Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática **Informatika Fakultatea**

TITULACIÓN: Ingeniería en Informática

Implementación de algoritmos de Inteligencia Swarm con fines didácticos

Alumno/a: Maider Gaztelu Aza
Profesores: Blanca Rosa Cases Gutierrez y Abdelmalik Moujahid

Proyecto Fin de Carrera, julio de 2013

Implementación de algoritmos Swarm con Fines Didácticos

Resumen

Con este proyecto hemos querido proporcionar un conjunto de recursos útiles para la impartición de un curso de Swarm Intelligence enfocado a la Particle Swarm Optimization (PSO). Estos recursos constan de una aplicación en NetLogo para poder experimentar, ejecutar y visualizar los diferentes modelos de la PSO, un review de la Swarm Intelligence profundizando en la PSO y una ontología de PSO que incluye los recursos bibliográficos necesarios para la investigación y la escritura de artículos.

Palabras Clave

Inteligencia de enjambre; modelos pso; optimización de enjambre de partículas; pájaros; problemas de búsqueda

Irakasteko Helburuarekin Egindako Swarm Algoritmoen Inplementazioa

Laburpena

Proiektu honekin Particle Swarm Optimization-ean(PSO) zentraturiko Swam Intelligence inguruko master mailako ikastaro bat emateko probetxugarri diren zenbait baliabide eskaini nahi izan ditugu. Aipatutako baliabideak onako hauek dira: alde batetik, NetLogo-n programatutako aplikazio bat, PSO-ko modelo desberdinak bistaratu eta hauekin zenbait esperimentu egin ahal izateko; beste aldetik, Swarm Intelligence inguruko *review* bat eta, azkenik, gai honi buruz ikertu eta artikuluak idazteko beharrezkoak diren baliabide bibliografikoak dituen PSO-ko ontologia bat.

Giltza-Hitzak

bilaketa problemak; partikula-multzoen optimizazioa; pso ereduak; swarm adimena; txoriak

Implementation of Swarm Algorithms with a Teaching Purpose

Abstract

With this project we have liked to provide some useful resources for giving a master level course of Swarm Intelligence, focused on Particle Swarm Optimization(PSO). This resources are: a NetLogo aplication for making experiments, executions and visualize the different models of PSO, a review about Swarm Intelligence, also focused on PSO and an ontology of PSO, which includes the necessary bibliographic resources for the investigation and the writing of articles of this topic.

Keywords

birds; particle swarm optimization; pso models; search problems; swarm intelligence

Índice general

Índice de figuras

Índice de cuadros

Parte I

MEMORIA

Índice de la Memoria

Capítulo 1

Introducción

Este documento presenta la Memoria del Proyecto de Fin de Carrera de Ingeniería Informática. A saber, Implementación de Algoritmos de Inteligencia Swarm con Fines Didácticos. Reúne por tanto, la información necesaria para comprender en su totalidad dicho proyecto. La estructura del presente documento se acoge a la norma española UNE 157801.

Capítulo 2

Objeto

En este proyecto de fin de carrera se recopilan los recursos necesarios para apoyar la impartición de un curso de Swarm Intelligence al nivel de máster, siendo necesario para este propósito:

- Elaborar un "review" o revisión bibliográfica del estado del arte de la llamada Swarm Intelligence (SI) centrado en Particle Swarm Optimization (PSO). Se pretende hacer un resumen de lo que es Swarm Intelligence, Ant Colony System y PSO, profundizando en éste último.
- Elaborar recursos informáticos para facilitar la escritura de artículos.

  - Recursos bibliográficos: Se dispone de una amplia bibliografía sobre PSO en diversas librerías digitales a las que está suscrita la UPV/EHU, como son IEEE Xplore, Scopus, Science Direct y CiteULike. Hay, en primer lugar, que constatar la importancia científica que tiene este tema consultando diversas medidas bibliométricas. Por tanto, se pretende proporcionar una recopilación bibliográfica de las publicaciones más relevantes que resulten útiles como punto de partida en la investigación y aprendizaje del tema que nos ocupa. La información útil para las personas que trabajan en PSO se puede catalogar en:
    - La relativa al modelo básico de PSO. En este caso los pocos artículos de los autores de quienes parte las primeras publicaciones (el psicólogo social James Kennedy y el ingeniero Russell C. Eberhart, ambos norteamericanos, así como el libro "Particle Swarm Optimization" en el que divulgan su modelo junto con .Yuhui H. Shi.) se centran en los siguientes temas:
      - Parámetros: obtener valores para un mejor rendimiento del sistema.
      - Simplificaciones: estudiar el modelo dinámico que resulta de fijar parámetros o binarizar el sistema.
    - Las variantes del modelo que surgen a medida que se avanza en los aspectos teóricos. En esta segunda categoría se recopilan:

        - ◇ Ampliaciones: como la llamada differential PSO.
        - ◇ Hibridaciones: que no contemplamos aquí, pero nos centraremos en las más referenciadas.
    - Ecuaciones de los modelos en LaTex: A la hora de comparar los diferentes artículos una de las dificultades que surgen es la identificación de los cambios que se hacen en el modelo original de PSO, si es que los hay.

- Ontología Protegé Frames: Para simplificar el trabajo de comprensión lectora y escritura de artículos científicos, se implementa una ontología en la que se relacionan los modelos recogidos en el Review con las ecuaciones, algoritmos y bibliografía. Dicha ontología se exporta a formato html permitiendo la navegación por clases e instancias siguiendo las relaciones de la ontología.

- Implementación en NetLogo: A la hora de escribir artículos sobre PSO se suelen utilizar recursos de programación recogidos en páginas de benchmark. Estas páginas ofrecen el software ya utilizado por los autores. A saber: Matlab, Python, etc.

  Pensamos que NetLogo (lenguaje para la programación de sistemas multiagente) es el lenguaje más adecuado, tanto para enseñar como para la experimentación científica en el caso de la PSO. En este trabajo se ha querido profundizar en los aspectos avanzados de Netlogo usando extensiones que permitan trabajar con comodidad. En nuestro caso se implementan algunas de las variantes recogidas en la ontología de forma que baste con elegir el modelo correspondiente para que los controles que usa la aplicación queden modificados, funcionalidad que ofrece la extesión Goo de Netlogo.

Capítulo 3

Antecedentes

Si iniciamos una búsqueda de documentos en Scopus sobre PSO obtenemos más de 23000 resultados, como se muestra en la Figura 3.1. Algunos de estos documentos han sido citados en numerosos artículos, siendo un *conference paper* de Kennedy y Eberhart titulado "*Particle Swarm Optimization*" el más destacado con más de 12000 citaciones, tal como se muestra en la Figura 3.2. Desde los primeros documentos presentados en 1995 este algoritmo a suscitado cierto interés entre investigadores que, a lo largo de los años han ido publicando numerosos experimentos en torno a este algoritmo tratando de comprender mejor su comportamiento e ncluso proporcionando desde pequeñas variaciones hasta una gran variedad de hibridaciones que ofrecen mejores resultados que muchos de los algoritmos ya conocidos. En la Figura 3.3 podemos ver que, a partir de 2003 se produce un aumento significativo del número de documentos publicados, alcanzando en 2012 la cifra más alta con más de 4000 artículos.

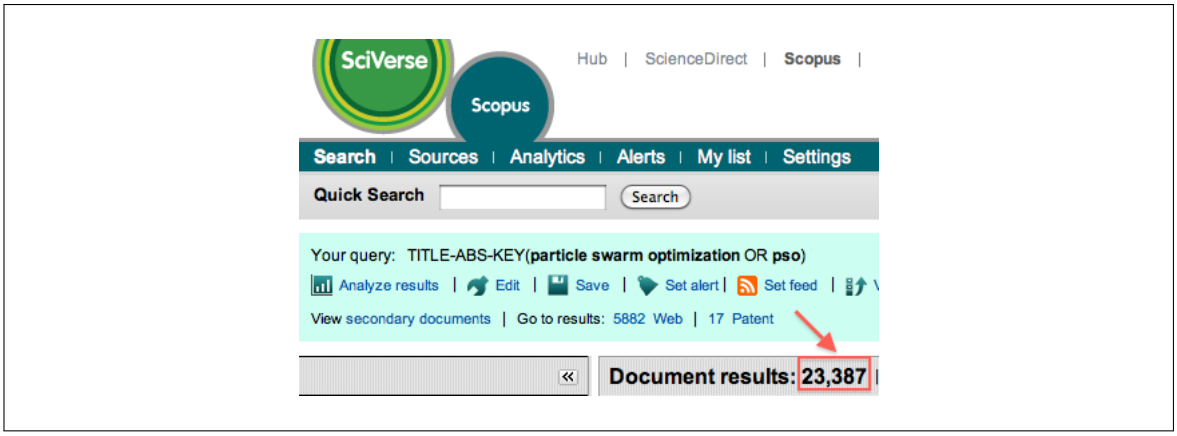
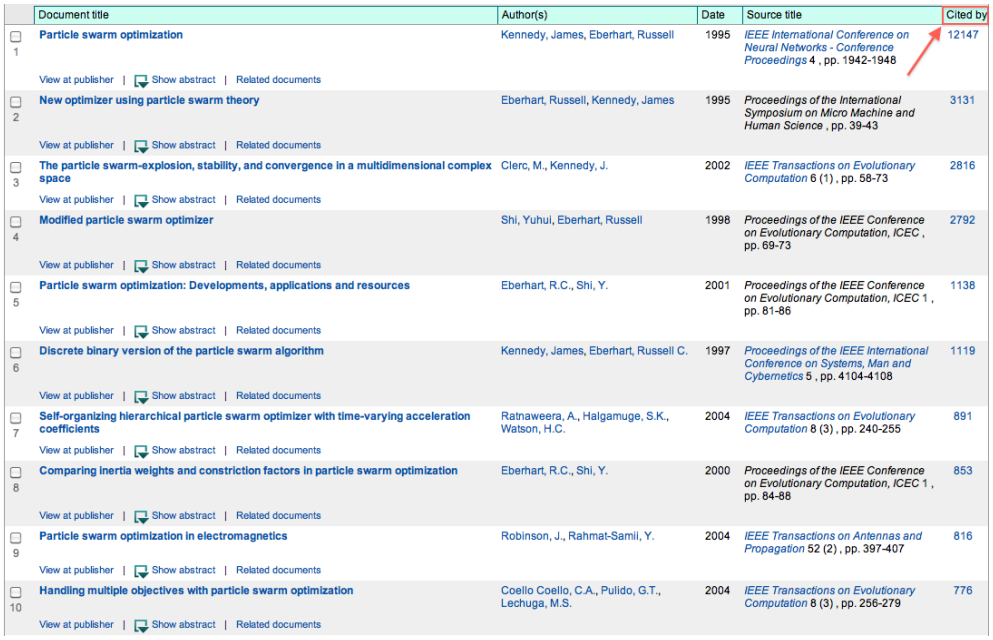


Figura 3.1: Número de artículos sobre PSO. Fuente: Scopus-SciVerse



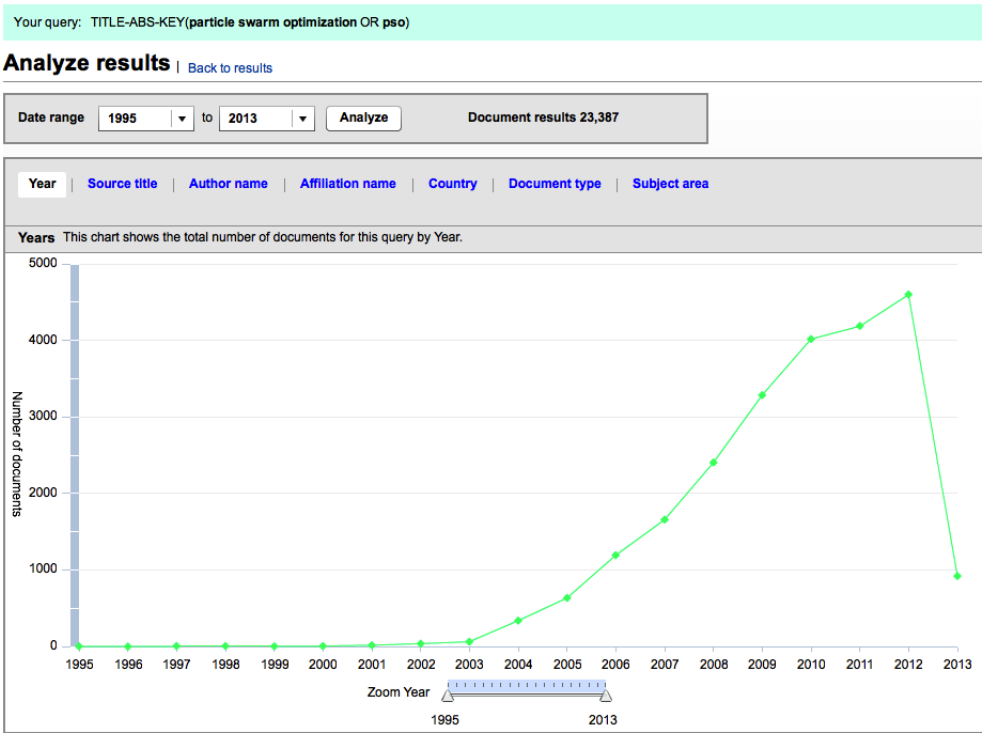
	Document title	Author(s)	Date	Source title	Cited by
1	**Particle swarm optimization** View at publisher | Show abstract | Related documents	Kennedy, James, Eberhart, Russell	1995	*IEEE International Conference on Neural Networks - Conference Proceedings* 4 , pp. 1942-1948	12147
2	**New optimizer using particle swarm theory** View at publisher | Show abstract | Related documents	Eberhart, Russell, Kennedy, James	1995	*Proceedings of the International Symposium on Micro Machine and Human Science* , pp. 39-43	3131
3	**The particle swarm-explosion, stability, and convergence in a multidimensional complex space** View at publisher | Show abstract | Related documents	Clerc, M., Kennedy, J.	2002	*IEEE Transactions on Evolutionary Computation* 6 (1) , pp. 58-73	2816
4	**Modified particle swarm optimizer** View at publisher | Show abstract | Related documents	Shi, Yuhui, Eberhart, Russell	1998	*Proceedings of the IEEE Conference on Evolutionary Computation, ICEC* , pp. 69-73	2792
5	**Particle swarm optimization: Developments, applications and resources** View at publisher | Show abstract | Related documents	Eberhart, R.C., Shi, Y.	2001	*Proceedings of the IEEE Conference on Evolutionary Computation, ICEC* 1 , pp. 81-86	1138
6	**Discrete binary version of the particle swarm algorithm** View at publisher | Show abstract | Related documents	Kennedy, James, Eberhart, Russell C.	1997	*Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* 5 , pp. 4104-4108	1119
7	**Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients** View at publisher | Show abstract | Related documents	Ratnaweera, A., Halgamuge, S.K., Watson, H.C.	2004	*IEEE Transactions on Evolutionary Computation* 8 (3) , pp. 240-255	891
8	**Comparing inertia weights and constriction factors in particle swarm optimization** View at publisher | Show abstract | Related documents	Eberhart, R.C., Shi, Y.	2000	*Proceedings of the IEEE Conference on Evolutionary Computation, ICEC* 1 , pp. 84-88	853
9	**Particle swarm optimization in electromagnetics** View at publisher | Show abstract | Related documents	Robinson, J., Rahmat-Samii, Y.	2004	*IEEE Transactions on Antennas and Propagation* 52 (2) , pp. 397-407	816
10	**Handling multiple objectives with particle swarm optimization** View at publisher | Show abstract | Related documents	Coello Coello, C.A., Pulido, G.T., Lechuga, M.S.	2004	*IEEE Transactions on Evolutionary Computation* 8 (3) , pp. 256-279	776

Figura 3.2: Artículos mas citados

3.1. PSO en pocas palabras

- La PSO[1] fue inicialmente propuesto por Kennedy y Eberhart en 1995. Otros investigadores destacables son: Angeline, Peter J.; Carlisle, Anthony J.; Engelbrecht, Andries P.; Fukuyama, Yoshikazu; Hu, Xiaohui; Mohan, Chilukuri; Parsopoulos, Konstantinos; Shi, Yuhui; Sinclair, Mark C.; van den Bergh, Frans y Xie, Xiaofeng.
- Es un algoritmo de optimización estocástico que tiene sus raíces en la Inteligencia Artificial y la psicología social, así como la ingeniería y la informática.
- Es un ejemplo de la Swarm Intelligence.
- Resulta útil en la resolución de diferentes tipos de problemas : con espacio de búsqueda continuo, discreto o mixto, con varios mínimos locales, con restricciones, problemas multiobjetivo y de optimización dinámica.
- La PSO original no es un optimizador local y por tanto, no hay garantía de que la solución encontrada sea un mínimo local.
- Soporta diferentes topologías.

