as *dominated.*
- **Step 3:** If all solutions (that is, when $i = N$ is reached) in the set are considered, Go to Step 4, else increment i by one and Go to Step 1.
- **Step 4:** All solutions that are not marked *dominated* are non-dominated solutions.

All these non-dominated solutions are assumed to consitute the first non-dominated front in the population. These solutions are temporarily ignored from the population and the above step-by-step procedure is applied again. The resulting non-dominated solutions are assumed to constitute the second non-dominated front. This procedure is continued until all population members are assigned a front.

Sharing Procedure

Given a set of n_k solutions in the *k-th* non-dominated front each having a dummy fitness value f_k, the sharing procedure [6] is perfomed in the following way for each solution $i = 1,2,\ldots,n_k$:

- **Step 1:** Compute a normilized Euclidean distance measure with another solution j in the *k-th* non-dominated front, as follows:

$$d_{ij} = \sqrt{\sum_{p=1}^{P} \left(\frac{x_p^{(i)} - x_p^{(j)}}{x_p^u - x_p^l} \right)^2}$$

  where P is the number of variables in the problem. The parameters x_p^u and x_p^l are the upper and lower bounds of variable x_p

- **Step 2:** This distance d_{ij} is compared with a pre-specified parameter σ_{share} and the following *sharing function* value is computed:

$$Sh\left(d_{ij}\right) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^2 & \text{if } d_{ij} \leq \sigma_{share} \\ 0 & \text{otherwise} \end{cases}$$

- **Step 3:** Increment j. If $j \leq n_k$, go to Step 1 and calculate $Sh\left(d_{ij}\right)$. If $j > n_k$ calculate niche count for i-th solution as follows:

$$m_i = \sum_{j=1}^{n_k} Sh\left(d_{ij}\right)$$

- **Step 4:** Degrade the dummy fitness f_k of i-th solution in the k-th non-domination front calculate the shared fitness, f_i', as follows:

$$f_i' = \frac{f_k}{m_i}$$

This procedure is continued for all $i = 1, 2, \ldots, n_k$ and a corresponding f_i' is found. Thereafter, the smallest value f_i^{min} of all f_i' in the k-th non-dominated front is found for further processing. The dummy fitness of the next non-dominated front is assigned to be $f_{k+1} = f_i^{min} - \epsilon_k$ where ϵ_k is a small positive number.

The above sharing procedure requires a pre-specified parameter σ_{share} which can be calculated as follows:

$$\sigma_{share} \approx \frac{0.5}{\sqrt[p]{q}},$$

where q is the desired number of distinct Pareto-optimal solutions. Although the calculation of σ_{share} depends on this parameter q, it has been shown elsewhere that the use of above equation with $q \approx 10$ works in many test problems. Moreover, the performance of NSGAs is not very sensitive to this parameter near σ_{share} values calculated using $q \approx 10$.

3.4 Population Evolution: Genetic Operators

3.4.1 Selection: Stochastic Remainder Proportionate Selection

Stochastic Remainder Proportionate Selection [8] is used with the fitness values, where a solution is selected as a parent in proportion to its fitness value. With such a operator, solutions of the first *non-dominated* front have higher probability of being a parent than solutions of other fronts. This is intended to search for non-dominated regions, which will finally lead to the *Pareto-optimal* front.

3.4.2 CrossOver and Mutation Operators: Polynomial Probability Distribution [5]

CrossOver Operator

Two parent solutions $\vec{x}^{(1)}$ and $\vec{x}^{(2)}$ obtained from selection operator are crossed with a probability $p_c = 0.9$. For a crossover, the solutions are crossed variable-by-variable to create two new children solutions $\vec{y}^{(1)}$ and $\vec{y}^{(2)}$. The children solutions are created by using a *Polynomial probability distribution.* Each variable is crossed with a probability of 0.5 using the following step-by-step procedure [6]:

- **Step 1:** Create a random number u between 0 and 1.
- **Step 2:** Calculate β_q as follows:

$$\beta_q = \begin{cases} (u\alpha)^{\frac{1}{\eta_c+1}}, & \text{if } u \leq \frac{1}{\alpha}, \\ \left(\frac{1}{2-u\alpha}\right)^{\frac{1}{\eta_c+1}}, & \text{otherwise,} \end{cases}$$

  where $\alpha = 2 - \beta^{-(\eta_c+1)}$ and β is calculated as follows:

$$\beta = 1 + \frac{2}{x_i^{(2)} - x_i^{(1)}} \min\left[\left(x_i^{(1)} - x_i^{l}\right) \ \left(x_i^{u} - x_i^{2}\right)\right]$$

  where x_i^l and x_i^l are the lower and upper bounds of parameter x_i. The parameter η_c is the distribution index and can take any non-negative value. A small value of η_c allows solutions far away from parents to be created as children solutions and a large value restricts only near-parent solutions to be created as children solutions. In the simulations we use $\eta_c = 30$.

