MÉTODO DE EVALUACIÓN DINÁMICA DE PLANES EN SISTEMAS INTELIGENTES AUTÓNOMOS

Tesista

Esp. Lic. Ezequiel GONZÁLEZ

Directores

Dr. Ramón García-Martínez (UNLa) y Mg. Darío Rodríguez (UNLa)

TESIS DE MAESTRÍA

EN INGENIERÍA EN SISTEMAS DE INFORMACIÓN

ESCUELA DE POSGRADO

FACULTAD REGIONAL DE BUENOS AIRES

UNIVERSIDAD TECNOLÓGICA NACIONAL

Abril, 2015

DEDICATORIA

A mi esposa Rocio, por acompañarme,
por creer en mí y apoyarme incondicionalmente.

A mi hijo Teo, porque su existencia es mi fuente de inspiración.

AGRADECIMIENTOS

A la Vida, por darme la capacidad de asombro, de pensar, de investigar y de aprender.

A mis padres, por haber priorizado mi educación por sobre todas las cosas.

A mi padre José Manuel, por confiar en mí y ser mi principal patrocinador.

A mi madre Lidia, por apoyarme desinteresadamente en cualquier proyecto que decida emprender.

A mis hermanas Viviana y Fernanda, por su afecto y sus palabras de aliento.

A mi primo Eric, por haber estado presente durante este largo proceso y haberme dado ánimo cuando fue necesario.

A mis suegros Patricia y Rodolfo y a Wally, por su inmensa generosidad y colaboración.

A mis amigos Maxi, Fede y Pato, por permitirme compartir con ellos mi entusiasmo por la Inteligencia Artificial.

A mi director de Tesis Ramón García Martinez, por su calidad humana, sus consejos en pos de mi crecimiento profesional y su sentido del humor; y a mi codirector Darío Rodriguez, por su exhaustiva revisión del texto.

Al Dr. Jorge Ierache, por sus comentarios iniciales y por haberme facilitado parte del material informático utilizado al comienzo de la experimentación.

RESUMEN

La característica principal de los sistemas inteligentes autónomos es que son capaces de auto proponerse planes, de ejecutarlos y de retroalimentar su base de conocimiento a partir de la información que extraen del entorno. En esta tesis se presenta una revisión de los métodos de aprendizaje y planificación de dichos sistemas, para luego centrar la investigación sobre la arquitectura LOPE. Aunque publicaciones posteriores implementaron modificaciones y extensiones que lograron mejorar su rendimiento, se han identificado ciertos aspectos del modelo que aún no han sido abordados. Por tal motivo, se proponen mejoras para ser aplicadas dentro de los módulos de planificación y aprendizaje, como también refinamientos al proceso de control y ejecución. Además, se elabora un indicador que permite una evaluación integral de la arquitectura y un análisis comparativo de los resultados alcanzados con las distintas mejoras aplicadas, en comparación con el diseño original.

Palabras clave: sistemas inteligentes autónomos, aprendizaje por interacción con el entorno, exploración, planificación, formación y revisión de teorías, aprendizaje por refuerzo, necesidades y motivación.

ABSTRACT

The main feature of autonomous intelligent systems is their ability to self-propose plans, to execute them and to provide feedback to its knowledge base from the information that they extract from the environment. In this thesis, we present a review of the learning and planning methods of the above mentioned systems, in order to focus the research on the LOPE architecture later on. Although subsequent researches implemented modifications and extensions which got to improve its performance, we have identified certain aspects of the model that have not yet been addressed. Therefore, several improvements have been proposed in order to be applied within the planning and learning modules, as well as refinements to the control and execution process. Furthermore, we elaborate an indicator that allows us to get a global evaluation of the architecture and to perform a comparative analysis between the results obtained through each of the implemented enhancements, compared to the original design.

Key words: autonomous intelligent systems, environment-interaction based learning, exploration, planning, theory creation and revision, reinforcement learning, needs and motivation.

ÍNDICE

1. INTRODUCCIÓN 1
1.1 OBJETIVO DE LA TESIS 1
1.2 VISIÓN GENERAL DE LA TESIS 1
1.3 AGENTES INTELIGENTES 2
1.3.1 Inteligencia y aprendizaje 4
1.3.2 Características del entorno 5
1.4 SISTEMA INTELIGENTE AUTÓNOMO 6
1.5 LA JERARQUÍA DE NECESIDADES DE MASLOW 6

2. ESTADO DE LA CUESTIÓN 9
2.1 APRENDIZAJE AUTOMÁTICO 9
2.1.1 Taxonomía del aprendizaje 9
2.1.2 Aprendizaje por observación y descubrimiento 10
2.1.3 Aprendizaje por interacción con el entorno 10
2.1.4 Exploración 11
2.1.5 Aprendizaje por refuerzo 11
2.2 ARQUITECTURAS CON CONOCIMIENTO INCORPORADO POR EL PROGRAMADOR Y AJUSTADO POR EL SISTEMA 13
2.2.1 Sistema de Hayes-Roth 13
2.2.2 PRODIGY/EXPERIMENT 14
2.3 ARQUITECTURAS CON CONOCIMIENTO INCORPORADO POR EL SISTEMA 17
2.3.1 Sistema Inteligente Autónomo 17
2.3.2 LIVE 19
2.3.3 Sistema de Christiansen 21
2.4 EL MODELO LOPE 22
2.5 EXTENSIONES O MODIFICACIONES AL MODELO LOPE 26
2.5.1 SIA con aprendizaje basado en intercambio de operadores 26
2.5.2 SIA con ciclo de vida de aprendizaje 28
2.5.3 Método de ponderación basado en la productoria de probabilidad de éxito de acciones 29
2.5.4 Algoritmos genéticos aplicados a los SIA 30

3. DESCRIPCIÓN DEL PROBLEMA 33
3.1 EL CONTEXTO DEL MODELO LOPE 33
3.2 OCURRENCIA DE GIROS INEFICIENTES 34
3.3 GESTIÓN INCOMPLETA DE TEORÍAS MUTANTES 35
3.4 FALTA DE EVALUACIÓN DE LOS PLANES EJECUTADOS 36
3.5 ÍNDICE DE CONFIABILIDAD ESTÁTICO 36

4. SOLUCIÓN 39
4.1 CONTEXTO DE LAS SOLUCIONES ABORDADAS 39
4.2 IMPLEMENTACIÓN DEL SISTEMA SENSOR 40
4.3 PREVENCIÓN DE GIROS INEFICIENTES 40
4.3.1 Arquitectura propuesta 41
4.3.2 Algoritmos 43
4.4 ADMINISTRACIÓN COMPLETA DE TEORÍAS MUTANTES 46
4.4.1 Arquitectura propuesta 46
4.4.2 Algoritmos 48
4.4.3 Ejemplo integrador 51
4.5 CASTIGO A LAS TEORÍAS DE LOS PLANES ABORTADOS 53
4.5.1 Arquitectura propuesta 53
4.5.2 Algoritmos 56
4.5.3 Ejemplo integrador 58
4.6 ÍNDICE DE CONFIABILIDAD DINÁMICO 60
4.6.1 Arquitectura propuesta 62
4.6.2 Algoritmos 63
4.6.3 Ejemplo integrador 65
4.7 ÍNDICE DE MEJORA 68

5. EXPERIMENTOS 71
5.1 INTRODUCCIÓN 71
5.2 DISEÑO EXPERIMENTAL 71
5.2.1 Simulaciones 71
5.2.2 Escenarios 72
5.2.3 Variables independientes 74
5.2.4 Variables dependientes 75
5.2.5 Casos a evaluar 77

5.3 GRÁFICAS Y SU INTERPRETACIÓN 77
5.3.1 Puntos de energía, utilidad y descubrimiento del escenario 78
5.3.2 Situaciones y teorías 82
5.3.3 Planificación 83
5.3.4 Matriz de valores finales promedio 89
5.4 ÍNDICE DE MEJORA 90

6. CONCLUSIONES 93
6.1 RESULTADOS FINALES Y CONCLUSIONES 93
6.2 FUTURAS LÍNEAS DE INVESTIGACIÓN 96

7. REFERENCIAS 97

ANEXO 101

ÍNDICE DE FIGURAS

Figura 1.1	Agente que interactúa con el medioambiente	3
Figura 1.2	La pirámide de Maslow: la jerarquía de necesidades humanas	7
Figura 2.1	Dinámica del aprendizaje por refuerzo	12
Figura 2.2	Arquitectura del sistema propuesto por Hayes-Roth	13
Figura 2.3	Arquitectura del sistema inteligente autónomo	18
Figura 2.4	Arquitectura del modelo LOPE	23
Figura 2.5	El robot explorador y su sistema sensor	23
Figura 2.6	Ciclo de percepción-aprendizaje-planificación-ejecución del modelo LOPE	25
Figura 2.7	Arquitectura del SIA con intercambio de teorías	27
Figura 2.8	Ciclo de vida de aprendizaje del sistema inteligente autónomo LOPE-LLC	29
Figura 2.9	Arquitectura del SIA con algoritmos genéticos	31
Figura 4.1	Nuevo sistema sensor del robot explorador	40
Figura 4.2	Diagrama de proceso: mejora de la prevención de giros ineficientes	42
Figura 4.3	Algoritmo que obtiene las acciones habilitadas	43
Figura 4.4	Algoritmo que filtra los posibles planes del árbol	45
Figura 4.5	Diagrama de proceso: extensión de la administración de teorías mutantes	47
Figura 4.6	Algoritmo de alto nivel para la administración de las teorías mutantes	49
Figura 4.7	Algoritmo que obtiene las teorías mixtas	50
Figura 4.8	Diagrama de proceso: extensión del castigo a la teoría que fracasó	54
Figura 4.9	Algoritmo de alto nivel del castigo a la teoría del plan abortado	56
Figura 4.10	Algoritmo que premia a una determinada teoría	57
Figura 4.11	Algoritmo que incrementa el K de una teoría similar	57
Figura 4.12	Árbol de situaciones asociado	58
Figura 4.13	Robot en escenario	59
Figura 4.14	Diagrama de proceso: extensión del índice de confiabilidad dinámico	62
Figura 4.15	Algoritmo que actualiza el vector de planes ejecutados	64
Figura 4.16	Algoritmo que actualiza el índice de fracaso	64
Figura 4.17	Algoritmo que actualiza el índice de confiabilidad	65
Figura 4.18	Jerarquía de necesidades propuesta para el robot explorador	69

Figura 5.1	Escenario 1	73
Figura 5.2	Escenario 2	73
Figura 5.3	Escenario 3	73
Figura 5.4	Porcentaje de recorrido por puntos de energía vs. tiempo	78
Figura 5.5	Porcentaje de puntos de energía alcanzados vs. tiempo	79
Figura 5.6	Utilidad promedio vs. tiempo	80
Figura 5.7	Porcentaje de descubrimiento del escenario vs. tiempo	81
Figura 5.8	Cantidad de situaciones percibidas vs. tiempo	82
Figura 5.9	Cantidad de teorías creadas vs. tiempo	83
Figura 5.10	Porcentaje de planes exitosos vs. tiempo	84
Figura 5.11	Porcentaje de planes fracasados vs. tiempo	85
Figura 5.12	Porcentaje de planes de contingencia vs. tiempo	86
Figura 5.13	Longitud promedio de planes exitosos vs. tiempo	87
Figura 5.14	Longitud promedio de planes fracasados vs. tiempo	88
Figura 5.15	Porcentaje promedio de pasos exitosos en planes fracasados vs. tiempo	88
Figura A.1	Recorrido promedio por c/u de las posiciones de c/ escenario para cada caso	101

ÍNDICE DE TABLAS

Tabla 2.1	Casos de refinamiento de EXPERIMENT	16
Tabla 4.1	Las necesidades del robot y las dimensiones y ponderaciones del índice	70
Tabla 5.1	Detalle de las simulaciones ejecutadas para cada caso	72
Tabla 5.2	Mejoras implementadas en cada uno de los casos	77
Tabla 5.3	Matriz de los valores finales promedio para cada variable y cada caso	89
Tabla 5.4	Índice de mejora para cada uno de los casos evaluados	90

NOMENCLATURA

AC	Acción
AG	Algoritmo genético
ATM	Administración de teorías mutantes en planificación (mejora)
AV	Avanzar (acción del robot)
BIO	Operadores integrados (por sus siglas en inglés "Built-In Operators")
CST	Castigo a la teoría que provocó el fracaso del plan (mejora)
DES	Porcentaje de descubrimiento del escenario
EBL	Aprendizaje basado en explicaciones (por sus siglas en inglés "Explanation-based Learning")
GD	Girar a la derecha (acción del robot)
GI	Girar a la izquierda (acción del robot)
IC	Índice de confiabilidad
ICD	Índice de confiabilidad dinámico (mejora)
IF	Índice de fracaso
IM	Índice de mejora
K	Parámetro (K) de la teoría
LA	Límite de aceptabilidad
LGP	Longitud promedio de planes
LOPE	Modelo de aprendizaje a partir de la observación en ambientes de planificación (por sus siglas en inglés "Learning by Observation in Planning Environments")
LOPE-LLC	Modelo LOPE con ciclo de vida de aprendizaje (por sus siglas en inglés "Learning by Observation in Planning Environments – Learning Life Cycle")
OR	Orientación (N, S, E, O)
P	Parámetro (P) de la teoría
PGI	Prevención de giros ineficientes (mejora)
PI	Posición inicial
RNP	Resultado neto de planes
RPE	Porcentaje de recorridos por puntos de energía
SF	Situación final
SI	Situación inicial
SIA	Sistema inteligente autónomo

SFM	Situación mutante que surge de la intersección de dos situaciones finales
TBO	Operadores basados en entrenamiento (por sus siglas en inglés "Trained Based Operators")
TEO	Cantidad de teorías creadas
TP	Teoría utilizada por el plan
TS	Teoría armada por los sensores
U	Utilidad de la teoría
VPE	Vector de planes ejecutados
WIO	Operadores de interacción global (por sus siglas en inglés "World Interaction Operators")

1. INTRODUCCIÓN

En este capítulo se presenta el objetivo (sección 1.1) y la visión general (sección 1.2) de la tesis; algunas definiciones básicas sobre los agentes inteligentes (sección 1.3); una breve presentación de los sistemas inteligentes autónomos (sección 1.4); y una introducción a la jerarquía de necesidades de Maslow (sección 1.5).

1.1 OBJETIVO DE LA TESIS

El propósito de esta tesis es construir un método de evaluación dinámica de planes para sistemas inteligentes autónomos, particularmente para el modelo LOPE [García-Martinez y Borrajo, 1997; 2000], sistema cuyo aprendizaje se basa en la formación y ponderación de teorías. El método elaborado incluye mejoras en los módulos de planificación y aprendizaje principalmente, aunque también se refinan algunas tareas dentro del proceso de control y ejecución. La propuesta incluye, además, la construcción de un indicador que posibilita una evaluación integral de las distintas mejoras implementadas al modelo, como así también, la comparación de los resultados alcanzados por cada uno de los casos definidos.

El modelo LOPE ha servido como arquitectura base para una serie de publicaciones que han implementado modificaciones o extensiones al diseño original. A pesar de que cada una de estas propuestas ha conseguido mejorar su rendimiento, se han identificado ciertos aspectos del modelo que aún no han sido abordados, como también algunos procesos que pueden ser refinados. El tratamiento de estos temas es el objetivo del presente trabajo.

1.2 VISIÓN GENERAL DE LA TESIS

El contenido de la tesis se estructura del siguiente modo. En el capítulo 1 se introduce el concepto de *agente inteligente* (sección 1.3), se da una primera definición de *sistema inteligente autónomo* (sección 1.4) y se presenta la jerarquía de necesidades de Maslow (sección 1.5).

En el capítulo 2 se elabora una revisión de los métodos de aprendizaje y planificación de aquellos sistemas que representan su entorno a partir de la creación de teorías. Se presenta una taxonomía del aprendizaje y luego se detallan los métodos de *aprendizaje automático* más relevantes (sección 2.1). A continuación se describen algunos sistemas en los que el conocimiento del dominio es incorporado por el programador y ajustado por el sistema (sección 2.2); se presentan tres arquitecturas en las que el conocimiento del dominio es incorporado directamente por el sistema (sección 2.3); se describe el

modelo LOPE en detalle (sección 2.4); y se especifican todas las modificaciones o extensiones que se le han aplicado a dicho modelo en el transcurso de los últimos años (sección 2.5).

El capítulo 3 describe las oportunidades de mejora identificadas. La primera de ellas trata sobre algunos casos específicos en los que se suceden acciones ineficientes que podrían ser evitadas (sección 3.2); la segunda está relacionada con una administración incompleta de las teorías mutantes (sección 3.3); la tercera trata sobre la falta de evaluación de los planes que han sido ejecutados (sección 3.4); y por último, la cuarta mejora se refiere a la rigidez del índice que mide la calidad de los planes, característica que dificultaría la correcta adaptación a la dinámica del sistema (sección 3.5).

En el capítulo 4 se presenta la solución del trabajo. Aquí se describe la pequeña modificación del sistema sensor a utilizar (sección 4.2); se detalla la solución propuesta para cada una de las debilidades encontradas (sección 4.3 a 4.6); y se presenta un indicador (sección 4.7), el *Índice de Mejora*, el cual permite un análisis comparativo completo e integral de los resultados de la experimentación.

El capítulo 5 contiene los resultados de los experimentos. En él se detallan las características de las simulaciones, la configuración de los escenarios, las variables utilizadas y los distintos casos que se evalúan (sección 5.2). Luego se presentan los resultados con sus gráficas e interpretaciones correspondientes y la matriz de valores finales promedio (sección 5.3). Al final del capítulo se incluye el índice de mejora calculado para cada caso (sección 5.4).

Por último, en el capítulo 6 se presentan los resultados finales y las conclusiones de la tesis (sección 6.1), y las futuras líneas de investigación (sección 6.2).

1.3 AGENTES INTELIGENTES

A pesar de que no hay una definición universalmente aceptada para el concepto de *agente*, sí existe un consenso general acerca de una característica fundamental que debe presentar, la *autonomía* [Wooldridge, 2011]. De acuerdo con dicho autor, la definición más adecuada sería la siguiente:

> "Un agente es un sistema informático que interactúa en un determinado medioambiente y que es capaz de llevar adelante acciones autónomas en dicho medio, con el fin de cumplir sus objetivos"

El concepto de autonomía se refiere a la capacidad de decidir qué acciones tomar para la consecución de los objetivos, y está estrechamente vinculado a la información almacenada por el agente. Es decir, para decidir qué acciones tomar, el agente debe contar con una base de conocimiento que le permita

decidir sobre la secuencia de acciones a ejecutar. Esta información puede ser proporcionada por el diseñador, puede ser aprendida por el propio agente o puede ser una mezcla de ambas.
Otra definición [Russell y Norvig, 2004], que involucra algunos términos específicos de la inteligencia artificial, es la siguiente:

> "Un agente es cualquier unidad capaz de percibir su entorno a través de *sensores* y actuar en ese medio utilizando *actuadores*"

Un agente humano, por ejemplo, posee cinco sentidos para percibir (la vista, el oído, el olfato, el tacto y el gusto) y manos, piernas y boca para actuar. En el caso de un agente de software, éste recibe archivos, paquetes vía red, entradas a través del teclado, etc. y actúa sobre el medio mostrando mensajes en el monitor, creando archivos o enviando paquetes. La figura 1.1 ilustra esta simple idea.

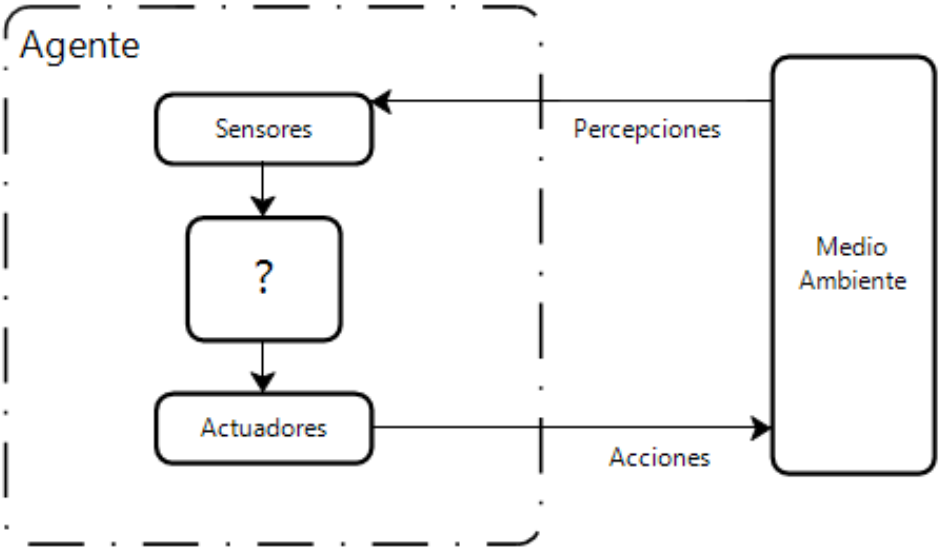

Figura 1.1 – Agente que interactúa con el medioambiente a través de sus sensores y actuadores (adaptado de [Russell y Norvig, 2004])

Parte de la dificultad en tratar de dar una definición general para el concepto de agente radica en que en la mayoría de los casos se le suele agregar atributos que sólo aplican al dominio particular en el que se está trabajando. Por ejemplo, para muchas aplicaciones, que el agente tenga la capacidad de aprender de sus experiencias es de vital importancia, sin embargo, existen situaciones en las que no sólo es indistinto sino que tampoco es deseable (ej.: sistema de control de tránsito aéreo) [Russell y Norvig, 2004].

1.3.1 Inteligencia y Aprendizaje

Ahora bien, ¿qué característica tiene que presentar un agente para ser considerado inteligente? Un agente *racional* o *inteligente* es aquel que hace lo correcto, y 'hacer lo correcto' es aquello que le es más conveniente, o en términos más concretos, aquello que le permite obtener el mejor resultado [Russell y Norvig, 2004]. Sin embargo, para evaluar si un resultado es mejor o peor que otro es necesario una medida de rendimiento, pero la dificultad radica en que no hay una única medida de rendimiento óptima para todos los agentes. Si se otorga la responsabilidad de la evaluación de sus acciones al mismo agente, muchos de ellos serían incapaces de responder o simplemente se engañarían a sí mismos. Por tal motivo, es importante la insistencia en medidas de rendimiento objetivas, que normalmente serán determinadas por el diseñador del sistema.
En [Russell y Norvig, 2004] se definen cuatros factores a analizar, que son necesarios para poder establecer si una acción ejecutada fue inteligente o no. Ellos son:

- la medida de rendimiento que define el criterio de éxito;
- el conocimiento del medio acumulado por el agente;
- las posibles acciones que el agente puede llevar a cabo;
- la secuencia de percepciones del agente hasta ese momento.

Teniendo en cuenta estos factores y lo descrito previamente, una definición adecuada y completa [Russell y Norvig, 2004] sería la siguiente:

> "Un *agente inteligente* es aquel que emprenderá aquella acción que supuestamente maximice su medida de rendimiento, basándose para ello, en las evidencias aportadas por la secuencia de percepciones y en el conocimiento que el agente mantiene almacenado".

Esta definición no supone omnisciencia (que el agente conozca el resultado de cada posible acción) ya que la acción racional depende del conocimiento disponible al momento de la toma de decisión, pero sí implica que el agente aprenda lo máximo posible de la información recopilada para evitar incurrir en un mal hábito a futuro. Es decir, un agente inteligente es aquel que, en la medida que van incrementándose las interacciones con el medioambiente, logra apoyarse más en el conocimiento surgido a partir de sus propias percepciones que en el conocimiento inicial. Dicha cualidad es un requisito fundamental para ser llamado agente inteligente, como también lo es que aprenda a compensar el conocimiento incompleto o parcial inicial cuando la toma de decisiones así lo requiera.
El modelo LOPE [García-Martinez y Borrajo, 1997; 2000], que se introduce en el siguiente capítulo, presenta un comportamiento inteligente y autónomo, ya que se ubica en la categoría de los sistemas

cuyo conocimiento del dominio es incorporado directamente por el sistema, es decir, que aprenden desde cero.

1.3.2 Características del Entorno

Al momento de diseñar un agente inteligente autónomo es imprescindible conocer las propiedades del entorno con el cual tendrá que interactuar. En [Russell y Norvig, 2004] se postulan varias categorías para definir el medioambiente, a saber:

- *Totalmente observable* vs. *Parcialmente observable*: si los sensores del agente le proporcionan un conocimiento completo del medio, entonces se dice que el entorno es totalmente observable. Es decir, para que sea totalmente observable los sensores tendrán que poder adquirir todos los aspectos relevantes para la toma de decisiones. Los entornos parcialmente observables pueden serlo debido al ruido de la información o a la baja calidad de los sensores;

- *Determinista* vs. *Estocástico*: se dice que el entorno es determinista si el siguiente estado del medio está totalmente determinado por el estado actual y la acción a ser ejecutada por el agente. En caso contrario se lo llama estocástico;

- *Estático* vs. *Dinámico*: si el medio puede cambiar mientras el agente está en el proceso de planificación, entonces el entorno es dinámico; de lo contrario, se lo define como estático. Los medios estáticos son más fáciles de tratar ya que el agente no necesita estar pendiente del medioambiente mientras está tomando una decisión;

- *Discreto* vs. *Continuo*: un ambiente es discreto si hay una cantidad finita y fija de acciones y percepciones relacionadas con él.

El caso más complejo es el de un entorno de trabajo *parcialmente observable*, *estocástico*, *dinámico* y *continuo* y, de hecho, la mayoría de las situaciones reales son de este modo. Dada la complejidad que implica un medioambiente de este tipo, los diseñadores de sistemas generalmente intentan resolver el problema que los atañe a partir de una especificación del medio lo más simple posible. En el caso del modelo LOPE, su arquitectura supone un entorno *estático*, que es *parcialmente observado* por el sistema y que es representado de manera *discreta* y *estocástica*.

1.4 SISTEMA INTELIGENTE AUTÓNOMO

De acuerdo a lo descrito en la sección previa y según lo definido en [Fritz *et al.*, 1989], un *sistema inteligente autónomo* es aquel que posee las siguientes características:

i) percibe su entorno a través de sus sensores y lo conceptualiza;
ii) actúa en el medio utilizando sus actuadores y lo hace en función de sus objetivos;
iii) sus objetivos se basan en la medida de rendimiento predefinida;
iv) posee una base de conocimiento sobre el dominio almacenada en memoria, la cual es principalmente aportada por el propio sistema;
v) planifica sus acciones teniendo en cuenta la situación percibida, sus objetivos y las experiencias almacenadas;
vi) ejecuta las acciones seleccionadas y se retroalimenta a partir de la observación de las situaciones resultantes, alcanzadas a partir de la ejecución de las acciones.

La arquitectura LOPE cumple con estos requisitos. Dicho modelo es un sistema inteligente autónomo que integra los procesos de percepción, aprendizaje, planificación y ejecución en un ciclo cerrado. El mismo puede ser descrito como un robot explorador que percibe su entorno, a partir de lo percibido registra una situación y arma una teoría local, planifica las acciones que llevará a cabo, ejecuta las acciones sobre el entorno y aprende a partir de su interacción con él.

1.5 LA JERARQUÍA DE NECESIDADES DE MASLOW

Uno de los objetivos de la presente tesis es el de diseñar y construir un indicador que permita la evaluación de las distintas mejoras a implementar, para así poder comparar sus resultados y tomar una decisión acerca de cuál de ellas es la que genera el mayor rendimiento. A diferencia de los resultados que se obtienen a través de las variables individuales, la ventaja que presenta el indicador (compuesto por varias variables) es que permite dar una respuesta clara y objetiva en aquellas situaciones en que resulta dificultoso tomar una decisión acerca de cuál es la implementación que genera el mejor resultado. Esto suele suceder cuando, al evaluar dos casos distintos, por ejemplo, uno de ellos presenta un valor muy alto en alguna de sus variables y muy bajo en otras y el otro caso presenta la situación contraria. En estas circunstancias, la presencia de un indicador termina resolviendo el conflicto ya que para cada uno de los casos evaluados se obtiene un valor único, a partir del cual se facilita la toma de decisiones.

Ahora bien, para la definición de un indicador son necesarios dos pasos, como mínimo. El primero de ellos es definir las variables que serán incluidas, y el segundo, definir la prioridad que se le dará a cada una ellas. Sin embargo, ambas definiciones implican la elección de un criterio previo por parte del diseñador, a partir del cual se pueda contestar a las siguientes preguntas: ¿por qué se han elegido esas variables y no otras? y ¿por qué algunas tienen mayor ponderación que otras? En este trabajo, el criterio adoptado se fundamenta en la teoría de necesidades humanas de Abraham Maslow.

De acuerdo con [Maslow, 1943], los seres humanos tienen una serie de necesidades, que van desde las más básicas a las más elevadas, cuya satisfacción es lo que motiva su comportamiento. Las características generales de su teoría son:

i) a medida que se van satisfaciendo las necesidades de orden inferior comienzan a desarrollarse necesidades más elevadas;

ii) sólo se atienden las necesidades superiores cuando se han satisfecho las necesidades inferiores;

iii) sólo las necesidades no satisfechas influyen en la conducta.

Maslow resume la jerarquía de las necesidades humanas de acuerdo a la figura 1.2.

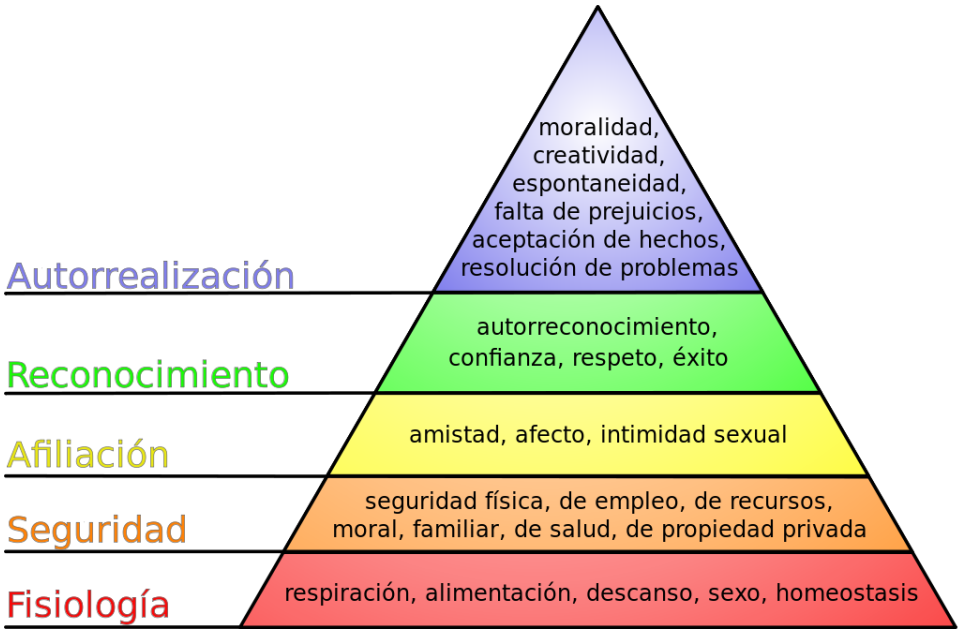

Figura 1.2 – La Pirámide de Maslow: la Jerarquía de necesidades humanas [Wikipedia, 2015]

En la base de la pirámide se encuentran las necesidades básicas o fisiológicas; un escalón arriba se ubican las necesidades de seguridad y protección; en el tercer nivel se encuentran las necesidades sociales o de afiliación; en cuarto lugar se sitúan las necesidades de estima o reconocimiento; y en la

cima de la jerarquía se hallan las necesidades de autorrealización. La figura incluye ejemplos para cada uno de los niveles.

El aporte que la teoría de Maslow ofrece a la tesis, es que ella puede ser utilizada de forma análoga para definir la jerarquía de necesidades del robot explorador. Y de acuerdo a las necesidades que hayan sido identificadas, se seleccionan las variables que permitan medir la satisfacción de ellas. Respecto a la ponderación que se le asigne a cada una de las variables, su valor debe ser acorde a la posición que su necesidad asociada ocupa dentro de la jerarquía. La elaboración de la jerarquía de necesidades para el caso del robot explorador se desarrolla en la sección 4.7.

2. ESTADO DE LA CUESTION

En este capítulo se incluye una breve introducción al aprendizaje automático (sección 2.1); se presenta una reseña de los sistemas cuyo conocimiento del dominio es incorporado por el programador y ajustado por el sistema (sección 2.2); y de aquellos sistemas en los que el conocimiento del dominio es incorporado enteramente por el sistema (sección 2.3). Luego se describe en detalle el modelo LOPE (sección 2.4) y, por último, se presentan las extensiones o modificaciones implementadas posteriormente a dicho modelo (sección 2.5).

2.1 APRENDIZAJE AUTOMÁTICO

En la presente sección se exhibe una taxonomía del aprendizaje (apartado 2.1.1), se describe el aprendizaje por observación y descubrimiento (apartado 2.1.2) y el aprendizaje por interacción con el entorno (apartado 2.1.3). Por último, se presenta el concepto de exploración (apartado 2.1.4) y el de aprendizaje por refuerzo (apartado 2.1.5).

2.1.1 Taxonomía del Aprendizaje

Con base en la reseña formulada en la tesis doctoral del Dr. Ramón García-Martinez [1997], se presenta una posible taxonomía sobre el aprendizaje [Carbonell *et al.*, 1983; Michalski, 1986; Kodratoff, 1988; Michalski y Kodratoff, 1990]:

- Aprendizaje por memorización;
- Aprendizaje por instrucción;
- Aprendizaje por deducción;
- Aprendizaje por analogía;
- Aprendizaje por inducción:
    - Aprendizaje por ejemplos;
    - Aprendizaje por observación-descubrimiento:
        - Observación pasiva;
        - Experimentación activa.

En este trabajo se obviarán los detalles relacionados a las primeras cuatro categorías y se focalizará en el aprendizaje por inducción. El aprendizaje inductivo tiene como objetivo obtener nuevo conocimiento a partir de hechos u observaciones. Es decir, el nuevo conocimiento que se genera no

debe estar implícitamente dentro del conocimiento disponible. Se pueden mencionar dos tipos de aprendizaje inductivo: (i) aprendizaje por ejemplos y (ii) aprendizaje por observación y descubrimiento. En el próximo apartado se describe con mayor detalle éste último tipo de aprendizaje.

2.1.2 Aprendizaje por Observación y Descubrimiento

El aprendizaje por observación y descubrimiento también recibe el nombre de *aprendizaje no supervisado*. Es una forma generalizada de aprendizaje inductivo que incluye sistemas de descubrimiento, tareas de formación de teorías o creación de criterios de clasificación en jerarquías taxonómicas.

El aprendizaje no supervisado requiere que el agente aprendiz realice más inferencias que en cualquier otro tipo de aprendizaje descrito. No recibe instancias específicas de un concepto particular, sino que debe clasificar la entrada de la información para formar conceptos. Es decir, las observaciones contienen conceptos que deben ser adquiridos por parte del agente.

Este tipo de aprendizaje puede clasificarse en términos de la interacción con el entorno. Los puntos extremos de esta clasificación son:

- Observación Pasiva: donde el aprendiz clasifica y taxonomiza las observaciones de los múltiples aspectos del entorno;
- Experimentación Activa: donde el aprendiz perturba el entorno y observa los resultados de esa perturbación. Las perturbaciones pueden ser aleatorias, guiadas por un interés específico o guiadas por restricciones.

2.1.3 Aprendizaje por Interacción con el Entorno

De acuerdo con [García-Martinez, 1997], la cantidad de tiempo requerido para programar el equivalente de la inteligencia humana es prohibitivamente alto. Un enfoque alternativo [Bock, 1985] apunta a que los sistemas aprendan a resolver problemas a través de un proceso iterativo de interacción con el entorno basado en ensayo y error, de la misma manera que actúan los humanos.

Para resolver un problema representado por un estado determinado del entorno, el sistema genera una respuesta basada en su experiencia. El entorno evalúa la corrección de la respuesta y envía esta evaluación al sistema. El proceso previo se repite hasta que la respuesta del sistema converge a la respuesta correcta.