[1]El estado del arte de la PSO se halla en el anexo III de esta documentación



Year	Documents
2013	923
2012	4600
2011	4188
2010	4020
2009	3286
2008	2405
2007	1658
2006	1193
2005	636
2004	337
2003	63
2002	38
2001	18
2000	7
1999	3

Figura 3.3: Número de documentos clasificados por año

Capítulo 4

Descripción de la situación actual

El punto de partida del proyecto es el siguiente:

- Software:
  - No existe software que unifique los diferentes modelos de la PSO para experimentar con ellos desde una misma interfaz.
  - Para la implementación se requiere *NetLogo v5.0* y la extensión *goo* incluida en él.
- Recursos humanos (alumna):
  - requiere un estudio de la PSO
  - requiere aprendizaje de programación con Netlogo
  - requiere aprendizaje del uso de la herramienta protégé
  - requiere aprendizaje de conocimientos básicos de LaTex
  - requiere aprendizaje de conocimientos básicos html y css

Capítulo 5

Definiciones y Abreviaturas

Nomenclatura

ACO Ant Colony Optimization

APSO Adaptive Particle Swarm Optimization

aPSO Acelerated PSO

DMS-PSO Dynamic Multi-Swarm Particle Swarm Optimizer

FAPSO Fuzzy Adaptive Particle Swarm Optimization

GA Genetic Algorithm

gbest Global Best

HC Hill Climbers

IA Inteligencia Artificial

lbest Local Best

LDIW Lineal Drecreasing Inertia Weight

pPSA Perturbed Particle Swarm Algorithm

PSO Particle Swarm Optimization

PSO Particle Swarm Optimization

PSO-DR Particle Swarm Optimization-Discrete Recombination

RANDIW Random Inertia Weight

SI Swarm Intelligence

SOC Self-Organized Criticality

Capítulo 6

Alcance

Este documento define todo lo que está incluido en el proyecto.

- Se define el siguiente alcance:
  - 1- Proporcionar una única aplicación que reúna un amplio catálogo de los modelos más significativos de la PSO. (NetLogo con extensiones)
  - 2- El software utilizado para la implementación será NetLogo.
  - 3- El proyecto se centrará en los autores más relevantes de la PSO, a saber: Kennedy y Eberhart.
  - 4- Realizar un review sobre la rama del Swarm Intelligence centrado en la PSO.
  - 5- Proporcionar recursos para facilitar la lectura, comprensión y escritura de artículos de esta rama elaborando una ontología en Protégé Frames que relacione cada modelo de PSO con sus referencias bibtex y ecuaciones en Latex.
- Se defienen los siguientes entregables:
  - Entregable número 1: review sobre el estado del arte de Swarm Intelligence.
    - Este entregable vendrá incluido como anexo en la documentación del proyecto.
    - Ofrecerá un breve resumen de lo que es Swarm Intelligence y profundizará en el tema de la Particle Swarm Optimization.
  - Entregable número 2: Ontología de la PSO. Se proporciona de 2 maneras.
    - Entregable 2.1: Ontología en Protégé Frames v. 3.5 beta
      - Se facilitaran los ficheros del proyecto que forma la Ontología en Protégé para poder ser ampliada a gusto del usuario.
      - Se proporcionarán las ecuaciones en LaTex de todos y cada uno de los modelos, así como su bibliografía.
    - Entregable 2.2: Ontología en html

        - ◇ Se proporcionará la Ontología en html para una lectura cómoda desde el navegador web. Se le aplicará el formato y estilo que el alumno/a considere apropiado.

- Entregable número 3: Implementación en NetLogo. Implementación de algunos de los modelos recogidos en el entregable número 1.
    - Entregable 3.1: Ejecutable de la aplicación.
    - Entregable 3.2: Carpeta de trabajo que contiene todos los ficheros necesarios para la implementación de la aplicación.
- Entregable número 4: Manual del Usuario de la Aplicación desarrollada en NetLogo.
    - Un documento en formato pdf que explique el manejo de la aplicación.

Capítulo 7

Estudio de alternativas y viabilidad

7.1. Codificación en NetLogo

7.1.1. ¿Qué es NetLogo?

NetLogo es un entorno de modelado programable para simular fenómenos naturales y sociales, que fue desarrollado por Uri Wilensky en 1999. Dicho entorno permite a estudiantes y profesores abrir simulaciones para "jugar" con ellas, explorando su comportamiento bajo diversas condiciones y ofrece la posibilidad de crear nuevos modelos. Es por tanto una poderosa herramienta para la investigación en diversos campos. NetLogo se ejecuta en la máquina virtual de Java, por lo que funciona en las principales plataformas (Mac, Windows, Linux, etc.). Además nos brinda la posibilidad de ejecutar los modelos como applets de Java en un navegador web.

7.1.2. Modelos swarm con agentes

Este tipo de programación permite dar instrucciones a cientos o miles de "agentes" que operan de manera independientemente. Esto hace que sea posible explorar la relación entre el comportamiento micro-nivel de los individuos y los patrones macro-nivel que surgen de su interacción. Si visualizamos cada individuo del enjambre como un agente, la programación del modelo se vuelve sencilla.

7.1.3. Paradigma multiagente

La PSO ofrece un lenguaje y una herramienta sencilla para codificar paradigmas de sistemas multiagente. Posee numerosos ejemplos y modelos de código abierto que facilitan la comprensión y aprendizaje del lenguaje al mismo tiempo que fomentan la reutilización de código. Además de todos los recursos software descargables desde la página principal[1] posee una extensa documentación y tutoriales, así como un manual[2] detallado acerca del funcionamiento de la aplicación y el lenguaje de programación.

[1] http://ccl.northwestern.edu/netlogo/

[2] http://ccl.northwestern.edu/netlogo/docs/NetLogo%20User%20Manual.pdf

7.2. Ontología en Protégé

7.2.1. Necesidad de una Ontología

Cuando realizamos una investigación, por muy pequeña que ésta sea, leemos, analizamos y contrastamos información de diversas fuentes. En busca de un conocimiento mayor y más preciso, emprendemos un camino donde saltamos de publicación en publicación y cuando uno mira atrás ya no sabe donde puso el pie, ni como llego hasta aquella valiosa información. Seguramente disponemos de toda esa información o, al menos la más relevante, en alguna carpeta ubicada en algún rincón de nuestro ordenador. Sin embargo, cuando llega el momento de escribir, citar, etc. ¿dónde están esas fuentes? ¿dónde esas referencias?... ¡vaya! seguramente nos queda un tedioso trabajo por delante buscando la "información sobre la información". Aun siendo una pequeñísima investigación la que nos ocupa, disponemos de bibliografía suficiente como para precisar alguna forma de organización que nos facilite toda la información necesaria para consultar las fuentes originales o escribir un artículo. Y el tiempo invertido en esta tarea crece de manera insospechada. Nuestra intención es por consiguiente, amenizar y facilitar este trabajo proporcionando una ontología en Protégé. De esta manera, todo alumno o profesor interesado en el tema puede acceder de manera rápida y sencilla a toda la bibliografía con la que se ha trabajado en este pequeño proyecto, organizada en artículos, conferencias, libros y tesis. Dispone de la URL de la fuente original, de la cita correspondiente y del BibText de cada publicación para incorporarlo a su base de datos. Además, para facilitar la escritura de artículos se proporcionan las ecuaciones de los diferentes modelos en código LaTex. El objetivo es permitir la reutilización de conocimiento de este dominio de manera sencilla, creando una base de conocimiento mediante clases, slots, restricciones, jerarquías, etc. que Protégé nos permite gestionar.

7.2.2. Protégé como herramienta de trabajo

Protégé es un editor de ontologías y bases de conocimiento gratuito open source basado en java. Ha resultado ser un software de apoyo muy útil y de fácil manejo. Nos ha permitido crear esta ontología de manera sencilla e intuitiva sin disponer de un gran conocimiento previo del manejo de esta herramienta. Además soporta Frames y OWL y es exportable a una gran variedad de formatos. Protégé está implementado en Java y dispone de numerosos plugins que permiten visualizar gráficamente una ontología, así como ejecutar comandos que modifiquen la misma. Una ontología puede gestionarse a través de comandos, ya que se dispone del plugin "Algernon in Java" para tal propósito.

Capítulo 8

Análisis de Riesgos

En este documento se procede a identificar y analizar los riesgos que pueden afectar al proyecto que nos ocupa. Para la identificación y descripción de los mismos emplearemos una estructura sencilla tal como EVENTO-CAUSA-EFECTO, donde se deduce que si tal CAUSA, puede ocurrir un EVENTO, provocando un EFECTO. Entendemos como "Evento" el riesgo en cuestión. El plan de gestión de riesgos se describe en el Anexo II: Planes de Gestión del Proyecto, concretamente en el capítulo Plan de Gestión de Riesgos.

8.1. Identificación de los riesgos

En el Cuadro 8.2 se muestra la lista de todos los riesgos identificados en este proyecto.

8.2. Evaluación de Probabilidad e Impacto de los Riesgos

Los riesgos han sido clasificados por orden de prioridad de acuerdo con sus implicaciones potenciales de tener un efecto sobre los objetivos del proyecto. En la Figura 8.1 se muestra el recuadro que indica la probabilidad de que un riesgo suceda y el impacto que podría tener si así fuera.

Los colores identifican la gravedad de los riesgos, siendo los verdes los menos preocupantes y los rojos los más graves. Si tuviéramos riesgos en la franja granate, la que pertenece al cuadrante inferior derecho, deberíamos replantearnos la viabilidad del proyecto.

Nº	Evento/Riesgo	Causa	Efecto
1	Baja por enfermedad(a corto o largo plazo).	Desconocida.	A corto plazo: ligera demora en el tiempo. A largo plazo: retraso significativo en el proyecto.
2	Problemas con las herramientas de comunicación.	Pérdida de mensajes a por fallo del servidor de correo.	Retraso en los hitos intermedios del proyecto
3	Problemas con el almacenamiento de la documentación(pérdida de documentos).	Fallos técnicos o fallos humanos.	Retraso en los hitos y gran retraso en el proyecto en general.
4	Cambios en el alcance frecuentes y/o desmesurados.	Proyecto demasiado ambicioso.	Retraso importante en los hitos y en el proyecto.
5	Reestructuración de la organización/replanificación.	Planificación no adecuada o demasiado optimista.	Invertir mas tiempo replanificando.
6	Aparición de tareas imprevistas no planificadas.	Análisis insuficiente,planificación no adecuada.	Invertir mas tiempo replanificando y aumento del tiempo de trabajo de los recursos humanos
7	Comunicación lenta o ausencia de feed-back.	Problemas humanos o tecnológicos.	Retrasos en las tareas.

Cuadro 8.2: Riesgos Identificados

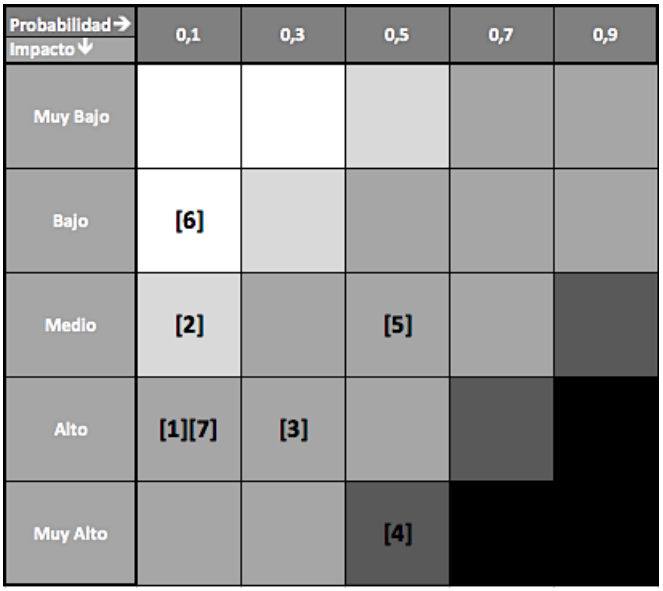


Figura 8.1: Matriz de probabilidad e impacto de los riesgos negativos

Capítulo 9

Organización y Gestión

9.1. Directrices para la Gestión de los Cambios

Para la gestión de cualquier cambio se realizarán las actividades que se muestran en le Figura 9.1 y que vienen numeradas a continuación:

1. Analizar viabilidad e impacto del cambio: Se reunirán la directora del proyecto y la alumna para analizar la viabilidad y evaluar el impacto del cambio a realizar.

2. Descubrir requisitos directamente afectados y dependientes: Se analizarán los requisitos que sean afectados con los cambios para una correcta implementación de cambios.

3. Redacción de propuesta de cambio de los requisitos: Realización de una propuesta de cambios para ser proporcionada a la directora del proyecto. Las modificaciones necesarias se comunicarán y realizarán.

4. Estimación de costes de los cambios: Realización por parte del alumno de una estimación del coste que conlleva el cambio y el informe asociado al mismo.

5. Viabilidad de costes: Valorar la factibilidad del cambio respecto del coste. La decisión final de aceptación o rechazo de los cambios residirá en la directora del proyecto.

6. Realizar cambios: La alumna comenzará las actividades designadas para realizar los cambios reflejados en la propuesta.

9.2. Directrices para el Seguimiento del Proyecto

Se diferencian los siguientes procesos de seguimiento y control:

1. Dar seguimiento y controlar el trabajo del proyecto: revisar, analizar y regular el avance, a fin de cumplir con los objetivos del proyecto. Para ello se realizan informes de estado, reuniones y mediciones del avance.

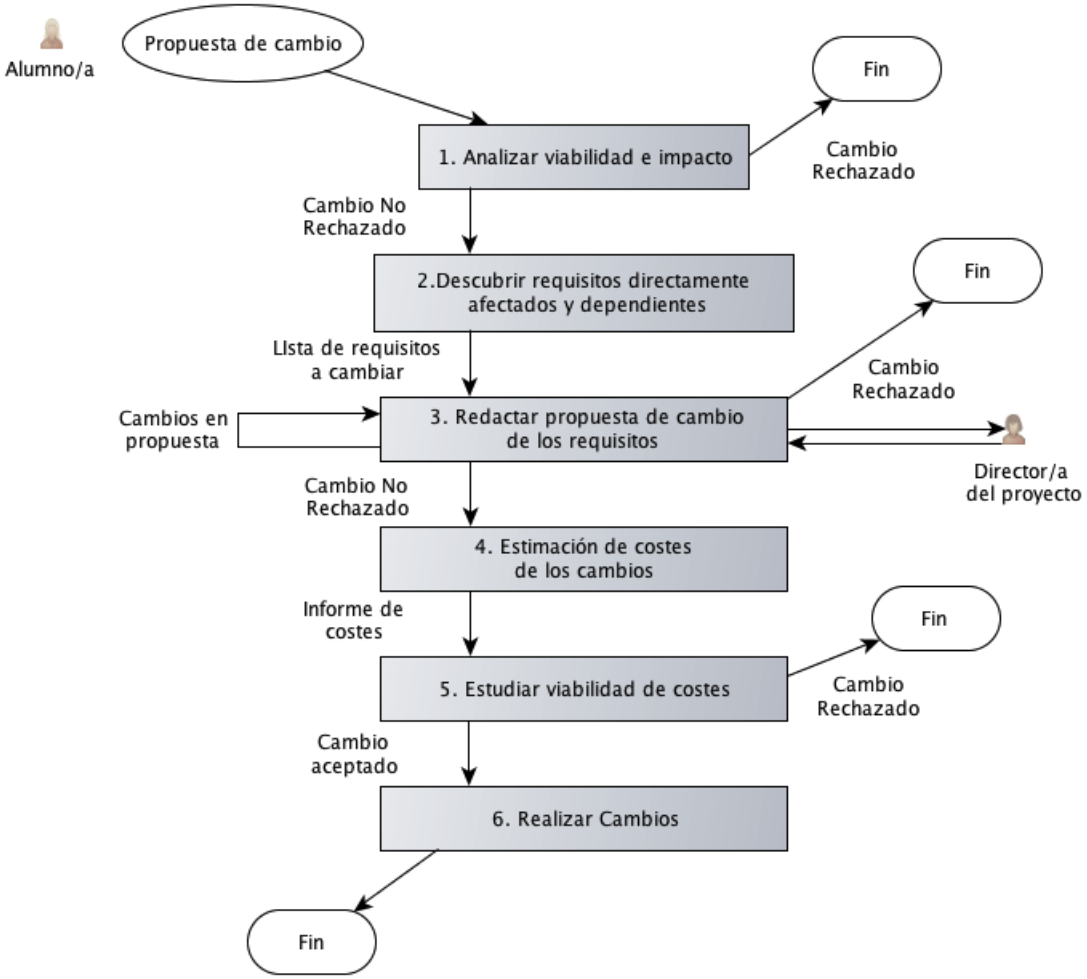


Figura 9.1: Diagrama de gestión de cambios

2. Realizar control integrado de cambios: revisar todas las solicitudes de cambios, aprobar los cambios y gestionar los cambios a los entregables, a los activos de los procesos de la organización, a los documentos del proyecto y al plan para la dirección del proyecto.