- **Step 3:** The children solutions are then calculated as follows:

$$y_i^{(1)} = 0.5\left[\left(x_i^{(1)} + x_i^{(2)}\right) - \beta_q \left|x_i^{(2)} - x_i^{(1)}\right|\right], y_i^{(2)} = 0.5\left[\left(x_i^{(1)} + x_i^{(2)}\right) + \beta_q \left|x_i^{(2)} - x_i^{(1)}\right|\right]$$

Mutation Operator

A **Polynomial Probability Distribution** [5] is used to create a solution $z_i^{(j)}$ in the vicinity of a parent solution $y_i^{(j)}$ The following procedure is used for each variable with a probability p_m:

- **Step 1:** Create a random number u between 0 and 1.
- **Step 2:** Calculate the parameter δ_q as follows:

$$\delta_q = \begin{cases} \left[2u + (1-2u)(1-\delta)^{\eta_m+1}\right]^{\frac{1}{\eta_m+1}}, & \text{if } u \leq 0.5, \\ 1 - \left[2(1-u) + 2(u-0.5)(1-\delta)^{\eta_m+1}\right]^{\frac{1}{\eta_m+1}} & \text{otherwise,} \end{cases}$$

  where $\delta = min\left[\left(y_i^{(j)} - y_i^l\right) \left(y_i^u - y_i^{(j)}\right)\right] / \left(y_i^u - y_i^l\right)$. The parameter η_m is the distribution index for mutation and takes any non-negative value. We use $\eta_m = 100 + t$ (where t is the iteration number) here.

- **Step 3:** Calculate the mutated child as follows:

$$z_i^{(j)} = y_i^{(j)} + \delta_q \left(y_i^u - y_i^l\right)$$

  The mutation probability p_m is linearly varied from $1/P$ till 10, so that, on an average, one parameter gets mutated in the beginning and all parameters get mutated at the end of a simulation run.

3.5 Use of Learning Algorithm

3.5.1 Interaction between Learning and Planning Algorithm

General Description

Actual human entity's behavior is not completely computed in real time as planning algorithms, rather a human entity learns from experience. Thus, a human uses *learning* to compute an accurate behavior. This is the main idea of our proposed method which combines a **Learning Algorithm** and a **Dynamic Planning Algorithm** (GA or MOEA) to compute the behavior of an avatar's arm. The *Learning Algorithm* is used to learn plans; the *Dynamic Planning Algorithm* is used to reach the objective whenever an obstacle makes the *computed learning plan* fail.

Learning Algorithm

1. *Markov Decision Process*

   First of all we need to represent the task as a finite Markov decision process (finite MDP) with the following statements:

   - $A = \{RotateShoulder10+, RotateShoulder10-, RotateElbow10+, RotateElbow10-\}$, is the *actions set*,
   - $S = \{Shoulderstates\} \times \{Elbowstates\}$ is the *states set*, where:
     - $Shoulderstates = \{0°, 10°, 20°, ..., 180°\}$
     - $Elbowstates = \{0°, 10°, 20°, ..., 180°\}$
   - Then we have: $|A| = 4, |S| = 361$.

- $P(s, a, s') = S \times A \times S \longrightarrow [0, 1]$ is the *joint probability* of making a transition to state s' if action a is taken in state s,
- $R(s, a, s') = S \times A \times S \longrightarrow R$ is an *immediate reward* for making a transition from s to s' by action a,
- and $\gamma \in [0, 1)$ is the *discount factor* for future rewards.

2. *Q-learning algorithm*

We chose a well-known *Reinforcement Learning* algorithm such as *one step Q-learning*, defined by the following action-value function

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

with the values of α and γ set to 0.5, and the *action-selection* rule set to $\epsilon - greedy$.

The reward function **R** is set to the **Euclidean distance** between the *effector's current position* and the *goal position*. Finally, in order to deal with the "exploration vs. exploitation" dilemma, we initialize ϵ with the numeric value of 1.0 and it is decreased at the beginning of each episode. Therefore, the value of ϵ is changed in a way inversely proportional to the number of elapsed episodes in the algorithm's execution.