Existen dos tipos de sistemas que incorporan conocimiento a través de su interacción con el entorno y que se diferencian en el origen de la información: aquellos donde el conocimiento sobre el dominio

es incorporado por el programador y ajustado por el sistema; y aquellos donde el conocimiento sobre el dominio es incorporado por el sistema. En la sección 2.2 se presentan dos arquitecturas pertenecientes al primer caso y en la sección 2.3, tres modelos del segundo tipo.

2.1.4 Exploración

La exploración (o búsqueda en línea – *online search*) es una forma de experimentación activa donde el aprendizaje se lleva a cabo a partir de la interacción con el entorno. Es el descubrimiento de las propiedades del medio con el que interactúa el agente, como consecuencia de la ejecución de acciones y de la observación de los estados alcanzados luego de su ejecución. Al explorar, el agente ejecuta una acción, observa el entorno y luego planifica la siguiente acción. Y así sucesivamente.
La exploración se suele llevar a cabo en medios estocásticos y/o dinámicos. En este tipo de entornos, los estados y acciones futuras son desconocidos para el agente. Aquí, el agente debe usar sus acciones como experimentos para determinar qué hacer después, y a partir de ahí, intercalar la planificación y la acción.
El ejemplo básico de exploración es un robot que es posicionado en un terreno desconocido, el cual debe ser explorado para construir una especie de mapa del mismo (exploración espacial). Este "mapa" equivaldría al conjunto de experiencias (percepciones) almacenadas en forma de tabla, y cada unidad de experiencia se representaría como una tupla cuyos elementos serían: la situación inicial, la acción ejecutada y la situación resultante.
Los problemas de exploración pueden resolverse únicamente por un agente que ejecuta acciones, más que por un proceso puramente computacional [Russell y Norvig, 2004]. Después de cada acción, el agente explorador recibe una nueva percepción y a partir de esta nueva información logra mejorar el "mapa" del medioambiente. En cada nuevo instante se utiliza la última actualización del "mapa" para decidir dónde ir después, es decir, para elegir que acción tomar.

2.1.5 Aprendizaje por Refuerzo

El aprendizaje por refuerzo se fundamenta en la psicología conductista. Se basa en la simple observación de que el hecho de recompensar conductas consideradas deseables y castigar aquellas indeseables, llevan a un cambio en el comportamiento. El término *aprendizaje por refuerzo* fue introducido en la Inteligencia Artificial en los primeros trabajos de Minsky [1954; 1963; Minsky y Selfridge, 1961]. Según el autor, el problema de este tipo de aprendizaje consiste en asignar premios y castigos a las acciones que el sistema realiza sobre su entorno.

El objetivo de un agente basado en refuerzo es descubrir la mejor relación entre los distintos estados del entorno y la ejecución de las acciones, de forma tal de maximizar la señal de refuerzo. La forma de lograr dicho objetivo es implementando el método de prueba y error. Es decir, en este tipo de aprendizaje el agente debe descubrir cuáles son las acciones que le generan una mayor recompensa, y para ello, el único método posible es probándolas. En la mayoría de los casos, las acciones ejecutadas no sólo influyen en el refuerzo inmediatamente posterior sino en los que percibirá más adelante. Estas dos características, prueba-error y refuerzo retardado, son los rasgos más distintivos de este tipo de aprendizaje.

En la figura 2.1 [Sutton y Barto, 1998] se describe la dinámica básica del refuerzo. En el momento *t* el agente se encuentra en el estado s_t y con un refuerzo r_t. La ejecución de la acción a_t resulta en un refuerzo r_{t+1} (puede ser una recompensa o castigo) y en un pasaje hacia el estado s_{t+1}.

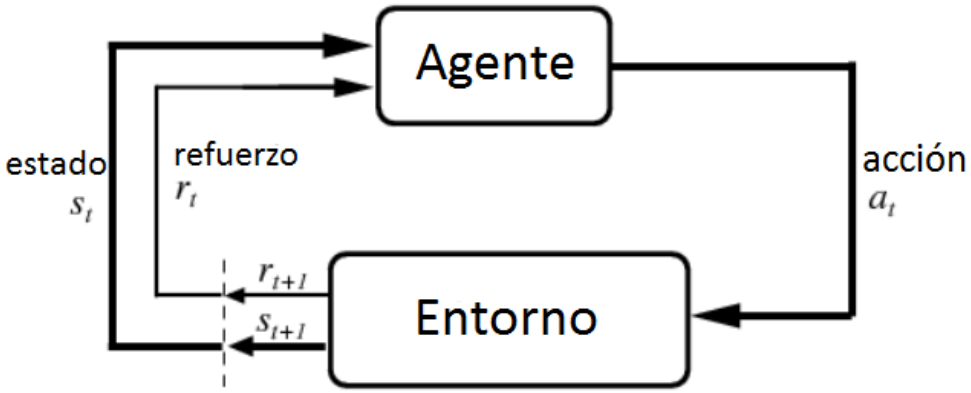

Figura 2.1 - Dinámica del aprendizaje por Refuerzo (adaptado de [Sutton y Barto, 1998])

Uno de los retos que se plantea el aprendizaje por refuerzo es el equilibrio entre la *exploración* y la *explotación*. El conflicto de intereses es claro: por un lado el agente prefiere llevar a cabo las acciones que se han ejecutado en el pasado y cuyas recompensas han demostrado ser altas; y por el otro lado, es evidente que para descubrir este tipo de acciones debe probar las que aún no se han ejecutado. En otras palabras, el agente debe explotar lo que sabe hasta el momento para obtener recompensas altas en el presente, pero también debe explorar nuevas acciones para poder conseguir altas recompensas en el futuro. Lo que se produce es un conflicto entre la recompensa de corto plazo y la de largo plazo. El dilema consiste en que ninguna de ambas tareas puede ser ejecutada de forma exclusiva sin que la otra falle. Por lo tanto, el agente deberá balancear entre una tarea y la otra para obtener el mejor rendimiento. Para ello, tendrá que incentivar la exploración probando una variedad de acciones nuevas, pero al mismo tiempo, tendrá que inducir la explotación de la información ya adquirida mediante una inclinación progresiva hacia aquellas acciones que lo recompensen mejor.

2.2 ARQUITECTURAS CON CONOCIMIENTO INCORPORADO POR EL PROGRAMADOR Y AJUSTADO POR EL SISTEMA

En esta sección se presentan dos sistemas en donde el conocimiento del dominio es incorporado por el programador y ajustado por el sistema. Ellos son: el sistema de Hayes-Roth (apartado 2.2.1) y el sistema PRODIGY/EXPERIMENT (apartado 2.2.2).

2.2.1 Sistema de Hayes-Roth

Para aprender en un mundo real, dinámico y observable, un sistema necesita conocer los efectos de las acciones sobre su entorno [Hayes-Roth, 1983]. A este tipo de conocimiento, el que encapsula los efectos de las acciones, Hayes-Roth llamó *teorías*. Un sistema de este tipo necesita construir planes, monitorear la ejecución de esos planes para detectar expectativas no cumplidas y diagnosticar y rectificar errores [García-Martinez, 1997]. La arquitectura del sistema se presenta en la figura 2.2.

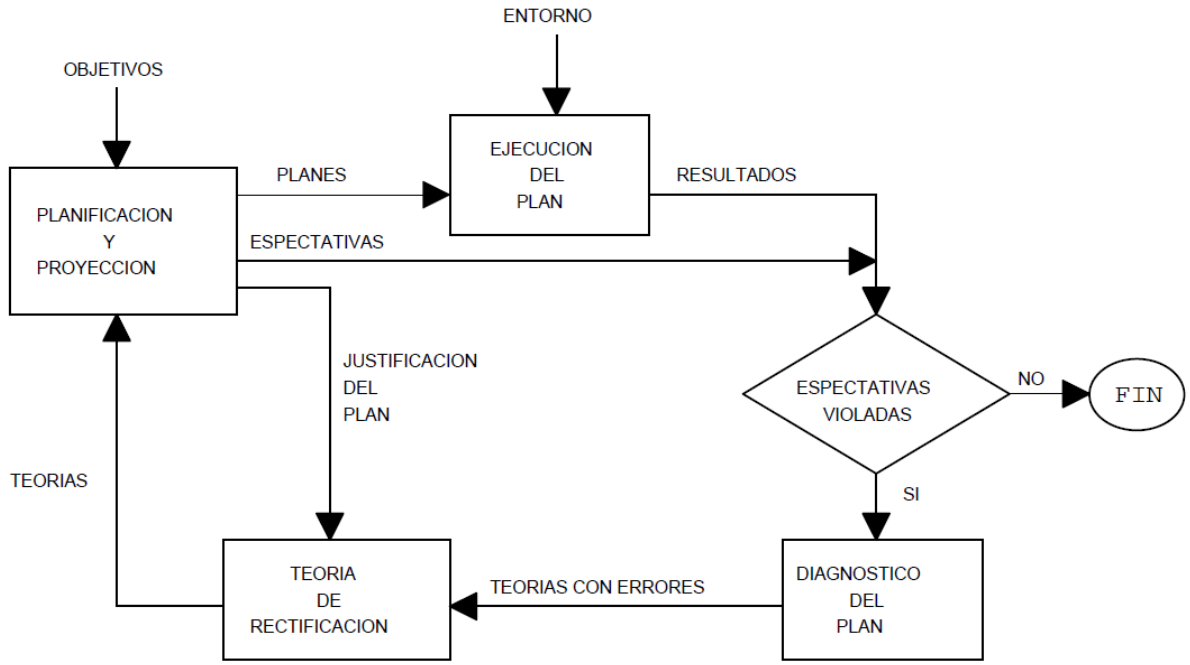

Figura 2.2 – Arquitectura del sistema propuesto por Hayes-Roth [García-Martinez, 1997]

En la medida que el agente interactúa con el medioambiente (gana experiencia), el conocimiento almacenado sobre el entorno (estructurado en forma de teorías) evoluciona gradualmente y es utilizado para tratar de alcanzar los objetivos.

El sistema genera los planes a partir de un conjunto de teorías. El armado del plan queda justificado si, al aplicarse al estado inicial, alcanza el estado objetivo. Si el plan falla, se supone implícitamente que hubo un error en las teorías asociadas.

A continuación se enumeran las causas más usuales de fallos [García-Martinez, 1997]:

- El plan fue injustificado, teniendo en cuenta las teorías y supuestos del sistema;
- El plan justificado previo no predijo cambios en las teorías asociadas;
- La justificación del plan fue equivocada.

Además de planificar con el fin de alcanzar sus objetivos, el sistema emplea métodos heurísticos para rectificar los errores de las teorías vigentes. Esta rectificación puede llevarse a cabo agregando o quitando restricciones. Para llevar adelante esta tarea de mejoramiento de las teorías locales aplicadas al proceso de planificación, Hayes-Roth [1983] propone alguna de las siguientes heurísticas:

- *Retracción*: restringe las predicciones de la teoría para que sea consistente con las observaciones;
- *Exclusión*: excluye la teoría que fue aplicada a la situación asociada al fallo;
- *Invalidación*: desecha las situaciones que niegan las predicciones de la teoría;
- *Aseguración*: incluye las situaciones que confirman las predicciones de la teoría;
- *Inclusión*: restringe la teoría incluyendo los casos que la confirman.

Estas heurísticas, que Hayes-Roth describe, son la base de gran parte del trabajo vinculado al aprendizaje dentro de la inteligencia artificial en general [Salzberg, 1985].

2.2.2 PRODIGY / EXPERIMENT

El sistema PRODIGY [Carbonell *et al.*, 1990; Veloso *et al.*, 1995] es un solucionador de problemas de uso general y también un planificador. A partir de una situación inicial definida, busca la mejor secuencia de operadores (mejor plan) para poder alcanzar su objetivo. Dicha búsqueda es guiada por un conjunto de reglas de control que se aplican en cada punto de decisión.
Como PRODIGY intenta resolver un problema, debe tomar decisiones sobre qué nodo expandir, sobre qué objetivo trabajar, qué operador aplicar y qué objetos usar. Estas decisiones pueden ser influenciadas por un control de reglas para incrementar la eficiencia en la búsqueda de solución del problema y para mejorar la calidad de las soluciones que se encuentran.
A continuación se enumeran los componentes propuestos por el sistema, a partir de los cuales se articula el aprendizaje:

- *Aprendiz* (Apprentice): interface de usuario que puede participar en un diálogo parecido al de un aprendiz [Joseph, 1989], permitiendo al usuario evaluar y guiar la resolución de problemas

y el aprendizaje del sistema. La interface es gráfica y está vinculada directamente al solucionador de problemas, de manera que los dos puedan adquirir conocimiento del dominio o aceptar consejos mientras el sistema está resolviendo un problema;

- *Módulo de aprendizaje basado en explicaciones* (EBL) [Minton, 1988; 1990]: permite adquirir reglas de control partiendo de la traza de resolución de un problema. Las explicaciones son construidas a partir de una teoría axiomática que describe tanto el dominio como los aspectos más relevantes de la arquitectura del solucionador de problemas;
- *Módulo de aprendizaje de reglas de control a partir de la descripción del dominio* (STATIC) [Etzioni, 1990]: analiza las descripciones del dominio y genera las reglas de control sin la utilización de ejemplos de entrenamiento;
- *Módulo de analogía derivacional* (Analogy) [Veloso y Carbonell, 1990]: reutiliza soluciones de problemas ya resueltos para resolver problemas futuros. El solucionador de problemas guarda las justificaciones de cada decisión durante su proceso de búsqueda. Estas justificaciones son usadas luego para guiar la reconstrucción de los próximos casos de resolución de problemas donde existan justificaciones similares;
- *Alpine*: módulo de aprendizaje y planificación de abstracciones [Knoblock, 1994]. El conocimiento axiomatizado del dominio es dividido en múltiples niveles de abstracción. Luego, durante la resolución del problema, PRODIGY primero encuentra una solución en un espacio abstracto y después usa esta solución para guiar la búsqueda de soluciones en espacios más concretos;
- *Módulo de aprendizaje por experimentación* (EXPERIMENT) [Carbonell y Gil, 1990]: refina el conocimiento sobre el dominio que es incompleto o que está incorrectamente especificado. La experimentación se dispara cuando el monitoreo del plan en ejecución detecta diferencias entre las expectativas internas y las observaciones.

Todos estos módulos de aprendizaje están en cierta medida integrados, ya que todos dan las mismas definiciones del dominio y comparten el mismo solucionador de problemas y la misma estructura de datos. Luego de que cada módulo obtiene sus conocimientos individualmente, estos son incorporados a la base de conocimiento del sistema.

En EXPERIMENT, el refinamiento del conocimiento sobre el dominio puede ser representado como el refinamiento de un conjunto de teorías sobre los operadores que modifican ese dominio. Cada operador tiene la siguiente estructura:

- Nombre del operador;
- Precondiciones que deben ser satisfechas por el estado del mundo;

- Postcondiciones que aparecerán/desaparecerán del nuevo estado del mundo.

Un plan está dado por una composición de operadores. Si el plan falla, se busca la causa en la definición de las precondiciones o las postcondiciones de los operadores que forman el plan. Si se interpreta al conjunto de los operadores como una teoría de los efectos que determinadas operaciones imprimen al estado del mundo, el fallo de un plan implica el fallo de la teoría, la cual debe ser refinada. La tabla 2.1, presentada en [García Martinez, 1997], estudia tres casos de refinamiento:

Resultado Esperado	**Comportamiento Observado**	**Estrategia de Recuperación**	**Método de Aprendizaje (Generador de Experimentos)**
Todas las precondiciones se satisfacen en el estado inicial.	Al menos una precondición es violada en el estado presente.	Planificar para obtener la precondición perdida.	Búsqueda binaria en la secuencia de operadores para establecer las precondiciones hasta el presente, agregando precondiciones negadas como postcondiciones del operador culpable.
Todas las precondiciones se satisfacen en el estado inicial.	Todas las precondiciones se satisfacen, pero el operador falla al ser aplicado.	Intentar planificar sin ese operador. Si falla esto: suspender el plan hasta que se realice el experimento.	Comparar el fallo presente con la última aplicación con éxito del operador, generar descripciones intermedias del mundo mediante una búsqueda binaria para identificar la parte necesaria del estado y agregarla a las precondiciones del operador.
El operador se aplica y todas las postcondiciones son satisfechas.	Al menos una postcondición falla en ser satisfecha.	Si la postcondición no satisfecha es incidental, ignorarla. Si es un estado objetivo, tratar con diferentes operadores.	Comparar con la última vez que todas las postcondiciones fueron encontradas. Desarrollar una búsqueda binaria en los estados del mundo para determinar la parte necesaria para satisfacer todas las postcondiciones. Luego, reemplazar con dos nuevos operadores, uno con la nueva precondición y todas las postcondiciones y el otro con la nueva precondición negada y sin la postcondición puesta en duda.

Tabla 2.1 – Casos de Refinamiento de EXPERIMENT [García-Martinez, 1997]

En el contexto de PRODIGY, Wang [1994; 1995] ha desarrollado el sistema OBSERVER, que aprende operadores a partir de las soluciones generados por expertos y luego los refina ejecutándolos en un simulador. El sistema automáticamente transforma los conocimientos generados por el simulador, en operadores que pueden ser utilizados dentro del proceso de planificación. Las precondiciones y postcondiciones de cada operador también son generadas automáticamente.

2.3 ARQUITECTURAS CON CONOCIMIENTO INCORPORADO POR EL SISTEMA

En esta sección se presentan tres arquitecturas en donde el conocimiento del dominio es incorporado directamente por el sistema. El primero de ellos es el Sistema Inteligente Autónomo (apartado 2.3.1); el segundo es el sistema LIVE (apartado 2.3.2); y el último, el sistema de Christiansen (apartado 2.3.3).

2.3.1 Sistema Inteligente Autónomo

De acuerdo con [García-Martinez, 1997], uno de los puntos involucrados en el problema de la modelización de Sistemas Inteligentes [Fritz, 1984; 1992; Fritz *et al.*; 1989] es lograr una base axiomática que describa formalmente los fenómenos que tienen lugar en este tipo de sistemas. Esta descripción apunta a proporcionar un instrumento para clasificar, medir y calcular en el campo de la inteligencia. Formalmente, no es relevante si el objeto de estudio es natural o artificial, sino que el propósito de dichos trabajos es abstraer los rasgos comunes, si los hay, de todos los procesos inteligentes. A partir de ésta abstracción, se clasificarán como inteligentes a los sistemas capaces de dar lugar a procesos inteligentes.

Un rasgo comúnmente asociado con la inteligencia es la capacidad de adquirir nuevos conocimientos. Esto se manifiesta en los procesos de aprendizaje, que aceptan ser descritos en términos de asimilación e incorporación de información extraída del contexto. Una forma de adquirir conocimiento nuevo es el llamado "método del ensayo-error", técnica que permite descubrir leyes simples cuya verdad se deduce a partir de la experiencia. En la teoría presentada por los autores mencionados, esta adquisición de conocimiento está centrada alrededor de la asimilación de experiencias, siendo las leyes empíricas las unidades de experiencia.

Un sistema inteligente autónomo (SIA) se define [Fritz *et al.*, 1989] como tal, si cumple con las siguientes condiciones: (i) transforma las percepciones de su entorno en situaciones (conjunto de datos esenciales del estado del medio); (ii) elige sus propios sub-objetivos guiado por su objetivo de diseño; (iii) construye sus propios planes para alcanzar sus objetivos, basándose en su propia experiencia (percepciones almacenadas en memoria); (iv) ejecuta el plan construido; y (v) aprende a partir de las interacciones con su entorno. Es decir, un SIA es aquel que percibe su entorno, que planifica sus acciones, que ejecuta los planes y que aprende a partir de las experiencias previas.

La interacción entre los distintos componentes del modelo se describe en la figura 2.3. El sistema parte de las percepciones del entorno y luego de conceptualizarlas, define la situación resultante. Esta

contiene los rasgos esenciales del estado del entorno, y en función de los objetivos del sistema, tratará de acceder a la nueva situación que le resulte más conveniente. Una vez hecho esto, el sistema recurre al conjunto de experiencias acumuladas para delinear el plan de acción. Cada unidad de experiencia se compone como mínimo de la situación inicial, la acción ejecutada, la situación final y el hecho de que las consecuencias de la acción hayan sido beneficiosas o no para lograr el objetivo. Este beneficio, o la falta del mismo, se traduce en la utilidad resultante. La acción que se llevará a cabo dependerá de que el sistema encuentre o no, en las experiencias previas, alguna relación con la situación actual. En caso afirmativo y considerando que el resultado de esa acción pasada haya resultado beneficiosa, el sistema tenderá a repetir la acción previa. En caso que el resultado de esa acción haya sido perjudicial, el sistema buscará acciones alternativas.

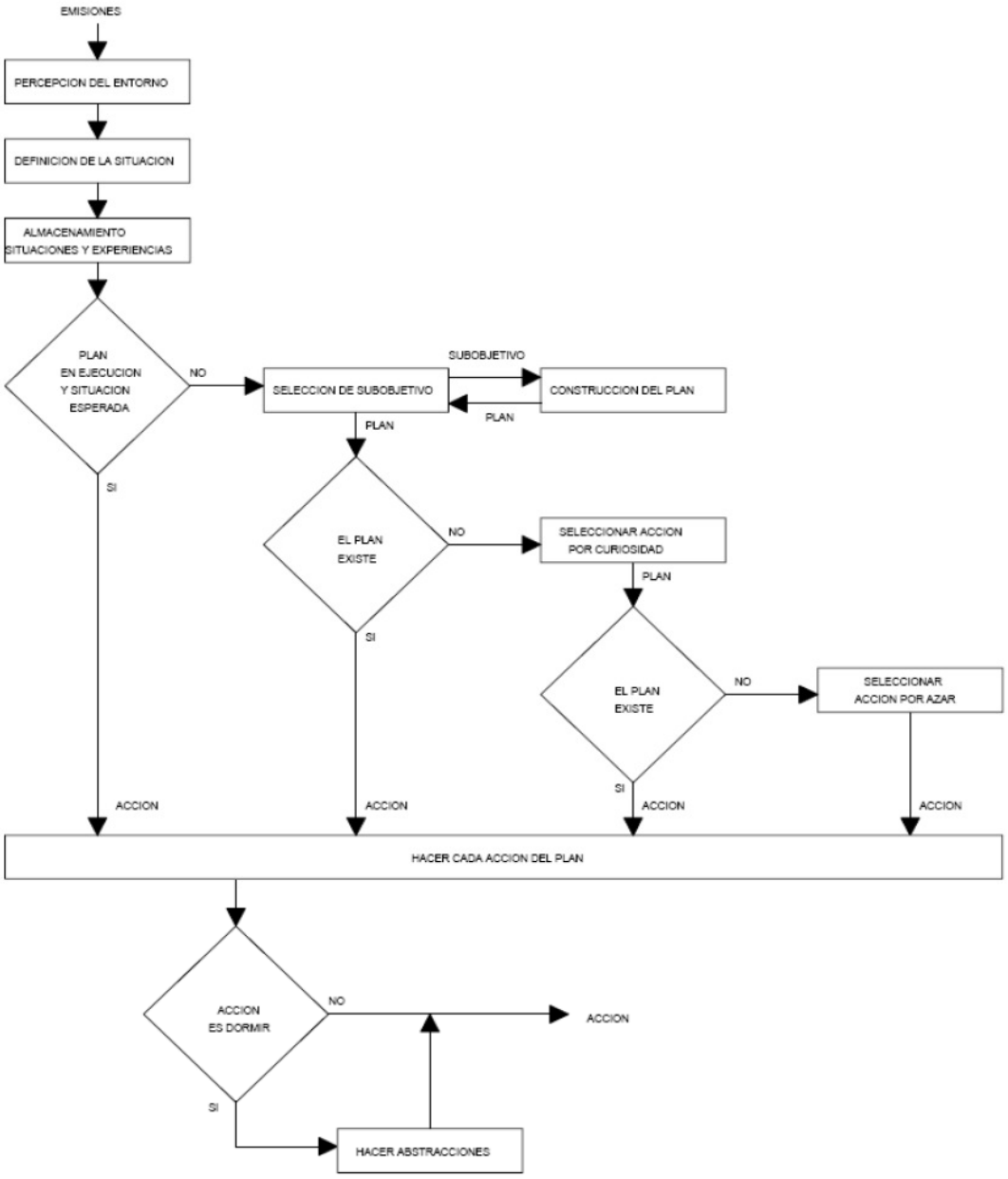

Figura 2.3 - Arquitectura del Sistema Inteligente Autónomo [García-Martinez, 1997]

Frente a situaciones conocidas, los sistemas inteligentes tienden a desarrollar una actuación que (por experiencia) consideran óptima (aunque no necesariamente es la óptima). Esta tendencia se denomina *hábito*. Un mal hábito se da cuando el sistema persiste en un cierto actuar, aun cuando éste ya no corresponde a la situación. Por otro lado, cuando el sistema se encuentra ante una situación nueva, este podrá actuar por azar, por intuición o basándose en experiencias previas.

Una de las características más importantes que se espera de un SIA es que aprenda lo máximo posible de lo que está percibiendo y de la forma más rápida. La importancia de este objetivo es más evidente cuando se trabaja con modelos en donde el agente no cuenta con ninguna información "a priori" del entorno donde se encuentra. En estos casos, no sólo la cantidad de interacciones será un factor preponderante en la eficiencia de su comportamiento, sino también la calidad de los procesos de aprendizaje proporcionados por el diseñador de software. En definitiva, en la medida que las interacciones con su entorno sean altas y que el algoritmo de aprendizaje sea más sofisticado, se verá disminuido el tiempo en que el sistema alcance un comportamiento exitoso y eficiente, o en otras palabras, la convergencia del modelo mejorará.

2.3.2 LIVE

Según se sostiene en [García-Martinez, 1997], el trabajo de Shen (sistema LIVE) enfoca el problema de aprendizaje a partir del entorno [Shen, 1989; Shen y Simon, 1989]. El objetivo es entender cómo un sistema resuelve problemas en un medio desconocido a partir de acciones innatas y de las percepciones. El aprendizaje es necesario ya que en un nuevo entorno un aprendiz no puede saber "a priori" las consecuencias de sus acciones ni las relaciones existentes entre las acciones y las percepciones de forma completa.

En este modelo el problema está motivado por dos hipótesis. La primera de ellas sostiene que el significado de los símbolos en un sistema está arraigado en las acciones del sistema y en las percepciones, es decir, la semántica es dependiente de la arquitectura. La segunda de las hipótesis de Shen consiste en que el aprendizaje no sólo es una de las actividades inteligentes básicas sino que tiene una importancia mayor que el razonamiento y la planificación. Sostiene que el aprendizaje está estrictamente ligado a las acciones y a las percepciones, y postula que, si es cierto que la evolución ha utilizado el 97% del tiempo en acciones y percepciones [Moravec, 1988], debe haber gastado por lo menos la misma cantidad de tiempo en aprender. En este sentido, la segunda hipótesis de Shen apoya la idea de que las acciones, las percepciones y el aprendizaje son bloques de construcción elementales de la inteligencia.

El aprendizaje del entorno requiere la integración de una variedad de actividades. Un sistema aprendiz debe ser capaz de explorar, planificar, adaptar, experimentar y descubrir. Afortunadamente, estas actividades tienen un propósito común: predecir las consecuencias de las acciones. Por esta razón, deben ser estudiadas de una manera coherente en la cual cada una ellas pueda beneficiar a las otras. Por ejemplo, la cooperación mutua entre predicción, explicación, resolución de problemas, experimentación y aprendizaje (descubrimiento) puede ser de la siguiente manera: la predicción es utilizada para evaluar un criterio de explicación; la explicación provee un significado a la predicción considerada; la resolución de problemas detecta cuándo y dónde los experimentos son necesarios; y el aprendizaje y el descubrimiento proveen más bloques de construcción para la predicción, la explicación y la resolución de problemas. Shen define el aprendizaje a partir del entorno de la siguiente manera:

> "Dado un conjunto de acciones y percepciones, el sistema aprendiz, mediante la aplicación de acciones y de la percepción del entorno, construye autónomamente un modelo de éste que le permite el cumplimiento de sus objetivos".

En esta definición las acciones son cambios físicos que ocurren dentro del aprendiz, por ejemplo, una señal que activa el motor que rota el brazo de un robot. Las consecuencias de las acciones pueden no ser conocidas y estas consecuencias pueden variar de un entorno a otro. Esta configuración, que separa las consecuencias de las acciones que las provocan, es un ingrediente clave para construir sistemas de aprendizaje que sean adaptables a distintos entornos.

Las percepciones son representaciones internas de la información que es percibida por el aprendiz desde el entorno, es decir, son la salida de los dispositivos de percepción del aprendiz. Dependiendo de estos dispositivos innatos las percepciones del aprendiz pueden ser al mismo tiempo muy precisas o muy vagas. Demasiada precisión puede exigir mucha potencia de procesamiento, mientras que demasiada vaguedad puede llevar a que no se capturen características importantes del mundo.

Para construir el modelo del entorno, el aprendiz es libre de usar todas sus acciones y percepciones. Sólo existe un único criterio para evaluar la calidad del modelo construido por el aprendiz: debe ser suficiente para que este logre alcanzar los objetivos de una manera deliberada. Es decir, debe predecir correctamente las consecuencias de cada una de las acciones que son empleadas en la solución.

Obsérvese que no se exige al sistema que construya un modelo exacto del entorno, sino que debe hacer hincapié en que los resultados de las acciones ejecutadas sean los esperados; ésta es una diferencia central entre la definición de Shen y la de la mayor parte del trabajo teórico existente en aprendizaje a partir del entorno [Rivest y Schapire, 1987].

2.3.3 Sistema de Christiansen

Según lo descrito en la tesis doctoral del Dr. García-Martinez [1997], el trabajo de Christiansen [1992] enfoca el aprendizaje automático de los efectos de las acciones de un sistema autónomo en un ambiente físico. El sistema interactúa con su entorno a través de sus sensores y sus efectores, y ambos le proveen un conjunto de valores que describen los efectos de las acciones sobre el medioambiente. Las imperfecciones en el sistema sensor-efector y las características de las acciones físicas, se combinan para generar resultados no deterministas y observados con ruidos. Por tal motivo, el aprendizaje exitoso de los efectos de las acciones requiere de algoritmos tolerantes al ruido y, para lograrlo, es necesario que el sistema sea capaz de razonar a partir de la incertidumbre y de predecir el efecto de sus acciones.

En este modelo, las acciones son representadas por operadores continuos llamados "funnels". Ellos son computados por un algoritmo de aprendizaje empírico que tolera el ruido y que asegura no realizar predicciones equivocadas. Con el fin de lograr sus objetivos, el sistema utiliza los sensores para obtener información del entorno y los efectores para ajustar el entorno de un modo tal que permita la concreción de los objetivos. En este contexto, el sistema debe ser capaz de predecir los efectos de las acciones que genera el subsistema efector.

El trabajo de Christiansen se concentra en dos aspectos del aprendizaje a partir del entorno, los cuales han recibido poca atención en la bibliografía: el tratamiento de las características del espacio continuo y el tratamiento del ruido y de la incertidumbre.

Desde el punto de vista del aprendizaje del sistema, el entorno es una caja negra. El sistema provee acciones como entradas al entorno y recibe estados percibidos a través de los sensores. Las salidas del entorno son llamadas estados, pero en general, estos estados solo reflejan parcialmente los estados verdaderos del entorno. En esta arquitectura, se asume que ambos, estados y acciones, son conjunciones de valores ordenados, correspondientes a medidas continuas de las características del entorno. Estos estados y acciones son descritos por puntos en un espacio multidimensional de características, en donde los espacios relevantes son los siguientes: (i) espacios de estados, S; y (ii) espacios de acciones, A. La situación actual se describe como un punto en el espacio <estado, acción>, el cual indica el estado actual y la acción a ser ejecutada.

El problema atacado por Christiansen es el de predecir el resultado de la ejecución de una acción desde el estado percibido. Asumiendo que el sistema usa los sensores antes y después de la ejecución de varias acciones, la información sobre los efectos de las acciones vuelve al sistema como una cadena de puntos [s_0, a_0, s_1, a_1, s_2, ...]. Otra forma posible de ver esto es como una secuencia de estados de transición [s, a, r] (estado, acción, estado resultante).

Un aspecto interesante de este tipo de aprendizaje es que el sistema solo recibe ejemplos de su tema de interés. Es decir, solo ve lo que puede ocurrir a partir del resultado de sus acciones, nunca ve lo que no puede conseguir. Una aproximación para la inferencia inductiva de los problemas de esta clase, es asumir familias de restricciones o formas funcionales para las respuestas, y aplicar técnicas de regresión para cambiar la mejor respuesta de acuerdo a algunas medidas de error. Estos métodos permiten tratar con ruido y con valores de características continuas.

Para que estas técnicas sean exitosas se requiere que la familia de respuestas aceptables sea limitada "a priori", limitación que constituye una forma de sesgo inductivo. Para los sistemas que poseen una gran cantidad de conocimiento inicial, tal como los sistemas de aprendizaje basado en explicaciones, el éxito del sistema de aprendizaje es altamente dependiente del conocimiento inicial elegido por el programador humano.

En este modelo se asume que el sistema, inicialmente, tiene poco conocimiento sobre el entorno y que no posee ninguna habilidad para predecir el efecto de las acciones. También asume que las características observadas (estado percibido) corresponden a medidas continuas de las características del entorno, y que las características controladas (correspondientes a los parámetros controlados de las acciones) representan cambios continuos con causa en los efectos de las acciones físicas. En adición, asume también que los cambios de estado del entorno solo son por la influencia de sus acciones. Por otra parte, supone que el entorno no es dinámico, es decir, que para el resultado de cada acción, el entorno permanece en un estado invariante durante el tiempo en que el sistema usa los sensores para medir los resultados de sus acciones. El supuesto de la invariancia del entorno en el tiempo permite al sistema generalizar sobre los estados de transición observados, independientemente de la historia específica de cada uno de ellos.

2.4 EL MODELO LOPE

El modelo LOPE (Learning by Observation in Planning Environments) [García-Martinez y Borrajo, 1997; 2000] es un sistema inteligente autónomo [Fritz *et al.*, 1989] con aprendizaje basado en formación y ponderación de teorías. Se ubica dentro de la categoría de los sistemas en los cuales el conocimiento sobre el dominio es incorporado por el sistema. Su arquitectura integra los procesos de planificación, control y ejecución, percepción y aprendizaje en un ciclo cerrado (figura 2.4).

Puede ser descrito como un robot de exploración que percibe el entorno a través del sistema sensor (figura 2.5) y registra teorías locales a partir de la situación previa, la acción ejecutada y la situación resultante. Dichas teorías son utilizadas para la construcción de planes que le permitirán alcanzar sus propios objetivos. A partir del conjunto de las teorías locales almacenadas, el sistema elabora una

representación del entorno que lo rodea y, basándose en ella, ejecuta el siguiente ciclo de planificación-ejecución-percepción-aprendizaje. En estos casos, como el robot sólo está percibiendo su entorno inmediato, la representación del medioambiente no deja de ser parcial. Aun así, es posible llevar adelante una extrapolación y utilizar el conjunto de teorías existentes como base para el próximo proceso de planificación y ejecución [Hayes-Roth, 1983].