3. Verificar el alcance: formalizar la aceptación de los entregables del proyecto que se han completado.

4. Controlar el alcance: dar seguimiento al estado del alcance del proyecto y del producto, así como gestionar cambios a la línea base del alcance.

5. Controlar el cronograma: dar seguimiento a la situación del proyecto para actualizar el avance del mismo y gestionar cambios a la línea base del cronograma.

6. Realizar control de calidad: dar seguimiento y registrar los resultados de la ejecución de actividades de control de calidad, a fin de evaluar el desempeño y recomendar cambios necesarios.

7. Informar del desempeño: recopilar y distribuir información sobre el desempeño, incluidos informes de estado, mediciones del avance y proyecciones.

8. Dar seguimiento y controlar los riesgos: dar seguimiento a los riesgos identificados, identificar nuevos riesgos y evaluar la efectividad del proceso contra riesgos a través del proyecto.

9. Procesos del grupo de proceso de cierre: aquellos procesos realizados para finalizar todas las actividades, a fin de completar formalmente el proyecto o una fase del mismo. Dichos procesos incluyen:

   a) obtener la aceptación de la directora del proyecto.
   b) realizar una revisión tras el cierre del proyecto o la finalización de una fase.
   c) archivar todos los documentos relevantes del proyecto en el sistema de información para ser utilizados como datos históricos.

Además, también se deberán atender los siguientes procesos:

- Controlar cambios y recomendar acciones preventivas para anticipar posibles problemas.

- Dar seguimiento a las actividades del proyecto, comparándolas con el plan para la dirección del proyecto y la línea base de ejecución del proyecto.

- Influir en los factores que podrían eludir el control integrado de cambios, de modo que únicamente se implementen cambios aprobados.

9.3. Directrices para la Recopilación y Distribución de Información del Proyecto

La distribución de la información del proyecto se realizará acorde al Plan de Comunicaciones que figura en el Anexo II del presente documento.

9.3.1. Distribución de la Información: Herramientas y Técnicas

- 1. Habilidades de Comunicación.

  Como parte del proceso de comunicación, el emisor es responsable de hacer que la información sea clara y completa para que el receptor pueda recibirla correctamente, así como confirmar que se ha entendido apropiadamente. El receptor es responsable de asegurarse de que la información sea recibida en su totalidad y entendida correctamente. La comunicación tiene muchas dimensiones.

- 2. Sistemas de Recopilación y Recuperación de Información.

  La información puede recopilarse y recuperarse a través de una gran variedad de medios, entre los que se incluyen: los sistemas manuales de archivo, las bases de datos electrónicas, el software de gestión de proyectos y los sistemas que permiten el acceso a documentación técnica como planos de ingeniería, especificaciones de diseño y planes de prueba.

- 3. Métodos de Distribución de la Información.

  La Distribución de la Información consiste en recopilar, compartir y distribuir información a los interesados en el proyecto de manera oportuna durante todo el ciclo de vida del mismo. La información del proyecto puede distribuirse mediante una gran variedad de métodos, entre los que se incluyen: reuniones del proyecto, herramientas de comunicación y conferencias electrónicas y herramientas electrónicas para la dirección de proyectos.

9.4. Lugar donde se realizará el trabajo.

El proyecto será realizado en el lugar que la alumna considere apropiado. Se dispondrá de un aula en la Facultad de Informática para tal propósito.

Capítulo 10

Planificación Temporal

10.1. Gestión del Tiempo

Las tareas a realizar se dividen en 7 grupos:

- 1- Planificación.
- 2- Análisis y Recopilación.
- 3- Investigación y Formación.
- 4- Diseño.
- 5- Desarrollo.
- 6- Validación y Pruebas.
- 7- Seguimiento y Control.

La estimación del tiempo necesario para cada sección es la siguiente:
El cuadro de control del tiempo queda parcialmente representado a continuación:

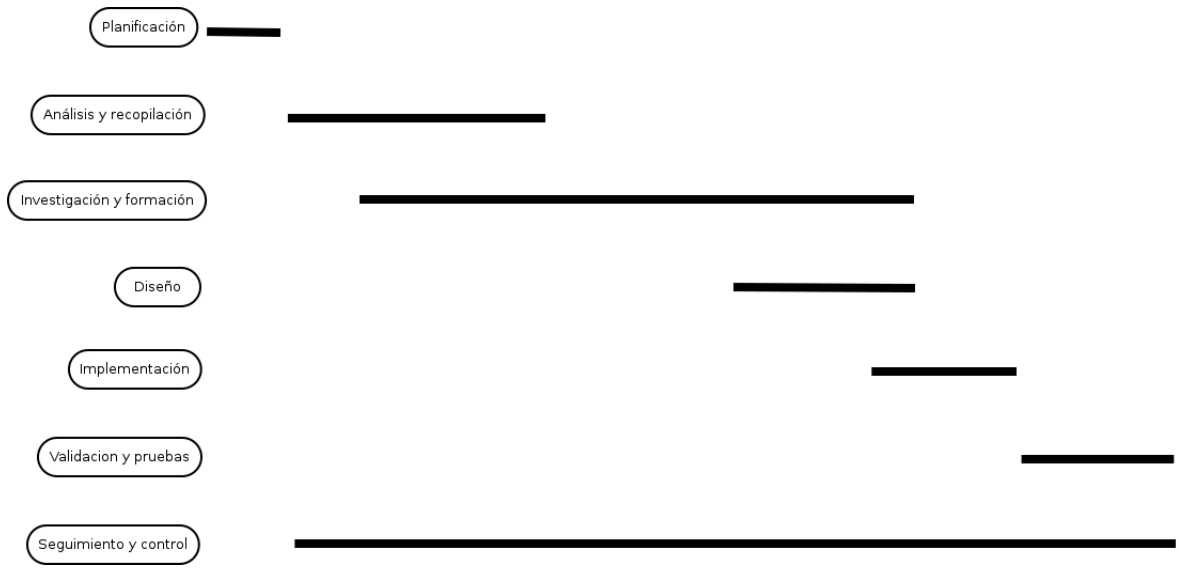


Figura 10.1: Diagrama de Gant

Tarea	**Tiempo estimado(h)**
Planificación	8
Análisis y recopilación	9
Investigación y formación	190
Diseño	11
Implementación	100
Validación y pruebas	10
Seguimiento y Control	20
Total:	348

Cuadro 10.1: Estimación del tiempo necesario

	enero			febrero
	semana 1	semana 2	semana 3	semana 4
1. Planificación	7,00			
2. Análisis y recopilación	4,00	3,00		
3. Investigación y formación	3	14	12	14
4. Diseño				3
5. Desarrollo				1
6. Validacion y pruebas				
7. Seguimiento y control	1,00	1,00	1,00	1,00
t. Real/ semana :	15,00	18,00	13,00	19,00
t. Estimado/semana :	17,00	17,00	17,00	17,00

junio				
semana 20	semana 21	T. Real	T.estimado	% completado
		8,00	8,00	100
		9,00	9,00	100
		197,00	190	98,5
		12,00	11	109,0909091
		106,00	100	106
5		5,00	10	50
1,00		19,00	20,00	95
6,00	0,00	356,00	348,00	
10,00	8,00	354,00		

Figura 10.2: Cuadro de control del tiempo

Parte II

ANEXOS

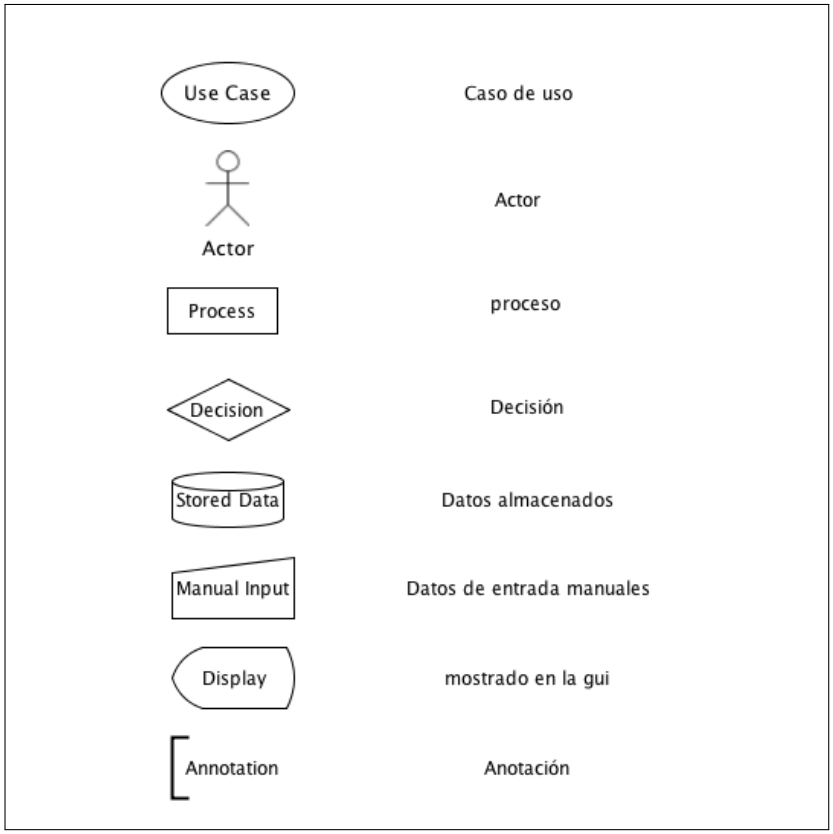
Índice de Anexos

Anexo A

Análisis y Diseño del Sistema

A.1. Análisis y diseño de la aplicación en NetLogo

A continuación mostraremos varias imágenes que especifican el diseño de la aplicación. Para que los elementos utilizados para la representación sean perfectamente comprensibles véase Cuadro A.1. Comenzaremos presentando el diseño con el conocido Diagrama de casos de uso en la Figura A.1, continuaremos con los diagramas de flujo en las 7 figuras siguientes y, por ultimo, mostraremos de manera gráfica las dos topologías que utiliza el programa para definir los vecindarios.



Cuadro A.1: Descripción de elementos de representación

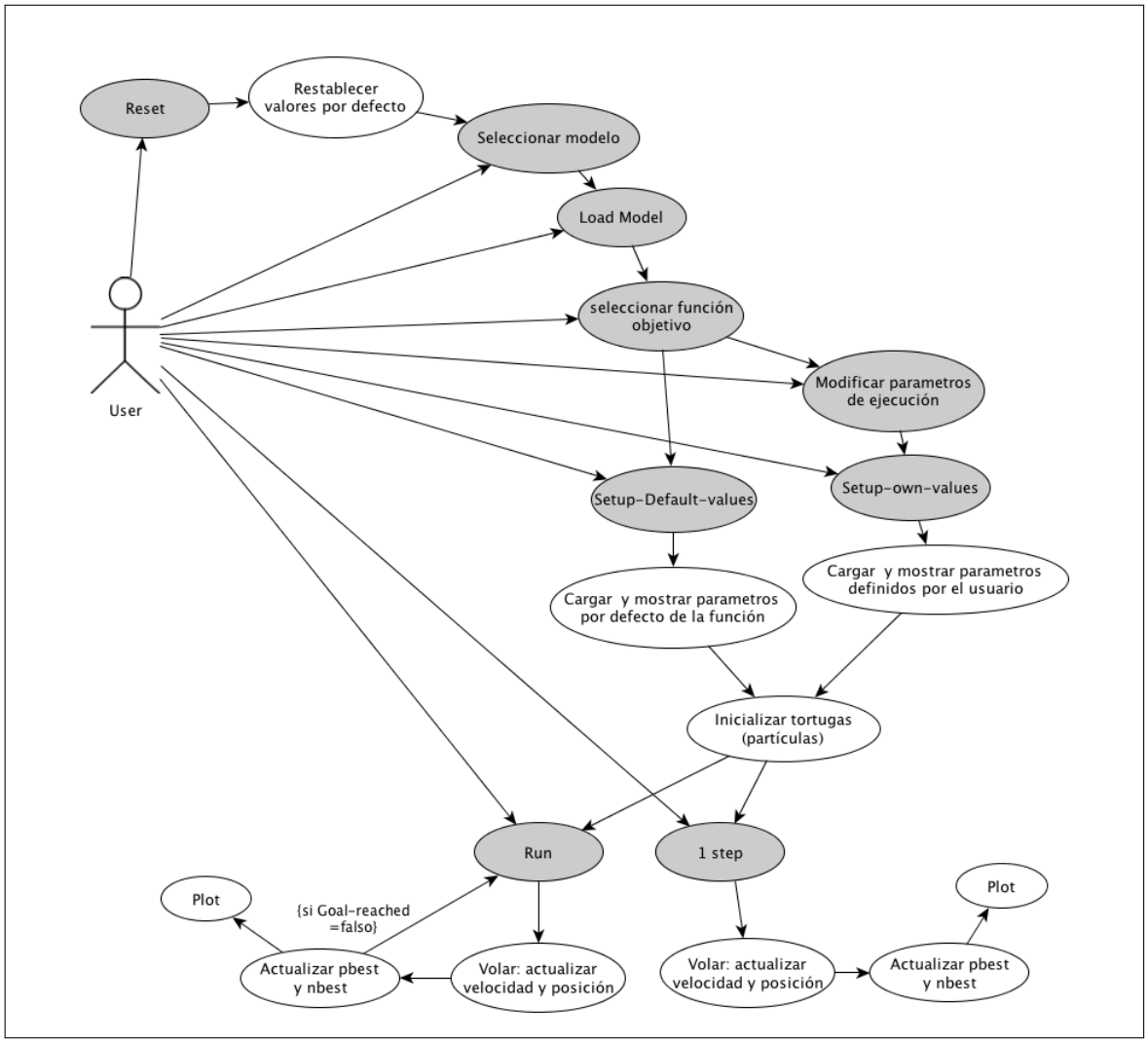


Figura A.1: Diagrama de casos de uso

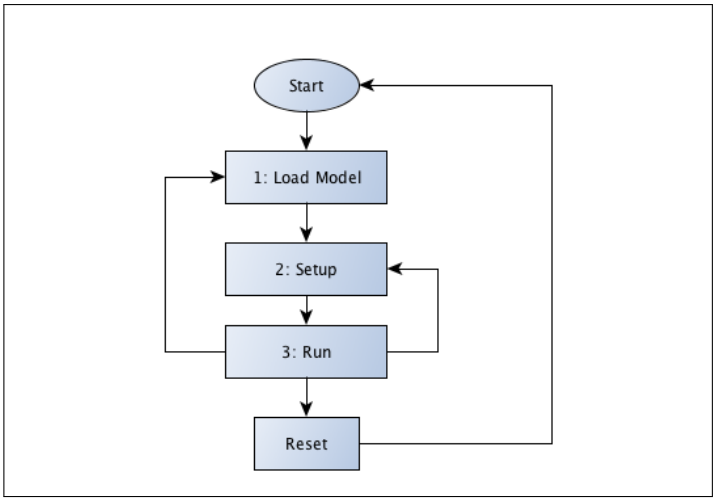


Figura A.2: Algoritmo general

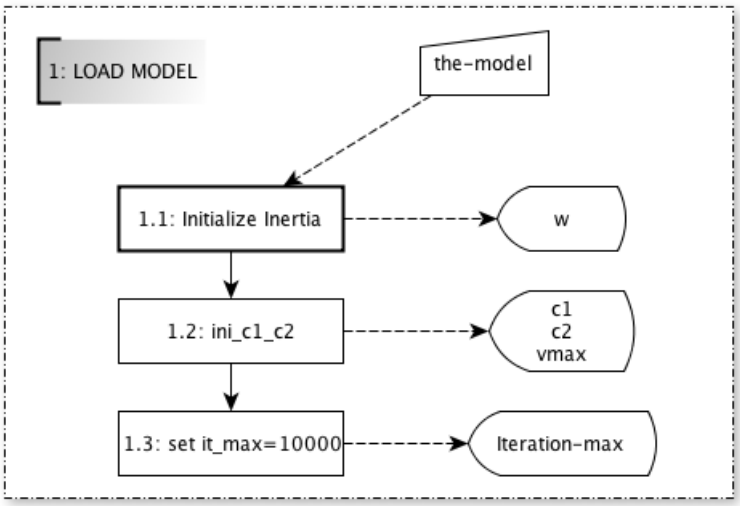


Figura A.3: Load Model (1)