3. *Planning-Learning Algorithms Fusion*

The architecture proposed (Fig 3) is constituted by three main modules: **Planning Algorithm**, **Learning Algorithm** and **Simulation Module**. The *Planning Algorithm*, described at the beginning of this section, is based on a MOEA and it is responsible of generating the necessary plans to move the effector to the goal position. The *Simulation Module* is composed by two routines: *Plan Execution* and *Collision Simulation*. This Module interacts directly with the Planning Algorithm in order to obtain a virtual representation of the plan that such algorithm generates. The Learning Algorithm's core is

a Reinforcement Learning (**RL**)algorithm which produces a Knowledge Base (**KB**). This KB is required by the Planning Algorithm to create a plan.

The information flow is explained as follows:

(a) Learning Algorithm creates a Knowledge Base.

(b) Planning Algorithm consults the actions stored in KB and generates a plan.

(c) The plan is executed by Simulation Module.

(d) Simulation Module simulates a possible collision, if a "simulated collision" occurs then Planning Algorithm generates a new plan, if not then continues with the Plan Execution.

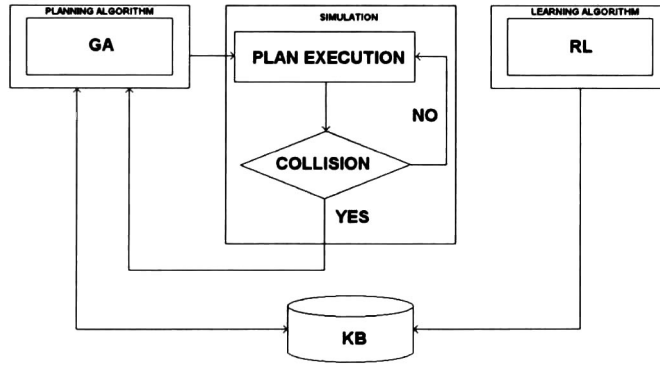


Figure 3.2: Planning - Learning Fusion with its respective modules.

3.6 Use Case: Avatar 's Arm

The study case that proposed consist of an avatar that must move one of his effectors from an initial position to a final position,(e.g. a baby who is learning how to carry his food from a

soup plate to his mouth). We focus on avatar's arm (Fig 3.6)which is composed by two bones: **Humerus** and **Forearm**; and these two bones are connected by two joints: **Shoulder** and **Elbow**. The possible actions that such arm can perform are *to increase or decrease the angle of each joint* ('shoulder' and/or 'elbow'). The joints'angles are bounded in the rank of (0°, 180°) in order to accomplish the minimal natural limitations of a human arm.

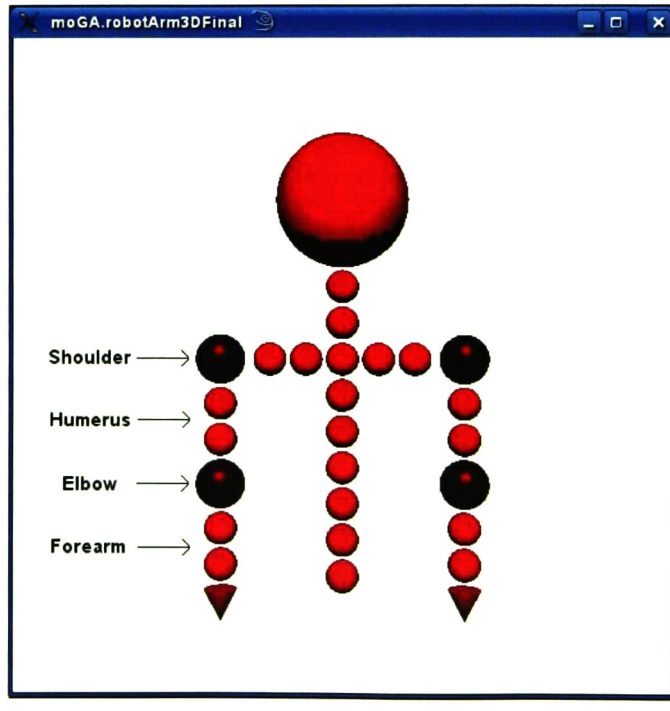


Figure 3.3: Description of Avatar's Model.

The arm starts its movement from *effector's initial position* (I_0) to get *effector's end position* (E_0) by using small plans (Planning Algorithm) which use the possible actions acquired by a previous knowledge (Learning Algorithm). Depending on effector's position, the effector's movement will be evaluated to consider whether is near enough to the end position or there are more possible actions to be performed. Such evaluation is based on the Euclidean Distance calculation.