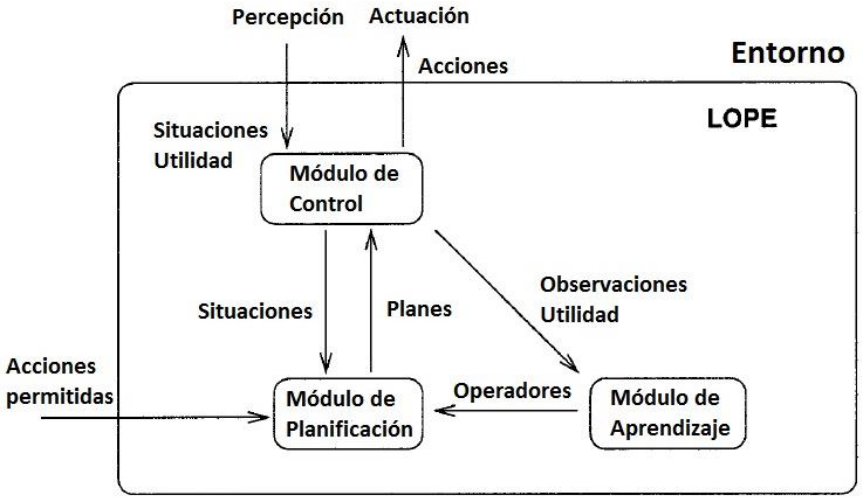

Figura 2.4 – Arquitectura del modelo LOPE
(adaptado de [García-Martinez y Borrajo, 1997; 2000])

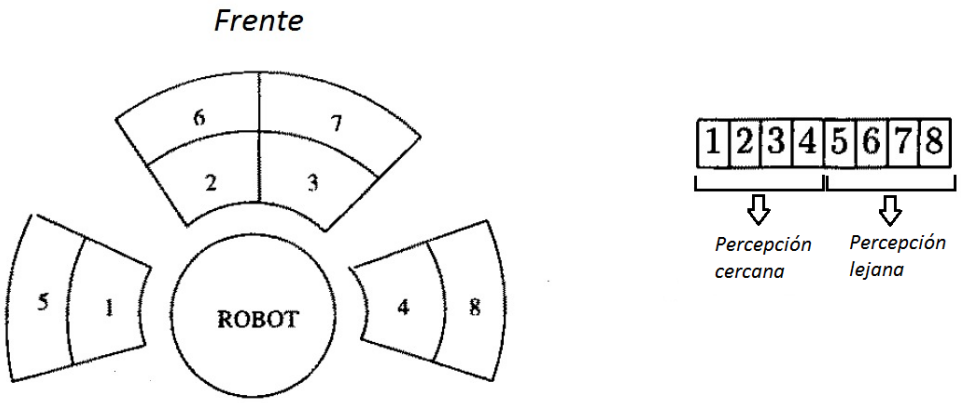

Figura 2.5 – El Robot Explorador y su sistema Sensor
(adaptado de [García-Martinez y Borrajo, 1997])

Con el fin de acelerar la convergencia del modelo se incluyen criterios heurísticos para hacer una generalización de las observaciones (percepciones) obtenidas. También se utilizan estimadores de probabilidad que permiten gestionar las contradicciones que se generan en las descripciones de los operadores.

El objetivo de la arquitectura LOPE es [Berlanga *et al.*, 1999] el estudio y la construcción de sistemas que, partiendo de conocimiento nulo, sean capaces de: auto-proponerse metas; establecer planes para

alcanzar dichas metas; ejecutar los planes; y aprender cómo afectan las acciones del robot al entorno (operadores) a partir de las desviaciones producidas al ejecutar las acciones de los planes.
En [Fritz *et al.*, 1989] las teorías (operadores) se arman a partir de la acción ejecutada y de lo percibido por el sistema sensor, antes y después de la acción. Cada observación o unidad de experiencia tiene la siguiente estructura:

[Situación Inicial, Acción, Situación Final]

Con este tipo de estructura, las percepciones son utilizadas directamente en el armado de la teoría local, sin ninguna clase de ajuste, es decir, la percepción y la teoría son una misma cosa, no hay ningún proceso de análisis en el medio. En el modelo LOPE, el concepto de percepción se mantiene idéntico pero la estructura de las teorías locales es extendida, ya que se le agregan ciertas características que permiten evaluar la confiabilidad de los planes antes de ser ejecutados. En este caso, la estructura de cada unidad experiencia asume la siguiente forma:

[Situación Inicial, Acción, Situación Final, *P*, *K*, Utilidad]

donde *K* representa la cantidad de veces que una teoría fue utilizada y *P* el número de instancias en que esa teoría fue utilizada con éxito. Por otro lado, la utilidad es el criterio utilizado para armar los planes, siendo el objetivo del robot o más precisamente de su proceso de planificación, el de alcanzar la situación de mayor utilidad. Es importante tener en cuenta que la utilidad no es una variable de entrada para el robot sino que está determinada implícitamente por el sistema en su conjunto. Es decir, cada coordenada espacial del entorno ya tiene asociada una utilidad definida “a priori” por el diseñador del sistema.
La figura 2.6 describe los procesos de percepción, aprendizaje, planificación y control y ejecución del modelo. Al comienzo el sistema no posee ningún conocimiento, por lo que simplemente percibe la situación inicial, elige una acción aleatoria y la ejecuta. Luego de esta primera acción, el agente percibe una nueva situación y arma una teoría local con la situación previa y la acción ejecutada. A partir de la comparación de la nueva teoría local armada y las teorías ya registradas en el sistema, se procede de alguna de las siguientes maneras:

- Si la teoría local es igual a alguna teoría registrada, ésta se refuerza. Para ello se incrementa el P y el K de la teoría registrada y se incrementa el K de las teorías similares;
- Si la teoría local es similar a alguna teoría registrada, se registra la nueva teoría local; se registra una nueva teoría mutante; se incrementa el P de la nueva teoría local y la mutante; se estandarizan los K de la teoría nueva y las teorías similares; y se asigna el nivel de utilidad a las nuevas teorías;

- Si no existe una teoría igual ni similar a la teoría local, se registra la nueva teoría local, se incrementa el P y el K de la nueva teoría y se le asigna el nivel de utilidad que le corresponde.

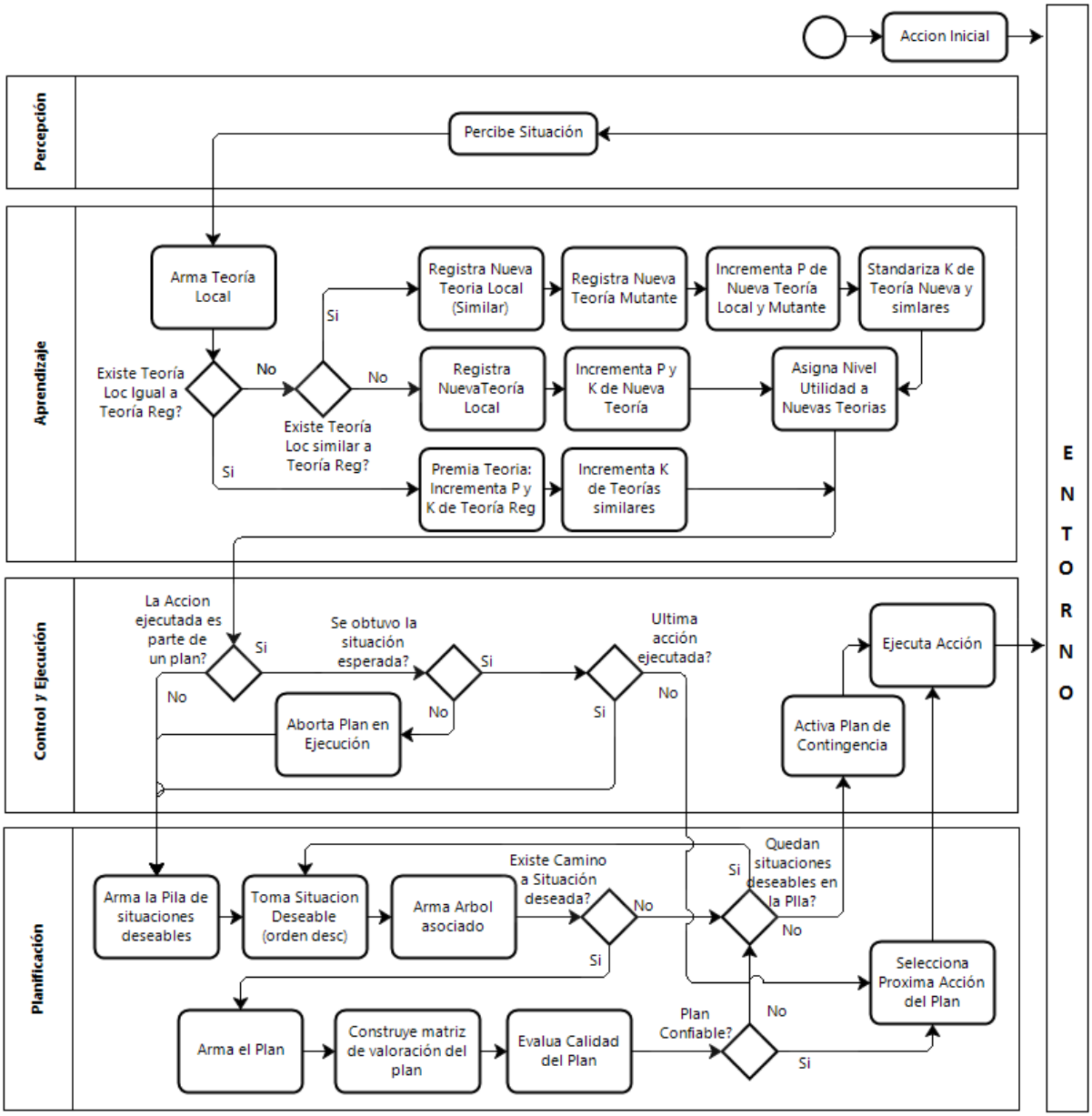

Figura 2.6 – Ciclo de Percepción-Aprendizaje-Planificación-Ejecución del modelo LOPE (adaptado de [García-Martinez, 1997])

Si existe un plan en ejecución, se verifica que la situación obtenida sea la esperada. Si esto ocurre, el control se pasa al módulo ejecutor, que selecciona la próxima acción y la ejecuta. En caso que la situación obtenida no haya sido la esperada, se aborta el plan y el control es devuelto al planificador.

Si no existe un plan en ejecución, ya sea porque la última acción del plan fue ejecutada o porque la acción ejecutada no era parte de un plan (acción inicial o plan de contingencia), entonces el control pasa al planificador y éste intenta armar un nuevo plan.

El planificador lleva adelante los siguientes pasos para la creación del plan: arma la pila de situaciones deseables; elige la situación con mayor nivel de utilidad; procede al armado del árbol de situaciones asociado; y verifica que exista un camino de mínimo recorrido entre la situación actual y la deseada. En caso que exista ese camino se arma el plan, se construye su matriz de valoración y, a partir de ella, se evalúa la calidad del plan. Si el plan resulta confiable (la probabilidad de éxito es superior al umbral de confiabilidad) se selecciona la acción correspondiente y se traslada el control al módulo ejecutor, quien es el responsable de ejecutarla. Si el plan no resulta confiable y quedan situaciones deseables en la pila se toma la siguiente situación deseable de ella y se repiten los mismos pasos. Si el plan no resulta confiable pero no quedan más situaciones deseables en la pila, se pasa el control al módulo ejecutor y éste ejecuta el plan de contingencia. Por último, si no existe un camino de mínimo recorrido entre la situación actual y la deseada y aún quedan situaciones deseables en la pila, se toma la siguiente situación deseable y se ejecutan los pasos ya descritos; si se agotan todas las situaciones deseables disponibles, se ejecuta el plan de contingencia.

2.5 EXTENSIONES O MODIFICACIONES AL MODELO LOPE

En esta sección se describen las cuatro modificaciones al modelo LOPE que se han elaborado en los últimos años: (i) SIA con aprendizaje basado en intercambio de operadores (apartado 2.5.1); (ii) SIA con ciclo de vida de aprendizaje (apartado 2.5.2); (iii) actualización del método de ponderación del modelo, basado en la productoria de probabilidad de éxito de acciones (apartado 2.5.3); y (iv) implementación de algoritmos genéticos al SIA (apartado 2.5.4).

2.5.1 Sistema Inteligente Autónomo con Aprendizaje basado en Intercambio de Operadores

De acuerdo a la reseña formulada en la tesis doctoral del Dr. Ierache [2010], esta arquitectura de sistema inteligente autónomo también percibe el entorno a través del sistema sensor, pero antes de realizar cualquier acción, se pregunta si es necesario intercambiar operadores con otro sistema inteligente autónomo [García-Martínez *et al.*, 2006]. Este proceso se lleva a cabo mediante un módulo de intercambio de operadores. Luego, registra la situación percibida del entorno y arma una teoría local con la situación previa y la acción ejecutada. En la figura 2.7 se presenta la arquitectura del sistema, donde se observa su interacción con el entorno y el funcionamiento general de los distintos componentes del modelo.

Si la teoría local es igual a alguna teoría registrada, ésta se refuerza; si no existe una teoría igual pero existen similares, éstas se ponderan y se generan teorías mutantes, las cuales son registradas y

ponderadas de la misma forma. Por último (luego del proceso de generación, registro y ponderación de teorías mutantes o si no existen teorías similares) se incorpora la teoría local y se pasa el control al subsistema controlador.

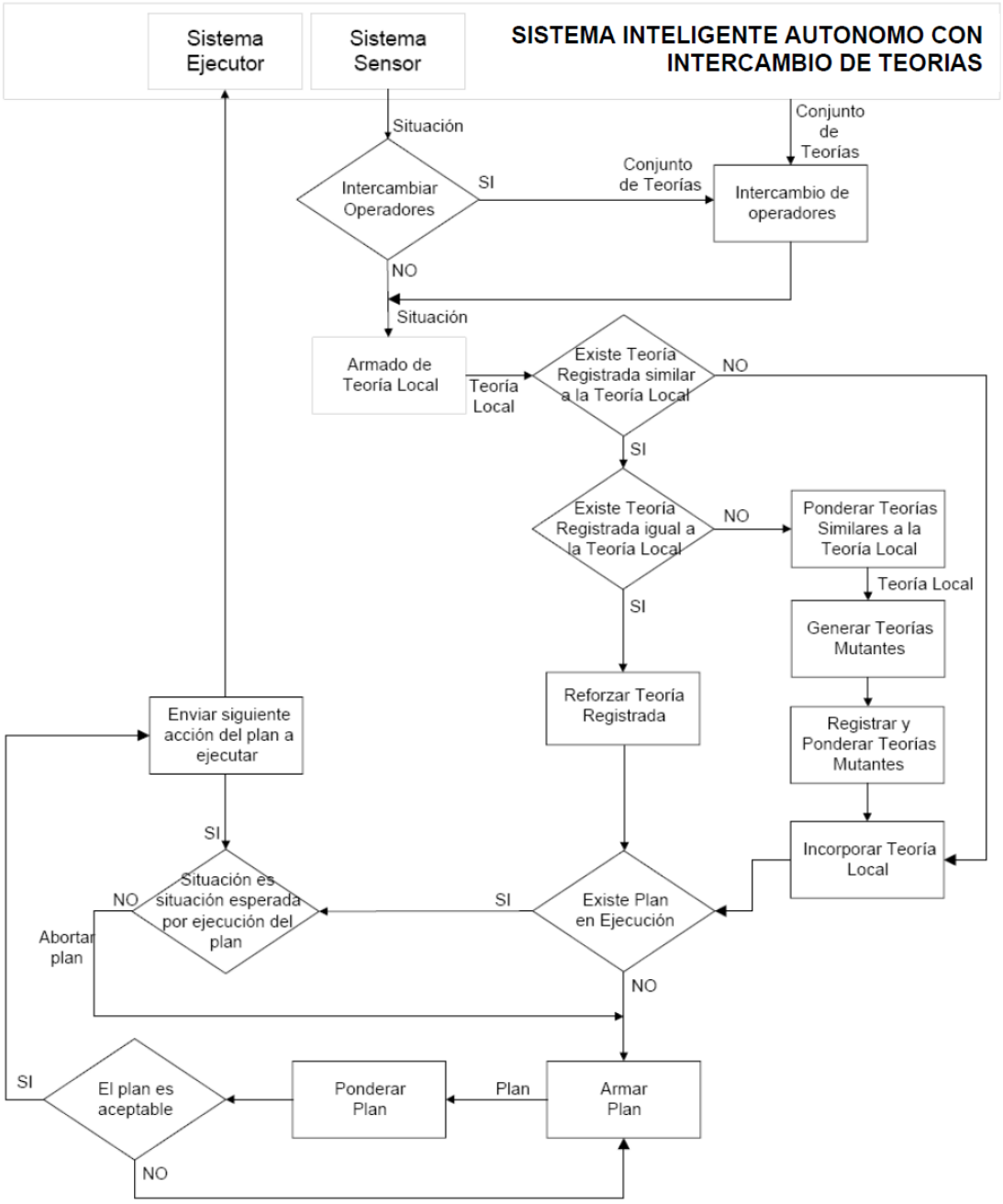

Figura 2.7 – Arquitectura del SIA con Intercambio de Teorías [Ierache, 2010]

Si existe un plan en ejecución, se verifica que la situación obtenida sea la esperada; si no ocurre esto, se aborta el plan y el control es devuelto al planificador. Si no existe un plan en ejecución, el planificador genera uno, lo envía al ponderador y mediante un criterio heurístico, determina si el plan es aceptable. En caso afirmativo, el controlador de planes en ejecución determina la siguiente acción a ser ejecutada, la cual es pasada al sistema ejecutor para que éste la aplique en el entorno.

2.5.2 Sistema Inteligente Autónomo con Ciclo de Vida de Aprendizaje

En [Ierache *et al.*, 2008] se presenta una extensión al modelo LOPE en la que se incluye un ciclo de vida de aprendizaje de tres capas:

1. *Operadores Integrados (BIO):* capa de aprendizaje donde los operadores (teorías) son implantados en el sistema por el programador;
2. *Operadores basados en entrenamiento (TBO)*: capa de aprendizaje donde las teorías son diseñadas previamente por el programador y luego se le aplican técnicas de aprendizaje evolutivo;
3. *Operadores de interacción global (WIO)*: capa de aprendizaje donde los operadores son aprendidos por la interacción con el medioambiente y con otros SIAs.

El ciclo de vida de aprendizaje propuesto para el modelo LOPE-LLC (LOPE Learning Life Cycle) se observa en la figura 2.8. El SIA "nace" con los operadores implantados por el programador, los que representan el conocimiento básico y que permiten el comportamiento reactivo inicial del sistema. La evolución de este conocimiento se lleva a cabo a través de los operadores aprendidos por entrenamiento. Dicho aprendizaje incluye las técnicas de refuerzo: se castigan los malos operadores y se recompensan los buenos. También se incluyen criterios heurísticos para crear algoritmos generalizados e incluirlos al conjunto de operadores existentes. Dado que la gran cantidad de operadores almacenados podría disminuir el rendimiento de los módulos de planificación y aprendizaje, el sistema automáticamente deja de lado los operadores con cocientes *P/K* bajos.

Al igual que el modelo LOPE original, el sistema LOPE-LLC busca aprender por sí mismo aquellos operadores que permitan predecir los efectos de sus acciones en el medioambiente; y esto lo consigue observando las consecuencias de sus acciones. Resumiendo, este sistema es capaz de: (i) proponerse sus propios objetivos; (ii) ejecutar los planes; (iii) encontrar la conducta correcta e incorrecta; (iv) aprender a partir de los BIO; (v) refinar el conocimiento del sistema a partir de la creación de los TBO usando métodos de refuerzo; y (vi) evolucionar a partir del intercambio de los WIO.

En la etapa de aprendizaje de los TBO, el sistema recibe las percepciones del entorno (situaciones), aplica las acciones y aprende a partir de la interacción con el ambiente de entrenamiento diseñado. En este caso, la situación inicial percibida por el sistema es representada a partir de sus BIO, pero luego selecciona una acción al azar, elegida a partir del conjunto de sus TBO. Tanto los BIO como los TBO, también son utilizados más adelante durante la etapa de aprendizaje de los WIO. Sobre la base de las tres capas del modelo LLC propuesto, el sistema evoluciona alcanzando cada uno de los

cuatro estados de madurez: (1) "nacido" (born), (2) "novato" (newbie), (3) "entrenado" (trained) y (4) "maduro" (mature).

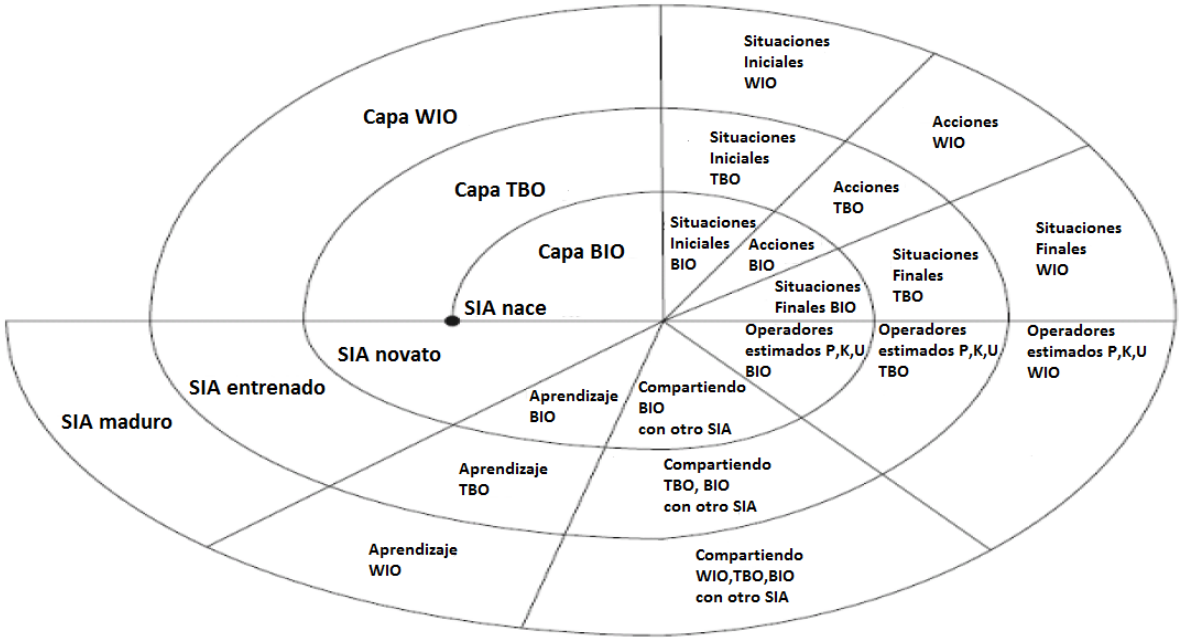

Figura 2.8 – Ciclo de Vida de Aprendizaje del Sistema Inteligente Autónomo LOPE-LLC (adaptado de [Ierache et al., 2008])

Cada una de las capas incluye las siguientes fases: (a) situación inicial del mundo (entorno y otros SIAs); (b) acciones basadas en los operadores del sistema, de acuerdo a los planes; (c) situación final prevista; (d) estimación de los operadores del sistema (*P*, *K*, *U*); (e) intercambio de operadores con otros SIAs; (f) proceso de aprendizaje; (g) evolución del sistema hacia el próximo estado.

Cuando el SIA "nace" (estado inicial), el programador le provee de operadores que le permitan iniciar su funcionamiento. El proceso dentro de la primera capa evoluciona a medida que se van compartiendo los BIO con otros SIAs y que se llevan adelante las tareas de aprendizaje. Llega un punto en que el sistema alcanza el nivel "novato". En esta nueva capa aprende a partir del entrenamiento y comparte los BIO y los TBO, lo que le permite alcanzar el nivel "entrenado". Finalmente, se inicia el camino de la última capa, en donde el sistema es capaz de compartir sus BIO, TBO y WIO, hasta que alcanza el estado "maduro".

2.5.3 Método de Ponderación basado en la productoria de probabilidad de éxito de acciones

En [López *et al.*, 2008] se propone un nuevo algoritmo de ponderación de planes con el objetivo de mejorar el rendimiento del sistema (porcentaje de planes exitosos). Para estimar la probabilidad de éxito de los planes, el método original de ponderación se basó en la teoría de autómatas estocásticos,

ya que el conocimiento que posee el sistema en un momento dado (el conjunto de teorías), puede ser visto como un modelo de cómo reaccionará el entorno frente a las acciones que ejecuta el sistema. O sea, si se consideran todos los estados en que puede encontrarse el autómata cuando recibe una entrada, puede definirse una matriz de transición que contenga la probabilidad de éxito del plan.
La primera modificación que lleva adelante López es dar una nueva representación para la definición de un plan. Mientras que en el modelo LOPE original, un plan tiene la siguiente estructura:

$$P_{ij} = A_1 \text{ o } A_2 \text{ o } \ldots \text{ o } A_r$$

donde A_1, A_2 y A_r identifican a cada una de las acciones necesarias para alcanzar la situación esperada; con la nueva propuesta, un plan se representa del siguiente modo:

$$P^*_{ij} = (S_{i1}, A_1, S_{f1}, P_1, K_1) \text{ o } (S_{i2}, A_2, S_{f2}, P_2, K_2) \text{ o } \ldots \text{ o } (S_{ir}, A_r, S_{fr}, P_r, K_r)$$

La diferencia entre ambas definiciones radica en que, si bien ambas representan un mismo plan, la segunda contiene mayor información, ya que expresa el plan como una composición de las *r* teorías en que se basó su construcción, haciendo explícitas las respectivas situaciones iniciales y finales esperadas en cada paso de la ejecución del plan.
En segundo lugar, mientras que la arquitectura LOPE original se basa en la matriz de transición para calcular la probabilidad de éxito de un determinado plan, en el método propuesto, la probabilidad estimada de que el plan P^*_{ij} aplicado a la situación S_i resulte en la situación S_j, se obtiene calculando el siguiente producto de números escalares:

$$P_{exito} = P_1/K_1 \,.\, P_2/K_2 \,.\, \ldots \,.\, P_r/K_r$$

Es decir, en este último caso, la probabilidad de éxito del plan es igual a la productoria de las probabilidades de éxito de cada uno de las acciones por separado.

2.5.4 Algoritmos genéticos aplicados a los sistemas inteligentes autónomos

En [Steinhilber *et al.*, 2009] se propone la aplicación de algoritmos genéticos al modelo LOPE. Un Algoritmo Genético (AG) es una técnica de búsqueda basada en la teoría de la evolución de Darwin, en la cual los individuos más aptos de una determinada población son los que sobreviven, ya que pueden adaptarse más fácilmente a los cambios que se producen en su entorno. Algunas de sus características son: (i) requieren muy poca información específica del problema; (ii) resuelven en forma rápida y eficiente problemas con espacio de búsqueda muy grandes; (iii) describen una alta dimensionalidad de la función de aptitud; y (iv) presentan funciones de aptitud no lineales [García-Martínez *et al.*, 2003].

Para operar, un AG simula la evolución de una población de individuos. Para ello, se genera una población inicial y luego se ejecuta un proceso iterativo de selección, cruzamiento y mutación. En la selección, se imita el mecanismo de selección natural, eligiendo los individuos más aptos del conjunto para que su información genética permanezca en las siguientes generaciones. El cruzamiento simula la reproducción sexual de los individuos, permitiendo que los de mayor aptitud generen nuevos individuos (hijos) combinando sus genes. Esta mezcla de código genético, provee un método de búsqueda altamente eficiente en el espacio de estructuras [García Martínez *et al.*, 2003]. Por último, la mutación contribuye a la diversidad genética de la población, alterando de manera aleatoria el valor de uno o más genes de cualquier individuo existente.

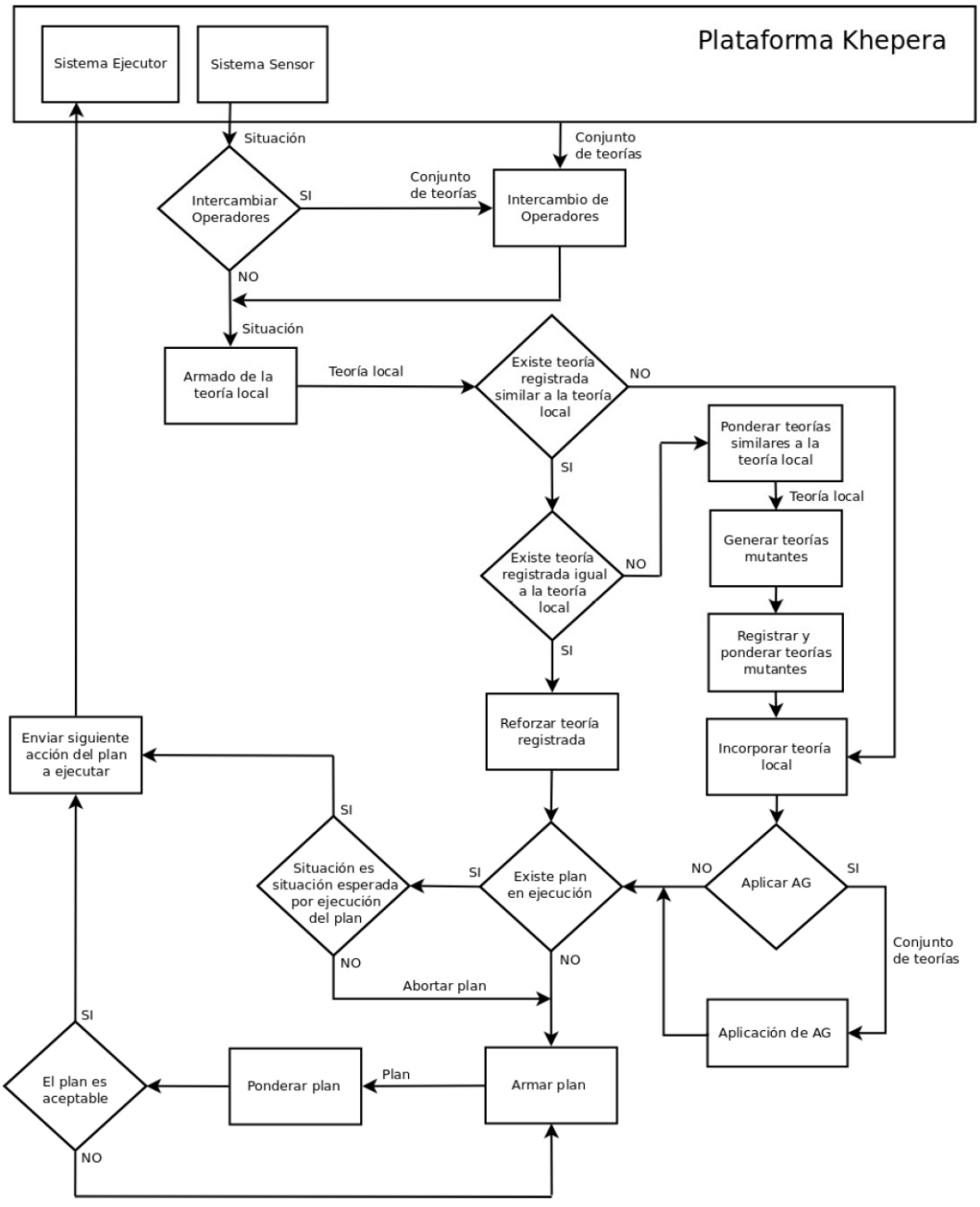

Figura 2.9 – Arquitectura del SIA con Algoritmos Genéticos [Steinhilber et al., 2009]

En la figura 2.9 se describe la nueva arquitectura propuesta. En esta versión modificada del modelo LOPE, el AG es implementado del siguiente modo. Una vez que el sistema percibe una nueva situación, se procede al armado de la teoría local. Si existe al menos una teoría similar y no hay teorías iguales a la nueva teoría local, se ponderan las teorías similares, se generan las teorías mutantes, se registran y ponderan las teorías mutantes y se incorpora la teoría local. Luego, si se sucedió una cantidad mínima de ciclos sin aplicar el AG y se iguala o supera una cantidad específica de teorías, se aplica el algoritmo genético al conjunto de teorías del SIA.

El AG acelera los tiempos de aprendizaje del SIA, ya que al combinarlo con otras estrategias, provoca un aumento en la cantidad de teorías que el sistema adquiere. También se observa un gran aumento de teorías cuando se combina mutación, intercambio, ponderación y AG, en comparación con los resultados obtenidos sin AG.

3. DESCRIPCIÓN DEL PROBLEMA

Se presenta en este capítulo el contexto del modelo LOPE (sección 3.1) y en las secciones subsiguientes las debilidades y oportunidades de mejora identificadas. En la sección 3.2 se detalla la debilidad que presenta el modelo en relación a la ejecución de giros ineficientes; en la sección 3.3 se describe la oportunidad de mejora que surge a partir de la detección de una administración incompleta de las teorías mutantes; en la sección 3.4 se presenta la debilidad vinculada a la falta de evaluación de los planes ejecutados; y por último, en la sección 3.5 se refiere la oportunidad de mejora que presenta el modelo en relación al índice de confiabilidad estático.

3.1 EL CONTEXTO DEL MODELO LOPE

En el capítulo previo se ha llevado a cabo una revisión de los métodos de aprendizaje y planificación de aquellos sistemas que se basan en la creación de operadores (o teorías) para representar el entorno y poder así, planificar sus acciones. El trabajo se centra en los llamados Sistemas Inteligentes Autónomos, concepto presentado por primera vez en [Fritz et al., 1989] y definido como aquel sistema que: (i) transforma las percepciones de su entorno en situaciones; (ii) elige sus propios sub-objetivos guiado por su objetivo de diseño; (iii) construye sus propios planes para alcanzar sus objetivos, basándose en su propia experiencia; (iv) ejecuta el plan construido; y (v) aprende a partir de las interacciones con su entorno.

Dentro de este tipo de sistemas, que aprenden a partir de la interacción con el entorno, es posible hablar de dos categorías: (a) aquellos en donde el conocimiento sobre el dominio es incorporado por el programador y ajustado por el sistema y (b) aquellos en donde el conocimiento sobre el dominio es incorporado puramente por el sistema [García-Martinez, 1997]. Esta tesis se focaliza en este último grupo, aunque se han descrito algunos sistemas importantes de ambos tipos para contextualizar los conceptos introducidos.

En línea con el trabajo de Fritz [1989], años más tarde se presenta el modelo LOPE [García-Martínez y Borrajo; 1997; 2000], un sistema inteligente autónomo con aprendizaje basado en formación y ponderación de teorías. Dicho modelo pertenece a la categoría (b) recientemente citada y puede ser definido como un robot de exploración que percibe el entorno a través de su sistema sensor y registra teorías locales a partir de la situación previa, la acción ejecutada y la situación resultante. La característica nueva de este modelo es que las teorías son ponderadas de acuerdo al número de veces que éstas fueron utilizadas con éxito en el proceso de planificación. Dicha ponderación sirve, luego, para evaluar la calidad de los planes antes de ser iniciados y de esta forma evitar la ejecución de

planes con baja probabilidad de éxito. A partir de la publicación de la arquitectura LOPE, en la última década se han elaborado varias modificaciones al modelo que han logrado mejorar el rendimiento de su aprendizaje y de su planificación. Ellas son:

1) En [García-Martínez et al., 2006] se incorporó una arquitectura multiagente que permite el intercambio de teorías;
2) En [Ierache et al., 2008] se utilizó un sistema multiagente pero en este caso se implementó un ciclo de vida de aprendizaje y se definieron distintos perfiles de agentes, cada uno de los cuales con un determinado método de adquisición y transmisión de conocimiento;
3) En [López et al., 2008] se propuso un mecanismo de ponderación de planes distinto, basado en la productoria de probabilidad de éxito de acciones;
4) En [Steinhilber et al., 2009] se tomó como marco de trabajo el algoritmo de ponderación elaborado por López pero se aplicaron algoritmos genéticos para mejorar el rendimiento.

A pesar de que cada una de estas modificaciones o extensiones aplicadas al sistema LOPE ha mejorado el rendimiento del mismo, hay ciertos aspectos del modelo que aún no han sido abordados y también algunos procedimientos que podrían ser refinados. El tratamiento de estos aspectos es el objetivo de esta tesis. En las secciones subsiguientes se describen las cuatro oportunidades de mejora que se han identificado a la largo de la investigación.

3.2 OCURRENCIA DE GIROS INEFICIENTES

La primera de las debilidades se ubica tanto en el sistema planificador como en el ejecutor. Ella se refiere a la ineficiencia que puede generarse en la ejecución de acciones por parte del robot, dado que el modelo carece de un mecanismo que permita evitar los giros ineficientes. Es decir, la arquitectura no tiene en cuenta la acción previamente ejecutada a la hora de decidir la próxima acción a ejecutar, ya sea cuando el sistema esté actuando por contingencia o cuando la acción a ejecutar sea parte de un plan armado.

En relación a la modalidad de Contingencia esta debilidad se ve fácilmente, ya que el algoritmo para seleccionar la acción subsiguiente sólo se basa en el resultado de una función de aleatoriedad, a partir del cual se elige automáticamente alguna de las tres posibles acciones. Pero al ser un mecanismo azaroso puede suceder que en reiteradas oportunidades ocurran casos en los que se Gira a la Izquierda (GI) y luego se Gira a la Derecha (GD), o viceversa. Dicho situación, a pesar de no afectar seriamente al modelo, genera una pérdida innecesaria de tiempo de cómputo.