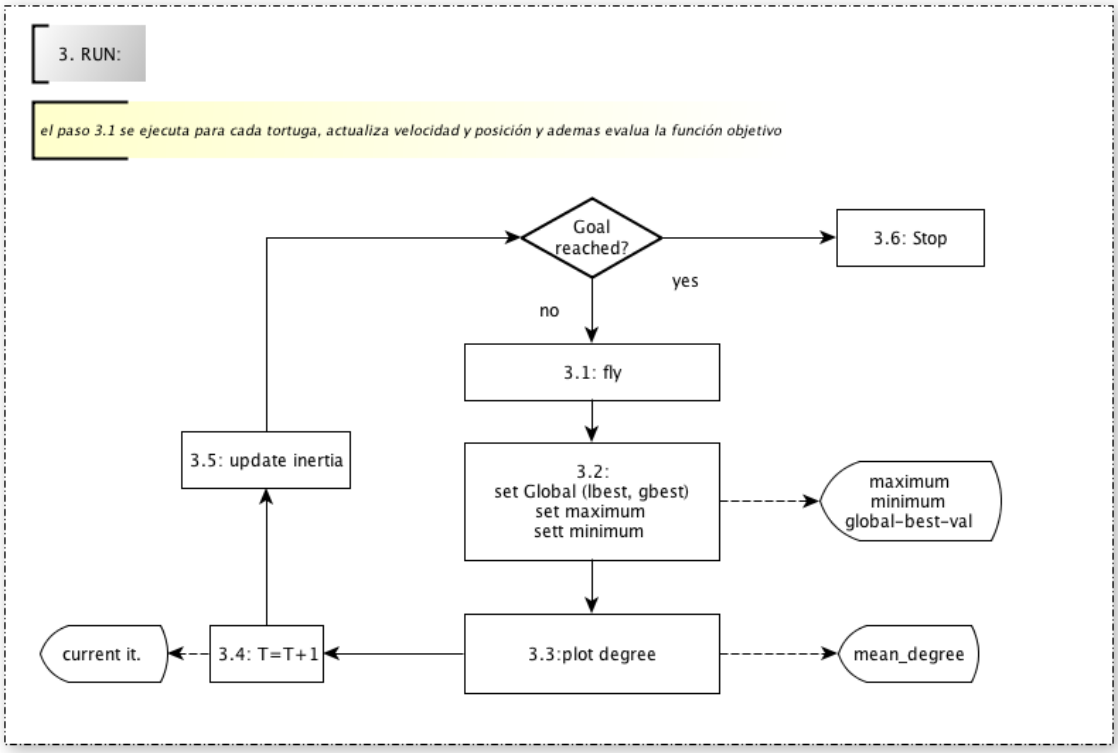


Figura A.5: Run (3)

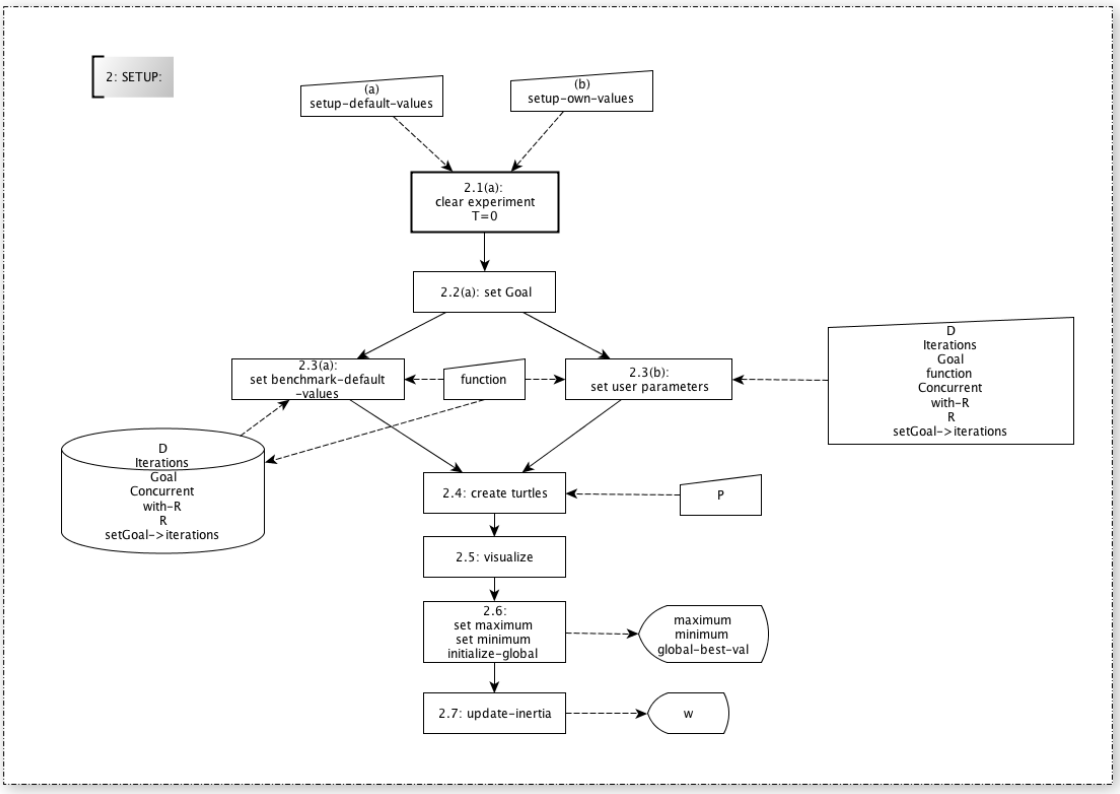


Figura A.4: Setup (2)

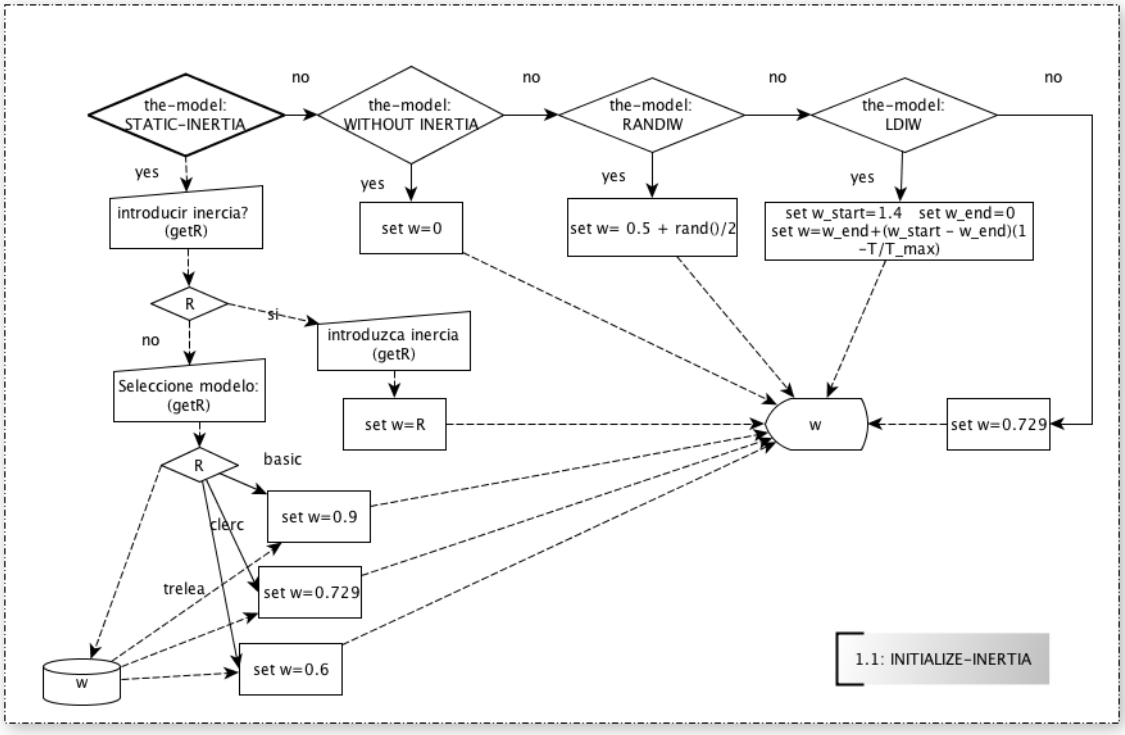


Figura A.6: Initialize Inertia (1.1)

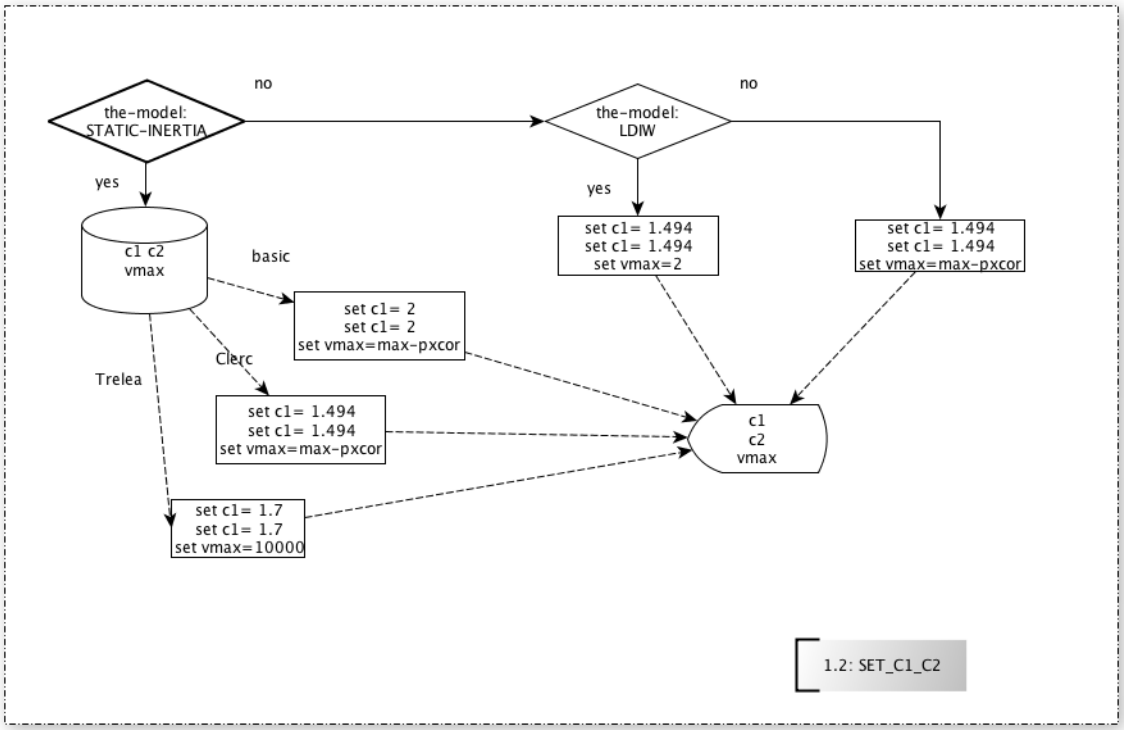


Figura A.7: Set_c1_c2(1.2)

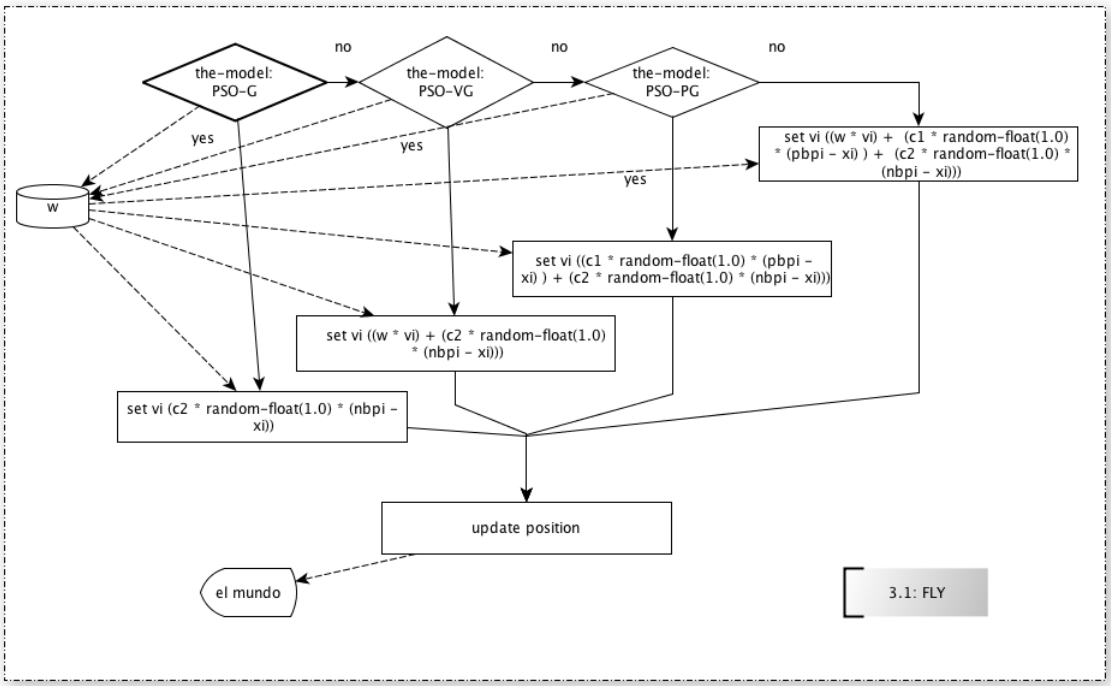


Figura A.8: Fly (3.1)