3.7 Discussion

After we have presented the complete description of our proposal, we have to remember that our objective is to present an approach where we combine two techniques: *MOEA* and some type of *Learning technique*. We do not focused on prove that this is the best approach, we just present one element of the complete set of solutions that this problem could have.

In the following chapter we present the numeric results of our investigation, where we make a comparision between the implementation of a **Steady State Genetic Algorithm (ssGA)** and our implementation of NSGA, and at the same time we compare them with the a merged implementation, where we combine these two algorithms with a Learning technique.

Chapter 4

Implementation Results

4.1 Introduction

For this implementation we used a Notebook with a Pentium Centrino Processor at 1.5 GHz and 512 MB in RAM. The application runs over Linux Suse 10.2 and was implemented in Java.

The shoulder is fixed to permanent position by the time remaining components change their position. A variation in the Shoulder's angle represents a complete movement of the bones and the Elbow; a change in the Elbow's angle only represents a movement of the Forearm by the time the Humerus stays in the same position.

4.2 ssGA with Knowledge Vs ssGA

The graphic (Fig 4.2) shows the results obtained after executing 100 simulations with 100 random E_0 in order to observe the behavior of our Planning-Learning Fusion and comparing it with the single execution of Planning Algorithm.

The results show that complementing the Planning Algorithm with the Learning Algo-

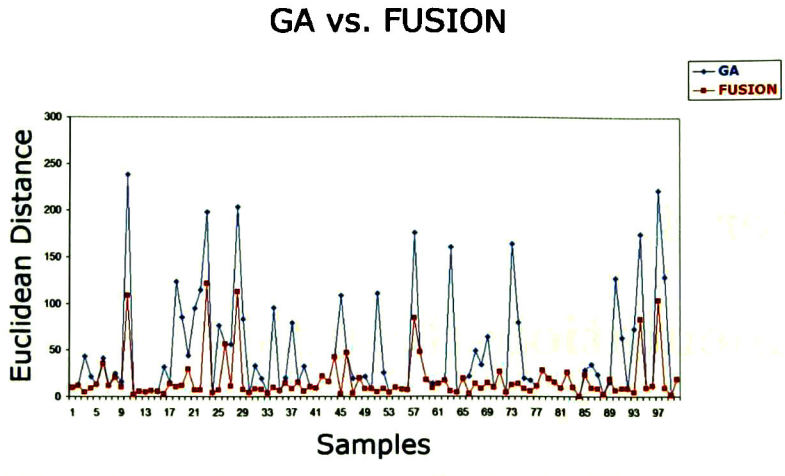


Figure 4.1: Genetic Algorithm Vs GA-Learning Fusion: Implementation Results.

rithm (Q-Learning in this case) decreases the Euclidean Distance between the E_0 and the near-optimal solution obtained by our algorithms fusion. The execution time in the Planning-Learning Fusion is beneath or equal (in the worst case) to the single Planning Algorithm execution time.

4.3 NSGA with Knowledge Vs NSGA

The graphic (Fig 4.3) shows the results obtained after executing 100 simulations with 100 random E_0 in order to observe the behavior of our Planning-Learning Fusion and comparing it with the single execution of Planning Algorithm, as we made in the last experiment, but now we change the algorithm to NSGA.

In this case, the results obtained vary a little bit from the results generated by the GA-implementation. First of all,we can observe that in some cases the solution generated by the Fusion implementation do not represent a reduction in the distance. This could be a not-desirable behavior, but if we look at the number and type of movements that the Avatar

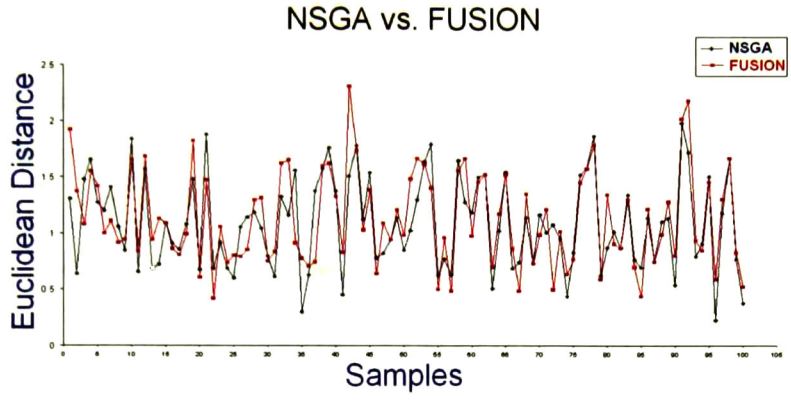


Figure 4.2: NSGA Vs NSGA-Learning Fusion: Implementation Results.