En la otra modalidad, cuando el sistema se encuentra en el proceso de armado de un plan, cabe la posibilidad de que suceda algo similar, pero en este caso se detectan más casos de ineficiencia. Suponiendo, por ejemplo, que el robot armó un plan de 5 pasos, podrían darse las siguientes situaciones: (i) que el primer paso del plan sea GD y que la última acción ejecutada (ya sea de un plan de Contingencia o de un Plan Armado) sea GI; (ii) que la segunda acción del plan sea GI y la tercera sea GD; (iii) que cada una de las cinco acciones sean todas GD o GI, resultando en un giro de 450°.

Todos los casos mencionados, bajo cualquiera de las modalidades descritas, son plausibles de que ocurran si no se implementan mecanismos para evitarlos. Un posible abordaje a este tipo de problemas es la implementación de algoritmos que tengan en cuenta las acciones pasadas y que también permitan descartar, de antemano, planes que contengan acciones inútiles. Dicha solución se describe en la sección 4.3.

3.3 GESTIÓN INCOMPLETA DE TEORÍAS MUTANTES

La segunda debilidad se manifiesta en el sistema de aprendizaje del modelo, y está vinculada a la administración incompleta de las teorías mutantes. De acuerdo al modelo original, el aprendizaje del sistema comienza con la percepción del entorno y el posterior armado de la teoría local correspondiente. Una vez que ésta ha sido armada, el sistema la compara con la base de teorías registradas para decidir la acción que llevará a cabo: (a) creación de una nueva teoría, (b) premiación de dicha teoría o (c) creación de una teoría similar y su correspondiente mutante. Ahora bien, tal como ha sido diseñado el sistema, las teorías mutantes sólo pueden ser creadas (caso *c*) pero no premiadas. Esto es así porque para que una teoría sea premiada, deben darse dos condiciones. La primera de ellas es, evidentemente, que la situación percibida coincida con la situación esperada del plan. La segunda condición es que el robot verifique que la teoría local armada ya se encuentra registrada dentro de su base de conocimiento. Mientras que la primera condición no genera ningún inconveniente, la segunda pasa por alto algo importante, que la teoría local armada nunca será de tipo mutante, ya que lo que perciben los sensores son situaciones específicas y "reales", no situaciones creadas por el sistema. Es decir, la construcción de una teoría mutante es el resultado de la aplicación de una determinada heurística por parte del módulo de aprendizaje, o sea, el procesamiento es posterior a la acción de observación y percepción. Por lo tanto, si por definición una teoría local no puede ser mutante, nunca se buscará una teoría de este tipo en la base de teorías registradas y, en consecuencia, nunca se premiará una teoría mutante.

A partir de lo expuesto, es evidente que el modelo LOPE hace hincapié en el uso de teorías normales (no mutantes) para el armado de los planes, ya que a medida que se ejecutan los ciclos, el *P* de cada teoría mutante siempre será igual a 1 y, conforme vayan surgiendo nuevas teorías similares, irá disminuyendo la probabilidad de éxito (*P/K*) de cada mutante. Esto repercutirá negativamente sobre la probabilidad de que una teoría mutante sea elegida para formar parte de un plan, y con el tiempo serán despreciadas por el planificador.

El autor propone la elaboración de un nuevo algoritmo que permita un mayor aprovechamiento de las teorías mutantes y una premiación pareja en comparación con las otras teorías. Esto se considera importante ya que una teoría mutante bien utilizada puede aumentar la probabilidad de éxito de un plan, debido a que cada situación esperada mutante, incluye al menos dos observaciones que permitirían alcanzar lo buscado, y por ende incrementaría de antemano la probabilidad de éxito de la acción a ejecutar. La solución a este problema se aborda en la sección 4.4.

3.4 FALTA DE EVALUACIÓN DE LOS PLANES EJECUTADOS

La tercera de las debilidades se dispara dentro del módulo de control y ejecución, más precisamente en la verificación que realiza el sistema al momento de finalización de un plan (con éxito o fracaso). De acuerdo al modelo, una vez que concluye el armado de un plan y se ejecutan todos los subprocesos del planificador, se selecciona la primera acción y se la ejecuta sobre el entorno. El control pasa entonces al sistema sensor que percibe la nueva situación, luego al módulo de aprendizaje, el cual arma la teoría local y premia o registra nuevas teorías según corresponda, y finalmente al controlador de ejecución. En esta última instancia se realiza la verificación que comprueba si la acción ejecutada resultó en la situación esperada o no. En caso afirmativo, el plan continúa, caso contrario se aborta el plan en ejecución y se procede a armar uno nuevo.

De lo descrito se deduce que en el caso en que uno de sus pasos no haya logrado la situación esperada, el sistema sólo se limita a abortar el plan y a devolver el control al planificador. De esta forma se pierde la oportunidad de refinar y retroalimentar el aprendizaje del modelo a partir de la asignación de premios y/o castigos a las teorías involucradas en los planes exitosos o abortados. La solución a este problema se describe en la sección 4.5.

3.5 INDICE DE CONFIABILIDAD ESTÁTICO

La cuarta debilidad se encuentra en el módulo de planificación y está relacionada con la etapa de *Evaluación de Calidad del Plan*. Esta tarea, previa a la fase de inicialización del plan armado, tiene

como objetivo evitar que el sistema ejecute planes que tengan un índice de confiabilidad bajo, o en otras palabras, que tiendan a fracasar. La evaluación se lleva cabo a partir de una comparación entre la probabilidad de éxito obtenida a partir de la matriz de transición asociada (matriz que mide la probabilidad de llevar a cabo todas las acciones del plan y alcanzar la situación esperada) y el valor del indicador de confiabilidad predefinido por el sistema. En caso que la probabilidad de éxito calculada para el plan sea mayor o igual al índice de confiabilidad, el sistema inicia la ejecución del plan; caso contrario, intenta armar un nuevo plan eligiendo la siguiente situación deseable de la pila armada.

En relación al criterio utilizado para evaluar la calidad (o probabilidad de éxito), se observa que el parámetro que sirve de medida de comparación (índice de confiabilidad) es un valor estático a lo largo del ciclo de vida del robot. Pero cabe preguntarse, ¿qué sucedería si como diseñadores del sistema observáramos que una gran cantidad de planes ha fracasado? En ese caso, ¿no sería conveniente que el umbral de confiabilidad se incremente para evitar que se ejecuten planes condenados al fracaso? El autor considera que la respuesta es afirmativa y que la solución propuesta (sección 4.6) debe incluir un indicador de confiabilidad dinámico.

4. SOLUCIÓN

Se presenta en este capítulo el contexto de las soluciones abordadas (sección 4.1); la modificación del sistema sensor implementado (sección 4.2); la mejora implementada para evitar la ejecución de giros ineficientes (sección 4.3); la extensión al modelo que permite una administración completa de las teorías mutantes (sección 4.4); la extensión que castiga a las teorías que han provocado el fracaso de los planes (sección 4.5); la extensión que incorpora un índice de confiabilidad dinámico (sección 4.6); y por último, la propuesta del índice de mejora que permitirá evaluar y comparar los resultados de la experimentación (sección 4.7).

4.1 CONTEXTO DE LAS SOLUCIONES ABORDADAS

De acuerdo a lo descrito en el capítulo anterior, el modelo LOPE [García-Martínez y Borrajo; 1997; 2000] presenta cuatro oportunidades de mejora para optimizar, principalmente, los procesos de planificación y aprendizaje del sistema.

La primera de las mejoras está relacionada con la prevención de la ejecución de giros ineficientes. En este caso, los nuevos algoritmos son incorporados tanto en el módulo de planificación como de control y ejecución.

La segunda mejora consiste en la incorporación de una completa administración de teorías mutantes, que incluye la premiación de las mismas cuando correspondiese y la creación de teorías mixtas, surgidas como resultado de la concatenación entre las teorías involucradas en la planificación y las obtenidas a partir de la percepción del medio. Esta extensión es implementada enteramente en el módulo de aprendizaje del sistema.

La tercera de las mejoras está vinculada a los resultados de los planes ejecutados, o mejor dicho, a los planes que han sido abortados. En este sentido, el mecanismo que se incorpora se ocupa de castigar a la teoría que provocó el fracaso del plan y de actualizar la ponderación de las teorías similares y complementarias. La implementación de esta extensión se ubica en el módulo de aprendizaje del sistema, siendo disparada desde el módulo de control y ejecución.

La cuarta y última mejora consiste en un refinamiento de la etapa de evaluación del plan a partir de la definición de un índice de confiabilidad dinámico, para así aumentar la probabilidad de éxito de los planes ejecutados. Su implementación se desarrolla por completo dentro del módulo de planificación y es disparada cuando el plan ha finalizado, ya sea exitosamente o cuando ha sido abortado.

4.2 IMPLEMENTACIÓN DEL SISTEMA SENSOR

La arquitectura base del sistema, a la cual se le agregan las cuatro mejoras indicadas, es idéntica al modelo original salvo por una modificación en el sistema de percepción. En el modelo original, la observación del medioambiente da como resultado un vector de 8 valores (figura 2.5), mientras que en la implementación actual, la percepción del entorno se resume en 5 valores. En la figura 4.1 se presenta el esquema del nuevo sistema sensor.

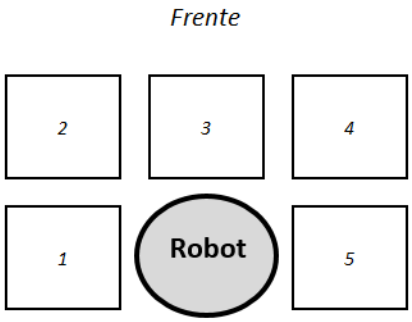

Figura 4.1 – Nuevo sistema sensor del robot explorador

Las cuadrículas 1 a 5 del sensor se corresponden con los índices del vector de la situación percibida, cada uno de las cuales devuelve un 1 (uno) en caso de percibir un objeto o un 0 (cero) en caso contrario. Si el robot encontrase un objeto a su izquierda, ello será representado por el siguiente vector de situación: [1,0,0,0,0]. En el caso que perciba un objeto a su derecha, [0,0,0,0,1] será su vector correspondiente. Por último, la identificación de un objeto delante de él se representará con el vector [0,0,1,0,0].

4.3 PREVENCIÓN DE GIROS INEFICIENTES

La mejora descrita en la presente sección tiene como objetivo incrementar la eficiencia del modelo en lo que respecta a la ejecución de acciones, particularmente a los giros que realiza el robot. Las modificaciones se implementan dentro de las dos modalidades posibles: i) acción por contingencia y ii) acción a partir de un plan armado.

El primer caso surge cuando no se ha armado ningún plan y el sistema tiene que decidirse por alguna de las tres posibles acciones de forma aleatoria (*Avanzar, Girar a la Derecha* o *Girar a la Izquierda*). Como se mencionó en el capítulo previo, la existencia de aleatoriedad puede ocasionar que el robot gire primero hacia un lado y en el paso siguiente gire nuevamente pero en el sentido contrario. Este comportamiento no produce ningún tipo de información nueva para el sistema, sino que sólo genera

pérdida de tiempo de procesamiento. Por lo tanto, con la nueva implementación, antes de elegir la acción por azar el sistema determina cuáles de ellas se encuentran habilitadas, de acuerdo a la historia de las acciones ejecutadas. Una vez que se dispone de dicha información, se aplica la función de aleatoriedad al conjunto de acciones que ya ha sido filtrado y se obtiene la acción que será ejecutada.

La mejora implementada para la segunda modalidad también tiene como objetivo la prevención de giros ineficientes pero, en este caso, la prevención se lleva a cabo dentro del módulo de planificación. Es decir, por construcción, un plan podría resultar en el siguiente ordenamiento de acciones: *GirarDerecha-GirarDerecha-GirarIzquierda-GirarIzquierda*. Al igual que en el caso anterior, si este plan se ejecutase completamente, las últimas dos acciones no producirían ninguna información provechosa para el sistema. Por lo tanto, en este caso la mejora desarrollada analiza el interior de cada uno de los posibles planes que forman parte del árbol de situaciones asociado, y decide accionar de acuerdo al siguiente protocolo:

Un plan es descartado si presenta alguna de las siguientes características:

- Su acción inicial es un giro y la acción previa (anterior al plan actual) fue un giro en sentido contrario;
- Alguna de sus acciones es un giro y la acción siguiente es un giro en sentido contrario;
- Está compuesto por cinco o más acciones y cada una de ellas son giros en el mismo sentido, resultando en una rotación final de 450° y 540°.

En los siguientes apartados se describe la arquitectura y el diagrama de la mejora descrita (apartado 4.3.1) y se introducen sus algoritmos de alto nivel (apartado 4.3.2).

4.3.1 Arquitectura propuesta

La arquitectura que se presenta a continuación (figura 4.2) incluye las implementaciones para ambas modalidades descritas, la primera dentro del módulo de control y ejecución y la segunda en el módulo de planificación. La diferencia respecto al modelo LOPE se encuentra en los dos subprocesos resaltados en la figura: *Obtiene Acciones Habilitadas* y *Filtra Posibles Planes*. Todos los procesos anteriores a esos agregados son idénticos al del modelo original y también lo son aquellos que se ejecutan posteriormente. Para que se inicie la tarea que permite obtener las acciones habilitadas, es necesario que previamente se haya activado el plan de Contingencia. Por otra parte, para que se dispare el filtrado de los posibles planes, sólo se requiere que se inicie normalmente el proceso de planificación. A continuación se describen los posibles caminos que permiten la ejecución de los nuevos subprocesos incorporados.

Para iniciar la etapa de planificación hay tres posibilidades: (1) que la acción previa no haya sido parte de un plan en ejecución, es decir, que anteriormente se haya actuado por contingencia; (2) que la acción previa sea la última de un plan en ejecución y que la misma haya alcanzado la situación esperada (plan finalizado exitosamente); y (3) que la acción previa sea parte de un plan en ejecución pero la misma no haya alcanzado la situación esperada (plan abortado). En cualquiera de los tres casos, el próximo paso que ejecuta el sistema es el de armar la pila de situaciones deseables. Luego, toma la situación deseable con mayor utilidad, arma el árbol asociado y a continuación ejecuta el primer subproceso de la mejora: el filtrado de los posibles planes según el protocolo indicado en el apartado previo. En el paso siguiente, el sistema verifica que exista un camino hacia la situación deseada. En caso que esto no ocurra, toma la siguiente situación deseable y repite el procedimiento. En caso afirmativo, arma el plan, construye su matriz de valoración y evalúa la calidad del mismo. Si el plan no resultase confiable se busca la siguiente situación deseable de la pila y se repiten los pasos ya descritos; en caso que el plan supere el umbral de confiabilidad, se selecciona la acción del plan a ejecutar, se pasa el control al sistema ejecutor y se ejecuta la acción sobre el entorno. En cualquiera de los dos casos en que el sistema regresa a buscar una nueva situación deseable en la pila armada, puede suceder que no queden más situaciones disponibles. En tales casos, el robot procede a activar el plan de Contingencia y a continuación ejecuta el segundo de los subprocesos de la mejora actual: la obtención de las acciones habilitadas antes de que sea ejecutada la función de aleatoriedad.

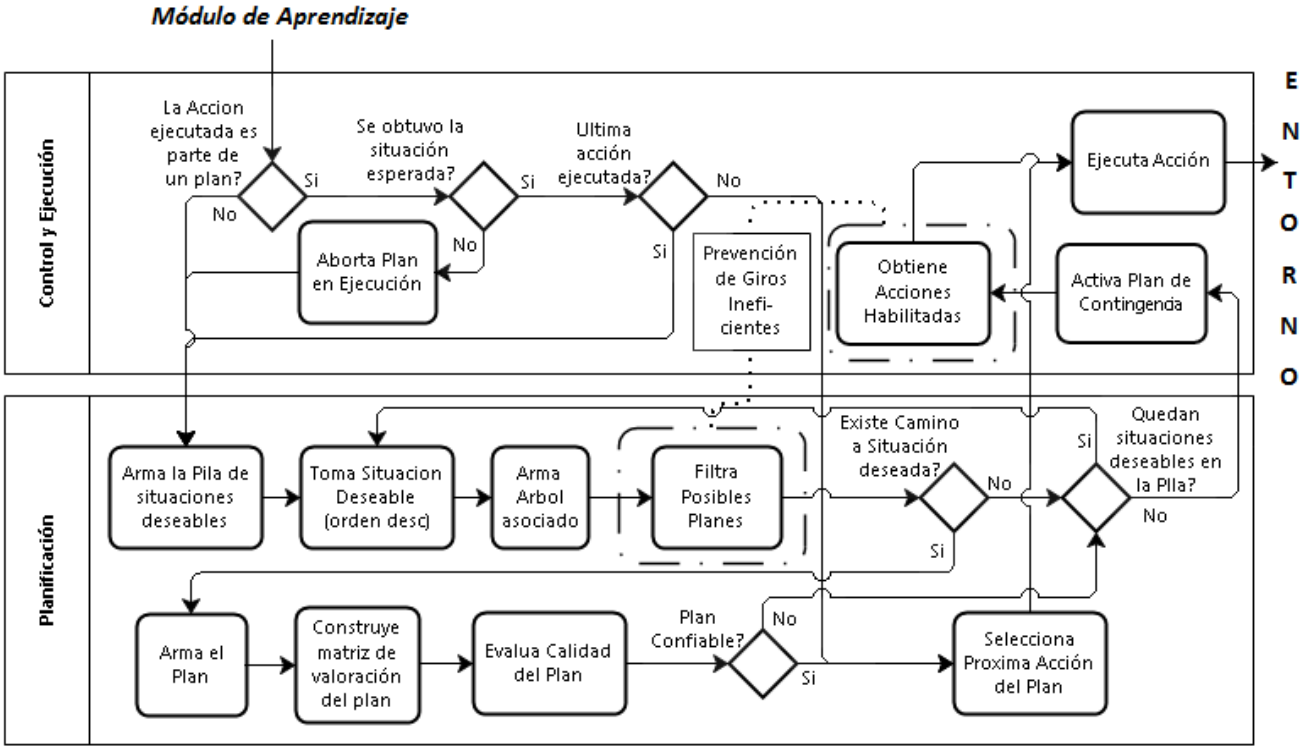

Figura 4.2 – Diagrama de Proceso: Mejora de la Prevención de Giros Ineficientes

4.3.2 Algoritmos

A continuación se presentan dos algoritmos de alto nivel pertenecientes a la mejora de la prevención de giros ineficientes. El primero de ellos, *ObtieneAccionesHabilitadas* (figura 4.3), corresponde a la mejora desarrollada para los casos de acción por Contingencia y es implementado dentro del módulo de control y ejecución. El segundo, *FiltraPosiblesPlanes* (figura 4.4), es parte de la prevención de giros ineficientes dentro del módulo de planificación.

Función ObtieneAccionesHabilitadas (*st,ac,H*): *H*

st: situación percibida
ac: acción ejecutada en el instante previo
H: vector de acciones habilitadas
ad: Verdadero si hay un bloque adelante
de: Verdadero si hay un bloque a la derecha
iz: Verdadero si hay un bloque a la izquierda

```
// i) Verifica si hay obstáculos Adelante, Derecha e Izquierda
ad = ExisteObstaculo(st,3)
de = ExisteObstaculo(st,5)
iz = ExisteObstaculo(st,1)

// ii) Deshabilita Avanzar, si corresponde
Si ad Es Igual a VERDADERO
     DeshabilitaAccion(H,"AV")
Fin Si

// iii) Deshabilita Girar a la Derecha, si corresponde
Si  (ac Es Igual a "GI")  O
    (ad Es Igual a FALSO Y ac Es Distinto de "AV")  O
    (ad Es Igual a VERDADERO Y de Es Igual a VERDADERO Y
     iz Es Igual a FALSO Y ac Es Distinto a "GD")
        DeshabilitaAccion(H,"GD")
Fin Si

// iv) Deshabilita Girar a la Izquierda, si corresponde
Si  (ac Es Igual a "GD")  O
    (ad Es Igual a FALSO Y ac Es Distinto de "AV")  O
    (ad Es Igual a VERDADERO Y iz Es Igual a VERDADERO Y
     de Es Igual a FALSO Y ac Es Distinto a "GI")
         DeshabilitaAccion(H,"GI")
Fin Si

Devuelve H
```

Figura 4.3 – Algoritmo que Obtiene las acciones Habilitadas

La función *ObtieneAccionesHabilitadas* (figura 4.3) está compuesta por cuatro tareas distintas. La primera tarea es comprobar la existencia o no de obstáculos a su alrededor. Para ello se ejecuta la función *ExisteObstaculo* para cada una de las tres ubicaciones posibles (adelante, derecha e izquierda), la cual devuelve una variable booleana: *verdadero* si encontró un obstáculo o *falso* en caso contrario. Esta última función recibe dos argumentos: el primero de ellos es el vector de la situación percibida (*st*) y el segundo es la posición a evaluar dentro ese vector. De acuerdo a la figura 4.1, para comprobar la existencia de un bloque a la izquierda es necesario pasar un 1 como argumento, cuando se verifica la existencia de un obstáculo delante se deberá pasar un 3, y en el caso de la comprobación del sector derecho del robot se pasará un 5 como argumento. La ejecución de estas tres primeras funciones conforma la primera tarea de la función *ObtieneAccionesHabilitadas*.

La segunda tarea consiste simplemente en deshabilitar la acción *Avanzar* (AV) si existe un obstáculo delante. Para ello se utiliza el resultado de la función *ExisteObstaculo(st,3)*.

La tercera tarea corresponde al procedimiento utilizado para deshabilitar la acción *Girar a la Derecha* (GD). Las tres condiciones para deshabilitar dicha acción son las siguientes: (i) la acción previa es igual a *Girar a la Izquierda* (GI); (ii) no existe un obstáculo delante y la acción previa no fue AV (esta condición está orientada a evitar que el robot gire 180° y vuelva sobre sus pasos, por ejemplo, cuando haya llegado al final de un pasillo del cual podría girar, avanzar y salir); y (iii) existe un obstáculo delante y otro a la derecha y la acción previa no fue GD (esta condición evita que el robot gire sobre su eje por querer girar hacia el lado en el que no hay un obstáculo).

La cuarta y última tarea se refiere al procedimiento para deshabilitar la acción *Girar a la Izquierda.* El criterio utilizado en este caso es análogo al utilizado para deshabilitar la acción GD.

El segundo de los algoritmos desarrollados para la mejora de prevención de giros ineficientes corresponde a la función *FiltraPosiblesPlanes* (figura 4.4). Esta función es implementada dentro del módulo planificador, luego de que haya concluido la construcción del árbol. El algoritmo analiza cada uno de los planes incluidos en el árbol y decide deshabilitarlos o no dependiendo del resultado de las comprobaciones. Los planes evaluados se descartan si alguna de las siguientes condiciones resulta verdadera: (i) la acción anterior fue un giro y la primera acción del plan evaluado es un giro en sentido contrario; (ii) en el plan evaluado existe alguna secuencia de acciones del tipo GD-GI o GI-GD; y (iii) en la composición del plan se observa alguna combinación de acciones del tipo GD/GI x 5 (equivalente a una rotación final de 450°. En este caso la última acción no aporta nada nuevo, sería más eficiente realizar un sólo giro GD/GI) o GD/GI x 6 (equivalente a una rotación final de 540°. Sucede lo mismo que en el caso anterior con las últimas dos acciones, sería más eficiente ejecutar solamente dos giros GD/GI).

Función FiltraPosiblesPlanes (*P*,*ac*): *P*

P: vector de Posibles Planes
ac: acción ejecutada en el instante previo

Para Todo $p_i \in P$

// i) Descarta planes cuyo acción inicial es GD/GI cuando la última acción ejecutada fue GI/GD
Si (*ac* **Es Igual a** "GD" **Y** $p_i(A_1)$ **Es Igual a** "GI") **O**
(*ac* **Es Igual a** "GI" **Y** $p_i(A_1)$ **Es Igual a** "GD")
DeshabilitaPosiblePlan*(P,i)*
Fin Si

// ii) Descarta planes con acciones consecutivas GI-GD y/o GD-GI
Para Todo $A_i \in A$ **tal que** $1 =< i <= 5$
Si (A_i = "GD" **Y** A_{i+1} = "GI") **O**
(A_i = "GI" **Y** A_{i+1} = "GD")
DeshabilitaPosiblePlan*(P,i)*
Fin Si
Fin Para Todo

// iii) Descarta planes que pretenden dar rotaciones totales de 450° y 540°

// Giros de 450°
Para Todo $A_i \in A$ **tal que** $1 =< i <= 2$
Si (A_i = "GD" **Y** A_{i+1} = "GD" **Y** A_{i+2} = "GD"
Y A_{i+3} = "GD" **Y** A_{i+4} = "GD") **O**
(A_i = "GI" **Y** A_{i+1} = "GI" **Y** A_{i+2} = "GI"
Y A_{i+3} = "GI" **Y** A_{i+4} = "GI")
DeshabilitaPosiblePlan*(P,i)*
Fin Si
Fin Para Todo

// Giros de 540°
Si (A_i ="GD" **Y** A_{i+1} ="GD" **Y** A_{i+2} ="GD" **Y** A_{i+3} ="GD"
Y A_{i+4} = "GD" **Y** A_{i+5} = "GD") **O**
(A_i ="GI" **Y** A_{i+1} ="GI" **Y** A_{i+2} ="GI" **Y** A_{i+3} ="GI")
Y A_{i+4} = "GI" **Y** A_{i+5} = "GI") **O**
DeshabilitaPosiblePlan*(P,i)*
Fin Si

Fin Para Todo

Devuelve *P*

Figura 4.4 – Algoritmo que Filtra los posibles Planes del árbol

4.4 ADMINISTRACIÓN COMPLETA DE TEORÍAS MUTANTES

La siguiente extensión al modelo tiene como objetivo mejorar y ampliar el alcance de la administración de las teorías mutantes. La mejora se implementa enteramente en el módulo de aprendizaje y se dispara cuando la acción previamente ejecutada es parte de un plan y su teoría asociada es del tipo mutante. Una teoría se define como mutante cuando su situación inicial y/o final lo son.

La solución desarrollada se divide en dos fases. La primera de ellas aborda la falta de premiación de las teorías mutantes y la segunda se ocupa de la creación y premiación de un nuevo tipo de teorías llamadas *mixtas.* Para lanzar la mejora, un paso previo indispensable es que el sistema identifique si el tipo de teoría involucrada en el plan en ejecución es mutante o no. En el caso que se trate de una teoría normal se sortea la mejora y el sistema procede a ejecutar todas las tareas del módulo de aprendizaje del mismo modo que lo hiciera bajo el modelo original. Ahora bien, el caso de interés aparece cuando la teoría ejecutada por el plan es efectivamente una mutante. De ser así, y suponiendo que fue exitosa su ejecución, el sistema ejecuta todas las tareas que aplican sobre la teoría percibida por los sensores (correspondiente a la arquitectura original) y a ellas adiciona el procedimiento de premiación de la teoría mutante. Con esta primera fase, se logra salvar la diferencia de criterio que se observaba en la distribución de premios sobre las teorías normales y las mutantes.

Por otro lado, con el fin de refinar la explotación de la información que recibe el sistema y con ello incrementar el número de teorías, se propone una segunda fase dentro de la solución. Ella se refiere al entrecruzamiento de las situaciones iniciales y finales de las teorías armadas por los sensores y las teorías mutantes que fueron ejecutadas por los planes. A estas teorías se les da el nombre de *teorías mixtas* y a cada una de ellas se le aplica el mismo tratamiento que a las anteriores, es decir, dependiendo de cuál sea el caso, se registrará un nueva teoría mixta, se registrará una nueva teoría mixta similar o se la premiará, en caso que corresponda.

En los siguientes apartados se describe la arquitectura y se presenta el diagrama detallado de la mejora (apartado 4.4.1), se detallan los algoritmos más importantes (apartado 4.4.2) y se incluye un ejemplo integrador (apartado 4.4.3).

4.4.1 Arquitectura propuesta

La nueva arquitectura se describe en la figura 4.5 y se la distingue con un recuadro especial. La figura muestra las dos fases de la mejora: la primera de ellas compuesta por los subprocesos *Premia Teoría Mutante* e *Incrementa K de Teorías Similares* y la segunda por los subprocesos *Obtiene*

Teorías Mixtas en adelante. En la parte superior de la figura se indican las tareas pertenecientes al modelo original del proceso de aprendizaje (que permanece sin cambios), el cual sigue siendo aplicado a las teorías locales armadas a partir de los sensores.

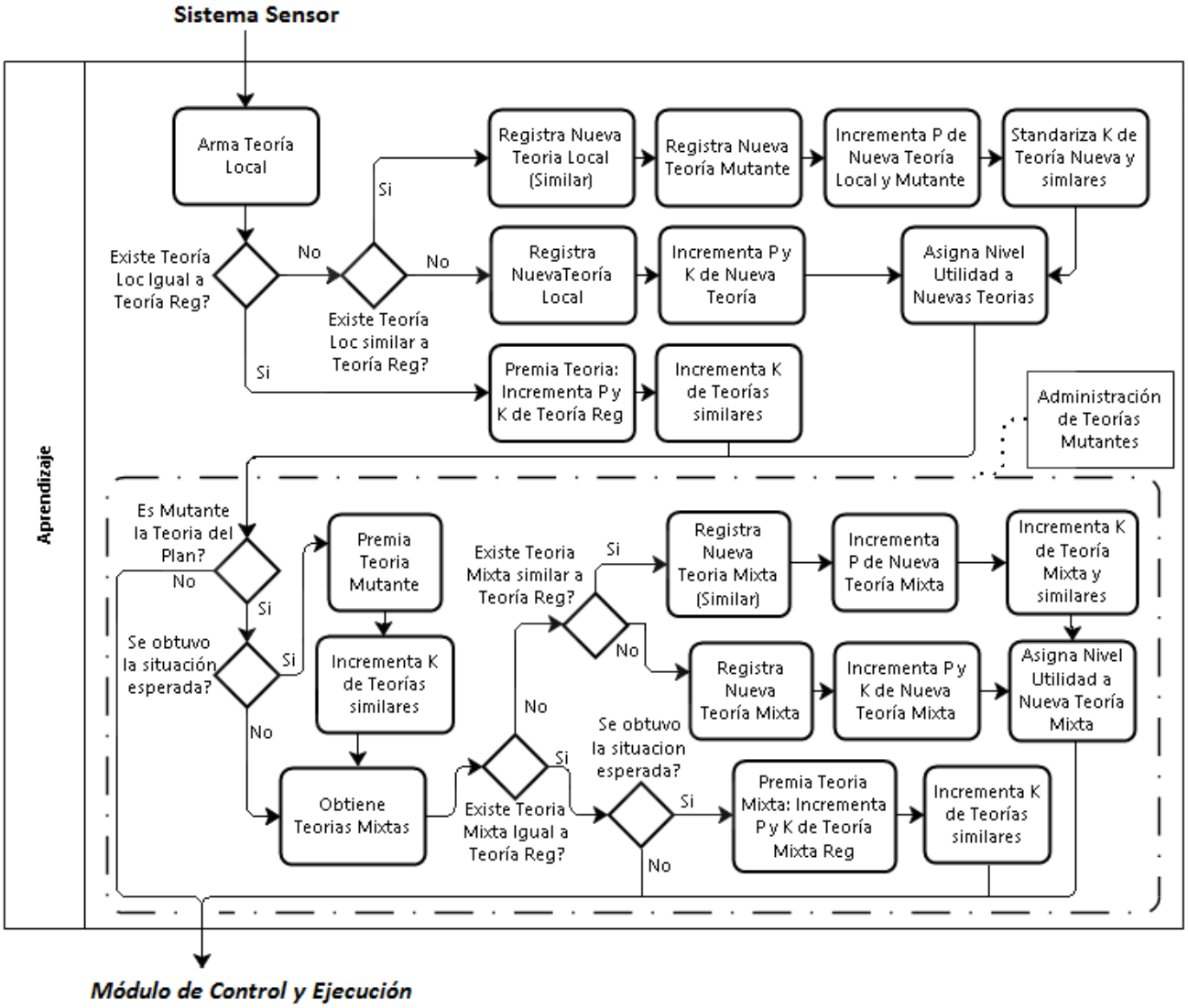

Figura 4.5 – Diagrama de Proceso: Extensión de la Administración de Teorías Mutantes

Como se observa en la figura, la mejora se dispara cuando el sistema percibe que la acción ejecutada por el plan corresponde a una teoría mutante. En caso de no serlo, el flujo de ejecución pasa directamente del procedimiento convencional de administración de teorías al módulo de control y ejecución. Cuando la teoría es del tipo mutante se inicia la nueva implementación. En esta situación, la primera verificación que ejecuta el sistema es la comprobación de haber alcanzado o no la situación esperada. En caso afirmativo, se activa la primera etapa, en la cual se premia la teoría mutante y luego se incrementa su *K* y el de sus teorías similares, si es que presenta. A continuación

se ejecuta la segunda fase, la cual también es iniciada en caso que la verificación mencionada resulte negativa y se pase directamente a ella. Esta segunda etapa lleva adelante la obtención de las teorías mixtas. Dicha tarea puede resultar en la identificación de 0 (cero) a 4 teorías mixtas, dependiendo del caso (el algoritmo presentado en la figura 4.7 describe este procedimiento con mayor detalle). En caso de haber obtenido al menos una teoría mixta, la ejecución continúa de forma análoga al proceso de aprendizaje estándar, verificando si existe alguna teoría igual a ella, si existe alguna teoría similar y registrando nuevas teorías o premiándolas dependiendo de cuál sea el caso.

4.4.2 Algoritmos

En este apartado se presentan los dos algoritmos pertenecientes a la mejora en cuestión. El primero de ellos, *AdministraTeoriasMutantes* (figura 4.6), engloba la solución completa de la extensión propuesta. El segundo, *ObtieneTeoríasMixtas* (figura 4.7), describe en detalle el mecanismo por el cual se obtiene cada una de las posibles teorías mixtas.

La función *AdministraTeoriasMutantes* está compuesta por dos tareas, sin embargo, ninguna de ellas inicia su ejecución si la teoría ejecutada recientemente por el plan no es una teoría mutante. Esta verificación la realiza la función *EsTeoriaMutante(tp).* Si la comprobación resulta verdadera, se procede a ejecutar la primera tarea: la premiación de la teoría mutante. Dicha acción sólo se lleva a cabo, como es de suponer, si se ha alcanzado la situación esperada. En tal caso, se premia la teoría aumentando en 1 su *P*, se obtiene la identificación común de sus teorías similares (situación inicial + acción) y se actualiza el *K* de la teoría que se premió y sus similares, si es que tuviese.