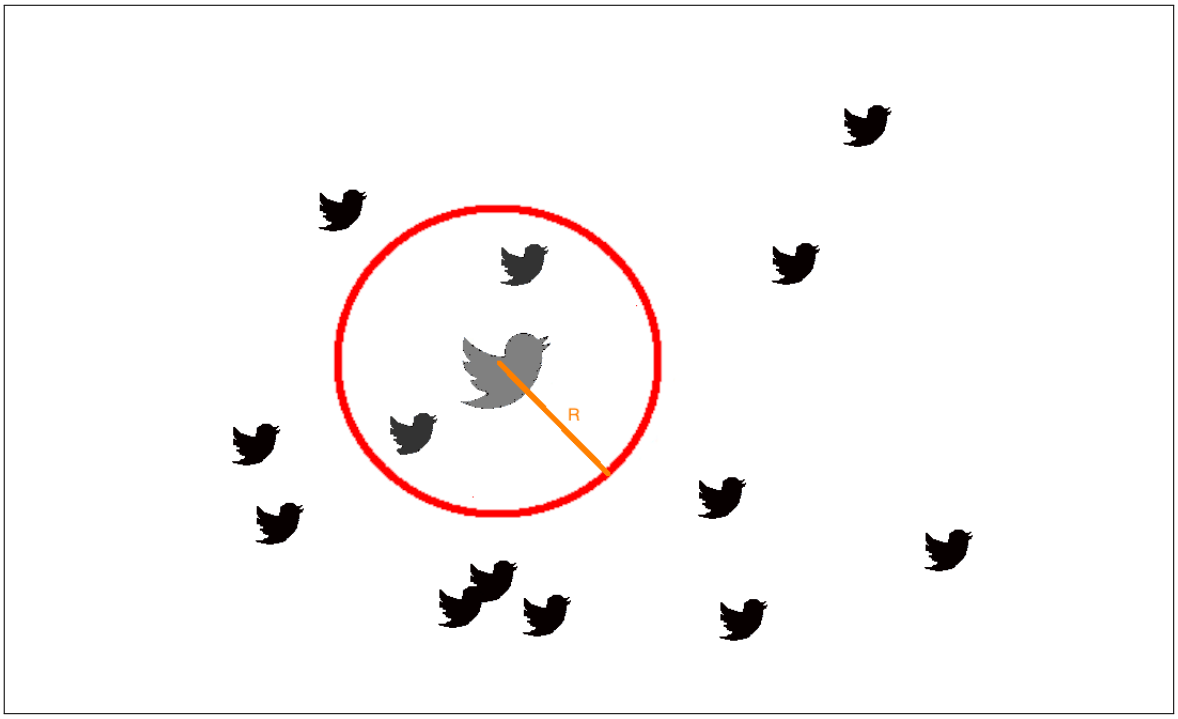


Figura A.9: Vecinos with-R 'On'

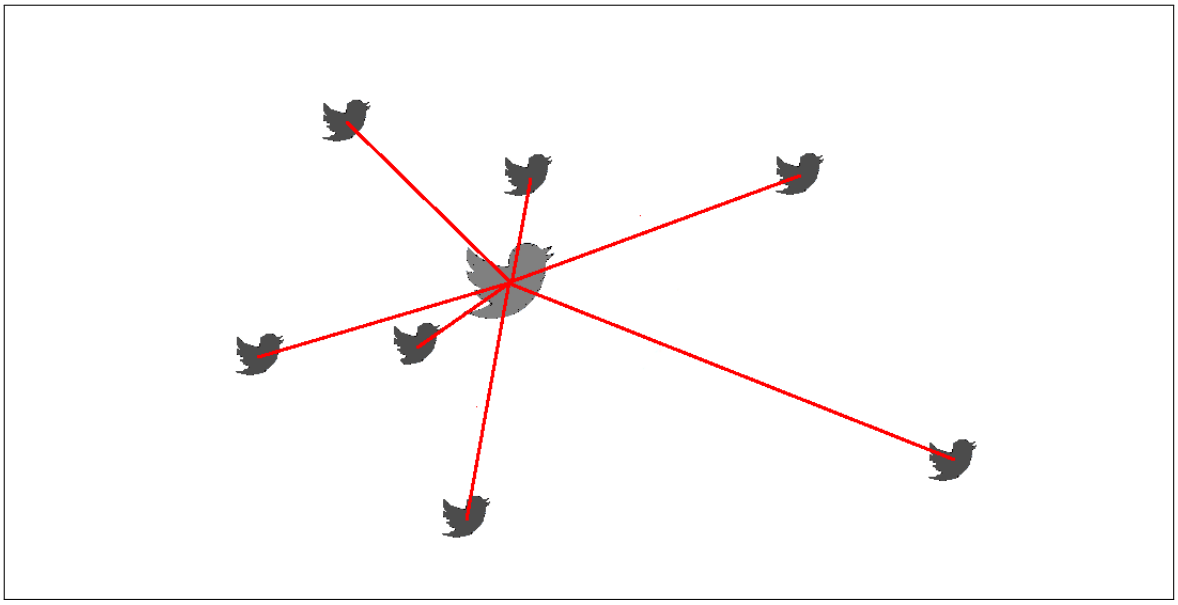


Figura A.10: Vecinos with-R 'Off' (full connected topology)

A.2. Análisis y diseño de la ontología en Protégé

En las figuras que se muestran a continuación se puede ver el diseño de la ontología que hemos realizado con Protégé. Las 4 primeras figuras se corresponden con la jerarquía de clases y las relaciones que existen entre ellas. Las tres figuras siguientes muestras los slots que contiene cada una de las clases principales y, por ultimo, se muestran algunos ejemplos de la vista que proporciona la ontología en html.