performs, we realize that the complete solution generated by the Fusion implementation is composed by big movements at the beginning, and small movements at the end. This represent a reduction in the time execution, which is the "*reward*" for a increase of the distance between the effector and E_0

4.4 Comparation of both Implentations

As we can see, the NSGA implementation represents a more complex method, mainly by the genetic operators and its classification procedures, but when we are dealing with a problem where the number of data to compute is considerably big, the time execution decrease considerably too, in contrast with a simple ssGA.

In both cases, the ssGa and NSGA obtain a considerable improvement in their results by using the Learning Algorithm, which means that the Fussion of Stochastic Methods and Learning Algorithms represent an acceptable option to solve problems where the space of possible solutions is considerably extensive.

4.5 Discussion

As we can see, our work represents an acceptable option to solve problems with a wide set of solutions, and the techniques that we present are already used in many subjects, and the main objective of our work is to present a Fussion of those technique.

In the following chapter, we present our main conclusions and the future work that this investigation could have eventually.

Chapter 5

Future Work and Conclusions

As we could see in the last chapters, our work is just a proposal to solve a defined problem: Avatar's Arm motion. And as every proposal, we just presented the initial stage of an investigation work that could be extended in many areas.

In the following sections we present the main areas where we think is possible to extend our work (Future Work Section) and also the more remarkable results that we obtained in this work (Conclusions).

5.1 Future Work

5.1.1 Use of Quaternions

In the Individual's representation we will obtain a more detailed model by using **quaternions**. The main reason to use quaternions is because of the nature of the problem where we have a 3D model linked to *Motion planning*.

The use of quaternions will also simplify the data processing task.

5.1.2 Implementing a more Detailed Translation Routine

In a future work, we could define a more structured proposal by improving the interpretation of the information that our algorithms share.

One possible solution could be the usage of **recursivity**. By using a *recursive interpretation* we could compress/decompress the information contained in every chromosome's allele.

5.1.3 Use of Collision Avoidance System (CAS)

In this work we presented a Collision detection system. In a future approach, we could include a more realistic scenario by adding a CAD to every object or agent that we need to trace.

5.1.4 Increase the Degrees-Of-Freedom (DOF)

In this work we used only two DOF: Shoulder and Elbow. In a future investigation we could add, at least, one DOF that would represent the *Wrist*. By adding this DOF, we would give to the Avatar's Arm a more realistic behavior.

5.2 Conclusions

In this work we have just presented one element from the complete set of solutions that Avatar's Arm Animation problem could have.Here we present the main conclusions of our work:

- First of all, we have presented a *NSGA* implementation, where there are a two classifica-

tion processes: Ranking and Sharing. Also, as we talked about Genetic Algorithms, we used Genetic Operator as: Selection, Crossover and Mutation.

- We defined an *Individual´s Representation*, where every allele represents a movement of the Avatar's Humerus or Forearm.
- In addition, we presented an *Evaluation Function*, which consist on Euclidean Distance function, so we are evaluating "how far from a *Goal position* the effector is"
- In this work, we have combined a Multi-Objective Evolutive Algorithm (MOEA) with a Reinforcement Learning Algorithm, particularly a Q-Learning Algorithm.
- Finally, we performed two Implementation Tests: simple Genetic Algorithm with / without Q-Learning , and NSGA with / without Q-Learning.
- The *Implementation Results* demostrate that even when simple Genetic Algorithms (GA)or NSGA with Q-Learning do not generate always the best solution (in numeric value), they represent an improvement in the Solution-Generation Time, because the movements generated by the Genetic and Learning Algorithm Fussion are more accurate than the ones generated by GA or NSGA alone.

Bibliography

[1] M. E. Aydin and T. C. Fogarty. Simulated annealing with evolutionary processes in job shop scheduling. *in Giannakoglou, K., Tsahalis, D., Periaux, J., Papailiou, K. and Fogarty, T. C. (eds.) Evolutionary Methods for Design, Optimisation and Control, (Proc. of EUROGEN 2001)*, September 2002.

[2] D.A. Van Veldhuizen C.A. Coello Coello and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer Academic/Plenum Pub, 2002.