La segunda tarea consiste, primeramente, en obtener las identificaciones de las posibles teorías mixtas. Para ello se ejecuta la función *ObtieneTeoriasMixtas(tp,ts)*. Esta función (en detalle en la figura 4.7) intenta construir cuatro teorías nuevas. La primera de ellas estaría compuesta por la situación inicial (*si*) de la teoría armada por los sensores (*ts*) y la situación final (*sf*) de la teoría que corresponde al paso actual del plan (*tp*). La segunda se armaría a partir de la *si* de la *tp* y la *sf* de la *ts*. Para construir cada una de estas dos teorías son necesarias dos condiciones. En el primer caso es requisito que la *sf* de la *tp* sea mutante y, en el segundo caso, se necesita que la *si* de la *tp* también lo sea. Esto es así ya que sino, se estarían creando teorías redundantes. Luego se sigue con la construcción de la tercera y cuarta teorías mixtas, pero en este caso es necesario un paso previo, la obtención de la situación mutante entre la situación final de la *ts* y la situación final de la *tp*. Esta tarea se lleva a cabo mediante la función *IDMutante(ts(sf),tp(sf))*. Habiendo sido creada esta nueva situación final *(sfm)*, que es la intersección entre ambas situaciones finales, se continúa con las

últimas dos teorías mixtas. La tercera de ellas se arma a partir de la *si* de la *ts* y la *sfm*, mientras que la cuarta se obtiene de la *si* de la *tp* y la *sfm*.

```
Función AdministraTeoriasMutantes (T,se,tp,ts): T
-------------------------------------------------
T:  conjunto de teorías aprendidas (incluye las mutantes).
se: Verdadero si se ha alcanzado la situación esperada
tp: ultima teoría ejecutada del plan
ts: teoría percibida por los sensores
X:  conjunto de Teorías Mixtas
s:  identificador de teoría similar
-------------------------------------------------
Si EsTeoriaMutante(tp) Es Igual a VERDADERO
    // i) Gestiona las Teorías Mutantes pertenecientes al Plan
    Si se Es Igual a VERDADERO
        PremiaTeoria(T,tp,1)
        s = IDTeoriaSimilar(tp)
        Para Todo t_i ∈ T tal que IDTeoriaSimilar(t_i) Es Igual a s
            IncrementaKTeoriaSimilar(t_i,1)
        Fin Para Todo
    Fin Si
    // ii) Gestiona las Teorías Mutantes Mixtas
    X = ObtieneTeoriasMixtas(tp,ts)
    Para Todo x_i ∈ X
        Si No Existe x_i ∈ X tal que x_i ∈ T
            InsertaNuevaTeoria(T,x_i)
            s = IDTeoriaSimilar(x_i)
            Para Todo t_i ∈ T tal que IDTeoriaSimilar(t_i) Es Igual a s
                IncrementaKTeoriaSimilar(t_i,1)
            Fin Para Todo
        Sino
            Si se Es Igual a VERDADERO
                PremiaTeoria(T,x_i,1)
                s = IDTeoriaSimilar(x_i)
                Para Todo t_i ∈ T tal que IDTeoriaSimilar(t_i) Es Igual a s
                    IncrementaKTeoriaSimilar(t_i,1)
                Fin Para Todo
            Fin Si
        Fin Si
    Fin Para Todo
Fin Si
Devuelve T
```

Figura 4.6 – Algoritmo de alto nivel para la Administración de las Teorías Mutantes

Cada una de estas nuevas teorías identificadas es cargada al vector de las teorías mixtas mediante la función *CargaTeoriaMixta(x_i,X)*. Luego, al devolver el flujo de ejecución a la función *AdministraTeoríasMutantes* se utiliza este vector para ir verificando si cada una de ellas existe dentro del conjunto de teorías registradas o no. En caso que no exista se crea y se verifica si existen teorías similares para ajustar sus correspondientes *K*s. En caso contrario, primero se confirma que se ha alcanzado la situación esperada. Si este ha sido el caso se premia la teoría y se procede a actualizar los *K*s; caso contrario, se saltea este paso.

Función ObtieneTeoriasMixtas (*ts,tp*): *X*

ts: teoría armada a partir del sistema sensor
tp: teoría que corresponde al paso actual del plan en ejecución
X: conjunto de Teorías Mixtas
x_i*:* teoría mixta a agregar
sfm: ID Mutante entre las situaciones finales de la *Ts* y de la *Tp*

```
// i) Mixta 1: Teoría formada con la SI de Ts y la SF de Tp
Si EsMutante(tp(sf)) Es VERDADERO
    xi = IdentificaTeoriaMixta(ts(si), ts(ac), tp(sf))
    X = CargaTeoriaMixta(xi,X)
Fin Si

// ii) Mixta 2: Teoría formada con la SI de Tp y la SF de Ts
Si EsMutante(tp(si)) Es VERDADERO
    xi = IdentificaTeoriaMixta(tp(si), tp(ac), ts(sf))
    X = CargaTeoriaMixta(xi,X)
Fin Si

sfm = IDMutante(ts(sf), tp(sf))

// iii) Mixta 3: Teoría formada con la SI de Ts y la SFM
xi = IdentificaTeoriaMixta(ts(si), ts(ac), sfm)
X = CargaTeoriaMixta(xi,X)

// iv) Mixta 4: Teoría formada con la SI de Tp y la SFM
xi = IdentificaTeoriaMixta(tp(si), tp(ac), sfm)
X = CargaTeoriaMixta(xi,X)

Devuelve X
```

Figura 4.7 – Algoritmo que obtiene las Teorías Mixtas

4.4.3 Ejemplo Integrador

Se considera la siguiente situación. El robot se encuentra en una posición donde la situación percibida es la 00110. En dicho instante, t=0, el sistema no tiene un plan en ejecución por lo que procede a armar un plan. El plan resultante consta de 2 pasos. La primera acción a ejecutar es GD y la situación esperada es 0-010 (mutante), la segunda acción es AV y la situación esperada es 100-- (también mutante). El resumen del plan se presenta debajo.

Paso	SI	AC	SF	Teoría utilizada por el plan (TP)
1	00110	GD	0-010	00110GD0-010
2	0-010	AV	100--	0-010AV100--

El robot inicia la ejecución del plan y en el instante t=1 percibe la nueva situación, 01010. Como la situación esperada coincide con la percibida (0-010 incluye a 01010), el plan permanece activo y se ejecuta el segundo paso. El robot realiza la última acción y en el instante t=2 percibe la situación 11010. En este caso, la situación percibida no está incluida en la mutante 100--, por lo tanto el plan es abortado. Ahora bien, desde el punto de vista del aprendizaje (administración de las teorías), si no se implementa la mejora el sistema construye las siguientes teorías locales.

t	SI	AC	SF	Teoría armada por los sensores (TS)
1	00110	GD	01010	00110GD01010
2	01010	AV	11010	01010AV11010

En el instante t=1 el robot arma la teoría local 00110GD01010 y comprueba si existe o no. Suponiendo que la teoría ya existe, se premia la teoría y luego prosigue. En el instante t=2 el sistema arma la teoría local 01010AV11010 y nuevamente verifica su existencia dentro de las teorías registradas. Si se vuelve a suponer que la teoría local armada ya fue registrada previamente, en este caso el sistema omite la premiación ya que la teoría no ha sido exitosa.

En este ejemplo se observa claramente que el sistema no tiene en cuenta en absoluto la teoría 00110GD0-010 (primera teoría que compone al plan actual) ni la 0-010AV100-- (segunda teoría del plan) dentro de su proceso de aprendizaje. Así como la teoría 00110GD01010 fue exitosa y por eso se la premió, también lo fue la teoría 00110GD0-010, pero con ella no se procedió del mismo modo. Esto sucede porque la gestión del aprendizaje sólo se lleva a cabo a partir de las situaciones percibidas por los sensores y no por las situaciones que componen los planes. Esta discrepancia no aparecería si las mutantes no existiesen, pero sí sucede cuando ellas están presentes, ya que lo que percibe el robot nunca puede ser una mutante; la mutante es el resultado de un proceso interno del

modelo. En fin, al no tenerse en cuenta la teoría 00110GD0-010, el sistema no la premia, lo que resulta en un absurdo ya que la parte del plan que operó de acuerdo a lo esperado se debió a ella. Pero no sólo eso, el robot podría mejorar su aprendizaje si armara teorías combinadas que incluyan a las situaciones percibidas por los sensores y a las situaciones mutantes incluidas dentro de los planes.
Si se implementa la mejora presentada, el primer paso será la premiación de la teoría mutante 00110GD0-010, que fue la que resultó exitosa. En segundo lugar, se procederá a obtener las identificaciones de las teorías mixtas de acuerdo al algoritmo definido en la figura 4.7, que en este caso serán:

En el instante t = 1:

#	SI	AC	SF	Teoría Mixta	Acción
1	00110	GD	0-010	00110GD0-010	Ninguna. Ya existe, es igual a TP
2	N/A	N/A	N/A	N/A	N/A. La SI de TP no es mutante
3	00110	GD	0-010	00110GD0-010	Ninguna. Ya existe, es igual a TP
4	00110	GD	0-010	00110GD0-010	Ninguna. Ya existe, es igual a TP

En el instante t = 2:

#	SI	AC	SF	Teoría Mixta	Acción
1	01010	AV	100--	01010AV100--	Se registra nueva teoría Mixta
2	0-010	AV	11010	0-010AV11010	Se registra nueva teoría Mixta
3	01010	AV	1-0--	01010AV1-0--	Se registra nueva teoría Mixta
4	0-010	AV	1-0--	0-010AV1-0--	Se registra nueva teoría Mixta

Por lo tanto, con el nuevo proceso de aprendizaje propuesto, en el instante t=1 el sistema armará tres teorías mixtas de las cuatro posibles ya que una de ellas no cumple con la condición de ser mutante. Pero ninguna de las tres se registrará como una nueva teoría dado que ya existe, es la misma teoría utilizada en el primer paso del plan. Sin embargo, en el instante t=2 el sistema logra armar las cuatro teorías mixtas posibles y en este caso puede registrarlas a todas en su base de conocimiento ya que no existen. La forma de estas cuatro teorías adicionales (01010AV100--, 0-010AV11010, 01010AV1-0-- y 0-010AV1-0--) surgen de la aplicación de los pasos *i* a *iv* del algoritmo definido en la función *ObtieneTeoriasMixtas,* presentada en la figura 4.7.

4.5 CASTIGO A LAS TEORÍAS DE LOS PLANES ABORTADOS

La extensión al modelo LOPE que se presenta a continuación tiene como objetivo mejorar el aprendizaje del sistema a partir del castigo de aquellas teorías que provocan el fracaso de los planes. Para aquellos planes que hayan sido ejecutados correctamente, el proceso será idéntico al de la arquitectura original.

De acuerdo a la solución que se propone, cuando una de las teorías componentes del plan en curso fracasa, es decir, cuando la situación esperada difiere de la situación percibida, el sistema procede de la siguiente forma:

- Aumenta en 1 el *P* de la teoría percibida;
- Aumenta en 2 el *P* de la teoría mutante correspondiente al entrecruzamiento de la teoría percibida y de la teoría esperada que fracasó. En caso que aún no haya sido creada dicha teoría mutante, se la crea y luego se le aumenta el *P*;
- Aumenta en 3 el *K* de la que teoría que fracasó, de la teoría percibida, de la mutante relacionada y de las otras teorías similares, si es que existiesen.

Es decir, el sistema castiga a la teoría que fracasó premiando a la teoría percibida y a su mutante asociada. De esta forma, disminuye la ponderación o la valoración de la teoría fracasada dentro de la base total de teorías registradas.

En los siguientes apartados se detalla la arquitectura de ésta extensión al modelo (apartado 4.5.1), se describe su algoritmo de alto nivel (apartado 4.5.2) y se agrega un ejemplo integrador para clarificar la explicación (apartado 4.5.3).

4.5.1 Arquitectura propuesta

La arquitectura que se presenta a continuación (figura 4.8) incluye las tareas involucradas en el castigo a las teorías de los planes fracasados. La diferencia respecto del modelo LOPE se encuentra en los tres subprocesos que se ejecutan luego de que el plan es abortado, debido a que no se obtuvo la situación esperada. Hasta ese punto la arquitectura es idéntica al modelo original, y también lo es luego de que se ejecute la extensión incorporada.

Con el fin de contextualizar la mejora y de comprenderla integralmente, en los párrafos siguientes se describen paso a paso (desde que el robot inicia su operación) tres ciclos de percepción-aprendizaje-planificación-ejecución. Muchas de las tareas que se mencionan no se incluyen en el siguiente gráfico, ya que éstas son idénticas a las del modelo original y están contenidas en la figura 2.6.

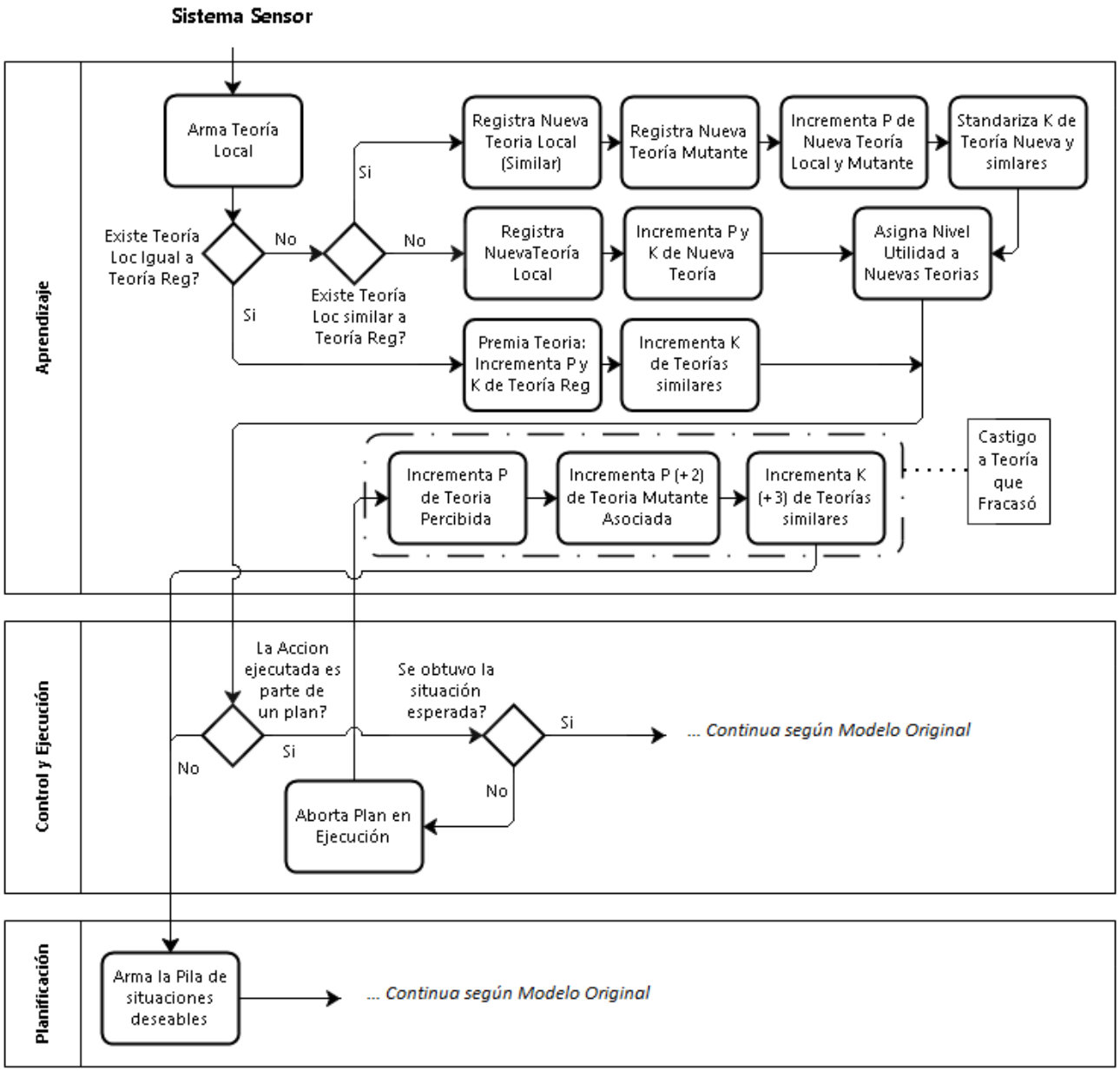

Figura 4.8 – Diagrama de Proceso: Extensión del Castigo a la Teoría que Fracasó

Al comienzo el modelo no presenta ninguna teoría armada, por lo tanto el agente ejecuta la acción inicial, que por defecto es *Avanzar*. Al cambiar su posición el sistema sensor percibe una nueva situación y con ella arma una nueva teoría local. Como el sistema aún no cuenta con una base de teorías almacenadas, se registra la nueva teoría local, se incrementa el *P* y el *K* de la nueva teoría y se le asigna el nivel de utilidad correspondiente. Dado que aún no existe un plan en ejecución, se pasa el control al módulo de planificación. Allí se arma la pila de situaciones deseables, se selecciona la situación con mayor nivel de utilidad, se arma el árbol asociado y se comprueba si existe un camino que vaya de la situación actual a la situación deseada. En caso que exista más de uno, se toma el de mínimo recorrido, se arma el plan, se construye la matriz de valoración del mismo y se evalúa su calidad. En caso que resulte ser un plan confiable (que la probabilidad de éxito sea mayor o igual a

un umbral de confiablidad previamente definido) se selecciona la próxima acción, se traspasa el control al sistema ejecutor y se lleva adelante la acción. En este punto se supone que se ha armado un plan, que consta de un sólo paso y cuya acción es la única ejecutada hasta el momento, *Avanzar*.
El agente interactúa con el entorno por segunda vez y percibe una nueva situación.
En el segundo ciclo el sistema se encuentra en una situación distinta a la inicial ya que ahora posee una teoría registrada. Por tal motivo, cuando se percibe la nueva situación, el agente tendrá que elegir entre tres posibles caminos: (1) si la nueva situación percibida resulta en una teoría igual a la registrada, entonces se premia dicha teoría, es decir, se incrementa el *P* y el *K* de la teoría registrada y se incrementa el *K* de las teorías similares (en este caso no hay teorías similares almacenadas); (2) si la nueva situación percibida resulta en una teoría similar a la registrada, entonces se registra la nueva teoría local, se registra una nueva teoría mutante, se incrementa el *P* de la nueva teoría local y mutante, se estandariza el *K* de la nueva teoría y sus similares y se asigna el nivel de utilidad a las nuevas teorías; (3) si la nueva situación percibida resulta en un teoría local que no tiene ninguna teoría igual o similar registrada, se procede de la misma forma que en el caso inicial, es decir, se registra la nueva teoría local, se incrementa el *P* y el *K* de la nueva teoría y se asigna el nivel de utilidad correspondiente. Habiendo seleccionado alguna de las tres posibles alternativas y habiendo finalizado los subprocesos descritos en cada una de ellas, el módulo de control y ejecución comprueba si la última acción ejecutada fue parte de un plan. En este caso sí lo fue, por lo tanto a continuación se verifica si se obtuvo la situación esperada. Suponiendo que la comprobación es exitosa, el sistema se pregunta si la acción ejecutada fue la última del plan. En este caso, la respuesta es afirmativa ya que este sólo incluía una acción, por lo tanto se trasfiere el control al planificador y se repiten todas las acciones mencionadas en el primer ciclo. Todo esto resulta en un nuevo plan, el cual se supone que aprueba la evaluación de calidad y, en consecuencia, el plan procede a ejecutar su primera acción sobre el entorno.
Al iniciarse el tercer ciclo también existe un plan ejecución. Por lo tanto, luego de observar la nueva situación del entorno y de ejecutar todos los subprocesos incluidos en alguno de los tres posibles caminos, se verifica si la situación percibida coincide con la situación esperada. Con el propósito de describir los nuevos procesos que incorpora la extensión propuesta, se supone que esta verificación falla. Al fallar, se procede a abortar el plan, pero a diferencia del modelo original donde el control es devuelto al planificador, en este caso la ejecución vuelve al módulo de Aprendizaje, ya que aquí es donde se incorpora el procedimiento de castigo a la teoría que provocó el fracaso del plan. Este proceso se divide en tres pasos: el primero de ellos consiste en incrementar el *P* de la teoría percibida en 1; el segundo, en incrementar el *P* de la teoría mutante asociada en 2; y el tercero, en aumentar en 3 los *K*s correspondientes a dichas teorías y sus similares.

Concluido el nuevo proceso, el sistema prosigue de acuerdo al modelo original: armará la pila de situaciones deseables; seleccionará la mejor situación deseable; armará el árbol asociado; en caso de existir un camino de mínimo recorrido entre la situación actual y la deseable se armará el nuevo plan, se construirá la matriz de valoración, se evaluará la calidad del plan y, en caso que el plan supere el umbral predefinido, se seleccionará la primera acción a ser ejecutada. Luego, se invocará al módulo de ejecución, se percibirá nuevamente el entorno y se iniciará un nuevo ciclo.

4.5.2 Algoritmos

A continuación se presenta el algoritmo de alto nivel del proceso de castigo a los planes abortados. Dicho algoritmo se ejecuta inmediatamente después de que alguna de las teorías del plan en ejecución produjo el fracaso del plan. La función *CastigaTeoriaFracaso* (figura 4.9) lleva adelante los tres pasos que han sido incorporados al modelo y que fueron descritos en la figura 4.8. Una vez finalizado el proceso, el flujo del proceso vuelve al planificador para armar la pila de situaciones deseables y todas las tareas subsiguientes ya mencionadas.

Función CastigaTeoriaFracaso (*T*,*f*,*l*): *T*

T: conjunto de teorías aprendidas (incluye las mutantes).
f: teoría que fracasó y abortó el plan
l: teoría local percibida por los sensores
m: teoría mutante asociada
s: identificador de teoría similar

```
// Verifica la existencia de la teoría mutante asociada
m = IdentificaTeoriaMutante(f,l)
Si No Existe m ∈ T
    InsertaNuevaTeoria(T,m)
Fin Si
// Premia Teoría Percibida y Mutante asociada
PremiaTeoria(T,l,1)
PremiaTeoria(T,m,2)
// Incrementa K de teorías premiadas y similares
s = IDTeoriaSimilar(f)
Para Todo t_i ∈ T tal que IDTeoriaSimilar(t_i) Es Igual a s
    IncrementaKTeoriaSimilar(T,t_i,3)
Fin Para Todo
Devuelve T
```

Figura 4.9 – Algoritmo de alto nivel del Castigo a la Teoría del Plan Abortado

Como se mencionó en el apartado previo, el algoritmo consta de tres etapas: (1) incrementar en 1 el *P* de la teoría percibida; (2) incrementar en 2 el *P* de la teoría mutante correspondiente; y (3) actualizar el *K* de las teorías premiadas y similares, incrementando su valor en 3. La función que se encarga de llevar a cabo estas tareas, *CastigaTeoriaFracaso*, toma como argumentos al conjunto de teorías aprendidas (*T*), a la teoría que fracasó y abortó el plan (*f)* y a la teoría local percibida (*l*). Previo a la ejecución de los tres pasos mencionados, la función lleva a cabo la identificación de la teoría mutante asociada al caso y la verificación de su existencia en la base de teorías registradas. En caso que ella no exista procede a crearla. A continuación se ejecuta el primer y segundo paso a través de la implementación de la función *PremiaTeoria*, la cual tiene como objetivo actualizar los *P* de acuerdo al tipo de teoría a la cual está siendo aplicada. Esta función toma como argumentos al conjunto de teorías aprendidas (*T*), a la teoría que premiará (*l* o *m)* y al valor que indica la magnitud del incremento que se le aplicará a su *P*. El algoritmo detallado de esta última función se presenta en la figura 4.10.

Función PremiaTeoria(*T*,*t*,*n*)

T: conjunto de teorías aprendidas (incluye las mutantes).
t_i = *[Si, AC, Sf, P, K, U]*
t: teoría a premiar
n: incremento a ser aplicado en *P*
p: *P* de la teoría a premiar

p = ObtienePTeoria*(T,t)*
p = p + n
ActualizaPTeoria*(T,t,p)*

Figura 4.10 – Algoritmo que Premia a una determinada teoría

Función IncrementaKTeoriaSimilar*(T,t,n)*

T: conjunto de teorías aprendidas (incluye las mutantes).
t_i = *[Si, AC, Sf, P, K, U]*
t: una de las teorías similares a actualizar
n: incremento a ser aplicado en *K*
k: *K* de la teoría a actualizar

k = ObtieneKTeoria*(T,t)*
k = k + n
ActualizaKTeoria*(T,t,k)*

Figura 4.11 – Algoritmo que Incrementa el K de una Teoría Similar

Luego se obtiene el identificador (*IDTeoriaSimilar*) que agrupa tanto a la teoría fracasada, a la mutante y a sus complementarias (si existen) y se procede a incrementar cada uno de sus *K*s en 3. La función *IncrementaKTeoriaSimilar* (figura 4.11) es la que realiza ésta última actualización.

4.5.3 Ejemplo Integrador

El siguiente ejemplo continúa el ejercicio integrador descrito en [García-Martínez; 1997]. Al final del mismo, el sistema presenta las siguientes condiciones:

- Situación actual: 00001111
- No existe un plan en ejecución
- Teorías construidas:

Teoría 1		(00000000, AVANZAR, 00000000, 1, 4, 1/3)
Teoría 2		(00000000, AVANZAR, 00001111, 2, 4, 1/2)
Teoría 3	M	(00000000, AVANZAR, 0000 - - - -, 1, 4, 1/2)
Teoría 4		(00001111, AVANZAR, 00001111, 2, 4, 1)
Teoría 5		(00001111, AVANZAR, 00000000, 1, 4, 1/3)
Teoría 6	M	(00001111, AVANZAR, 0000 - - - -, 1, 4, 1/3)

Dado que no existe un plan en ejecución, el planificador arma, a partir de las situaciones conocidas, una nueva pila de situaciones deseables dada por <situación, utilidad>. En este caso, la pila de situaciones deseables (ordenada de mayor a menor utilidad), queda expresada del siguiente modo:

Pila de Situaciones Deseables:
< 00001111, 1 >
< 00001111, 1/2 >
< 0000 - - - -, 1/2 >
< 00000000, 1/3 >
< 0000 - - - -, 1/3 >

El sistema toma la primera situación deseable y arma el árbol de situaciones asociado (figura 4.12):

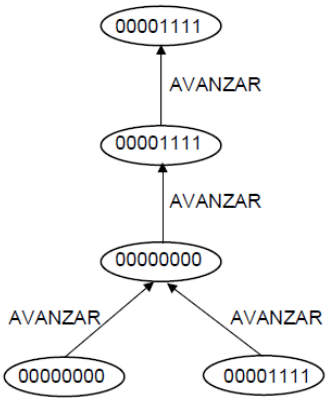

Figura 4.12 – Árbol de Situaciones asociado [García-Martinez, 1997]

Dado que la situación actual es la 00001111 y existe un camino de mínimo recorrido entre la situación actual y la situación deseable se arma el plan:

Plan = AVANZAR

Armado el plan se procede a construir la matriz de valoración para evaluar la confiabilidad del plan. En este ejemplo se asocia la fila y la columna 1 a la situación 00000000, la fila y la columna 2 a la situación 00001111 y la fila y la columna 3 con la situación 0000 - - - -.

$$M_{AVANZAR} = \begin{vmatrix} 1/4 & 1/2 & 1/4 \\ 1/4 & 1/2 & 1/4 \\ 0 & 0 & 0 \end{vmatrix}$$

Luego, según el conocimiento del que dispone el sistema, la probabilidad estimada de que aplicando el plan *Avanzar* a la situación 00001111 se obtenga la situación 00001111, es 1/2. Asumiendo que el plan es confiable (que 1/2 sea mayor o igual al índice de confiabilidad prefijado), se ejecuta la única acción del plan, que es *Avanzar*.

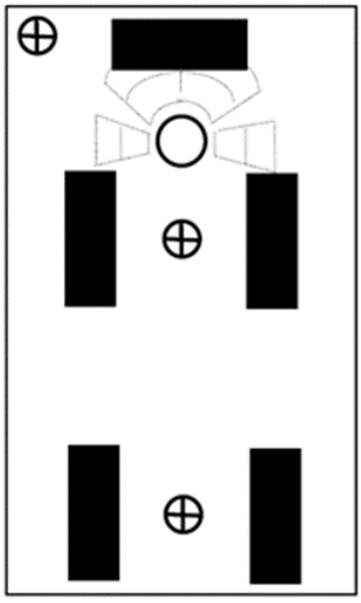

Figura 4.13 – Robot en Escenario (adaptado de [García-Martinez, 1997])

La nueva situación percibida (figura 4.13) es: 00000110
No existe una teoría registrada que sea igual, pero sí existen teorías similares: teoría 4, 5 y 6. Por lo tanto:

1) Se registra la nueva teoría local:
   (00001111, AVANZAR, 00000110, ?, ?, ?)

2) Se registra la nueva teoría mutante:
   (00001111, AVANZAR, 0000-11-, ?, ?, ?)

3) Se incrementa el P de la nueva teoría local y de la mutante:
   (00001111, AVANZAR, 00000110, 1, ?, ?)
   (00001111, AVANZAR, 0000-11-, 1, ?, ?)

4) Se estandariza el *K* de la teoría nueva y las similares y se asignan los niveles de utilidad. Por lo tanto, el conjunto total de teorías del sistema queda expresado de la siguiente forma:

Teoría 1		(00000000, AVANZAR, 00000000, 1, 4, 1/3)
Teoría 2		(00000000, AVANZAR, 00001111, 2, 4, 1/2)
Teoría 3	M	(00000000, AVANZAR, 0000 - - - -, 1, 4, 1/2)

Teoría 4		(00001111, AVANZAR, 00001111, 2, 6, 1)
Teoría 5		(00001111, AVANZAR, 00000000, 1, 6, 1/3)
Teoría 6	M	(00001111, AVANZAR, 0000 - - - -, 1, 6, 1/3)
Teoría 7		(00001111, AVANZAR, 00000110, 1, 6, 1/3)
Teoría 8	M	(00001111, AVANZAR, 0000 -11- , 1, 6, 1/3)

Como el plan todavía se encuentra en ejecución, el sistema analiza si se ha obtenido la situación esperada. La respuesta es negativa, ya que la situación esperada era la 00001111 y el sistema sensor obtuvo la 00000110. Por tal motivo se aborta el plan en ejecución y se procede a castigar a la teoría 4, que es la teoría que fracasó:

1) Se incrementa en 1 el *P* de la nueva teoría percibida, la teoría 7;
2) Se incrementa en 2 el *P* de la teoría mutante asociada, la teoría 8;
3) Se suma 3 a cada uno de los *K*s de la teoría percibida, de la mutante y del resto de las teorías similares a ellas.

El nuevo conjunto de teorías es el siguiente:

Teoría 1		(00000000, AVANZAR, 00000000, 1, 4, 1/3)
Teoría 2		(00000000, AVANZAR, 00001111, 2, 4, 1/2)
Teoría 3	M	(00000000, AVANZAR, 0000 - - - -, 1, 4, 1/2)
Teoría 4		(00001111, AVANZAR, 00001111, 2, 9, 1)
Teoría 5		(00001111, AVANZAR, 00000000, 1, 9, 1/3)
Teoría 6	M	(00001111, AVANZAR, 0000 - - - -, 1, 9, 1/3)
Teoría 7		(00001111, AVANZAR, 00000110, 2, 9, 1/3)
Teoría 8	M	(00001111, AVANZAR, 0000 -11- , 3, 9, 1/3)

4.6 ÍNDICE DE CONFIABILIDAD DINÁMICO

En el modelo original, el Índice de Confiabilidad (IC) constituye el umbral a partir del cual se determina la calidad del plan antes de que sea ejecutado. La calidad del mismo equivale a la probabilidad de éxito que resulta de la matriz de valoración del plan. Si la probabilidad que arroja la matriz es mayor o igual al IC, el plan se ejecuta normalmente; en caso contrario, se aborta la ejecución del plan para así lograr evitar que se lleven adelante planes con una alta probabilidad de fracaso.

La propuesta de mejora que se presenta en esta sección mantiene lo descrito en el párrafo precedente pero incorpora al proceso de evaluación un índice de confiabilidad dinámico. Al comienzo, el IC se inicializa con un valor determinado a priori por el diseñador, que para los fines de esta tesis toma el valor de 0,60. A medida que se van sucediendo los ciclos de observación-aprendizaje-planificación-ejecución, el IC se irá ajustando (aumentando o disminuyendo) de acuerdo a los resultados de los

últimos cinco planes ejecutados. Es decir, si de las últimas cinco ejecuciones, un "alto" porcentaje ha fracasado, tiene sentido considerar elevar el IC para que la evaluación de los planes sea más "exigente". En el otro extremo, si de los últimos cinco planes ejecutados ninguno ha fracasado, quizás sea conveniente disminuir el IC ya que pueden haber planes que no están siendo ejecutados y que podrían ser exitosos. A la luz de los cambios introducidos, resulta claro que es necesario definir un límite para lo que el sistema considera "bajo" o "alto". Este parámetro, definido a priori por el sistema, se define como el Límite de Aceptabilidad (LA). Para el propósito del presente trabajo, el valor del LA se fija en 0,20 (o 20%), por lo que se considera aceptable que de cada cinco planes, como máximo uno fracase.

Los resultados de los últimos cinco planes ejecutados se almacenan en el Vector de Planes Ejecutados (VPE), en cuyas posiciones se asigna un 1 en caso de fracaso o un 0 (cero) en caso de éxito. De acuerdo al contenido del vector se calcula el Índice de Fracaso (IF), que luego será comparado con el LA para realizar o no la actualización del IC. A continuación se presenta un ejemplo de la estructura del VPE y su IF correspondiente:

Vector de Planes Ejecutados

i=0	i=1	i=2	i=3	i=4
0	0	1	0	1

El VPE indicado arriba resulta en un IF = 2/5 = 0,40.

El VPE y su respectivo IF se actualizan cada vez que el sistema finaliza la ejecución de un plan, ya sea porque el plan se abortó debido a una diferencia entre la situación esperada y percibida o porque ya han sido ejecutadas todas las acciones del plan. En ambos casos, dicha actualización se produce en el instante inmediatamente anterior al proceso de actualización del IC, que en caso que corresponda, se llevará a cabo de acuerdo al criterio que se indica debajo. La siguiente regla define la magnitud de la variación que será aplicada al IC, de acuerdo al valor del IF, en comparación con el LA.

Índice de Fracaso (IF)	Variación del IC
IF = 0	- 2%
0 < IF <= LA	sin variación
IF > LA	+ 2%

En los próximos apartados se presenta la arquitectura del nuevo proceso (apartado 4.6.1), se detalla el nuevo algoritmo (apartado 4.6.2) y se incorpora un caso integrador (apartado 4.6.3) para ejemplificar la dinámica del IC y sus implicancias en futuras evaluaciones de planes.

4.6.1 Arquitectura propuesta

La nueva arquitectura desarrollada se describe en la figura 4.14. En ella se incorporan las tareas que fueron presentadas en los párrafos previos. Todas ellas se ejecutan luego de que haya finalizado la ejecución del plan, tanto para terminaciones exitosas como interrumpidas. En el apartado 4.5.1 se han descrito con sobrados detalles todos los posibles caminos a recorrer y acciones a ejecutar previas al proceso que atañe aquí, por lo tanto, en este caso la explicación se inicia con la terminación del plan actual.

El procedimiento de modificación del índice de confiabilidad (IC) se dispara desde el módulo de control y ejecución y se inicia en el módulo de planificación con el subproceso *ActualizaVPEyIF* (que incluye dos tareas). Es decir, una vez que el plan finaliza o es abortado, el sistema incorpora al VPE el resultado de la última planificación.

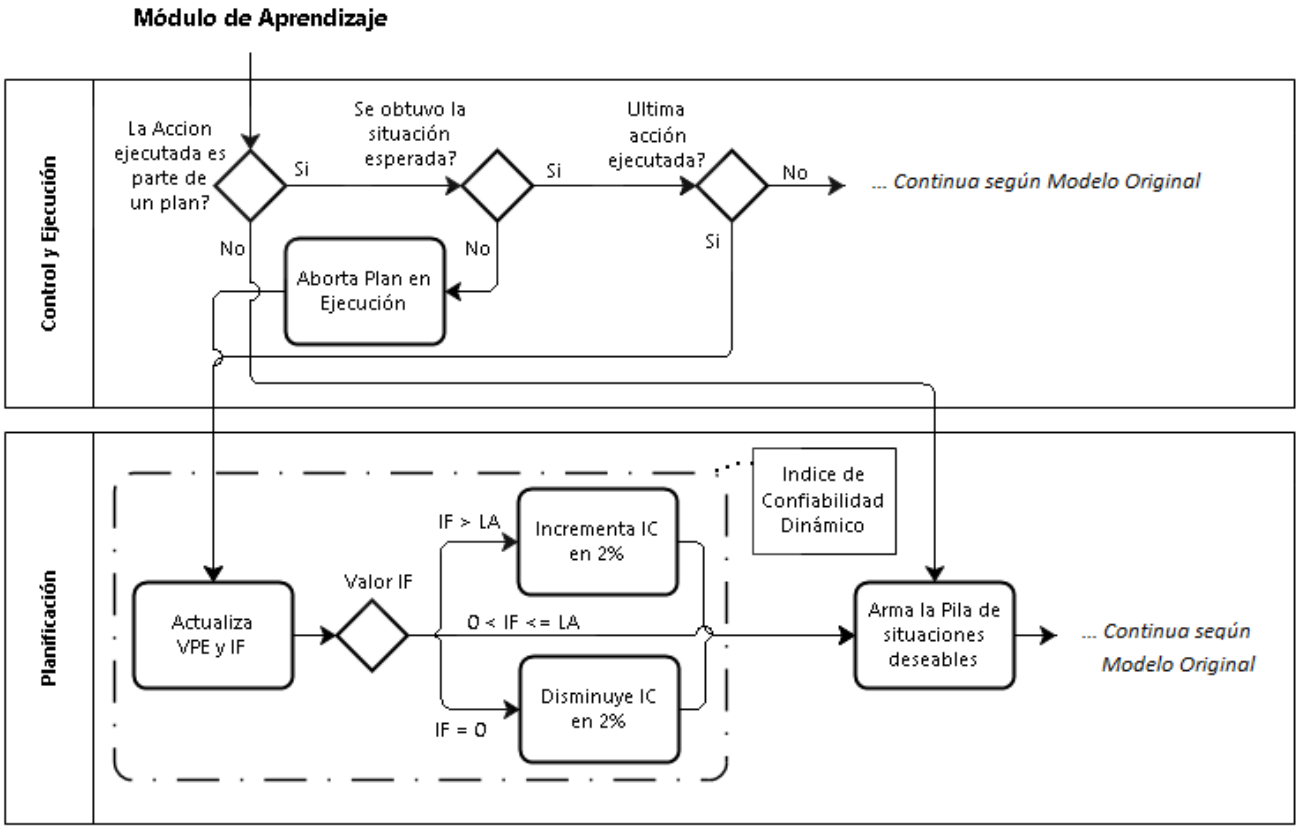

Figura 4.14 – Diagrama de Proceso: Extensión del Índice de Confiabilidad Dinámico

En caso que el plan haya sido exitoso se agrega un 0 (cero) en la primera posición vacía del vector, o de lo contrario, se asigna un 1 (uno). El vector almacena como máximo cinco valores (el último resultado se ingresa en el índice 0 y el primer resultado ingresado, en la posición 4), por lo tanto, cuando ya han sido ejecutados cinco planes y se deba cargar un nuevo resultado, se elimina el

resultado del primer plan ingresado (ubicado en la posición 4), se corren una posición a la derecha los valores de las posiciones 0 a 3 y se ingresa el resultado del nuevo plan ejecutado en la posición 0. Este mecanismo se repite a medida que nuevos planes van finalizando su ejecución. En el instante siguiente a la actualización del VPE se lleva adelante la actualización del IF, índice que mide el porcentaje de planes fracasados a partir de los resultados almacenados en el vector.