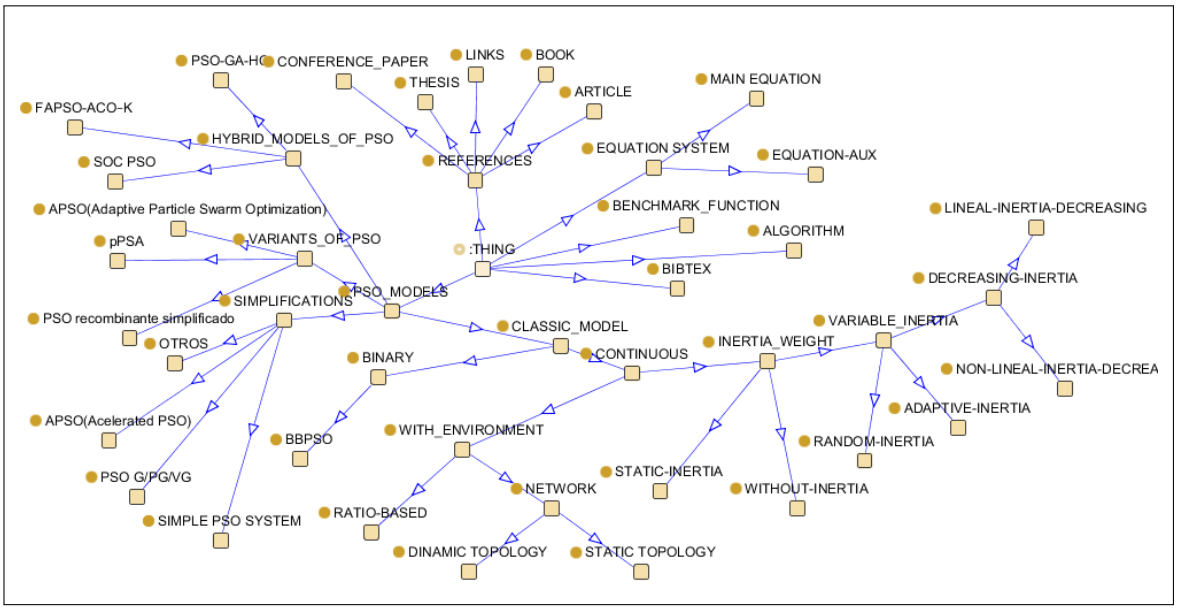


Figura A.11: Jerarquía de clases

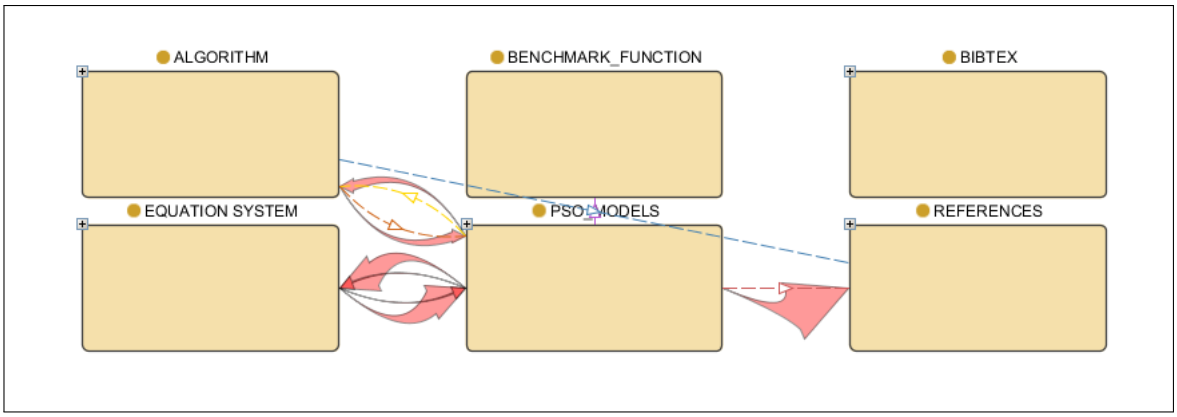


Figura A.12: Relaciones entre clases

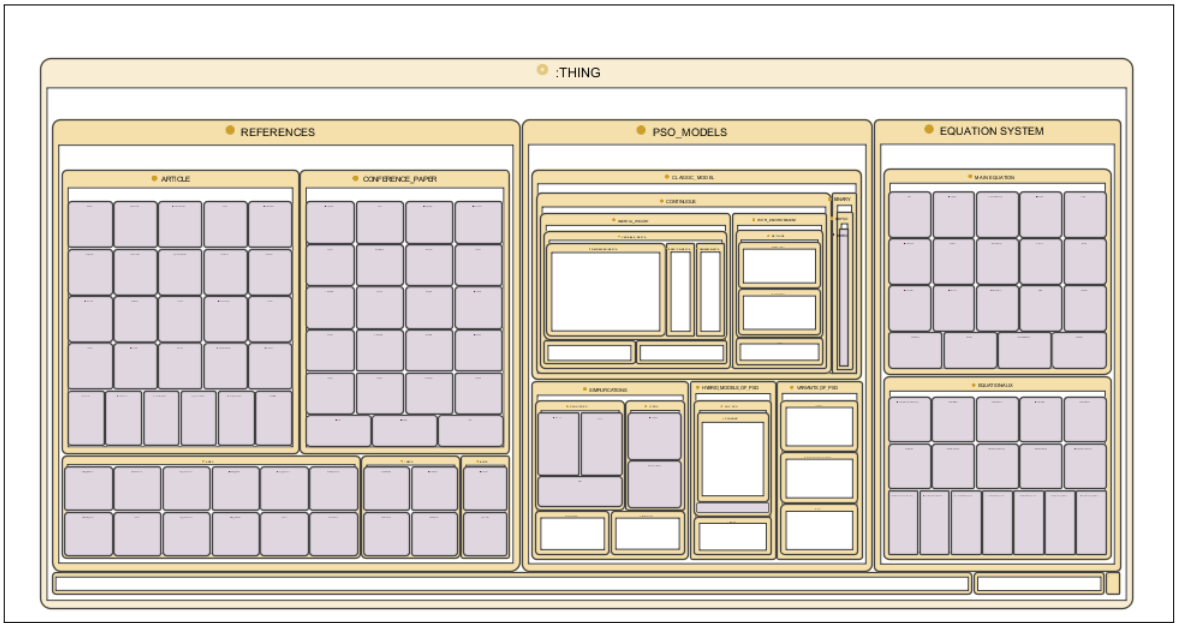


Figura A.13: Vista anidada clases e instancias

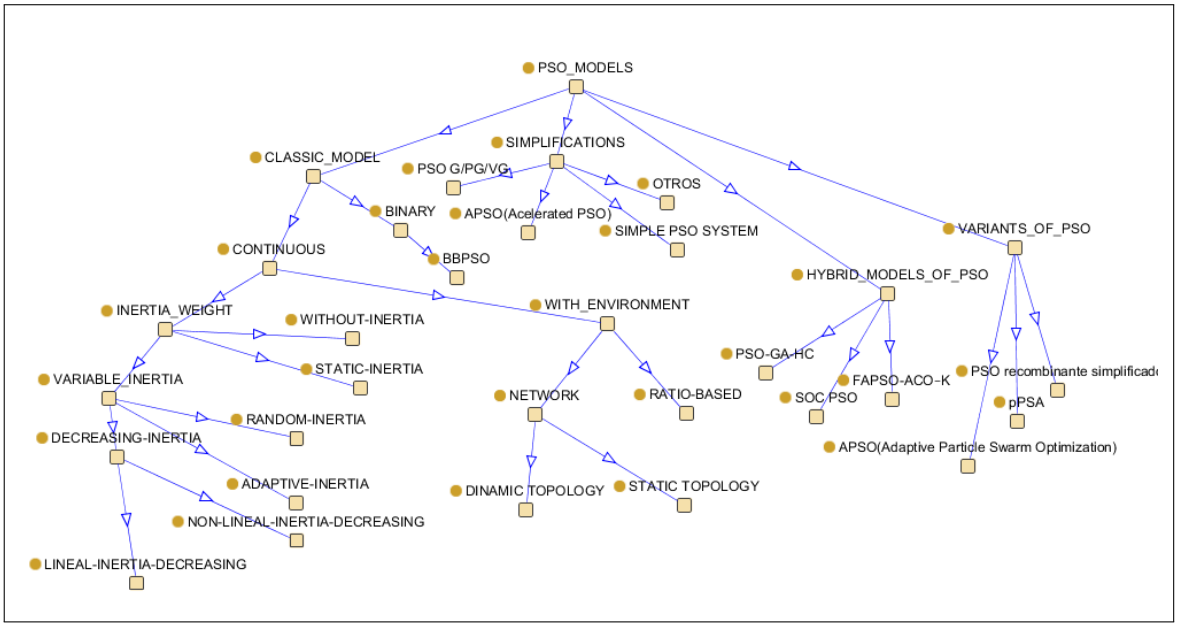


Figura A.14: Modelos de PSO

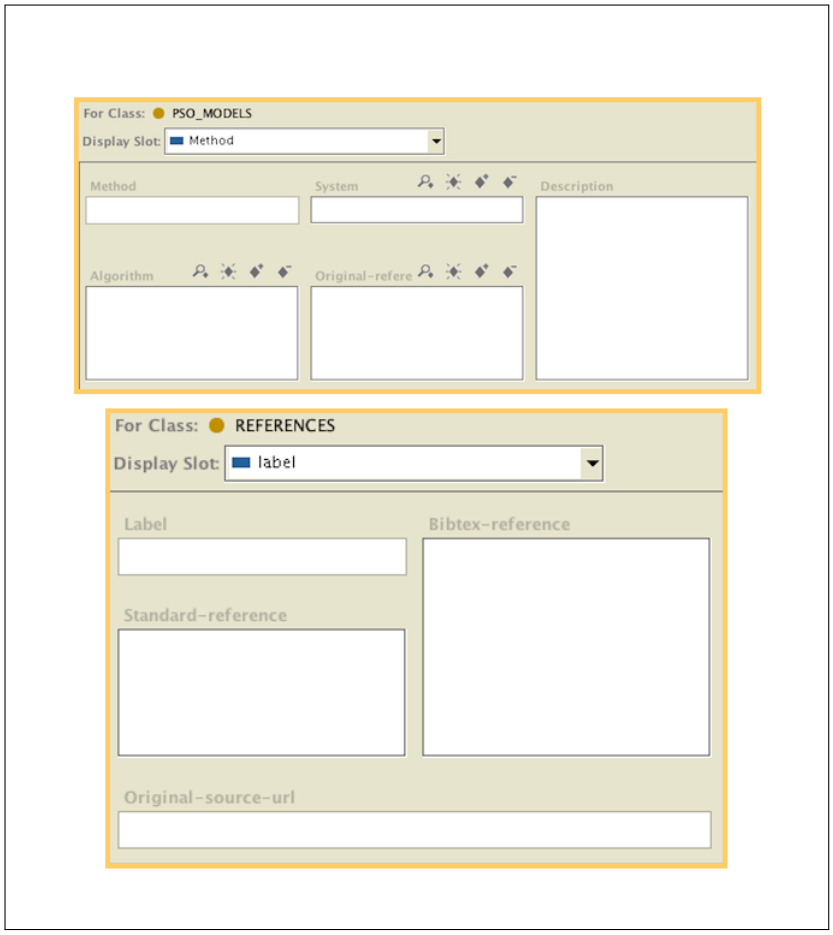


Figura A.15: Classes: PSO_MODELS and REFERENCES

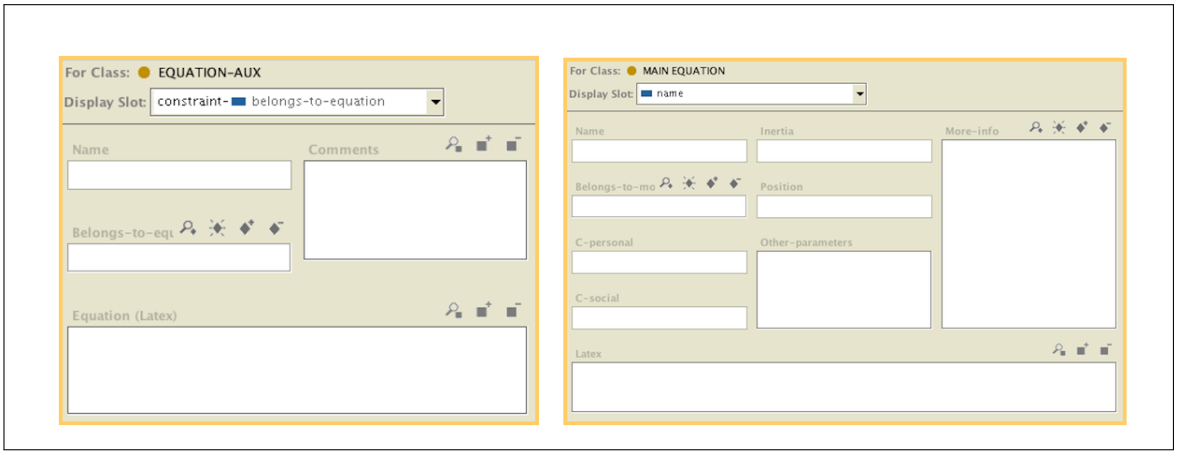


Figura A.16: Classes: EQUATION-AUX and MAIN EQUATION

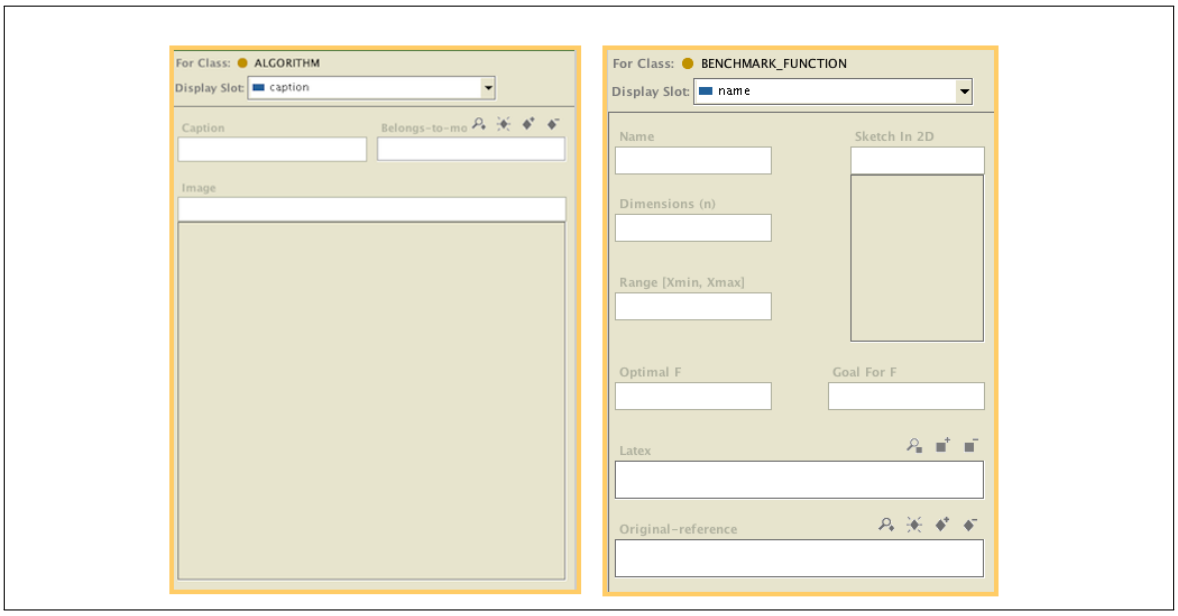


Figura A.17: Classes: ALGORITHM and BENCHMARK_FUNCTION

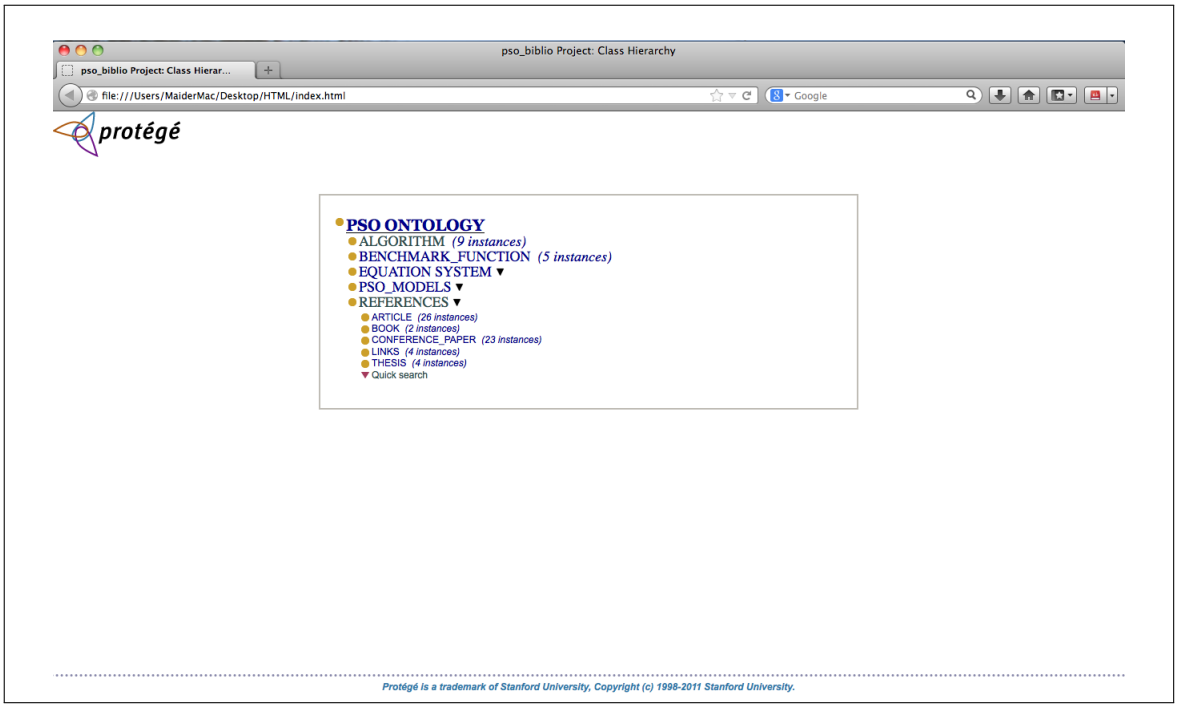


Figura A.18: Vista html: Index

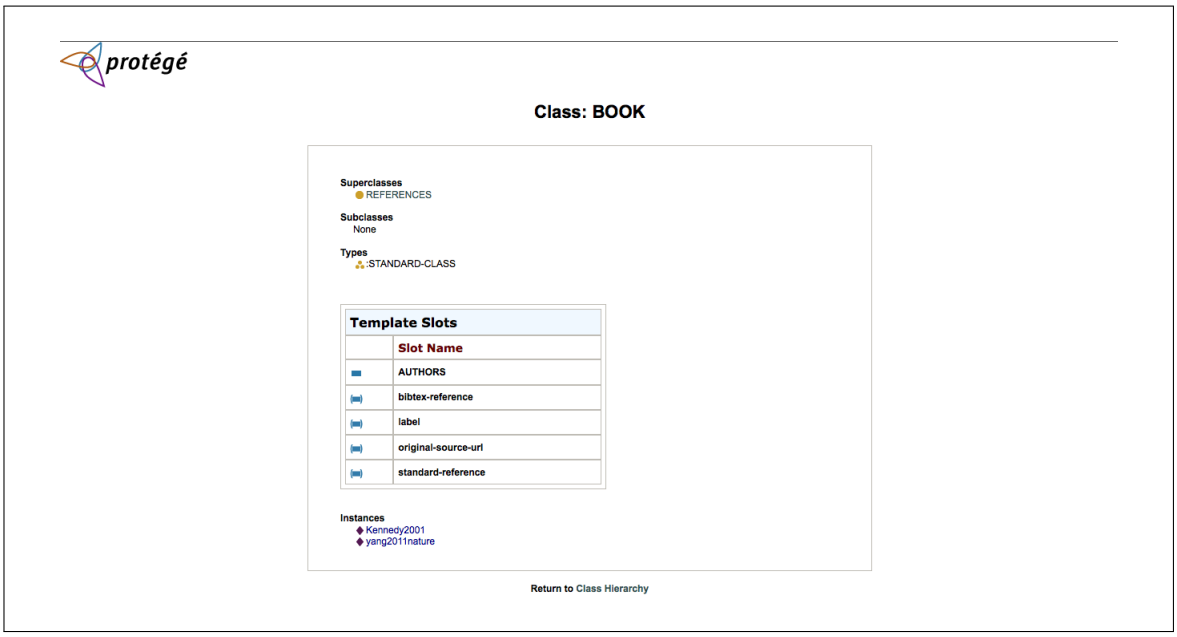


Figura A.19: Vista html: Class Book

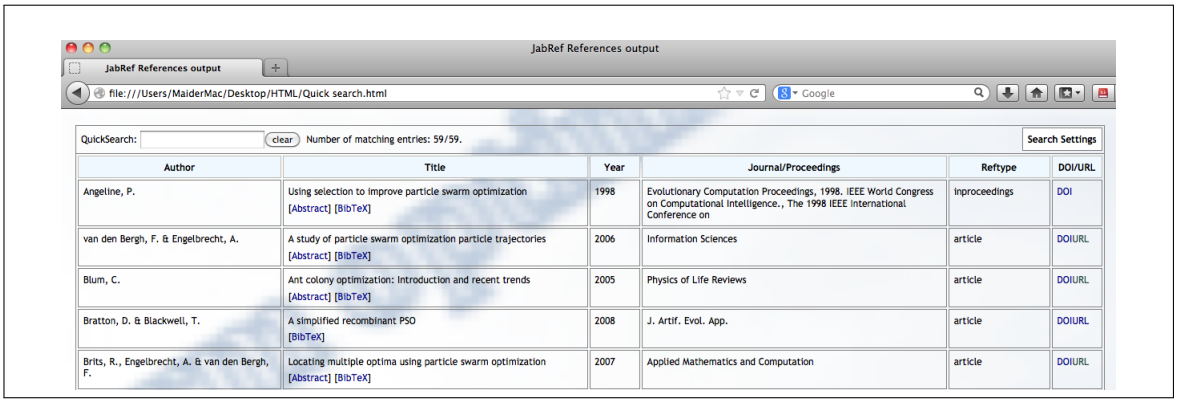


Figura A.20: Vista html: Quick Search

Anexo B

Planes de Gestión del Proyecto

B.1. Plan de Gestión del Alcance

Se plantean 5 procesos básicos para la gestión del alcance:
1. Recopilar requisitos
2. Definición del alcance
3. Creación de EDT
4. Verificación del alcance
5. Control del alcance.

B.1.1. Recopilación de requisitos

Se definen los siguientes requisitos para el proyecto:

- Review sobre la rama de Swarm Intelligence centrado en la PSO. (Bibliografía de unas 30-40 publicaciones)
- Facilitar una ontología que permita estudiar los distintos modelos de PSO.
- Implementar algunos de los modelos más significativos de la PSO en Netlogo.
- Realizar experimentos desde la aplicación implementada y comparar los resultados con los de los artículos.

B.1.2. Definición del alcance

El Alcance del Proyecto está definido en el Capítulo 7 de la Memoria del Proyecto. Las tareas a realizar en este proyecto se pueden desglosar en los siguientes apartados:

1. Planificación del Proyecto
   - *a*) Definir el Alcance
   - *b*) Desarrollar el EDT
   - *c*) Planificación Temporal
   - *d*) Cronograma
   - *e*) Plan de Gestión de Recursos
   - *f*) Plan de Gestión de las Comunicaciones
   - *g*) Plan de Gestión de la Calidad
2. Análisis y recopilación
   - *a*) Análisis de requisitos, factibilidad del proyecto
   - *b*) Recopilación de material bibliográfico (Scopus y Science Direct)
   - *c*) Generar base de datos Bibliográfica (.bib)
3. Investigación y formación.

   a) Estudio del estado del arte: Review sobre la rama de Swam Intelligence centrado en la PSO
   b) Estudio del software existente
   c) Aprendizaje del software y lenguaje de programación que se utilizará (NetLogo)
   d) Estudio de las extensiones adicionales necesarias para el problema (Extensión goo)
   e) Aprendizaje de nociones básicas de Protégé
   f) Aprendizaje de nociones básicas de LaTex
   g) Aprendizaje de html y css

4. Diseño:

   a) Identificar y definir los modelos de PSO que se van a codificar
   b) Determinar la presentación de los modelos (selección de modelos, selección de parámetros) dentro de la interfaz gráfica
   c) Diseño de casos de uso
   d) Diagramas de Flujo
   e) Diseñar una ontología (Protégé & html) que permita estudiar los distintos modelos de PSO y proporcionar algunos recursos de escritura como:
      1) las ecuaciones de los modelos en Latex
      2) las referencias bibliográficas en BibTex

5. Desarrollo

   a) Implementar el código perteneciente a los modelos seleccionados
   b) Fusionar los modelos dentro de una misma interfaz donde el usuario pueda escoger con qué modelo y con qué parámetros trabajar
   c) Crear Ontología en Protégé
   d) Crear Ontología en html

6. Validación y pruebas

   a) Comprobar que los resultados del programa coinciden con los obtenidos en los experimentos de los artículos originales
   b) Análisis de Resultados (benchmark)

7. Seguimiento y Control

   a) Aseguramiento y Control de la Calidad

b) Seguimiento y Control de los Cambios en el Alcance

c) Seguimiento y Control de las Comunicaciones

d) Registro de Actividades

e) Reuniones

B.1.3. EDT

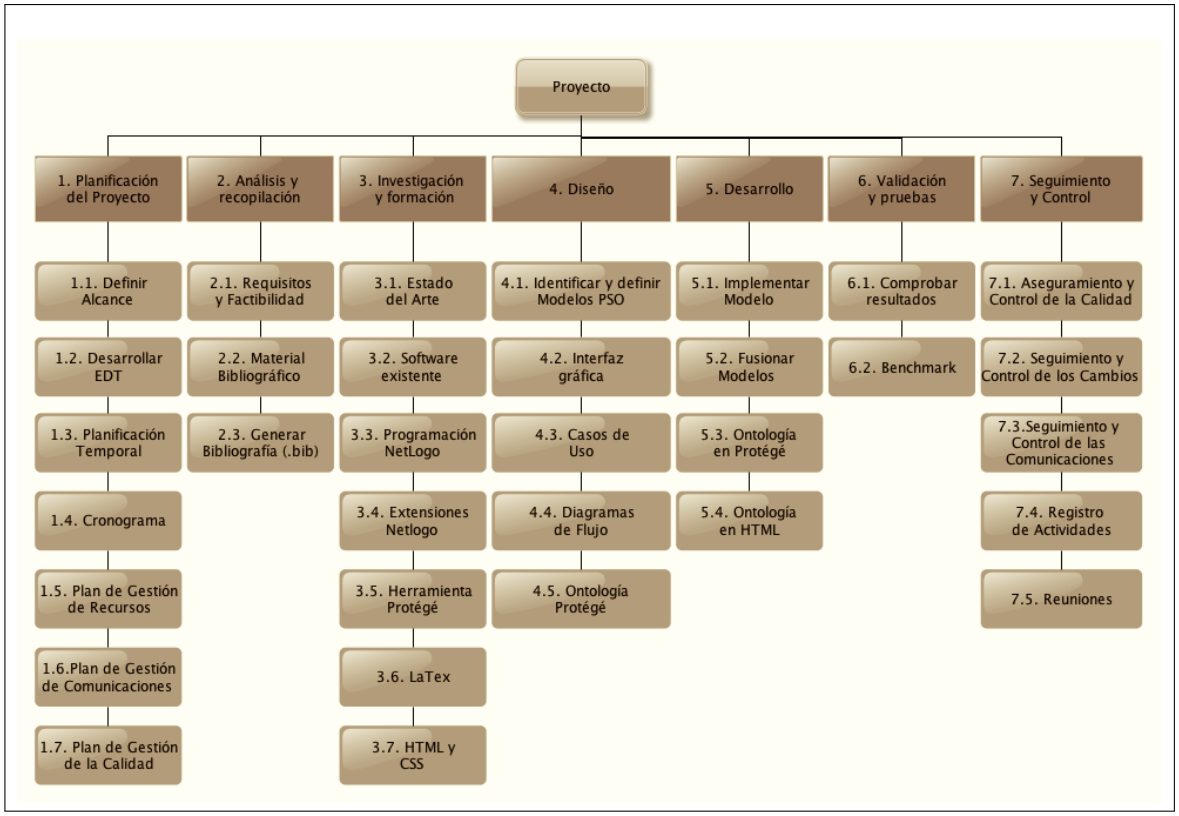


Figura B.1: EDT

B.1.4. Verificación del Alcance

Este proceso consiste en formalizar la aceptación de los entregables del proyecto que se han completado. Por tanto se deberán revisar los entregables por parte de la directora del proyecto para asegurarse de que se han completado satisfactoriamente y para obtener de ellos su aceptación formal.