[3] J. Xuan D. Chafekar and K. Rasheed. Constrained multi-objective optimization using steady state genetic algorithms. *Genetic and Evolutionary Computation -GECCO*, vol. 2723/2003, 2003.

[4] M. H. Overmars D. Nieuwenhuisen, A. Kamphuis. High quality navigation in computer games. *Science of Computer Programming*, March 2006.

[5] K. Deb and R.B. Agrawal. Simulated binary crossover for continous search space. *Complex Systems*, pages 115–148, 1995.

[6] Kalyanmoy Deb. Non-linear goal programming using multi-objective genetic algorithms, technical report no. ci-60/98. Master's thesis, Department of Computer Science/XI, University of Dortmund, Germany, Dortmund, Germany, 1998.

[7] M. Garber and M. C. Lin. Constraint-based motion planning using voronoi diagrams. *Proc. Fifth International Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2002.

[8] D.E. Goldberg. *Genetic algorithms for search, optimization, and machine learning.* Addison-Wesley, 1989.

[9] Z.M. Lu S.H. Sun H.M. Hang H.C. Huang, J.S. Pan. Vector quantization based on genetic simulated annealing. *Signal Processing*, pages 1513–1523, 2001.

[10] J.J. Lee I.K. Jeong. Adaptive simulated annealing genetic algorithm for system identification. *Engineering Applications of Artificial Intelligence*, pages 523–532, 1996.

[11] D. Manocha J. Cohen, M. Lin and M. Ponamgi. I-collide: An interactive and exact collision detection system for large-scale environments. *In Proc. Of ACM Interactive 3D Graphics Conference*, pages 189–196, 1995.

[12] P. Bessière E. Mazer J.M. Ahuactzin, E. Talbi. Using genetic algorithms for robot motion planning. *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 671–675, 1992.

[13] L. Jaillet J.P. van den Berg, D. Nieuwenhuisen and M. H. Overmars. Creating robust roadmaps for motion planning in changing environments. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1053–1059, August 2005.

[14] L. E. Kavraki and Jean Claude Latombe. Randomized preprocessing of configuration space for fast path planning. *IEEE Conference on Robotics and Automation*, pages 2138–2145, 1994.

[15] L. E. Kavraki and Jean Claude Latombe. Probabilistic roadmaps for robot path planning. *PAGE PROOFS for John Wiley and Sons Ltd.*, pages 22–49, February 1998.

[16] D.Z. Zheng L. Wang. An effective hybrid optimization strategy for jobshop scheduling problems. *Computers and Operations Research*, pages 585–596, 2001.

[17] M.C. Lin. Efficient collision detection for animation and robotics. Master's thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, California, 1993.

[18] J. Cohen et al. M. C. Lin, D. Manocha. Collision detection: Algorithms and applications. *Proceedings of the 2nd Workshop on Algorithmic Foundations of Robotics*, 1996.

[19] T. C. Fogarty M. Emin Aydin. A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application. *Journal of Intelligent Manufacturing*, Vol. 15(No. 6):805–814, December 2004.

[20] M. Rosendbluth A. Teller E. Teller N. Metropolis, A. Rosenbluth. J. chem. phys. *J. Chem. Phys.*, (No. 21):1087, 1953.

[21] M. C. Lin R. Gayle and D. Manocha. Constraint-based motion planning for deformable robots. *Proc. of IEEE International Conference on Robotics and Automation*, 2005.

[22] Khaled Rasheed. Gado: A genetic algorithm for continuous design optimization, technical report dcs-tr-352. Master's thesis, Department of Computer Science, The State University of New Jersey, New Brunswick, NJ, 1998.

[23] M. Lin S. Gottschalk and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. *In Proc. of ACM Siggraph '96*, 1996.

[24] Jr. M.P. Vecchi S. Kirkpatrick, C. D. Gelatt. Optimization by simulated annealing. *Science*, (No. 4598), May 1983.

[25] J.P. van den Berg and M. H. Overmars. Roadmap-based motion planning in dynamic environments. *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1598–1605, September 2004.

[26] J.P. van den Berg and M. H. Overmars. Prioritized motion planning for multiple robots. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, August 2005.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Animación del Brazo de un Avatar Usando Planeación Dinámica

del (la) C.

Angel Rafael RODRÍGUEZ MORENO

el dia 13 de Marzo de 2008.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González Pico
Investigador CINVESTAV 2A
CINVESTAV Unidad Guadalajara

Dr. Ricardo Vilalta López
Investigador CINVESTAV
CINVESTAV