Una vez concluidas estas dos primeras tareas, el sistema evalúa el IF resultante. En caso que su valor sea mayor al límite de aceptabilidad (LA), incrementa en un 2% el índice de confiabilidad (IC); en caso que sea igual a 0, lo disminuye en un 2%; y por último, en caso que sea mayor a cero y menor o igual al LA, no actualiza el IC y pasa directamente al siguiente subproceso, el armado de la pila de situaciones deseables. En fin, con la inclusión de este nuevo mecanismo, se logra una actualización constante del IC, el cual trae como consecuencia un sistema "más exigente" a medida que aumentan los planes que fracasan o "más tolerante" en caso contrario.

4.6.2 Algoritmos

A continuación se presentan los algoritmos relacionados con la arquitectura abordada. En la figura 4.15 se describe la función *ActualizaVPE*, la cual se encarga de la primera etapa descrita en el apartado anterior. Esta función toma como argumentos al vector de planes ejecutados (*V*) y al resultado del último plan (*r*). El primer paso lo lleva a cabo ejecutando la función *ObtienePosicionVacia*, a través de la cual comprueba si existe alguna posición vacía dentro del vector. En caso afirmativo, ingresa el resultado del plan en dicha posición. De lo contrario, si ya han sido ejecutados al menos cinco planes y el vector tiene todas las posiciones ocupadas, toma el valor correspondiente a cada posición (haciendo uso de la función *ObtieneElemento*) e inserta dicho valor en su posición siguiente (índice *i+1*) con la función *InsertaElemento*. Posteriormente, asigna el resultado del último plan ejecutado en la posición inicial (índice 0). Al finalizar su ejecución, la función devuelve el vector actualizado para ser utilizado en la actualización del IF.

La función *ActualizaIF* (figura 4.16) define la tarea de actualización del índice de fracaso. En primer lugar, se recorren todas las posiciones del vector, almacenando la suma de sus valores en la variable local *s*. Una vez concluido este procedimiento, se presentan dos posibles caminos. En caso que se hayan ejecutado al menos cinco planes, el valor de la sumatoria se divide por cinco. En caso contrario, el valor total de la suma se divide por la cantidad de planes que fueron ejecutados (esta cantidad es equivalente al valor del índice que corresponde a la posición vacía del VPE –*v*-, ya que el vector comienza en la posición 0, no en la 1). Por lo tanto, dado que los planes fracasados se

identifican con el valor 1, al dividir la suma del contenido del VPE por la cantidad de planes ejecutados (sean 5 o menos), se obtiene automáticamente el porcentaje de planes fracasados.

```
Función ActualizaVPE(V,r): V

V: vector de planes ejecutados
r: resultado del último plan ejecutado (0= Éxito, 1= Fracaso)
i: índice del vector
e: elemento del vector

i = ObtienePosicionVacia(V)
// Si el vector tiene una posición vacía, inserta el nuevo resultado allí
Si i No Es Igual a NULL
    InsertaElemento(V,i,r)
Sino
    // Corre hacia la derecha los elementos de la posición 0 a 3
    Para Todo i tal que 0 =< i <= 3
        e = ObtieneElemento(V,i)
        InsertaElemento(V,i+1,e)
    Fin Para Todo
    // Inserta el resultado del último plan ejecutado en la posición 0
    InsertaElemento(V,0,r)
Fin Si
Devuelve V
```

Figura 4.15 – Algoritmo que actualiza el Vector de Planes Ejecutados

```
Función ActualizaIF(V,v): f

V: vector de planes ejecutados
v: índice de la posición vacía (NULL en caso que no exista)
f: índice de fracaso (asociado al vector de planes ejecutados)
i: índice del vector
s: suma de los elementos del vector

Inicializa(s,0)
Para Todo i tal que 0 =< i <= 4
    s = ObtieneElemento(V,i) + s
Fin Para Todo
Si v Es Igual a NULL
    f = s / 5
Sino
    f = s / v
Fin Si
Devuelve f
```

Figura 4.16 – Algoritmo que actualiza el índice de fracaso

El paso que resta es actualizar el índice de confiabilidad. Una vez que ya han sido ejecutadas las funciones *ActualizaVPE* y *ActualizaIF*, el próximo y último paso es ejecutar la función *ActualizaIC* (figura 4.17). Dicha función recibe como argumentos al índice de fracaso actualizado a partir de la ejecución del último plan, al límite de aceptabilidad del sistema y al valor del índice de confiabilidad previo a su actualización. La modificación se lleva adelante a partir de la siguiente pauta: si el índice de fracaso es igual a cero, entonces se aplica una disminución de un 2% al índice de confiabilidad; si el índice de fracaso es mayor al límite de aceptabilidad, entonces se incrementa el IC en un 2%.

Función ActualizaIC*(f,l,c)*: *c*

f: índice de fracaso actualizado con el resultado del último plan
l: límite de aceptabilidad
c: índice de confiabilidad

```
// Si el IF es igual a 0, disminuye 2% el IC
Si f Es Igual a 0
    ModificaIC(c,-0.02)
Sino
    // Si el IF es mayor al límite de aceptabilidad, incrementa 2% el IC
    Si f Es Mayor que l
        ModificaIC(c,0.02)
    Fin Si
Fin Si
Devuelve c
```

Figura 4.17 – Algoritmo que actualiza el Índice de Confiabilidad

4.6.3 Ejemplo Integrador

El ejemplo presentado a continuación comienza a partir de lo dejado por el ejemplo integrador de [García-Martínez; 1997]. A diferencia del abordaje del mismo en el apartado 4.5.3, aquí se omiten los detalles relacionados con el proceso de construcción de nuevas teorías y de elaboración del plan (armado de la pila de situaciones deseables y del árbol de situaciones asociado) para focalizar en el proceso de evaluación de calidad del plan y la posterior actualización del índice de confiabilidad (IC). Al comienzo del ejemplo (t=0) se supone que: (a) ya se ha armado el plan, cuya acción es *Avanzar* y su SI (00001111) es igual a su SF; y (b) ya se ha construido su matriz de valoración.

Plan = AVANZAR

$$M_{AVANZAR} = \begin{vmatrix} 1/4 & 1/2 & 1/4 \\ 1/4 & \textcircled{1/2} & 1/4 \\ 0 & 0 & 0 \end{vmatrix}$$

Según el conocimiento del que dispone el sistema, la probabilidad estimada de que aplicando el plan *Avanzar* a la situación 00001111 se obtenga la situación 00001111, es 1/2. Es decir, la probabilidad de éxito del plan armado es 0,50. En esta instancia el sistema debe determinar la confiabilidad del plan comparando la probabilidad de éxito del plan con el valor actual del IC, que para los fines del presente ejemplo se supone en 0,40. En definitiva, como la probabilidad de éxito es superior al índice de confiabilidad, se ejecuta la única acción del plan, que es *Avanzar*.

En t=1 el agente percibe una nueva situación. Suponiendo que no existe una teoría registrada que sea igual pero sí teorías similares se procede a: (i) registrar la nueva teoría local; (ii) registrar la nueva teoría mutante; (iii) incrementar el *P* de la nueva teoría local y de la mutante; (iv) estandarizar el *K* de la nueva teoría y de sus similares; y por último, (v) asignar los niveles de utilidad correspondientes. Como el plan todavía se encuentra en ejecución, el sistema analiza si se ha obtenido la situación esperada. En este punto se ha decidido evaluar diferentes situaciones para así poder estudiar los tres posibles caminos que puede tomar el sistema con la mejora propuesta. En línea con esto se presentan tres casos, de los cuales en los dos primeros se supone que la nueva situación percibida es distinta a la esperada, mientras que en el último se supone lo contrario. En cada uno de ellos se describe en detalle las nuevas tareas incluidas en la mejora, que son:

1) la actualización del VPE y su IF asociado;
2) la actualización del IC, en caso que correspondiese.

Además, con el fin de enriquecer los ejemplos, cada caso presenta distintas estructuras para sus Vectores de Planes Ejecutados (VPE) y, en consecuencia, distintos IF. Respecto al Límite de Aceptabilidad (LA), se supone un valor de 0,20 para todos, es decir, se acepta como máximo que el 20% de los planes fracasen. Por último, cómo ya se acordó previamente, el Índice de Confiabilidad inicial se asume en 0,40.

Caso 1

Se considera que antes del comienzo de la ejecución del plan descrito recién, el sistema presenta las siguientes características:

VPE = [0,0,1,0,1]
IF = 0,40

Es decir, de los últimos 5 planes ejecutados 2 han fracasado. Ahora bien, una vez finalizado el plan en ejecución el primer paso corresponde a la actualización del Vector de Planes Ejecutados y su IF asociado. Como en este primer caso se supone que el último plan fracasó, la nueva estructura del VPE queda de la siguiente forma:

VPE = [1,0,0,1,0]

IF = 0,40

Se observa que a pesar de que la estructura del vector ha variado, su IF se mantiene constante. Esto se debe a que por un lado se ha eliminado del histórico un plan que ha fracasado, pero por otro lado, ha ingresado al vector el resultado del último plan, cuyo resultado ha sido negativo.

El segundo paso es la actualización del IC. Como el nuevo IF (0,40) es mayor que el LA (0,20) se incrementa el IC en un 2%, resultando en el siguiente valor:

$IC_{resultante} = 0,408$

Por lo tanto, para las próximas evaluaciones de planes, el umbral de calidad será leventemente mayor, o en otros términos, el sistema será un poco más exigente a la hora de decidir si un plan debe ejecutarse o no.

Caso 2

En este caso se considera que previo al comienzo del último plan, de los 5 planes ejecutados ninguno ha fracasado. El sistema presenta los siguientes valores:

VPE = [0,0,0,0,0]

IF = 0

A continuación se procede a la actualización del VPE y del IF. Como en este segundo caso también se supone que el último plan ha fracasado, la nueva estructura del VPE es la siguiente:

VPE = [1,0,0,0,0]

IF = 0,20

A diferencia del caso anterior, aquí se ha modificado tanto el VPE como el IF. Este último ha aumentado dado que ahora existe un plan fracasado que es tenido en cuenta para el cálculo.

El segundo paso es la actualización del IC. En este caso, el nuevo IF (0,20) es igual al LA. Por tal motivo no se lleva adelante la actualización del IC y su valor permanece invariable:

$IC_{resultante} = 0,40$

Aquí se observa que la "exigencia" del modelo a la hora de evaluar los planes no ha sido alterada.

Caso 3

En éste último caso se supone la siguiente estructura para el VPE y su correspondiente IF:

VPE = [0,0,0,0,1]

IF = 0,20

Es decir, antes de que el último plan se haya iniciado, de los últimos 5 planes ejecutados sólo uno ha fracasado. A diferencia de los dos casos anteriores, aquí se considera que el último plan ejecutado resultó exitoso, por lo tanto, los valores actualizados son los siguientes:

VPE = [0,0,0,0,0]

IF = 0

Luego de la actualización se observa que la nueva estructura del VPE resulta en un IF igual a 0. A raíz de esto, el siguiente paso que ejecutará el sistema será reducir en un 2% el IC, alcanzando el siguiente valor:

$IC_{resultante} = 0,392$

Por consiguiente, la disminución del IC se verá reflejado en que en las futuras evaluaciones de los planes el sistema actuará de un modo más "tolerante".

4.7 ÍNDICE DE MEJORA

Hasta aquí se han descrito las cuatro mejoras propuestas para mejorar el rendimiento de la arquitectura LOPE original. Lo que resta, y eso es el propósito de esta sección, es definir un criterio que permita realizar una evaluación integral de los resultados que se obtengan. Con este objetivo a la vista se construye un indicador, el *Índice de Mejora*, con el cual se podrá llevar a cabo una comparación rigurosa entre los resultados que se alcancen con el modelo original y aquellos que se logren con cada una de las extensiones implementadas.

Se considera que no sólo es importante la medición del porcentaje de planes exitosos y fracasados, sino que además, dado que el agente es un robot explorador, resulta vital, por ejemplo, conocer el porcentaje del escenario que logra descubrir. Debido a esto, el índice que se ha construido está compuesto por cinco variables. Pero antes de proceder a identificarlas, es conveniente explayarse acerca de las "motivaciones" que posee el robot cuando se encuentra en funcionamiento, argumentación que se fundamenta en el concepto de *necesidad*, utilizado dentro del contexto de la teoría sobre la motivación humana de Maslow. En línea con esto, se puede afirmar que el robot a lo largo de su actuación, tiene una serie de necesidades que busca cubrir y que no todas ellas tienen la

misma importancia, sino que están ordenadas según una jerarquía predefinida. Haciendo un paralelismo entre la pirámide de Maslow, que analiza la jerarquía de necesidades humanas, han sido identificadas cinco necesidades para el caso del robot explorador que está siendo estudiado, ellas se presentan en la figura 4.18:

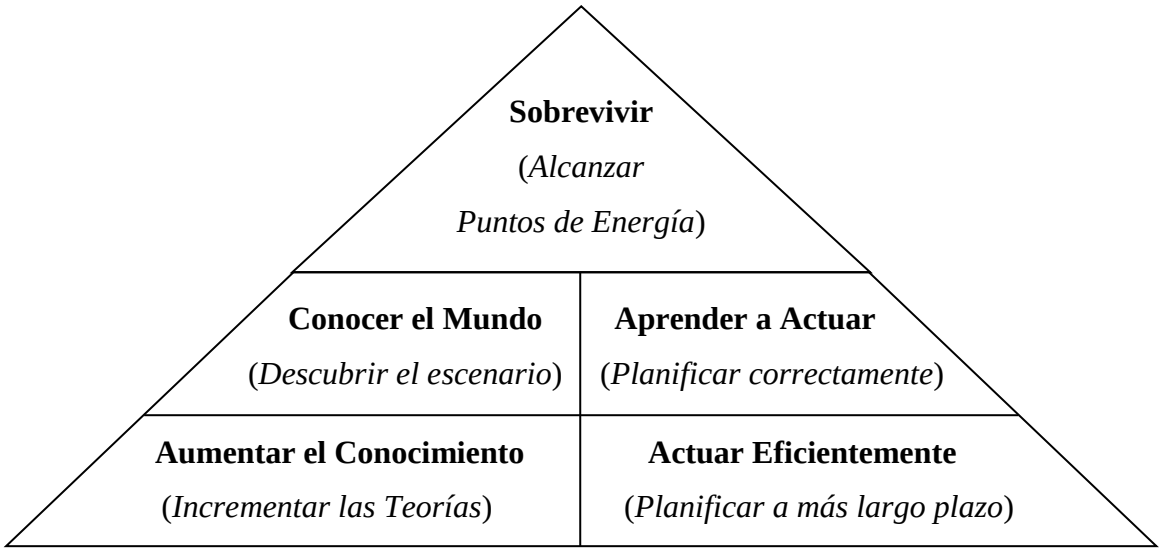

Figura 4.18 – Jerarquía de Necesidades propuesta para el Robot Explorador

La primera necesidad identificada es la de *Sobrevivir*. Esto, traducido al lenguaje del sistema, significa que el robot necesita encontrar los puntos de energía y situarse sobre ellos periódicamente para poder recargar su batería, que es la que le permite continuar funcionando. La segunda y tercera necesidad, *Conocer el Mundo* y *Aprender a Actuar*, se consideran que están al mismo nivel. Esto se sustenta en que por una parte el robot necesita descubrir el escenario para cumplir con la exploración para la cual ha sido diseñado, pero al mismo tiempo es imprescindible que sepa cómo actuar sobre el mismo, es decir, debe ser capaz de poder armar un plan. O sea, así como es vital que el robot conozca el medio para luego poder elegir la situación que desea alcanzar, es importantísimo también que el robot demuestre pericia en el armado de los planes, ya que dicha funcionalidad le permitirá alcanzar el objetivo que se ha trazado. En fin, una y otra necesidad, son igualmente importantes.

De forma similar, la tercera y cuarta necesidad, *Aumentar el Conocimiento* y *Actuar Eficientemente*, se suponen al mismo nivel. Ambas pueden verse como un refinamiento de las dos necesidades anteriores. Por un lado, el hecho de que el robot incremente la cantidad de teorías registradas, mejora sin duda el conocimiento acerca del escenario o de su mundo. Por otro lado, si el robot no sólo aprende a actuar sobre el medio sino que además, logra hacerlo de una forma más eficiente (arma planes a mayor plazo) estaría indudablemente mejorando su proceso de planificación.

Ahora bien, habiendo presentado la escala de necesidades vinculadas al modelo LOPE, es necesario identificar las cinco variables a las que se ha hecho referencia al comienzo de la sección. Estas

dimensiones permitirán medir el grado en que cada una de esas necesidades es satisfecha. Cada una de las variables, a su vez, tiene asignada una determinada ponderación, cuyo valor está relacionado con la posición que ocupa su necesidad asociada dentro de la pirámide presentada previamente. En la tabla 4.1 se detallan las necesidades del robot, sus dimensiones asociadas y la ponderación que le corresponde a cada una de ellas dentro del índice de mejora (IM).

Prio-ridad	**Necesidad del Robot**	**Dimensión a medir**	**Pond. IM**
1	Alcanzar Puntos de Energía	% de Recorridos por Puntos de Energía (RPE)	0,30
2	Descubrir el Escenario	% de Descubrimiento del Escenario (DES)	0,25
2	Planificar correctamente	Resultado Neto de Planes (RNP)	0,25
3	Incrementar las Teorías	Cantidad de Teorías Creadas (TEO)	0,10
3	Planificar a más largo plazo	Longitud Promedio de Planes (LGP)	0,10

Tabla 4.1 – Las Necesidades del robot y su relación con las Dimensiones y Ponderaciones del Índice

En la tabla se observa que la mayor ponderación (0,30) se le asigna a la dimensión *Porcentaje de Recorridos por Puntos de Energía*, cuya necesidad asociada (*Alcanzar Puntos de Energía*) se sitúa en el pináculo de la pirámide mencionada. Le siguen las dimensiones *Porcentaje de Descubrimiento del Escenario* y *Resultado Neto de Planes*, ambas con una ponderación levemente inferior de 0,25. Por último, a las dimensiones *Cantidad de Teorías Creadas* y *Longitud Promedio de Planes* se les ha asignado una ponderación de 0,10. Los valores asignados a estas cinco magnitudes están de acuerdo con el ordenamiento que se le ha dado a cada una de sus respectivas necesidades en la figura 4.18.
Por último, cabe mencionar que el índice pentadimensional que se ha propuesto será calculado a partir de la variación de cada una las magnitudes indicadas, respecto al modelo base (modelo LOPE sin ninguna extensión implementada). Es decir, el IM que se obtendrá para cada uno de los casos a evaluar, será computado de acuerdo a la siguiente expresión:

$$IM = a\,\Delta RPE + b\,\Delta DES + c\,\Delta RNP + d\,\Delta TEO + e\,\Delta LGP$$

donde *a, b, c, d* y *e* representan las ponderaciones incluidas en la tabla 4.1 y Δ representa el porcentaje de variación de la magnitud en cuestión, respecto del modelo original.

5. EXPERIMENTOS

Se presentan en este capítulo los rasgos generales del método de experimentación adoptado (sección 5.1); una descripción detallada del diseño experimental, que incluye las características de las simulaciones, la configuración de los escenarios elegidos, la definición de las variables independientes y dependientes, y la presentación de los casos a evaluar (sección 5.2); las gráficas resultantes, su interpretación y la matriz de valores finales promedio (sección 5.3); y por último, el índice de mejora obtenido para cada uno de los casos evaluados (sección 5.4).

5.1 INTRODUCCIÓN

Se ejecutan las simulaciones con el objetivo de evaluar la incidencia de la aplicación de las distintas mejoras propuestas al modelo. Los resultados de cada uno de los experimentos se analizan desde una perspectiva integral, en la cual se tienen en cuenta una amplia variedad de factores. Por ejemplo, no sólo se considera importante el porcentaje de planes exitosos sino también el porcentaje de descubrimiento del escenario, el porcentaje de puntos de energía alcanzados, la longitud promedio de los planes, entre otros. Las mejoras son introducidas de modo incremental de forma tal de poder identificar claramente la variación en las variables medidas. La experimentación incluye distintos escenarios, cada uno de los cuales con una configuración distinta respecto de la posición y/o cantidad de obstáculos y puntos de energía.
La programación del SIA se desarrolla en *VBA 7.1 (Microsoft Visual Basic For Applications 7.1)* y se implementa en una planilla *Microsoft Excel 2013*. El software utilizado es propiedad del tesista.

5.2 DISEÑO EXPERIMENTAL

En esta sección se definen las características generales de las simulaciones ejecutadas (apartado 5.2.1); se describe la configuración de los distintos escenarios (apartado 5.2.2); se definen las variables independientes (apartado 5.2.3) y las variables dependientes (apartado 5.2.4); y se detallan las mejoras incluidas en cada uno de los casos a evaluar (apartado 5.2.5).

5.2.1 Simulaciones

A continuación se describe el procedimiento experimental llevado a cabo:

- Se ejecutaron y compararon 5 casos, sumando un total de 540 simulaciones y 108.000 ciclos;

- Cada caso consta de 108 simulaciones. Cada simulación incluye 200 iteraciones. Cada iteración es un ciclo completo de percepción-aprendizaje-planificación-ejecución;
- En cada ciclo el sistema ejecuta alguna de las siguientes acciones: Avanzar (AV), Girar a la Derecha (GD) o Girar a la Izquierda (GI);
- Para cada caso se utilizaron 3 escenarios, ejecutándose 36 simulaciones por cada escenario;
- En cada escenario el robot inicia su operación en 3 posiciones distintas, ejecutándose 12 simulaciones por cada una de ellas;
- En cada posición inicial (Pi) el robot alterna las 4 posibles orientaciones (Or): Norte, Este, Sur y Oeste. Por cada orientación se ejecutaron 3 simulaciones.

En la tabla 5.1 se presenta la configuración de las simulaciones ejecutadas para cada caso.

Escenario 1			**Escenario 2**			**Escenario 3**		
Pi	**Or**	**# Sim**	**Pi**	**Or**	**# Sim**	**Pi**	**Or**	**# Sim**
1	N	3	1	N	3	1	N	3
	E	3		E	3		E	3
	S	3		S	3		S	3
	O	3		O	3		O	3
2	N	3	2	N	3	2	N	3
	E	3		E	3		E	3
	S	3		S	3		S	3
	O	3		O	3		O	3
3	N	3	3	N	3	3	N	3
	E	3		E	3		E	3
	S	3		S	3		S	3
	O	3		O	3		O	3
SubTot E1		**36**	**SubTot E2**		**36**	**SubTot E3**		**36**
						Total		**108**

Tabla 5.1 – Detalle de las Simulaciones Ejecutadas para cada Caso

5.2.2 Escenarios

Cada escenario tiene un tamaño de 7 x 7. Todos ellos están compuestos por 5 puntos de energía y un número definido de obstáculos y posiciones libres. A continuación se presentan los 3 escenarios utilizados en la experimentación.

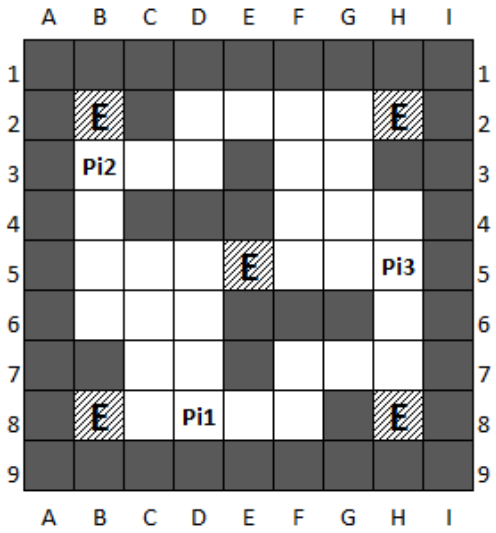

Figura 5.1 – Escenario 1

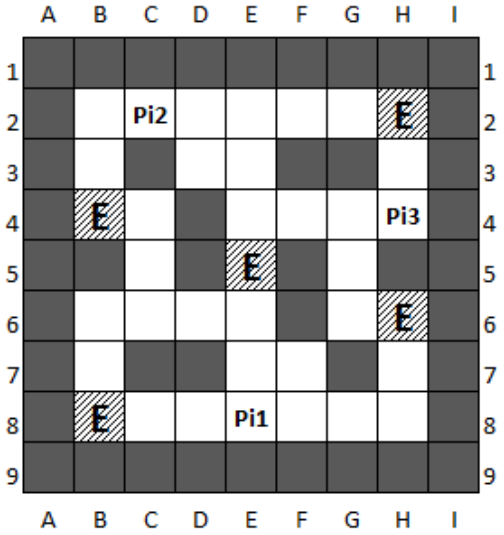

Figura 5.2 – Escenario 2

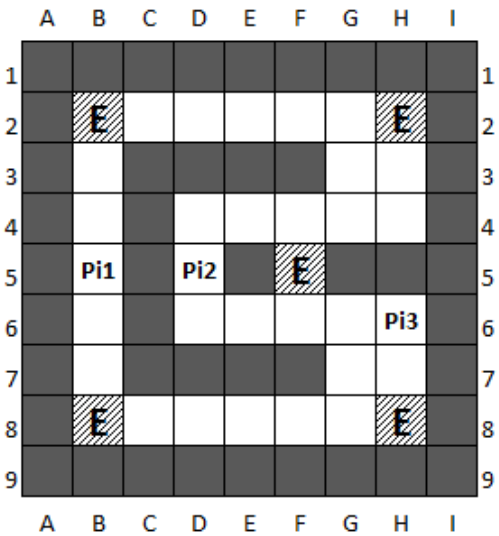

Figura 5.3 – Escenario 3

El escenario 1 (figura 5.1) consta de 37 ubicaciones libres y 12 posiciones con obstáculos. Las tres posiciones iniciales son D8 (Pi1), B3 (Pi2) y H5 (Pi3). Los cinco puntos de energía se ubican en las posiciones B2, B8, E5, H2 y H8.

El escenario 2 (figura 5.2) consta de 37 ubicaciones libres y 12 posiciones con obstáculos. Las tres posiciones iniciales son E8 (Pi1), C2 (Pi2) y H4 (Pi3). Los cinco puntos de energía se ubican en las posiciones B4, B8, E5, H2 y H6.

El escenario 3 (figura 5.3) consta de 35 ubicaciones libres y 14 posiciones con obstáculos. Las tres posiciones iniciales son B5 (Pi1), D5 (Pi2) y H6 (Pi3). Los cinco puntos de energía se ubican en las posiciones B2, B8, F5, H2 y H8.

5.2.3 Variables Independientes

Se consideran las siguientes variables independientes:

- *Tiempo*: se define como unidad de tiempo al lapso transcurrido entre una percepción del entorno y otra. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,200].
- *Posición inicial*: indica la ubicación inicial del robot dentro del escenario. Es una variable cualitativa cuyo valor corresponde a las coordenadas del escenario, del tipo (X,Y). La coordenada X corresponde a las letras incluidas en el intervalo [B,H] y la coordenada Y corresponde a los números pertenecientes al intervalo [2,8].
- *Orientación*: indica la orientación del robot dentro del escenario. Es una variable cualitativa cuyo valor corresponde a los puntos cardinales: Norte (N), Este (E), Sur (S) y Oeste (O).
- *Acción Ejecutada*: indica la última acción ejecutada por el robot. Es una variable cualitativa cuyo valor corresponde a las tres posibles acciones del modelo: Avanzar (AV), Girar a la Derecha (GD) y Girar a la Izquierda (GI).
- *Plan*: define el conjunto de acciones a ejecutar y sus situaciones esperadas asociadas, necesarias para alcanzar una determinada situación deseable. La cantidad máxima de acciones que pueden ser concatenadas en un plan son 6. Es una variable cualitativa representada por un vector cuyo tamaño pertenece al intervalo [1,6].
- *Azar:* indica el valor, obtenido aleatoriamente por el sistema, a partir del cual el robot elige una de las tres posibles acciones cuando se encuentra en el modo de acción por contingencia. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].
- *Índice de Fracaso*: indica el porcentaje de planes fracasados sobre el total de los últimos 5 planes ejecutados (incluidos en el Vector de Planes Ejecutados - VPE), para cada instante dado por la variable *tiempo*. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].
- *Índice de Confiabilidad*: indica la probabilidad mínima que un plan armado tiene que alcanzar para poder ser ejecutado. Funciona como una medida de la probabilidad de éxito del plan. Su valor inicial es de 0,60, el cual permanece constante si la mejora ICD no es implementada o varía en caso

contrario. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].

- *Límite de Aceptabilidad:* indica el porcentaje de planes fracasados que se considera aceptable. A los fines de esta tesis se parametriza en 0,20. Es una variable cuantitativa cuyo valor pertenece el intervalo [0,1].
- *Mejora de Prevención de Giros Ineficientes*: esta variable cualitativa adopta el valor "Activa" si se implementa la mejora que previene la ocurrencia de giros ineficientes o "Inactiva" si no se incorpora.
- *Mejora de Administración de Teorías Mutantes*: esta variable cualitativa adopta el valor "Activa" si se implementa la mejora de la administración completa de las teorías mutantes o "Inactiva" si no se incorpora.
- *Mejora de Castigo de Teorías en Planes Abortados*: esta variable cualitativa adopta el valor "Activa" si se implementa la mejora del castigo a la teoría que generó el fracaso del Plan o "Inactiva" si no se incorpora.
- *Mejora de Índice de Confiabilidad Dinámico*: esta variable cualitativa adopta el valor "Activa" si se implementa la mejora del índice de confiabilidad dinámico o "Inactiva" si no se incorpora.

5.2.4 Variables Dependientes

Se consideran las siguientes variables dependientes:

- *Porcentaje de Recorridos sobre Puntos de Energía*: indica el porcentaje de recorridos sobre posiciones con puntos de energía sobre el total de iteraciones de una simulación. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].
- *Porcentaje de Puntos de Energía alcanzados*: indica el porcentaje de puntos de energía alcanzados al finalizar la simulación. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].
- *Utilidad Promedio*: indica el promedio de la utilidad percibida por el sistema para un determinado instante dado por la variable *tiempo*. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].
- *Porcentaje de Descubrimiento del Escenario*: indica el porcentaje de posiciones recorridas sobre el total de las posiciones libres del escenario, para un

determinado instante dado por la variable *tiempo*. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].

- *Porcentaje de Planes Exitosos*: indica el porcentaje de planes que han sido ejecutados exitosamente sobre el total de planes ejecutados. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].
- *Porcentaje de Planes Fracasados*: indica el porcentaje de planes que han fracasado en su ejecución sobre el total de planes ejecutados. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].
- *Porcentaje de Planes de Contingencia*: indica el porcentaje de planes de contingencia disparados sobre el total de planes ejecutados. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].
- *Cantidad de Situaciones*: indica la cantidad de situaciones que han sido percibidas y reconocidas por el sistema en un determinado instante dado por la variable *tiempo*. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,100].
- *Cantidad de Teorías*: indica la cantidad de teorías que han sido creadas por el sistema en un determinado instante dado por la variable *tiempo*. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,400].
- *Longitud Promedio de Planes Exitosos*: indica la cantidad promedio de pasos de los planes ejecutados exitosamente. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,6].
- *Longitud Promedio de Planes Fracasados*: indica la cantidad promedio de pasos de los planes que han fracasado en su ejecución. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,6].
- *Porcentaje Promedio de Pasos Exitosos en los Planes Fracasados*: indica la cantidad promedio de pasos exitosos de los planes que han fracasado en su ejecución. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,5].
- *Resultado Neto Promedio de Planes:* indica el resultado global del proceso de planificación. Su valor resulta de restar el porcentaje promedio de planes fracasados al porcentaje promedio de planes exitosos. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].
- *Longitud Promedio de Planes:* indica la longitud promedio global de los planes ejecutados. Su valor resulta de promediar la longitud promedio de los planes

fracasados y de los planes exitosos. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,6].

- *Índice de Mejora:* indica el rendimiento integral del sistema. Es un indicador pentadimensional que permite comparar los resultados de los distintos casos evaluados entre sí. Es una variable cuantitativa cuyo valor pertenece al intervalo [0,1].

5.2.5 Casos a Evaluar

Se evaluaron 5 casos en total. Para cada uno de ellos se ejecutaron 108 simulaciones. Los resultados de las 540 simulaciones se reunieron en una matriz en donde se detallan los valores promedio de las variables dependientes mencionadas previamente. La configuración de cada uno de los casos ejecutados se detalla en la tabla 5.2.

# Caso	Mejoras Implementadas
0	Modelo Original. Sin Mejoras implementadas.
1	i) Prevención de Giros Ineficientes (PGI) ii) Administración Completa de Teorías Mutantes (ATM)
2	i) Prevención de Giros Ineficientes (PGI) ii) Administración Completa de Teorías Mutantes (ATM) iii) Castigo a la Teoría que provocó el Fracaso del Plan (CST)
3	i) Prevención de Giros Ineficientes (PGI) ii) Administración Completa de Teorías Mutantes (ATM) iii) Índice de Confiabilidad Dinámico (ICD)
4	i) Prevención de Giros Ineficientes (PGI) ii) Administración Completa de Teorías Mutantes (ATM) iii) Castigo a la Teoría que provocó el Fracaso del Plan (CST) iv) Índice de Confiabilidad Dinámico (ICD)

Tabla 5.2 – Mejoras Implementadas en cada uno de los Casos

5.3 GRÁFICAS Y SU INTERPRETACIÓN

A continuación se presentan las gráficas correspondientes a las variables que integran la matriz general, acompañadas de la interpretación de los resultados obtenidos. Se analizan las variables

dependientes relacionadas con los Puntos de Energía, la Utilidad Promedio y el Porcentaje de Descubrimiento del Escenario (apartado 5.3.1); las Situaciones Percibidas y las Teorías Creadas (apartado 5.3.2); y las variables vinculadas a la Planificación (apartado 5.3.3). En cada uno de los gráficos se comparan los 5 casos evaluados. Por último, se presenta la matriz (apartado 5.3.4) que incluye los valores finales promedio de las variables estudiadas para cada uno de los casos, y en la cual se resaltan los mejores valores alcanzados en cada una de ellas.