B.1.5. Control del Alcance

Es el proceso por el que se monitorea el estado del alcance del proyecto y del producto, y se gestionan cambios a la línea base del alcance. Los cambios serán realizados siguiendo las Directrices para la Gestión de Cambios del Capítulo 10 de la Memoria.

B.2. Plan de Gestión de Calidad

A continuación se define el sistema de calidad del proyecto que se va a llevar a cabo para desarrollar la gestión de la calidad a lo largo de todo el proyecto, de forma que logremos los mejores resultados posibles, con el menor coste y cumpliendo los plazos estipulados. Para ello se han especificado los requisitos necesarios para lograr una calidad mínima que cubra las expectativas del director del proyecto.

Partiendo de la calidad mínima a cumplir, se han propuesto una serie de posibles mejoras, las cuales, se realizarán únicamente cuando se haya cumplido el nivel mínimo, nos encontremos dentro de los costes estimados y haya tiempo disponible para realizarlas.

A continuación se muestra la lista de control con la que se realizará el pertinente control de calidad para comprobar que se cumplen adecuadamente los niveles de calidad exigidos o, en su caso, las mejoras propuestas:

Actividad	Tarea	Tipo	Completado
Investigación y formación	Review sobre la rama de Swam Intelligence centrado en la PSO. (entre 30-40 artículos)	Requisito inicial	
Diseño	Facilitar una ontologíáa (Protege) que permita estudiar los distintos modelos de PSO	Requisito en la ampliación del alcance	
	Proporcionar las ecuaciones de los modelos en Latex	Valor añadido	
	Proporcionar las referencias bibliográficas para facilitar la escritura de artículos	Valor añadido	
Implementación	Implementar el código de programación de los modelos mas significativos de la PSO (NetLogo)	Requisito inicial	
	Ofrecer los modelos dentro de una misma interfaz gráfica	Valor añadido	
	Ofrecer un conjunto de ejercicios o experimentos a realizar por el usuario.	Valor añadido	
Validación y Pruebas	Realizar algunos experimentos incluidos en los artículos bibliográficos	Requisito inicial	
	Análisis de los resultados. Benchmarck de parámetros	Valor añadido	

T:totalmente completado; P:parcialmente completado; N:no completado

Figura B.2: Gestión de la calidad

B.3. Plan de Gestión de los Recursos

B.3.1. Humanos

Existen tres actores principales en la realización de este proyecto:

- Blanca Rosa Cases Gutierrez, profesora del Departamento de Lenguajes y Sistemas Informáticos de la Universidad del País Vasco/Euskal Herriko Unibertsitatea, en calidad de directora del proyecto.
- Abdelmalik Moujahid, profesor del departamento de Ciencia de la Computación e Inteligencia Artificial de la Universidad del País Vasco/Euskal Herriko Unibertsitatea, en calidad de codirector del proyecto.

- Maider Gaztelu Aza, estudiante de Ingeniería Informática en la Universidad del País Vasco/Euskal Herriko Unibertsitatea, autora del proyecto.

B.3.2. *Hardware*

- Ordenador personal:
  - Modelo: MacBook Pro 8,2
  - Procesador: 2.2 GHz Intel Core i7
  - Memoria: 4 GB 1333 MHz DDR3

B.3.3. *Software*

- Sistema operativo: Mac OS X v10.6.8.
- Ofimática:
  - LyX v2.0.1
  - Microsoft Excel para Mac 14.1.0
- Diseño:
  - yEd v3.10.2
  - Eclipse Modeling Tools versión Juno Service Release 2
- Navegador web:
  - Safari v5.1.1
  - Firefox v21.0
  - Google Chrome v27.0.1453.110
- Base de datos Bibliográfica: JabRef v2.3.1
- Procesado de Imágenes: Gimp v2.8.2
- Aplicación: NetLogo v5.0
- Ontología: Protégé v3.4.8 for MacOSX
- Editor HTML: Sublime Text 2.0.1

B.4. Plan de Gestión de las Comunicaciones

Las comunicaciones alumna-directora del proyecto se realizaran principalmente vía correo electrónico de la Universidad, siendo posible emplear el teléfono móvil para ocasiones puntuales de prioridad alta y, por tanto, urgentes.

Además, se realizarán reuniones periódicas para el correcto seguimiento del proyecto. La fecha y hora de cada reunión será acordada por los participantes de la misma con la mayor antelación posible por correo electrónico.

Las consultas y dudas se podrán resolver con la directora del proyecto bien vía correo electrónico o bien de manera presencial en una fecha y hora acordada previamente por correo electrónico.

B.5. Plan de Gestión de los Riesgos

Las estrategias para los riesgos negativos serán Evitar, Transferir, Mitigar o Aceptar; mientras que las estrategias para los riesgos positivos serán Explotar, Compartir, Mejorar y Aceptar. Los riesgos han sido descritos en Capítulo 9, Análisis de Riesgos de la Memoria del Proyecto.

Nº	**Riesgo**	**Medidas preventivas**	**Medidas Paliativas**
1	Baja por enfermedad(a corto o lago plazo).	-	Replanificar. Posible reorganización de los recursos humanos.
2	Problemas con las herramientas de comunicación.	Acuses de recibo. Disponer de medios de comunicación alternativos.	Utilizar medios de comunicación alternativos determinados en el plan de comunicaciones.
3	Problemas con el almacenamiento de la documenta-ción(pérdida de documentos).	Buena gestión de ficheros. Copias de seguridad en varios dispositivos y en la nube.	Medidas de restauración de datos.
4	Cambios en el alcance frecuentes y/o desmesurados.	Acordar un alcance realista y ceñirse al mismo.	Estudiar la viabilidad y replanificar.
5	Reestructuración de la organización/ replanificación.	Realizar una planificación adecuada y realista y ajustarse a ella en la medida de lo posible.	Replanificar.
6	Aparición de tareas imprevistas no planificadas.	Realizar una planificación adecuada y exhaustiva que determine todas las tareas necesarias para el éxito del proyecto.	Identificar y analizar las nuevas tareas, su prioridad y sus dependencias. Replanificar.
7	Comunicación lenta o ausencia de feedback.	Compromiso de feedback fluido. Aplicar el plan de comunicaciones.	Utilizar medios de comunicación alternativos.

Cuadro B.2: Estrategias para los riesgos

Anexo C

Swarm Intelligence Review

C.1. Concepto

Con el nombre de Swarm Intelligence (SI) nos referimos a los modelos del comportamiento colectivo representados en términos de sistemas descentralizados y autoorganizados ya sean naturales o artificiales. Es un concepto que se emplea principalmente en el ámbito de la Inteligencia Artificial (IA). Este término lo presentaron Gerardo Benii, profesor de la Universidad de California, Riverside y Jing Wang en 1989 [16].

La aplicación de los principios del "swarm" en robots se llama 'swarm robotics', mientras que 'Swarm Intelligence' se refiere al conjunto más general de los algoritmos. 'Swarm prediction' se ha utilizado en el contexto de los problemas de previsión.

Un Sistema SI se compone típicamente de una población de agentes simples o boids [1] que interactúan entre sí y con su medio ambiente. Por tanto, la inspiración de estos sistemas proviene de la naturaleza y son esencialmente sistemas biológicos o biológicamente inspirados.

Los agentes siguen unas reglas muy básicas y las interacciones entre ellos generan cierto comportamiento global "inteligente" desconocido para cada individuo de manera aislada. En consecuencia, dado un espacio de posibilidades, la población de un enjambre (swarm) puede encontrar soluciones variadas, patrones que resuelven problemas a través de un modelo de interación social simplificado.

C.2. Líneas

Como ejemplos naturales de SI podríamos mencionar las colonias de hormigas, bandadas de pájaros, el pastoreo de animales, el crecimiento bacteriano y los bancos de peces, aunque a lo largo de los años se han desarrollado muchos algoritmos más. Sin embargo, queremos resaltar la relevancia de dos algoritmos en particular: Ant Colony Optimization(ACO) y Particle Swarm Optimization(PSO). Este estudio se centrará en el segundo algoritmo, partiendo de los artículos de los padres de la PSO, Kennedy y Eberhart y mostrando las variaciones que diversos científicos han realizado sobre el algoritmo principal e implementando y analizando algunas de estas variantes.

C.2.1. Ant Colony Optimization (ACO):

Es una clase de algoritmos de optimización inspirados en las acciones de una colonia de hormigas. Estos algoritmos son realmente útiles a la hora de encontrar un camino de longitud mínima hacia un objetivo o meta. Las hormigas artificiales simuladas mediante agentes son capaces de localizar soluciones óptimas moviéndose por el espacio de búsqueda. Su comportamiento biológico es sencillo, las hormigas depositan unas feromonas en el suelo marcando un camino favorable que podrán seguir otros miembros de la colonia.

[1] boids: programa de IA que simula el comportamiento de los pájaros que, en bandadas, exploran su entorno en busca de comida. Este comportamiento se denomina "flocking".

En sus artículos Christian Blum, 2005[17] y Dorigo, 2006[26, 24] nos muestran las características de este modelo. Estos estudios se basan en la aportación de Goss et al. [31], que realizó algunos experimentos usando con colonias de hormigas reales. Cabe destacar dos experimentos:

- El primero se trataba de un hormiguero conectado a la fuente de alimento por dos caminos de igual longitud. Resultado: Después de algún tiempo la colonia

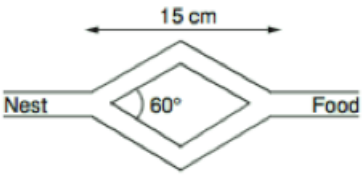


Figura C.1: Caminos de igual longitud

  entera converge hacia el uso del mismo camino. Repitiendo el experimento se comprueba que cada uno de los caminos se utiliza en aproximadamente el 50 % de los casos.

- El segundo se trata de un hormiguero conectado a la fuente de alimento por dos caminos de distinta longitud. Resultado: Las hormigas que eligen por casualidad

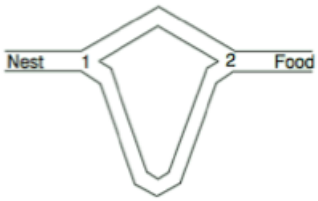


Figura C.2: Caminos de distintas longitudes

  el camino corto son las primeras en llegar al nido. Por tanto, el camino corto recibe antes la feromona y, en consecuencia, aumenta la probabilidad de que las demás hormigas seleccionen ese camino.

Se usa este comportamiento de las hormigas para resolver el problema del viajante (el clásico "Travelling Salesman Problem" o TSP), encontrando un circuito Hamiltoniano en un grafo, es decir, a partir de un nodo que representa a una ciudad, encontrar la ruta más corta que pasa por todas las ciudades y vuelve a la primera sin repetir ninguna de las intermedias. Fue además el primer problema que resolvió el primer algoritmo de optimización ACO (1991) [27, 25, 26]

El modelo que surge inspirado por este comportamiento es el siguiente:

- Toma de decisiones: elección del siguiente paso (nodo) de una hormiga tal como se muestra en la Figura 1.3.

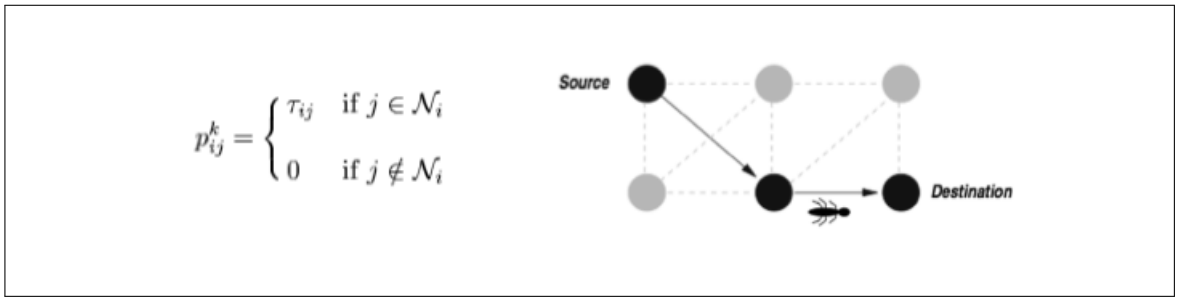


Figura C.3: Elección del siguiente nodo de una hormiga

La regla de decisión de una hormiga k situada en el nodo i usa la cantidad de feromona del camino τ_{ij} para calcular la probabilidad con la que elegiría el nodo $j \in N_i$ como siguiente nodo al que desplazarse, donde N_i es el conjunto de vecindarios que están a un paso.

- Incremento de feromona en un camino (Simple Ant Colony Optimization): Si consideramos una hormiga que en el momento **t** se desplaza del nodo **i** al nodo **j**, el valor de τ_{ij}(t) cambiaría de la siguiente manera, $\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau$. Las hormigas depositan la feromona en los arcos que ellas usan. En S-ACO las hormigas depositan una cantidad constante de feromona $\Delta\tau$. Esto incrementa la probabilidad con que ese arco será reutilizado por otra hormiga.

- Evaporación de la feromona en un camino: $\tau \leftarrow (1 - \rho)\,\tau$. Decrece la cantidad de feromona de manera exponencial en cada iteración del algoritmo. El parámetro $\rho \in (0, 1]$ regula la evaporación de la feromona. La evaporación favorece la exploración de diferentes arcos durante todo el proceso de búsqueda.

- Características generales de la colonia de hormigas:

  - Cada hormiga es suficientemente compleja como para encontrar una solución al problema, pero una solución de calidad solo puede obtenerse como resultado de la interacción colectiva.
  - Cada hormiga solo tiene información local del nodo que está visitando.
  - La comunicación entre hormigas es indirecta, mediante las lecturas/escrituras sobre las variables de cantidad de feromona.
  - Las hormigas no son adaptables.

C.2.2. Particle Swarm Optimization (PSO):

Particle Swarm Optimization (PSO) es un método de optimización estocástica basado en poblaciones y tiene sus raíces en dos metodologías principales. Por un lado, podemos percibir sus lazos con la Inteligencia Artificial en general, y con las teorías sobre bandadas de pájaros, bancos de peces (fish schooling) y enjambres en

particular, que desarrolla la Vida Artificial[2] y de donde procede la SI. Por otro lado, también está ligado a la computación evolutiva en la literatura.