5.3.1 Puntos de Energía, Utilidad y Descubrimiento del Escenario

En este apartado se presentan 4 gráficos: (i) el porcentaje de recorrido sobre puntos de energía para el total de ciclos ejecutados (figura 5.4); (ii) el porcentaje de puntos de energía alcanzados (figura 5.5); (iii) la utilidad promedio (figura 5.6); y (iv) el porcentaje de descubrimiento del escenario (figura 5.7). En el Anexo se detalla el recorrido promedio por cada posición de los distintos escenarios, para cada uno de los casos evaluados.

Se observa en el gráfico 5.4 que a medida que se ejecutan los ciclos, la cantidad de veces que el robot recorre puntos de energía aumenta. Este resultado es coherente con el objetivo del proceso de planificación, cuyas acciones se articulan para alcanzar situaciones con la mayor utilidad posible. Teniendo en cuenta que las posiciones que poseen las utilidades más altas (igual a 1) se corresponden con los puntos de energía del escenario, cabe esperar que a medida que transcurra el tiempo el robot continúe recorriendo puntos de energía (no necesariamente distintos) para lograr "sobrevivir".

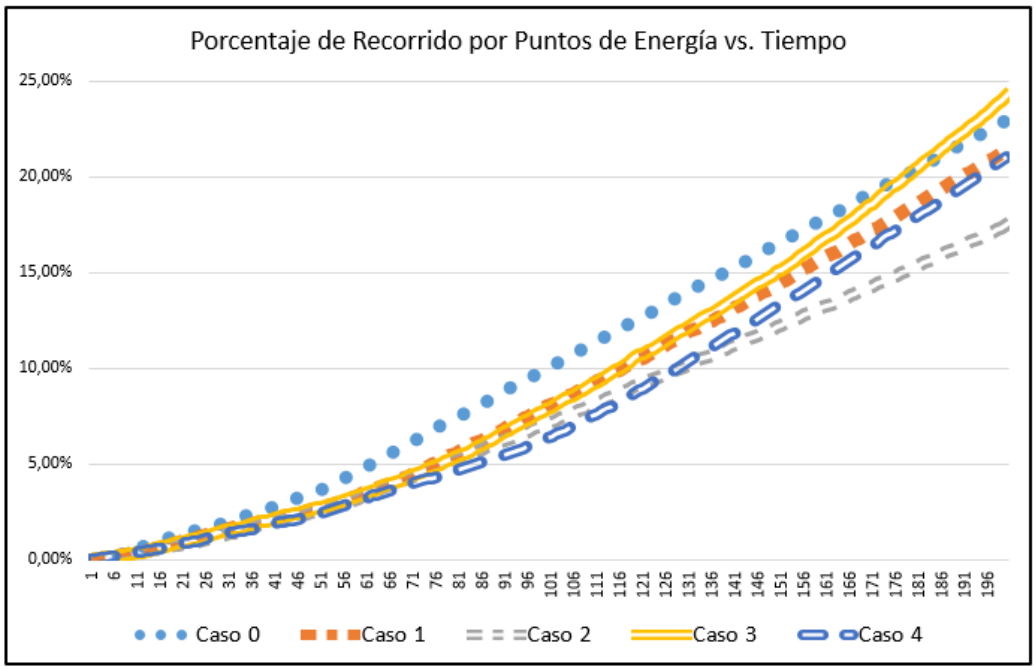

Figura 5.4 – Porcentaje de Recorrido por Puntos de Energía vs. Tiempo

Al final del ciclo, el caso 3 fue el que alcanzó el mayor porcentaje de recorridos sobre puntos de energía (24,18%), y en segundo lugar se posicionó el caso 0, con 22,96%. Los casos 1 (21,38%) y 4 (21,06%) terminaron con valores similares y en último lugar se posicionó el caso 2, con 17,46%.

En el gráfico 5.5 se observa que a medida que transcurre el tiempo se van descubriendo nuevos puntos de energía hasta converger asintóticamente a un determinado valor. Esta convergencia se da muy tempranamente al implementarse el modelo original (caso 0), alcanzando un valor de 14,81%. El resto de los casos oscilan entre 26,48% y 33,70%, siendo el caso 3 el del máximo valor obtenido. La significativa diferencia entre el caso 0 y el resto, puede explicarse a partir de la naturaleza de la mejora PGI, cuyo objetivo es lograr una eficiencia en la ejecución y en la planificación, y de esta forma poder alcanzar un mayor porcentaje de descubrimiento del escenario en cuestión.

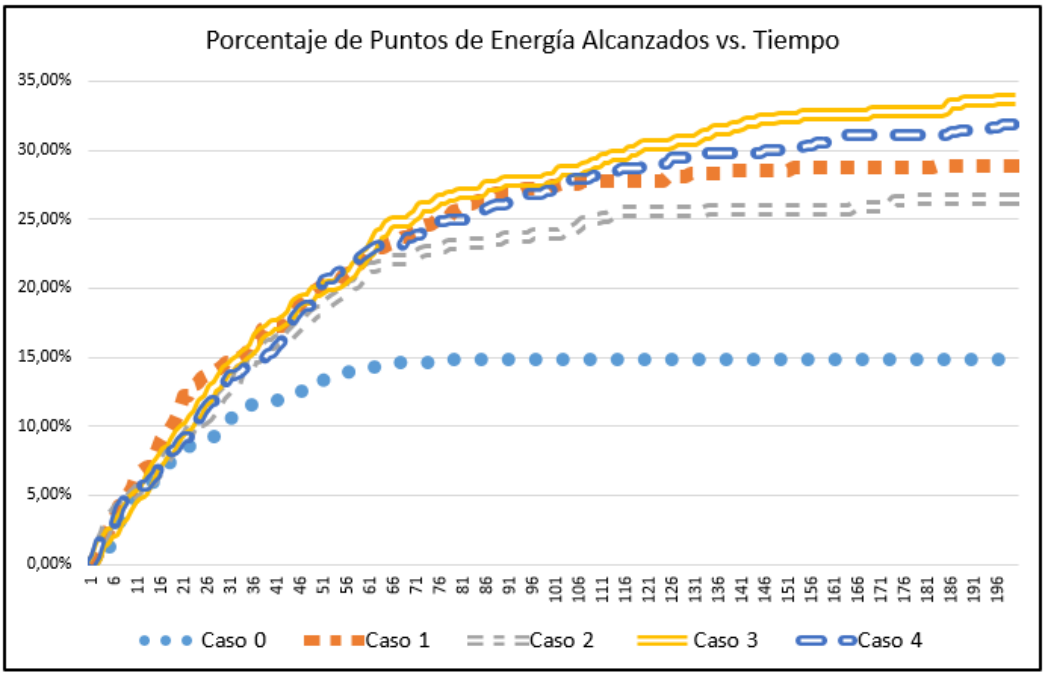

Figura 5.5 – Porcentaje de Puntos de Energía Alcanzados vs. Tiempo

Se observa en el gráfico 5.6 que la tendencia es creciente respecto a la utilidad promedio alcanzada, dinámica que es consecuente con el comportamiento observado en el gráfico 5.4. Efectuando una comparación entre ambas figuras, se observa que el orden de los valores finales obtenidos son todos correspondidos salvo el caso 0. Este último caso alcanza la segunda posición al analizarse el porcentaje de recorridos sobre puntos de energía (figura 5.4) pero desciende a la cuarta cuando se evalúa la utilidad promedio (figura 5.6). También se aprecia que durante una gran parte del tiempo, el caso del modelo original es el que obtiene la más alta utilidad promedio y luego hacia el final se ve superado por los casos 1, 3 y 4.

Estas dos observaciones se sintetizan en la siguiente interpretación. En primer lugar, es esperable que los gráficos 5.4 y 5.6 estén estrechamente correlacionados, ya que aquel mide el porcentaje de recorridos sobre puntos energía y este la utilidad. En consecuencia, se espera que a medida que se recorran más ubicaciones con centros de energía (posiciones con utilidad 1), la utilidad promedio aumente. Ahora bien, ¿a qué se debe, entonces, la variación observada en el caso 0? Para responder a esta pregunta es necesario recurrir al resultado acerca del porcentaje de puntos de energía alcanzados (figura 5.5), en el cual se observa que el modelo original es aquel que menos puntos de energía alcanza. Por lo tanto, si durante el caso 0 se recorre una gran cantidad de puntos de energía pero el porcentaje de puntos de energía descubiertos es bajo, se supone que una vez que el robot se encuentra posicionado sobre un punto de energía, procede a alejarse del mismo para alcanzar otros puntos pero, en reiteradas ocasiones, vuelve sobre sus pasos para posicionarse nuevamente en el punto de energía anterior. Este comportamiento (alejarse del punto de energía, girar y regresar al mismo) genera una utilidad promedio menor que los casos en que el robot se traslada directamente de un punto de energía a otro distinto, ya que el recorrido total es menor.

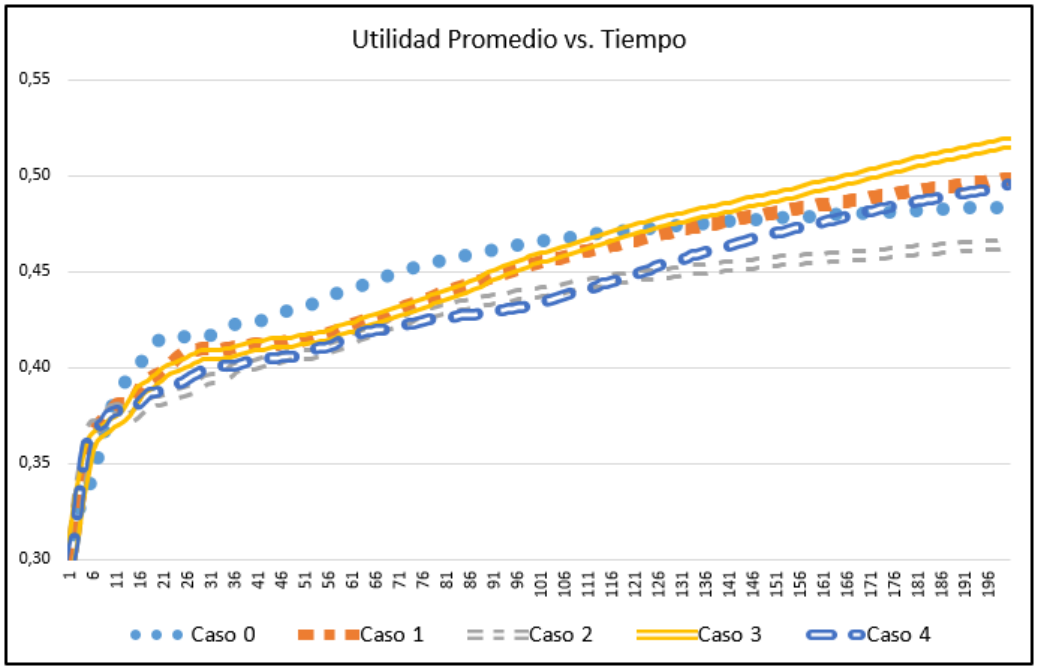

Figura 5.6 – Utilidad Promedio vs. Tiempo

En segundo lugar, el motivo por el cual al comienzo la utilidad promedio del caso 0 es mayor y luego decae, puede explicarse teniéndose en cuenta que las mejoras introducidas al sistema motivan un mayor porcentaje de exploración vs. explotación en los primeros ciclos. Esto trae como consecuencia que durante una gran parte del tiempo se priorice el descubrimiento de nuevas posiciones antes que

el acercamiento a los puntos de energía ya conocidos. Como contrapartida, en el caso del modelo original, el sistema es más propenso desde el comienzo a tratar de alcanzar los puntos de energía ya conocidos si los resultados de la planificación resultaron exitosos, ya que ello le garantiza la "supervivencia".

Por último, en relación a los valores obtenidos, se observa nuevamente que el caso 3 es el que alcanza el mejor resultado (utilidad promedio de 0,52), en segundo lugar se ubica el caso 1 y 4 con un valor de 0,50, el cuarto lugar lo ocupa el caso 0 con una valor de 0,48 y por último, el caso 2 con una utilidad promedio de 0,46.

En el gráfico 5.7 se observa un comportamiento bastante similar al de la figura que describe el porcentaje de puntos de energía alcanzados (figura 5.5). Esto resulta coherente ya que ambos gráficos están vinculados a la acción de descubrimiento; mientras que el primer gráfico se refiere al descubrimiento de los puntos de energía, el segundo se refiere al descubrimiento general del escenario, que incluye a los puntos de energía.

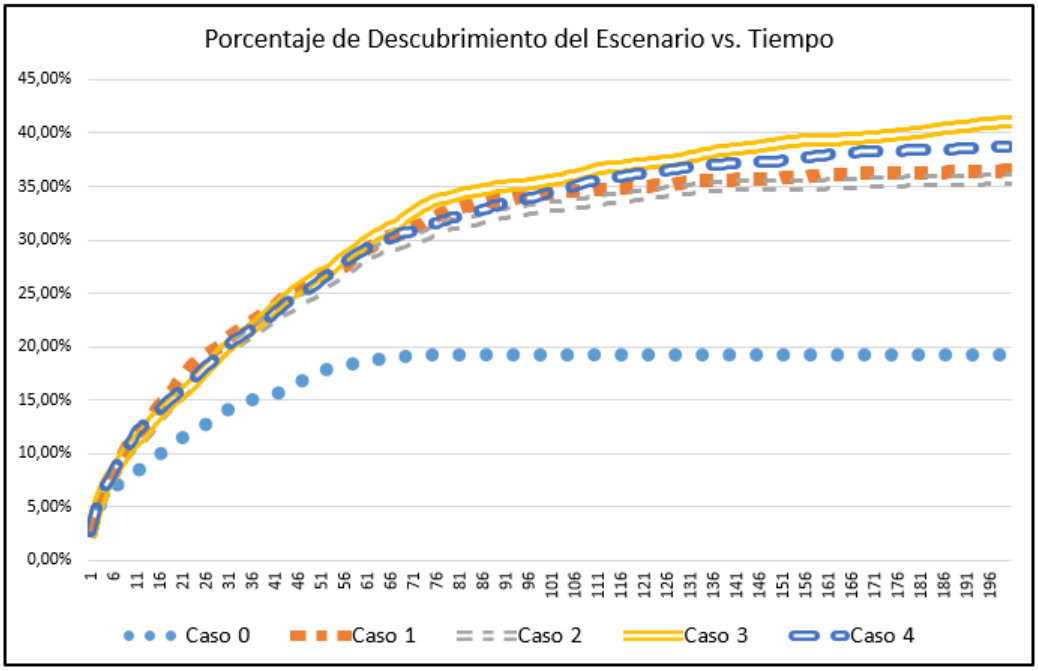

Figura 5.7 – Porcentaje de Descubrimiento del Escenario vs. Tiempo

Se observa que el orden de los valores finales alcanzados en la figura 5.5 se mantienen en la 5.7. También se mantiene la gran diferencia que existe entre los resultados obtenidos en el caso 0 (19,21%) vs. el resto. El caso 3 obtiene el mejor valor de descubrimiento, alcanzando el 41,04% del escenario (duplicando al resultado del modelo original), lo sigue el caso 4 con un valor de 38,78%, luego el caso 1 con 36,49%, le sigue el caso 2 con 35,71% y, por último, el caso 0. En concordancia

con lo anteriormente expuesto, se interpreta que los significativos progresos en los resultados alcanzados se deben a la aplicación de la mejora PGI, que genera una mayor exploración del entorno.

5.3.2 Situaciones y Teorías

Se observa en el gráfico 5.8 que a medida que se ejecutan los ciclos las situaciones se incrementan hasta converger asintóticamente a un determinado valor. El mismo comportamiento resulta en la figura 5.9, en la cual se describen las teorías construidas a medida que avanza el tiempo. El ordenamiento de los valores finales de los 5 casos se corresponde en los dos gráficos mencionados y también lo hace con los valores de la figura 5.7, que describe el porcentaje de descubrimiento del escenario. Esto es esperable ya que en la medida que el sistema va descubriendo posiciones inexploradas aumenta la probabilidad de encontrarse con situaciones nuevas y, en consecuencia, de construir nuevas teorías (cuya construcción se basa en situaciones y acciones). Se entiende que la mayor cantidad de situaciones que presenta el sistema en cualquiera de los casos en comparación con el modelo original, se debe a la implementación de la mejora PGI. Por otro lado, el aumento en la cantidad de teorías construidas se debe, por supuesto, a la mayor cantidad de situaciones percibidas, pero también a la utilización de la mejora ATM, que incrementa la creación de teorías a partir del procesamiento de las situaciones mutantes (iniciales y/o finales) que componen las teorías de los planes en ejecución.

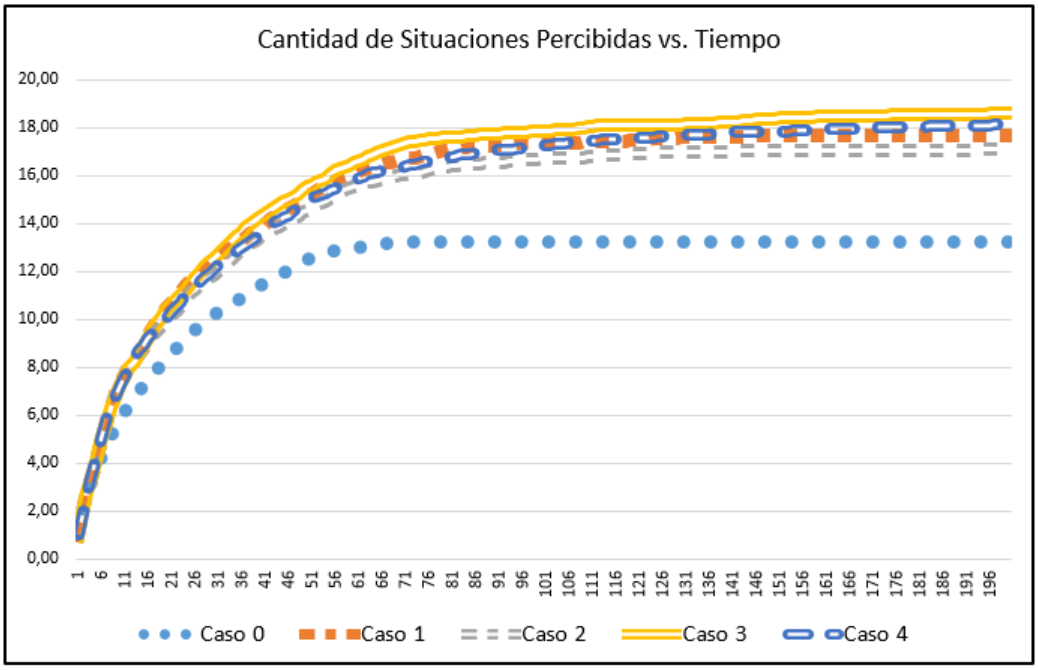

Figura 5.8 – Cantidad de Situaciones Percibidas vs. Tiempo

Los mejores indicadores se obtienen con el caso 3 (18,59 situaciones y 79,77 teorías), le sigue el caso 4 (18,14 situaciones y 77,09 teorías), luego el caso 1 (17,71 situaciones y 72,72 teorías), a continuación el caso 2 (17,10 situaciones y 68,13 teorías) y por último, el caso 0 (13,26 situaciones y 42,69 teorías). Es interesante observar que el caso 3 casi duplica la cantidad de teorías creadas con el modelo original.

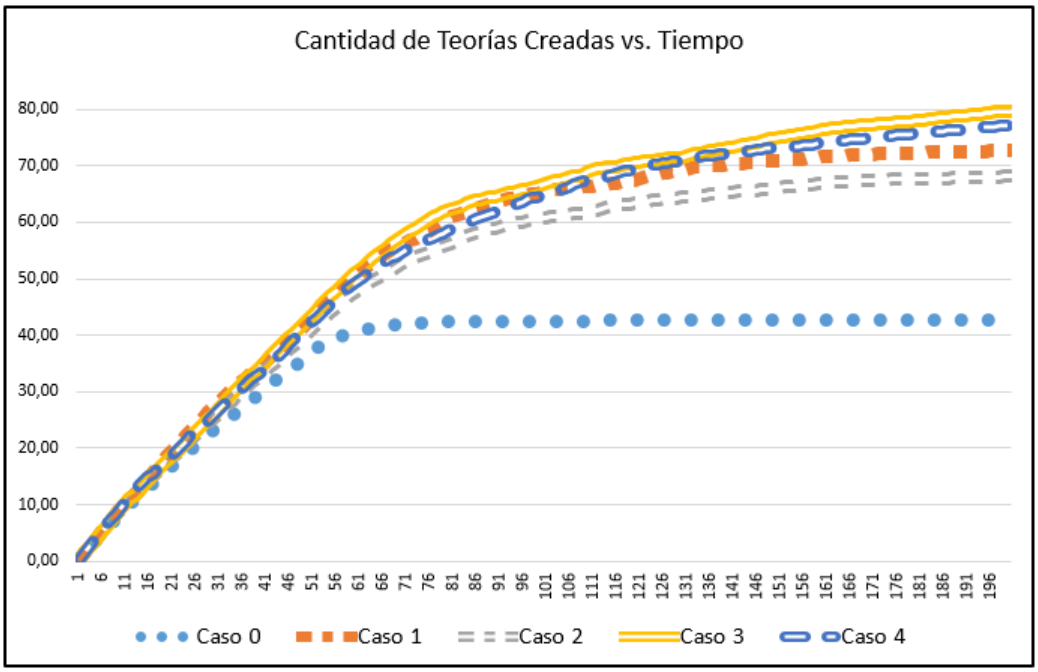

Figura 5.9 – Cantidad de Teorías Creadas vs. Tiempo

5.3.3 Planificación

Las tres posibles opciones en la ejecución de planes son: (a) plan armado que resulta exitoso, (b) plan armado que fracasa y (c) plan por contingencia. Este último no cuenta con un resultado asociado ya que no hay una situación esperada. Las primeras tres gráficas de este apartado se corresponden con la descripción de estas tres variables mencionadas.

En el gráfico 5.10 se describe el porcentaje de planes exitosos que presenta el sistema. Se observa que a medida que transcurre el tiempo, este porcentaje aumenta. Existe una importante diferencia entre el valor final alcanzado por el modelo original y el resto de los casos. En el caso 0 se alcanza el máximo porcentaje de planes exitosos, con un valor de 68,86%. A diferencia de las variables analizadas previamente, aquí el caso 3 presenta el mínimo valor, sólo un 39,67% de los planes resultan exitosos. El resto de los casos logran un porcentaje de éxito de 45,39% (caso 2), 41,55% (caso 1) y 40,83% (caso 4). Si se tienen en cuenta los resultados observados en gráficas previas, se desprende la siguiente interpretación. La diferencia significativa entre los resultados obtenidos en el

caso 0 y en el resto de los casos, se debe justamente a la diferencia que existe, en forma contraria pero con similar magnitud, en la gráfica 5.7 (porcentaje de descubrimiento del escenario) entre los casos con mejoras implementadas y el caso 0. Es decir, en dicha figura se observa que el sistema recorre un porcentaje del escenario bastante menor en el caso 0 en comparación con el resto. Resulta lógico entonces deducir que, si el sistema ha descubierto sólo una pequeña proporción del entorno, le será más fácil aprender a conducirse y a planificar sus acciones en dicha circunstancias, en comparación a ejecutar sus acciones en un escenario de mayor tamaño. Esto es así ya que la cantidad de teorías a partir de las cuales tendrá que elegir en este último caso será mayor, las combinaciones para concatenar las acciones también lo serán, y en consecuencia, la probabilidad de no acertar en la planificación aumentará.

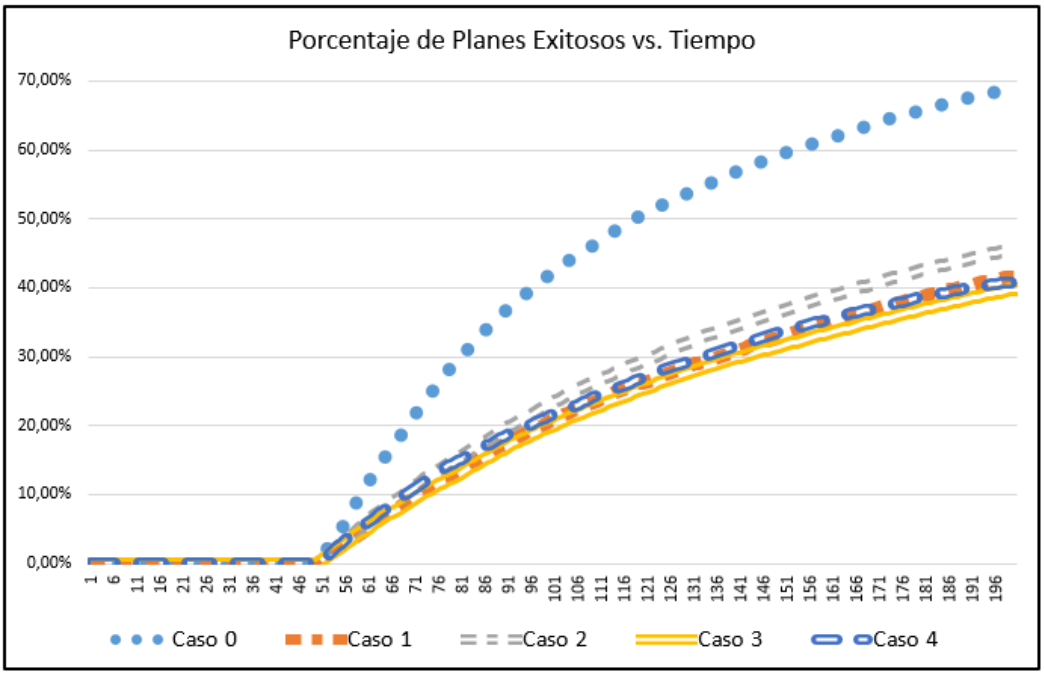

Figura 5.10 – Porcentaje de Planes Exitosos vs. Tiempo

La figura 5.11 describe la evolución del porcentaje de planes que ha fracasado para cada uno de los casos. En términos generales, la interpretación es similar a la gráfica anterior, pero hay un punto interesante que vale la pena mencionar. Mientras que los casos 0, 1 y 2 presentan una tendencia decreciente en la proporción de planes fracasados aproximadamente hacia la mitad del tiempo transcurrido, los otros dos casos mantienen una tendencia creciente hasta el final. Este resultado es coherente considerando que justamente los casos 3 y 4 son aquellos que incluyen la mejora ICD. La característica del ICD es que implementa un índice de confiabilidad dinámico que, por un lado es incrementado cuando una determinada cantidad de planes ha fracasado y, por el otro, y aquí se

encuentra el quid de la cuestión, es disminuido cuando se observa en las últimas ejecuciones un 100% de éxito. Por lo tanto, en aquellas situaciones en que se disminuye el índice de confiabilidad, implícitamente se está reduciendo la exigencia con la cual se evalúan los planes a ser ejecutados. Y esto trae como consecuencia dos posibilidades: (i) que aumente la probabilidad de fracaso del plan a ejecutar, o (ii) que se ejecuten planes que nunca antes habían sido ejecutados y que puedan resultar exitosos. Esto es así ya que, por construcción, el sistema tolera un porcentaje de fracaso mínimo (límite de aceptabilidad) con el objetivo de que el robot nunca deje de alternar la *exploración* (aunque sea mínima habiendo llegado a una determinada instancia) con la *explotación* del conocimiento adquirido.

Respecto a los valores finales alcanzados, el mínimo porcentaje de planes fracasados se logra en el caso 0 (0,92%), el caso 2 le sigue con un 3,26%, luego el caso 1 con un 4,00%, a continuación se posiciona el caso 3 con un 7,06% y en el último lugar se ubica el caso 4 con un 7,94%.

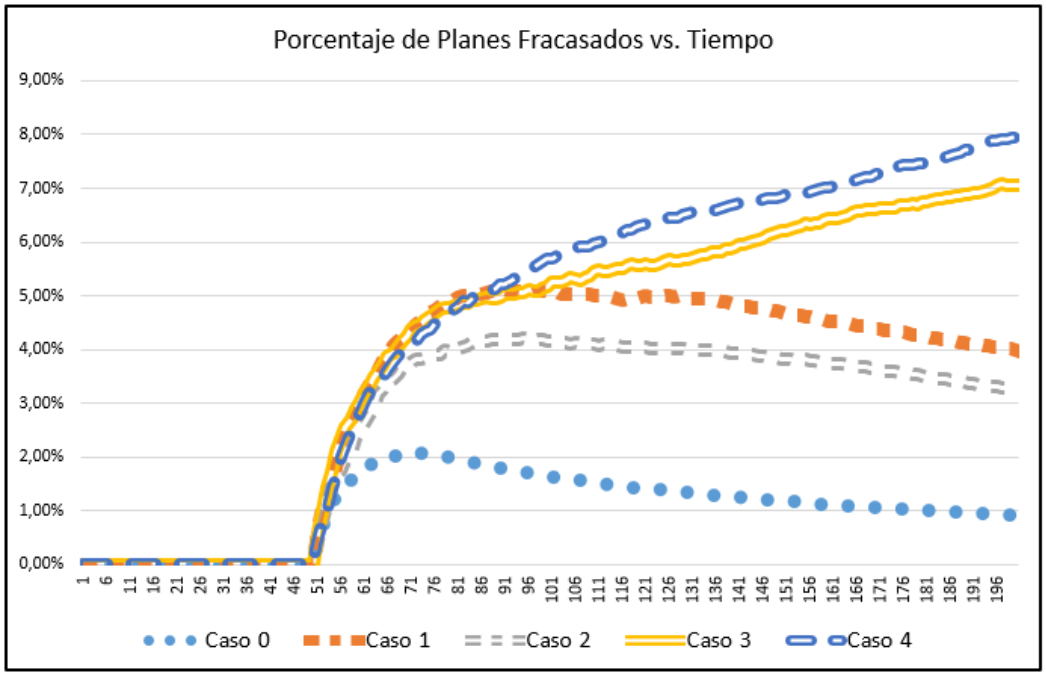

Figura 5.11 – Porcentaje de Planes Fracasados vs. Tiempo

El último de los gráficos relacionado con los distintos tipos de planes describe el porcentaje de planes de contingencia ejecutados (figura 5.12). Los resultados que se observan van de la mano con los ya estudiados en las dos gráficas anteriores. Durante los primeros 50 ciclos, por definición, el sistema actúa por contingencia. Esto se observa en los dos gráficos previos en donde hasta dicho instante, el porcentaje de planes exitosos y de planes fracasados es 0%. Aquí, por el contrario, hasta el instante 50 el 100% de los planes se ejecuta por contingencia para luego decrecer gradualmente. El

valor final de cada uno de los casos es equivalente a la diferencia que surge entre el 100% y la suma de los porcentajes de los planes fracasados y de los planes exitosos. El caso 0 finaliza con un valor de 30,22%, y el resto de los casos alcanzan un valor similar: el caso 4 un 51,24%, el caso 2 un 51,35%, el caso 3 un 53,27% y el caso 1 un 54,45%.

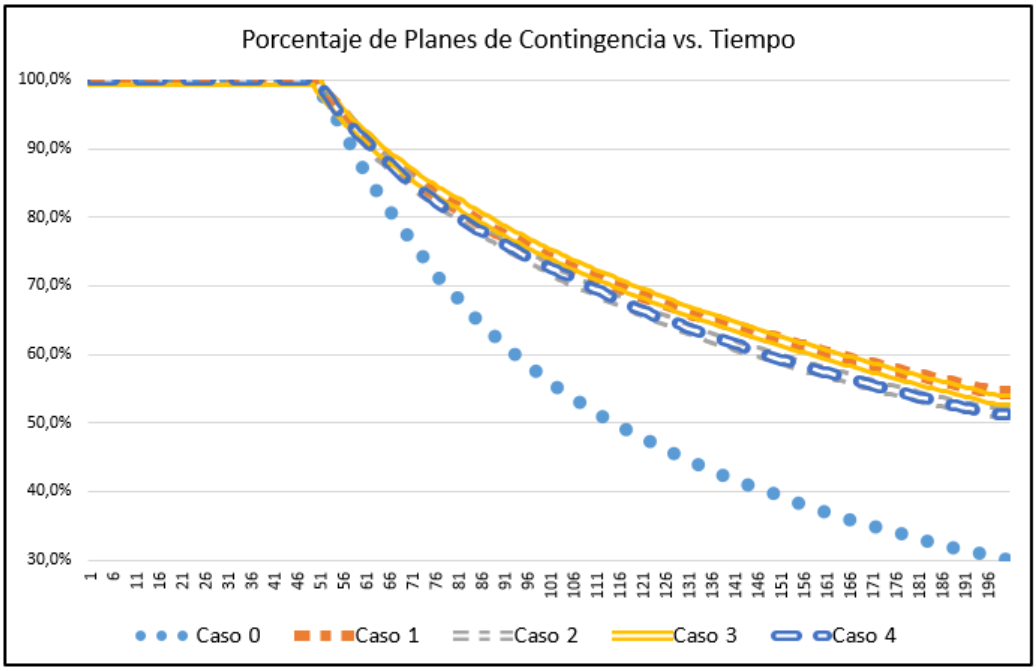

Figura 5.12 – Porcentaje de Planes de Contingencia vs. Tiempo

La figura 5.13 describe la longitud promedio de los planes exitosos. Se entiende que una planificación que consta de un mayor número de pasos y que logra ser exitosa, es reflejo de un proceso de planificación y aprendizaje más refinado. Se observa que el modelo original (caso 0) resulta en una longitud promedio menor que el resto (1,21 pasos) y que el caso 1 alcanza el mejor resultado con 1,67 pasos promedio. Esto es esperable considerando que la mejora ATM genera una mayor cantidad de teorías en donde al menos una de las situaciones involucradas es mutante (inicial y/o final). Por lo tanto, habría una mayor proporción de pasos que incluyen situaciones esperadas mutantes, lo que provocaría un aumento en la probabilidad de éxito del paso en cuestión (la probabilidad de éxito de un paso a ejecutar es mayor si la situación esperada es mutante, ya que como mínimo existen dos situaciones con la que se alcanza el éxito de dicho paso, en vez de una). En segundo lugar se posiciona el caso 2 con 1,54 pasos. En este caso la explicación también puede encontrarse en las teorías mutantes, ya que la mejora CST premia a la teoría mutante vinculada a la teoría que fracasó, repercutiendo en un aumento del *P* de dicha teoría. Este aumento incrementaría

las posibilidades de que ella sea incluida en el armado de un plan, y en consecuencia, aumentaría la probabilidad de éxito de su paso asociado, como ya se ha explicado.

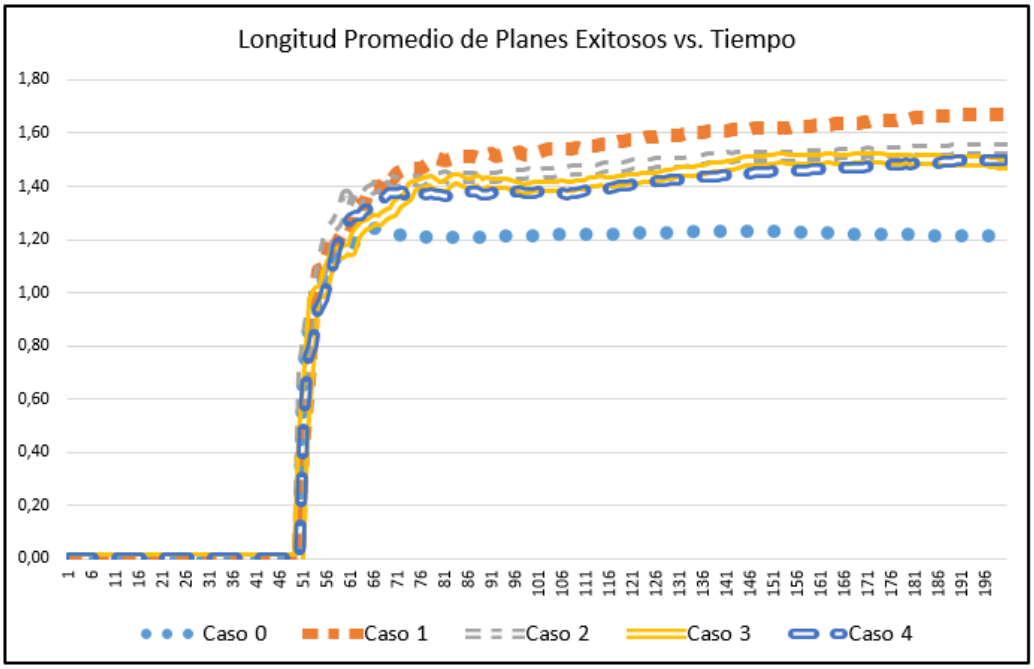

Figura 5.13 – Longitud Promedio de Planes Exitosos vs. Tiempo

La figura 5.14 presenta la longitud promedio de los planes fracasados. En principio, aquí también parecería deseable que la longitud de los planes fracasados sea mayor, ya que eso reflejaría un proceso de aprendizaje y planificación más ambicioso. Pero cabe preguntarse lo siguiente. ¿Es realmente preferible un plan de 4 pasos que fracase en el primer paso a uno de 1 paso que también fracase? Quizás lo es si se resalta el mérito de haber podido armar un plan de 4 pasos, pero también se puede argumentar que si fracasó en el primer momento, el proceso generó una pérdida de tiempo mayor en el armado, repercutiendo negativamente en el rendimiento del sistema. Por lo tanto, antes de responder a dicha pregunta es necesario conocer el porcentaje de pasos exitosos dentro de los planes fracasados, información incluida en la figura 5.15.

Luego de un análisis simultáneo de las gráficas 5.14 y 5.15, se puede concluir que una mayor longitud promedio de los planes fracasados es reflejo de un mejor rendimiento del sistema. Esto se verifica en que, los casos 4 y 3, que son los que mayor longitud promedio alcanzan (1,96 pasos y 1,87 pasos) son a su vez, los que mayor porcentaje de pasos exitosos tienen en los planes que han fracasado (12,62 % y 9,97%).

En relación al modelo original, se observa en la figura 5.14 que se posiciona en último lugar, con una longitud promedio de 1,03 y en la figura 5.15 en la anteúltima posición con un porcentaje de 8,08% de pasos exitosos en los planes fracasados. Sobre los mejores resultados obtenidos con los casos 4 y

3, la explicación puede encontrarse en que ambas incluyen la mejora ICD. Dicha mejora, como ya se comentó en apartados anteriores, es la única que genera una disminución del índice de confiabilidad cuando los últimos planes han sido exitosos en un ciento por ciento. En consecuencia, esta baja del umbral con el que se evalúan los planes, podría permitir que se ejecuten planes más largos, ya que la longitud del plan presenta una relación inversamente proporcional (en la mayoría de los casos) con su probabilidad de éxito.

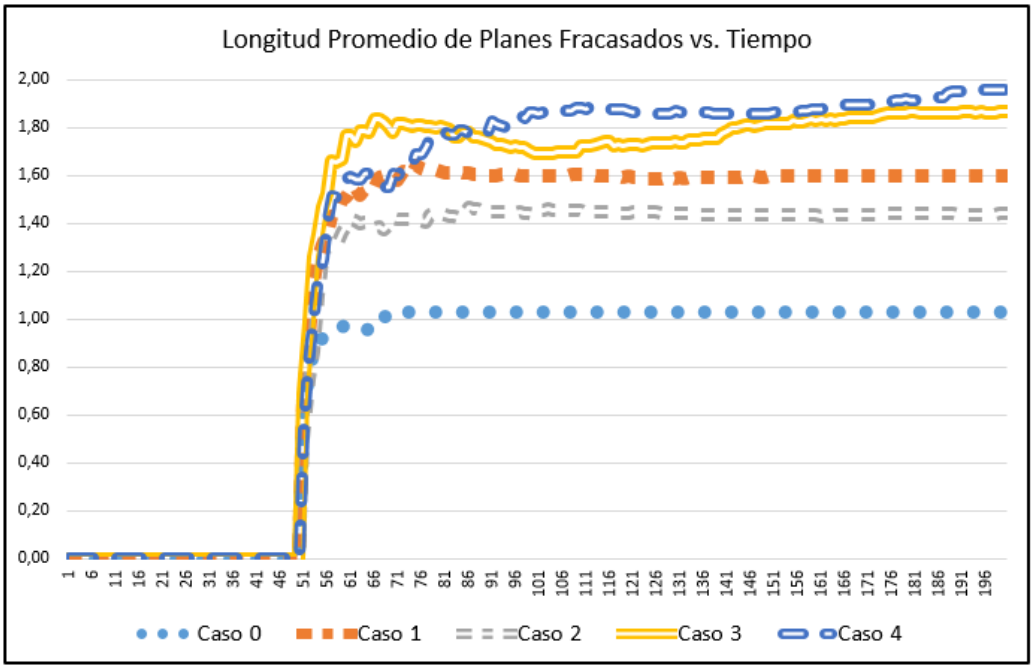

Figura 5.14 – Longitud Promedio de Planes Fracasados vs. Tiempo

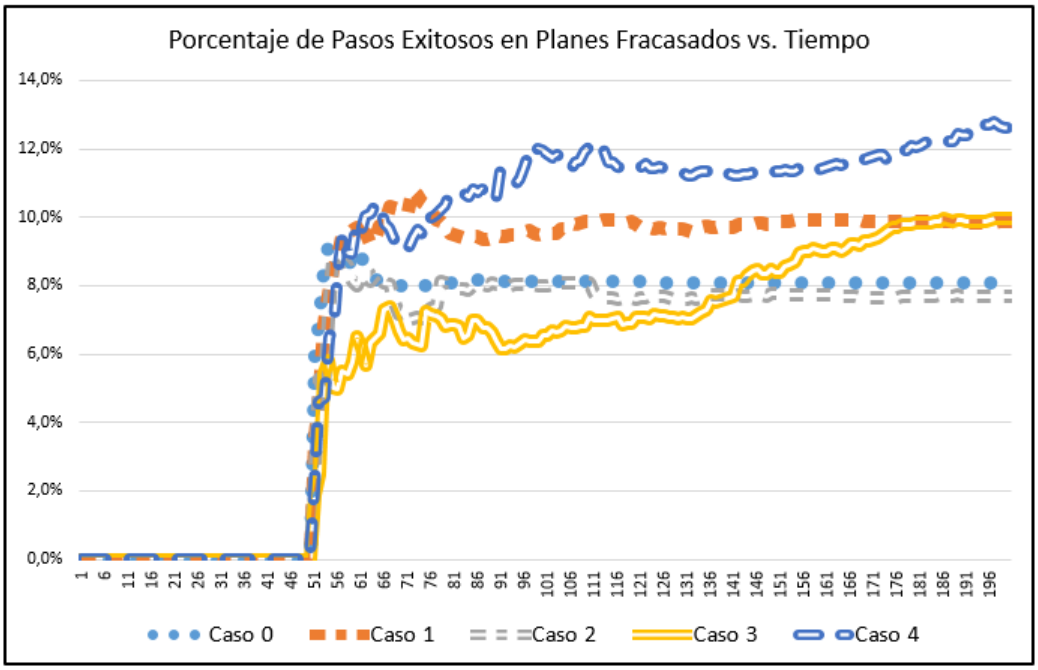

Figura 5.15 – Porcentaje Promedio de Pasos Exitosos en Planes Fracasados vs. Tiempo

5.3.4 Matriz de Valores Finales Promedio

En la tabla 5.3 se sintetizan los valores finales promedio de la totalidad de las variables dependientes que se han evaluado, graficado e interpretado previamente. En sombreado se identifican los valores más altos que se han alcanzado en cada una de ellas.

Grupo de Variables	*Variable Dependiente*	*Caso*				
		0	*1*	*2*	*3*	*4*
Puntos de Energía, Utilidad y Descubrimiento Escenario	*% Recorrido sobre Puntos de Energía*	22,96%	21,38%	17,46%	**24,18%**	21,06%
	% Puntos de Energía Alcanzados	14,81%	28,89%	26,48%	**33,70%**	31,85%
	Utilidad Promedio	0,48	0,50	0,46	**0,52**	0,50
	% Descubrimiento del Escenario	19,21%	36,49%	35,71%	**41,04%**	38,78%
Situaciones y Teorías	*Cantidad de Situaciones Percibidas*	13,26	17,71	17,10	**18,59**	18,14
	Cantidad de Teorías Creadas	42,69	72,72	68,13	**79,77**	77,09
Planificación	*% Planes Exitosos*	**68,86%**	41,55%	45,39%	39,67%	40,83%
	% Planes Fracasados	**0,92%**	4,00%	3,26%	7,06%	7,94%
	Longitud de Planes Exitosos	1,21	**1,67**	1,54	1,49	1,50
	Longitud de Planes Fracasados	1,03	1,60	1,44	1,87	**1,96**
	% Pasos Exitosos en Planes Fracasados	8,08%	9,87%	7,69%	9,97%	**12,62%**

Tabla 5.3 – Matriz de los Valores Finales Promedio para cada Variable y cada Caso

En relación al porcentaje de recorridos sobre puntos de energía, al porcentaje de puntos de energía alcanzados, a la utilidad promedio y al porcentaje de descubrimiento del escenario, el caso 3 es el que mejores valores presenta. Lo mismo sucede respecto a la cantidad de situaciones percibidas y a la cantidad de teorías creadas. Ahora bien, los resultados están divididos si se consideran las variables del proceso de planificación. Analizando el porcentaje de planes exitosos y el de planes fracasados, los mejores valores se obtienen con el caso 0 (modelo original). Si se evalúa la longitud de los planes exitosos, el mejor resultado se ubica en el caso 1. Por último, si se valora la longitud de los planes fracasados y el porcentaje de pasos exitosos en los planes fracasados, el mejor caso resulta ser el 4.

5.4 ÍNDICE DE MEJORA

En la presente sección se estudian los índices de mejora (IM) obtenidos para cada uno de los casos evaluados (tabla 5.4). Como se ha descrito en el capítulo anterior, el índice de mejora es un indicador pentadimensional compuesto por los siguientes elementos:

i- Porcentaje de Recorrido por Puntos de Energía (RPE);
ii- Porcentaje de Descubrimiento del Escenario (DES);
iii- Resultado Neto de Planes (RNP);
iv- Cantidad de Teorías Creadas (TEO);
v- Longitud Promedio de Planes (LGP).

La tabla 5.4 describe, para cada uno de estos 5 elementos: los valores obtenidos para cada caso (*Valor*); los valores obtenidos habiéndole aplicado la ponderación correspondiente (*Valor Pond*); los porcentajes de cambio de los valores obtenidos respecto al valor base (*Dif c/Base - Valor*); y los porcentajes de cambio ponderados (*Dif c/Base - Pond*). A partir de este último número, se obtiene el IM resultante para cada uno de los casos analizados.

ÍNDICE DE MEJORA							
Caso	Componente / Pond	RPE	DES	RNP	TEO	LGP	IM
		0,30	*0,25*	*0,25*	*0,10*	*0,10*	
0 (Base)	Valor	0,23	0,19	0,68	42,69	1,12	
	Valor Pond	0,07	0,05	0,17	4,27	0,11	
	Dif c/Base - Pond	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
1	Valor	0,21	0,36	0,38	72,72	1,63	
	Valor Pond	0,06	0,09	0,09	7,27	0,16	
	Dif c/Base - Valor	-6,90%	89,91%	-44,73%	70,33%	45,76%	
	Dif c/Base - Pond	**-2,07%**	**22,48%**	**-11,18%**	**7,03%**	**4,58%**	**20,83%**
2	Valor	0,17	0,36	0,42	68,13	1,49	
	Valor Pond	0,05	0,09	0,11	6,81	0,15	
	Dif c/Base - Valor	-23,96%	85,84%	-38,00%	59,57%	32,93%	
	Dif c/Base - Pond	**-7,19%**	**21,46%**	**-9,50%**	**5,96%**	**3,29%**	**14,03%**
3	Valor	0,24	0,41	0,33	79,77	1,68	
	Valor Pond	0,07	0,10	0,08	7,98	0,17	
	Dif c/Base - Valor	5,32%	113,57%	-52,00%	86,84%	49,56%	
	Dif c/Base - Pond	**1,60%**	**28,39%**	**-13,00%**	**8,68%**	**4,96%**	**30,63%**
4	Valor	0,21	0,39	0,33	77,09	1,73	
	Valor Pond	0,06	0,10	0,08	7,71	0,17	
	Dif c/Base - Valor	-8,27%	101,81%	-51,59%	80,57%	54,14%	
	Dif c/Base - Pond	**-2,48%**	**25,45%**	**-12,90%**	**8,06%**	**5,41%**	**23,55%**

Tabla 5.4 – Índice de Mejora para cada uno de los casos evaluados

La elección de las ponderaciones para cada uno de los elementos que componen el indicador, ha sido determinada a partir de las consideraciones, ya mencionadas, acerca de la jerarquía de necesidades de Maslow, personalizadas para el caso del robot explorador.

Se observa que las extensiones implementadas al modelo, todas ellas, resultan en una mejora al mismo. El caso 3 alcanza el valor más alto, un 30,63%, que representa una mejora en esa magnitud respecto del modelo original. En segundo lugar, se posiciona el caso 4, con una mejora del 23,55%, luego sigue el caso 1 con un valor de 20,83 % y, en último lugar, el caso 2 con una mejora del 14,03%.

Los resultados obtenidos permiten la siguiente interpretación. Cuando se decide estudiar el modelo desde un punto de vista pentadimensional, la mejora que genera el ICD conjuntamente con la PGI+ATM (implementadas en los casos 3 y 4) en la mayoría de las variables consideradas, compensa y supera largamente la caída del rendimiento en la planificación.

6. CONCLUSIONES

En este capítulo se presentan los resultados finales y las conclusiones del trabajo (sección 6.1), y las futuras líneas de investigación que de él se desprenden (sección 6.2).

6.1 RESULTADOS FINALES Y CONCLUSIONES

Luego del relevamiento documental sobre los sistemas inteligentes autónomos, particularmente de aquellos modelos basados en la creación de teorías para representar su entorno, ha sido la arquitectura LOPE la que ha captado el mayor interés del tesista. Dicho modelo presenta la característica distintiva de incluir un procedimiento de ponderación de teorías que se basa en la cantidad de éxitos que cada una de ellas ha cosechado a lo largo de sus ejecuciones. Esta particularidad permite evaluar la calidad de los planes antes de que estos sean ejecutados y evitar de este modo, que se ejecuten planes con baja probabilidad de éxito. A partir de la publicación del modelo LOPE, trabajos posteriores han propuesto modificaciones o extensiones con el fin de mejorar su rendimiento, y de hecho, lo han logrado. Sin embargo, el autor detectó cuatro oportunidades de mejora que aún no habían sido identificadas y propuso las soluciones correspondientes a cada caso.
A continuación se presenta el resumen de los resultados obtenidos y las conclusiones que de ellos se desprenden:

- El *índice de mejora* alcanzó un resultado positivo en cada una de las cuatro soluciones propuestas. Esto significa que, al margen de las diferencias, todas las mejoras y extensiones han logrado incrementar el rendimiento del sistema. El caso 3 obtuvo el valor más alto, con una mejora del 30,63% en comparación con el diseño original. En segundo lugar se ubicó el caso 4 con un incremento del 23,55%, luego se posicionó el caso 1 con un valor de 20,83% y por último, el caso 2 con un incremento del 14,03%.

  A partir de esto se concluye que la implementación conjunta de las mejoras ICD y PGI+ATM (aplicada a los casos 3 y 4) ha sido la que ha conseguido los mejores resultados.

- De las 11 variables medidas, todas ellas incluidas en la matriz de valores finales promedio (tabla 5.3), sólo dos han alcanzado un rendimiento mayor en la implementación del modelo LOPE original, es decir, sin ninguna mejora aplicada. Dichas variables son: (a) el porcentaje de planes exitosos, con un valor de 68,86%; y (b) el porcentaje de planes fracasados, con un valor de 0,92%.

De estos resultados se desprenden dos conclusiones:

(i) A nivel general, las extensiones y mejoras propuestas han alcanzado resultados por lo menos interesantes, ya que 9 de las variables han mostrado progresos;

(ii) A pesar de que la efectividad de la planificación adquiere su mejor nivel con la arquitectura base, el porcentaje de descubrimiento del escenario es notablemente inferior al resto de los casos (19,21% vs. 38% como valor promedio), motivo por el cual dicha apreciación tiende a debilitarse. ¿Por qué? La respuesta es sencilla. Si el sistema descubre sólo una pequeña proporción del entorno, le será más fácil planificar sus acciones en dichas circunstancias, en comparación con un escenario de mayor tamaño. Y esto es así dado que la cantidad de teorías a partir de las cuales tendrá que elegir en este último caso será mayor (cuanto más se descubre, mayor es el aprendizaje y mayor es la base de conocimiento), y en consecuencia, la probabilidad de no acertar en la planificación aumentará ya que tendrá más posibilidades de combinaciones de situaciones y acciones.

- La implementación de las mejoras PGI y ATM, que conjuntamente conforman el caso 1, presentó avances respecto al diseño original en 8 de las 11 variables medidas. Los mayores progresos se obtuvieron en el porcentaje de puntos de energía alcanzados, en el porcentaje de descubrimiento del escenario, en la cantidad de teorías creadas y en la longitud de planes fracasados y de planes exitosos. Esta última variable alcanzó, además, la mayor longitud de todos los casos evaluados.

  Con estos resultados se concluye que el caso 1 aporta resultados positivos al modelo y sería una buena decisión incluir la prevención de giros ineficientes y la nueva administración de teorías mutantes en futuras implementaciones.

- La implementación de la extensión CST conjuntamente con PGI+ATM (caso 2), presentó mejoras en 6 de las 11 variables medidas, en comparación con el modelo original. Sin embargo, comparando sus resultados con los del caso 1, sólo en dos de ellas se obtuvo mejor rendimiento: en el porcentaje de planes exitosos y en el porcentaje de planes fracasados.

  De lo referido se concluye que si bien la aplicación incremental de la extensión CST mejora el rendimiento del sistema en comparación con el caso base, no lo hace tanto como la PGI+ATM sola. Es más, éste caso obtuvo el índice de mejora más bajo de todos (aunque positivo de todas formas). Por tal motivo, se recomienda revisar el algoritmo de castigo en pos de lograr mejores resultados.

- El caso 3, que incluye las mejoras ICD y PGI+ATM, es aquel que obtuvo el mejor resultado de toda la experimentación. Bajo este escenario se lograron progresos en 9 de las 11 variables medidas. Además, se obtuvieron valores máximos para 6 de ellas: el porcentaje de recorrido sobre puntos de energía, el porcentaje de puntos de energía alcanzados, la utilidad promedio, el porcentaje de descubrimiento del escenario, la cantidad de situaciones percibidas y la cantidad de teorías creadas.

  Por lo tanto, se recomienda rotundamente la inclusión del índice de confiabilidad dinámico (además de PGI+ATM) para futuras implementaciones del modelo LOPE, ya que el incremento en el rendimiento del sistema ha sido sustancial.

- El caso 4, que incluye todas las mejoras propuestas, también logró progresos en 8 de las 11 variables medidas. Sus valores resultaron similares a los obtenidos por el caso 3 y en algunos casos a los obtenidos mediante el caso 1. De todas formas, lo distintivo de este caso es que mediante su implementación se han obtenido valores máximos para dos variables concernientes a la planificación: la longitud de planes fracasados (1,96) y el porcentaje de pasos exitosos en planes fracasados (12,62%).

  De lo descrito surgen dos conclusiones. En primer lugar, más allá de que en términos generales los resultados hayan mostrado un progreso respecto al diseño original, cuando se compara este caso con el 3, pareciera que la aplicación conjunta de las extensiones ICD y CST no resulta favorable. Sin embargo, con la información con la que se cuenta hasta el momento, tampoco es posible sentenciar que ello se deba exclusivamente a que su integración genera una dinámica contraproducente para el sistema. Esto es así dado que previamente se observó que la mejora CST no ha logrado alcanzar las expectativas. Por tal motivo, habría que volver a ejecutar este mismo caso si es que en algún futuro se desarrolla un nuevo algoritmo de castigo de planes. Recién ahí podría darse alguna interpretación adicional.

  La segunda conclusión se vincula con la anterior. Resulta curioso que se haya obtenido el valor máximo para la longitud de planes fracasados y para el porcentaje que mide la cantidad de pasos exitosos dentro de ellos, y no haya ocurrido algo similar con la longitud de los planes exitosos. En este caso, también se concluye que esto podría tener que ver con la introducción de la extensión CST, cuyos aportes al modelo no han sido del todo claros.

6.2 FUTURAS LÍNEAS DE INVESTIGACIÓN

Sobre la base de los resultados obtenidos se desprenden las siguientes posibilidades de investigación:

- A partir del bajo rendimiento de la extensión CST, surge la posibilidad de diseñar un nuevo algoritmo que lleve adelante el castigo de los planes que han fracasado. Eventualmente, podría definirse un mecanismo que se retroalimente del resultado de los planes, aunque no necesariamente involucre un castigo. La idea general es que el sistema no pase por alto los resultados de los planes, sino que los utilice de algún modo, no importa cuál, pero que este le permita incrementar las posibilidades de aprendizaje a partir de ellos.

- A pesar de que la mayoría de las variables analizadas han mostrado mejoras luego de haberse implementado las soluciones, los porcentajes de planes exitosos y de planes fracasados no han alcanzado los niveles esperados. Aun habiéndose dado detalladas explicaciones respecto al motivo de por qué esto sucede, es ineludible la clara oportunidad que se presenta para mejorar dichas magnitudes. El refinamiento de algún proceso ya existente o una nueva extensión que permita mejorar la evaluación de calidad de los planes podría ser el camino a seguir.

- La implementación del caso 1 es el resultado de la aplicación simultánea de las mejoras PGI y ATM. La primera de ellas busca evitar la ejecución de acciones redundantes para así lograr ahorrar tiempo de cómputo; la segunda, es una propuesta más sofisticada que propone una nueva administración de las teorías mutantes. Debido a esta diferencia, es decir, a la relativa simplicidad de una propuesta en comparación con la otra, es que el tesista decidió en su momento no implementar la mejora PGI por separado, sino incluirla junto con la ATM. Sin embargo, a la luz de los resultados y con el ánimo de lograr discernir con mayor precisión la contribución de cada una de ellas al resultado final, se considera que en futuras implementaciones sería conveniente estudiar ambas mejoras por separado.

7. REFERENCIAS

Berlanga, A., Borrajo, D., Fernández, F., García-Martínez R., Molina, J. & Sanchis, A. (1999). Robótica Cognoscitiva y Aprendizaje Automático. En A. M. García Serrano, R. Rizo, S. Moral y F. Toledo (eds.), *Libro de Actas de la CAEPIA'99-TTIA'99 VIII Conferencia de la Asociación Española para la Inteligencia Artificial / III Jornadas de Transferencia Tecnológica de Inteligencia Artificial* (pp. 1-8). Murcia, España: Comité Organizador CAEPIA-TTIA'99.

Bock, P. (fall, 1985). The emergence of artificial intelligence: Learning to learn. *AI Magazine*, *6*(3), 180-190.

Carbonell, J. G., Gil, Y. (1990). Learning by experimentation: The operator refinement method. En Y. Kodratoff y R. S. Michalski (eds.), *Machine Learning: An Artificial Intelligence Approach* (vol. 3, pp. 191–213). Palo Alto, CA, EEUU: Morgan Kaufmann.

Carbonell, J. G., Knoblock, C. A. y Minton S. (1990). PRODIGY: An integrated architecture for planning and learning. En K. VanLehn (ed.), *Architectures for Intelligence: The 22nd Carnegie Mellon Symposium on Cognition* (cap. 9, pp. 241-278). Hillsdale, NJ, EEUU: L. Erlbaum Associates.

Carbonell, J. G., Michalski, R. S. y Mitchell T. M. (1983). An overview of machine learning. En R. S. Michalski, J. G. Carbonell y T. M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach* (vol. 1, pp. 3-23). Palo Alto, CA, EEUU: Tioga.

Christiansen, A. D. (1992). *Automatic acquisition of task theories for robotic manipulation* (Tesis de Doctorado). School of Computer Science, Carnegie Mellon University. Pittsburgh, PA, EEUU.

Etzioni, O. (1990). *A structural theory of explanation-based learning* (Tesis de Doctorado). School of Computer Science, Carnegie Mellon University. Pittsburgh, PA, EEUU.

Fritz, W. (1984). The intelligent system. *ACM SIGART Bulletin*, *90*, 34-38.

Fritz, W. (1992). World view and learning systems. *Robotics and Autonomous Systems*, *10*(1), 1-7.

Fritz, W., García-Martínez, R., Rama, A., Blanqué, J., Adobatti, R. y Sarno, M. (1989). The autonomous intelligent system. *Robotics and Autonomous Systems*, *5*(2), 109-125.

García-Martínez, R. (1997). *Un modelo de aprendizaje por observación en planificación* (Tesis de Doctorado). Facultad de Informática, Universidad Politécnica de Madrid. Madrid, España.

García-Martínez, R. y Borrajo, D. (1997). Planning, learning and executing in autonomous systems. *Lecture Notes in Artificial Intelligence*, sub-serie of *Lecture Notes in Computer Science*, *1348*, 208-220.

García-Martínez, R. y Borrajo, D. (2000). An integrated approach of learning, planning and executing. *Journal of Intelligent and Robotic Systems, 29*(1), 47-78.

García-Martínez, R., Borrajo, D., Britos, P. y Maceri, P. (2006). Learning by knowledge sharing in autonomous intelligent systems. *Lecture Notes in Artificial Intelligence*, sub-serie of *Lecture Notes in Computer Science, 4140*, 128-137.

García-Martínez, R., Servente, M. y Pasquini, D. (2003). *Sistemas inteligentes*. Buenos Aires: Editorial Nueva Librería.

Hayes-Roth, F. (1983). Using proofs and refutations to learn from experience. En R. S. Michalski, J. G. Carbonell y T. M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach* (vol. 1, pp. 221-240). Palo Alto, CA, EEUU: Tioga.

Ierache, J. S. (2010). *Modelo de ciclo de vida para el aprendizaje basado en compartición de conocimientos en sistemas autónomos de robots* (Tesis de Doctorado). Facultad de Informática, Universidad Nacional de La Plata. La Plata, Buenos Aires.

Ierache, J. S., García-Martínez, R. y De Giusti, A. (2008), Learning life cycle in autonomous intelligent systems. En M. Bramer (ed.), *Artificial Intelligence in Theory and Practice II: IFIP 20th world computer congress* (pp. 451-455). Nueva York, EEUU: Springer.

Joseph, R. L. (1989). Graphical knowledge acquisition. En B. R. Gaines y J. H. Boose (eds.), *Proceedings of the 4th Knowledge Acquisition for Knowledge-Based Systems Workshop* (pp. 18.1-16). Calgary, Canada: SRDG Publications.

Knoblock, C. A. (1994). Automatically generating abstractions for planning. *Artificial Intelligence, 68*(2), 243-302.

Kodratoff, Y. (1988). *Introduction to machine learning*. San Mateo, CA, EEUU: Morgan Kaufmann.

López, D., Merlino, H., Ierache, J. S. y García Martínez, R. (2008). A method for pondering plans in autonomous intelligent systems. En *Anales del V Workshop de Inteligencia Artificial Aplicada a la Robótica Móvil* (pp. 98-104). Neuquén, Argentina: Universidad Nacional del Comahue.

Maslow, A. H. (1943). A theory of human motivation. *Psychological Review, 50*(4), 370-396.

Michalski, R. S. (1986). Understanding the nature of learning: Issues and research directions. En R. S. Michalski, J. G. Carbonell y T. M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach* (vol. 2, pp. 3-25). Los Altos, CA, EEUU: Morgan Kaufmann.

Michalski, R. S. y Kodratoff, Y. (1990). Research in machine learning: recent progress, classification of methods and future directions. En Y. Kodratoff y R. S. Michalski (eds.), *Machine Learning: An Artificial Intelligence Approach* (vol. 3, pp. 3-30). Palo Alto, CA, EEUU: Morgan Kaufmann.

Minsky, M. L. (1954). *Theory of neural-analog reinforcement systems and its application to the brain-model problem* (Tesis de Doctorado). Princeton University. Princeton, NJ, EEUU.

Minsky, M. L. (1963). Steps toward artificial intelligence. En E. A. Feigenbaum y J. Feldman (eds.), *Computers and Thought* (pp. 406-450). Nueva York, NY, EEUU: McGraw-Hill.

Minsky, M. L. y Selfridge, O. G. (1961). Learning in random nets. En E. C. Cherry (ed.), *Information Theory: Fourth London Symposium* (pp. 335-347). Londres, Inglaterra: Butterworths.

Minton, S. (1988). *Learning effective search control knowledge: An explanation-based approach* (Tesis de Doctorado). School of Computer Science, Carnegie Mellon University. Pittsburgh, PA, EEUU.

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence, 42*(2-3), 363-391.

Mondada, F., Franzi, E. y Guignard A. (1999). The development of Khepera. En A. Löffler, F. Mondada y U. Rückert (eds.), *Proceedings of the First International Khepera Workshop: Experiments with the Mini-Robot Khepera* (vol. 64, pp. 7-14). Paderborn, Alemania: Heinz Nixdorf Institut, Universität Paderborn.

Moravec, H. P. (1988). *Mind children, the future of robot and human intelligence*. Cambridge, MA, EEUU: Harvard University Press.

Rivest, R. L. y Schapire, R. (1987). Diversity-based inference of finite automata. En *28th Annual Symposium on Foundations of Computer Science* (pp. 78-87). Washington DC, EEUU: IEEE.

Russell, S. J., Norvig, P. (2004). Agentes Inteligentes. Resolver problemas mediante búsqueda. Búsqueda informada y exploración. En *Inteligencia artificial: un enfoque moderno. Segunda edición* (cap. 2-4, pp. 37-151). Madrid, España: Pearson Educación.

Salzberg, S. (1985). Heuristics for inductive learning. En A. K. Joshi (ed.), *IJCAI' 85 Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (vol. 1, pp. 603-609). San Francisco, CA, EEUU: Morgan Kaufmann.

Shen, W. M. (1989). *Learning from the environment based on percepts and actions* (Tesis de Doctorado). Carnegie Mellon University. Pittsburgh, PA, EEUU.

Shen, W. M. y Simon, H. A. (1989). Rule creation and rule learning through environmental exploration. En N. S. Sridharan (ed.), *IJCAI' 89 Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (vol. 1, pp. 675-680). San Francisco, CA, EEUU: Morgan Kaufmann.

Steinhilber, R. A., García-Martínez, R. y Kuna, D. (2009). Mutación de teorías en sistemas inteligentes autónomos basada en algoritmos genéticos. En J. S. Ierache (comp.), *Anales del*

VII Campeonato de Fútbol de Robots y Workshop de Sistemas Autónomos de Robots (pp. 24-33). Buenos Aires: Universidad de Morón.

Sutton, R. S. y Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA, EEUU: MIT Press.

Veloso, M. M. y Carbonell, J. G. (1990). Integrating analogy into a general problem-solving architecture. En M. Zemankova y Z. Ras (eds.), *Intelligent Systems* (pp. 29-51). Chichester, Inglaterra: Ellis Horwood.

Veloso, M. M., Carbonell, J. G., Pérez, A., Borrajo, D., Fink, E. y Blythe, J. (1995). Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence, 7*(1), 81-120.

Wang, X. (1994). Learning planning operators by observation and practice. En K. Hammond (ed.), *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems* (pp. 335-340). Menlo Park, CA, EEUU: AAAI Press.

Wang, X. (1995). Learning by observation and practice: An incremental approach for planning operator acquisition. En A. Prieditis y S. J. Russell (eds.), *Machine Learning: Proceedings of the 12th International Conference on Machine Learning* (pp. 549-557). San Francisco, CA, EEUU: Morgan Kaufmann.

Wikipedia (28 de febrero de 2009). Pirámide de Maslow [Archivo de Imagen]. Recuperado de http://es.wikipedia.org/wiki/Pir%C3%A1mide_de_Maslow. Página vigente al 17/03/2015.

Wooldridge, M. J. (2011). Intelligent agents. En *An Introduction to Multiagent systems* (cap. 2, pp. 21-47). Chichester, West Sussex, Inglaterra: Wiley.

ANEXO

La figura A.1 describe en detalle los recorridos promedio por cada posición perteneciente a los distintos escenarios, para cada uno de los casos evaluados. Cada caso consta de 108 simulaciones de 200 iteraciones cada una. Los números que se indican en cada una de las cuadrículas del gráfico representan la cantidad promedio de veces que el robot pasó por esa posición. Los bloques negros representan las paredes u obstáculos y los grises los puntos de energía.

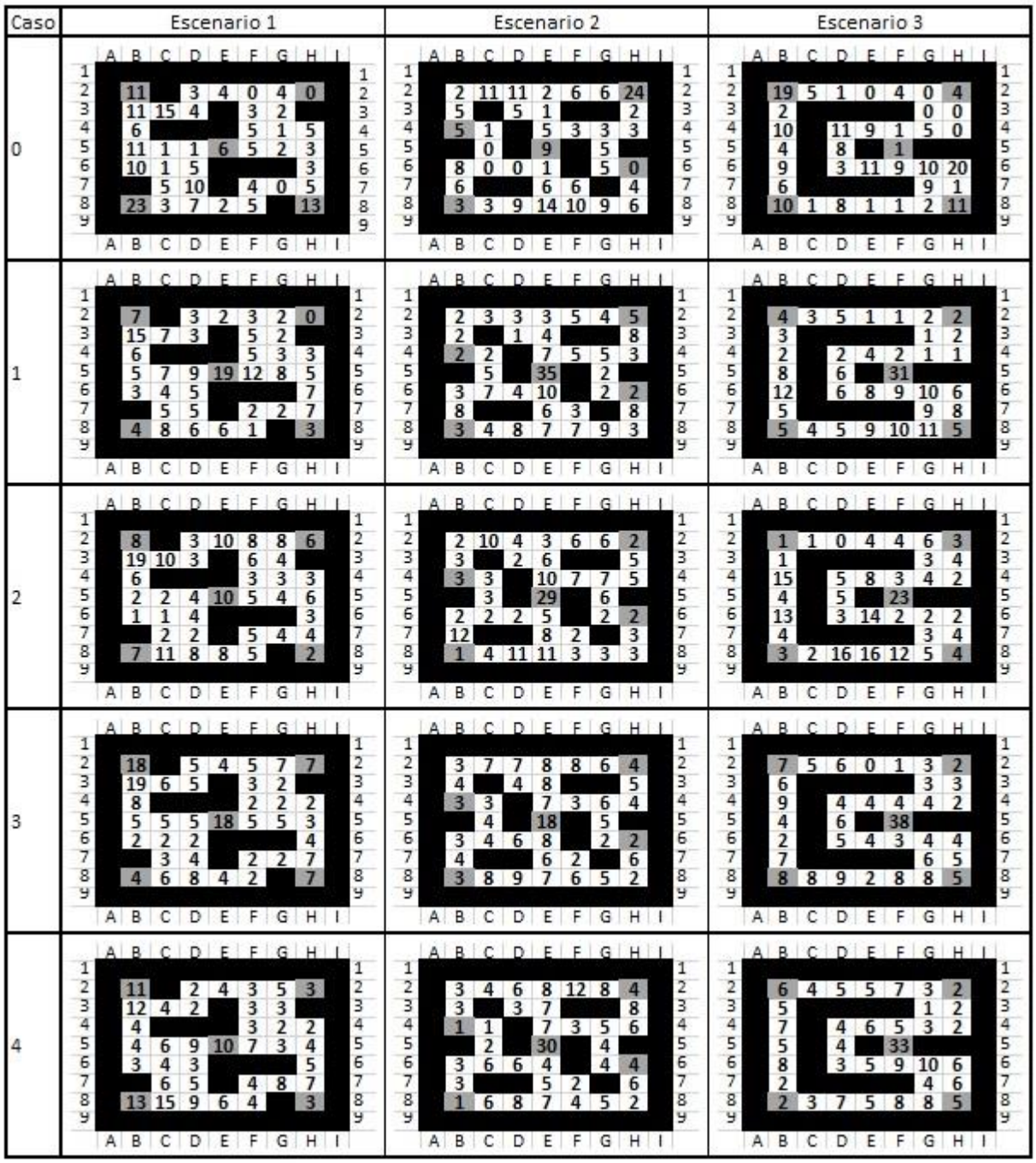

Figura A.1 – Recorrido Promedio por c/u de las posiciones de cada Escenario, para cada Caso