En los sistemas biológicos se llama sistemas evolutivos a aquellos que modelan la reproducción genética y la selección natural: se trata de la evolución a largo plazo o filogenética. En el caso de la SI se trata de una evolución ontogenética, durante la vida de una población, no de evolución en el sentido darwiniano. Por eso, sería mejor decir que la PSO es un modelo dinámico de poblaciones.

Se trata de un método computacional que optimiza un problema tratando, de manera iterativa, de mejorar una solución candidata con respecto a una determinada medida de calidad. PSO optimiza un problema teniendo una población de soluciones candidatas, representadas mediante partículas, y moviendo estas partículas a través del espacio de búsqueda de acuerdo a unas sencillas fórmulas matemáticas sobre la posición de la partícula y la velocidad.

El algoritmo PSO que James Kennedy y Rusell Eberhart descubrieron durante sus simulaciones sobre modelos sociales simplificados en el año 1995 [34] constituye un modelo muy simple, fácilmente programable. Tampoco necesita gran soporte tecnológico, ya que únicamente requiere operaciones matemáticas primitivas que no aportan un coste computacional elevado en términos de memoria y velocidad.

Durante años, numerosos científicos han tratado de simular comportamientos animales, como es el caso de Reynolds en 1987 [8, 57] y Heppner and Grenander en 1990 [33] que presentaron simulaciones sobre bandadas de pájaros mucho antes de que la PSO tomara forma.

Sus parámetros de control influyen mucho en su rendimiento. Varios estudios teóricos y simulaciones han demostrado la influencia de valores como el peso inercial y los coeficientes de aceleración a la hora de obtener un comportamiento convergente. Una mala elección de estos parámetros podría dar lugar a un comportamiento cíclico o divergente.

C.2.2.1. PSO original

Como ya hemos dicho, la PSO es un enfoque de optimización estocástica que mantiene un enjambre de soluciones candidatas, conocido como *partículas* [34]. Kennedy y Eberhart las nombraron de esa manera debido a su velocidad y aceleración. Cada partícula representa un individuo, individuos que interactúan entre sí mientras aprenden de su propia experiencia, de manera que poco a poco los miembros de la población se mueven en mejores regiones del espacio del problema. Los individuos de un enjambre de partículas pueden ser conceptualizados como células de un autómata celular, cuyos estados cambian de muchas dimensiones simultáneamente. En la PSO original, la posición X_i de cada i partícula se ajusta de la siguiente manera:

$$v_{ij}(t+1) = v_{ij}(t) + \phi_{1j}(t)(y_{ij}(t) - x_{ij}(t)) + \phi_2(\hat{y}_{ij}(t) - x_{ij}(t)) \tag{C.1}$$

[2]rama interdisciplinar en la intersección entre la biología y la informática que pretende establecer modelos formales y simulaciones de los sistemas vivos.

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{C.2}$$

para *i=1, ..., s* y *j=1, ..., n*; donde
$\phi_{1j}(t) = c_1 r_{1j}(t)$ *y* $\phi_{2j}(t) = c_2 r_{2j}(t)$,
s es el número total de partículas que hay en el enjambre,
n es la dimensión del problema, es decir, el número de parámetros de la función que se están optimizando,
c_1 y c_2 son los coeficientes de aceleración,
$r_{1j}(t)$, $r_{2j}(t) \sim U(0,1)$,
$x_i(t)$ es la posición de la partícula i en el instante de tiempo t,
$v_i(t)$ es la velocidad de la partícula i en el instante de tiempo t,
$y_i(t)$ es la mejor solución personal de la partícula i en el instante de tiempo t,
$\hat{y}_i(t)$ es la mejor solución encontrada por el vecindario de la partícula i en el instante de tiempo t.

En la *ecuación (2.1)* la velocidad de una partícula está determinada por tres factores:

- $v_i(t)$, que sirve como término de Momento para prevenir oscilaciones excesivas en la dirección de búsqueda.
- $\phi_1(t)(y_i(t) - x_i(t))$, que se refiere a la *componente cognitiva.* Esta componente representa la distancia a la que se encuentra una partícula de la mejor solución, $y_i(t)$, encontrada por ella misma. La componente cognitiva representa la tendencia natural de los individuos de volver al entorno donde experimentaron su mejor rendimiento.
- $\phi_2(t)(\hat{y}_i(t) - x_i(t))$, que se refiere a la *componente social.* Esta componente representa la distancia a la que se encuentra una partícula de la mejor solución, $\hat{y}_i(t)$, encontrada por su vecindario. La componente social representa la tendencia de los individuos de seguir el éxito de otros individuos.

De este primer modelo, James Kennedy y Rusell Eberhart [34] concluyeron que modelar comportamientos sociales proporciona una buena herramienta para solventar problemas de optimización de las ingenieras. El PSO cumple los 5 principios básicos del Swarm Intelligence:

1-principio de proximidad: Cálculos n-dimensionales llevados a cabo en una serie de pasos de tiempo.

2-principio de calidad: la población responde a los factores de calidad pbest y gbest.

3-principio de diversidad en la respuesta: la asignación de las respuestas entre el pbest y el gbest asegura una diversidad de respuesta.

4-principio de estabilidad: la población solo cambia su estado (modo de comportamiento) cuando el gbest cambia.

5-principio de adaptabilidad: la población se adapta, ya que cambia cuando el gbest cambia.

C.3. PSO

C.3.1. Resumen Libro "Swarm Intelligence" [37] de Kennedy y Everhart con Shi

C.3.1.1. El Enjambre de partículas (The particle Swarm, Capítulo 7)

En esta sección se presenta el Particle Swarm en forma binaria (A) y en números reales (B). El algoritmo se introduce en términos de comportamiento social y cognitivos, a pesar de ser utilizado como un método de resolución de problemas en la ingeniería y la informática.

El comportamiento universal de un individuo puede resumirse en tres principios:

- Evaluación: La tendencia de evaluar los estímulos para determinar si son positivos o negativos es una de las características de los individuos vivos. Sin ella, el aprendizaje no sería posible. Es necesario que el individuo tenga la capacidad de determinar si se trata de algo bueno o malo.

- Comparación: Basada en la teoría de la comparación social en la que Festinger (1954) describió algunas de las formas en que la gente utiliza a los demás como un estándar para la medición de sí mismos, y cómo las comparaciones con los demás pueden servir como una especie de motivación para aprender y cambiar. De la misma manera, los individuos del Particle Swarm se comparan con sus vecinos e imitan solo a aquellos que son superiores a ellos mismos.

- Imitación: Es una manera efectiva de aprender a hacer cosas, sin embargo muy pocos animales son capaces de hacerlo de manera real. La verdadera imitación es fundamental para la sociabilidad humana, y es fundamental para la adquisición y mantenimiento de las capacidades mentales.

Los tres principios se pueden combinar, incluso en los seres sociales simplificados en los programas informáticos, permitiéndoles adaptarse a complejos problemas ambientales, resolviendo problemas extremadamente difíciles.

A) Modelo de decisión Binario:

Consideremos un individuo "bare-bone"[3], un ser simple, con una sola cosa en su mente, un conjunto de decisiones que tomar, sí/no o verdadero/falso, decisiones binarias pero muy sutiles, donde es difícil decidir qué decisiones tomar. Para cada decisión, este individuo supersimplificado puede estar en un estado o el otro, ya sea en el estado sí, que se representa con un 1, o el estado no con un 0. Está rodeado por otros si/no individuos, que también están tratando de decidir. Todos quieren tomar las mejores decisiones.

Hay dos importantes tipos de información disponibles para estos seres primitivos. La primera es su propia experiencia, es decir, que han probado las opciones y saben qué estado ha sido el mejor hasta ahora, y cuán bueno era. Pero estos seres sociales

[3]elemento escueto, sencillo, limitado, básico, esencial

tienen una segunda consideración, de hecho, saben cuál de las opciones han encontrado más positivas sus vecinos y cuán positivas eran estas opciones. Si estos seres son algo similar a las personas, saben cómo sus vecinos han logrado sus objetivos gracias a observarlos y hablar con ellos.

La probabilidad de que el individuo elija "sí" para cualquiera de las decisiones es una función sobre cuán exitoso ha resultado el "sí" en el pasado con respecto al "no." Sin embargo se ve afectada por la influencia social. La teoría del impacto social dice que las decisiones de los individuos binarios tenderán a estar de acuerdo con la opinión mantenida por la mayoría de los demás, siendo ponderada por la fuerza y la proximidad de los mismos.

Para este modelo Kennedy y Eberhart se limitan a decir que los individuos tienden a ser influenciados por el mejor éxito de cualquier individuo que esté conectado, es decir, por el miembro de su vecindario sociométrico que haya tenido el mayor éxito hasta el momento. Además de su experiencia pasada y las aportaciones del entorno social, otro factor que afecta la decisión del individuo es su propensión o la posición actual sobre el tema. Si la posición inicial del individuo es extrema es menos probable que intente la otra alternativa. Este modelo se representa matemáticamente de la siguiente manera (Kennedy and Eberhart, 1997):

P(x_{id}(t) = 1) = f (x_{id}(t−1) , v_{id} (t − 1) , ρ_{id} , ρ_{gd})

donde:

- P(x_{id}(t) = 1) es la probabilidad con que el individuo i elija el 1 (la probabilidad de elegir 0 es 1-P) para el bit que se encuentra en la posición d del bitstring (string de bits).
- x_{id}(t) es el estado actual de la posición d del bitstring del individuo i.
- t indica el paso actual y t -1 el paso anterior.
- v_{id} (t − 1) es la medida de la predisposición del individuo, o la probabilidad actual de elegir el 1.
- ρ_{id} es el mejor estado encontrado hasta el momento, por ejemplo, es 1 si el mayor éxito se obtuvo cuando x_{id} era 1.
- ρ_{gd} es el óptimo del vecindario, 1 si el mayor éxito de cualquier miembro del vecindario resultó en el estado 1 y 0 en caso contrario.

El parámetro de predisposición de un individuo v_{id}(t) determinará un umbral de probabilidad. Cuanto mayor sea v_{id}(t) la probabilidad de que el individuo elija 1 será mayor y viceversa. Este umbral debe estar en el rango [0.0, 1.0]. Para lograr esto se aplica la siguiente función sigmoidea:

s(v_{id}) = $\frac{1}{1+exp(-v_{id})}$

Se desea ajustar la disposición del individuo hacia los éxitos del individuo y la comunidad, favoreciendo la mejor posición pero, no tanto que el individuo deje de buscar antes de tiempo. Para ello, usaron el símbolo φ (letra Griega phi) para representar un número aleatorio positivo extraído de una distribución uniforme con un

límite superior predefinido. En la versión binaria, el límite es algo arbitrario, y a menudo se establece de tal manera que los dos φ límites sumen 4.0. La formula de la decisión binaria es la siguiente:

$v_{id}(t) = v_{id}(t-1) + \varphi_1(\rho_{id}-x_{id}(t-1)) + \varphi_2(\rho_{gd}-x_{id}(t-1))$

if $\rho_{id} < \mathrm{s}(\mathrm{v}_{id}(\mathrm{t}))$ then $\mathrm{x}_{id}(\mathrm{t}) = 1$;

else $\mathrm{x}_{id}(\mathrm{t}) = 0$

donde ρ_{id} es un vector de números aleatorios, elaborado a partir de una distribución uniforme entre 0,0 y 1,0. En este estudio se reconoce la acción de las dos fuerzas (la experiencia individual y la influencia social) sin ideas preconcebidas acerca de relativa importancia. De manera que, cualquiera de las dos informaciones será dominante en unos casos y en otros lo será la otra.

Se puede establecer un parámetro constante V_{max} en el inicio de un ensayo para limitar el rango de v_{id} evitando así que el enjambre explore demasiado y salga de la región donde debe encontrar el óptimo. En la práctica se establece con un valor de ± 4.0, de manera que siempre hay al menos una posibilidad de $\mathrm{s}(\mathrm{v}_{max}) \approx 0.0180$.

Como los individuos tienden a gravitar probabilísticamente hacia las decisiones que han resultado exitosas por ellos y sus compañeros, el resultado es la optimización del vector de decisión de cada individuo y la convergencia de la población hacia un patrón de soluciones óptimas. Recalcar que, el rendimiento del algoritmo depende de la clase de problema. Sin duda, este tipo de optimización no es apropiada para cualquier problema que tratemos de abordar.

B) Particle Swarm en un espacio de Números Reales:

En esta vista de individuos como puntos en un espacio, los cambios a lo largo del tiempo se representan con el movimiento de los puntos, ahora realmente partículas. Desde un enfoque psicosocial, estos puntos tienden a moverse hacia los demás, para influir en ellos, son individuos que buscan un acuerdo con sus vecinos.

La posición de una partícula i es representada por el vector con símbolo $\vec{x_i}$. Puede haber cualquier número de partículas con vectores de cualquier dimensión. El cambio de la posición de una partícula se representa como $\vec{v_i}$ (velocidad). La velocidad es un vector de números que son sumados con el fin de mover las partículas en cada paso:

$\vec{x_i}(\mathrm{t}) = \vec{x_i}(\mathrm{t}-1) + \vec{v_i}(\mathrm{t})$

Como en la versión binaria, se define un vecindario para cada individuo, según su posición en el array que representa la topología de la población. Este array se implementa habitualmente como una anillo, donde el último miembro del array tiene como siguiente al primero.

La dirección de movimiento del individuo es una función que combina la posición y velocidad actual, la posición del mejor valor hallado previamente por el individuo y la mejor posición encontrada por cualquier miembro de su vecindario:

$\vec{x_i}(\mathrm{t}) = \mathrm{f}\ (\vec{x_i}(\mathrm{t}-1)\ ,\ \vec{v_i}(\mathrm{t}-1)\ ,\ \vec{p_i},\ \vec{p_g})$

Así como en el modelo binario el cambio de la partícula se definía mediante la probabilidad ahora, este cambio, se define en términos de velocidad y posición en R^n:

$$\begin{cases} \vec{v_i}((t) = \vec{v_i}((t-1) + \varphi_1(\vec{p_i}-\vec{x_i}(t-1)) + \varphi_2(\vec{p_g}-\vec{x_i}(t-1)) \\ \vec{x_i}(t) = \vec{x_i}(t-1) + \vec{v_i}(t) \end{cases}$$

donde, como antes, las variables φ son números aleatorios definidos por un límite superior.

Para que las partículas oscilen dentro de unos límites, sin tendencia a la convergencia o colapso de la nube hacia un punto se define un parámetro Vmax:

$$\begin{aligned} If\ v_{id} > v_{max} \quad then \quad v_{id} = v_{max} \\ ElseIf\ v_{id} < -v_{max} \quad then \quad v_{id} = -v_{max} \end{aligned}$$

En cuanto al número de partículas apropiado, se ha observado que el algoritmo trabaja bien asignando una población de entre 10 a 50 individuos.

C.3.1.2. Variantes y comparaciones (Variations and comparisons, Capítulo 8)

El paradigma del Particle Swarm ha sido analizado y experimentado por diversos investigadores, incluyendo matemáticos, ingenieros, físicos, bioquímicos y psicólogos. Esta sección se centra en la importancia de los parámetros ajustables del Particle Swarm.

Los parámetros más importantes son v_{max} y φ, que se establecen al comienzo del ensayo y permanecen constantes. Como ya hemos mencionado antes, al método más tradicional para evitar que la trayectoria de la partícula se expanda hasta el infinito es definir un parámetro Vmax con las características dadas anteriormente. Para comprender mejor el efecto de este parámetro o métodos de constricción similares se puede reducir la partícula a una sola dimensión y mostrar en un gráfico la trayectoria de la misma (Kennedy 1998). En este tipo de experimentos Clerc y Kennedy[22] comprobaron que las trayectorias convergen a un punto estable:

$\rho = \frac{\varphi_1 \rho_i + \varphi_2 \rho_g}{\varphi_1 + \varphi_2}$

donde ρ representa la media ponderada de las dos mejores marcas.

El parámetro de control $\varphi = \varphi_1 + \varphi_2$ a veces llamado "constante de aceleración" es importante para determinar el tipo de trayectoria. Controla el efecto de la fuerza de $(\rho - x)$ en la velocidad. Cuando φ es muy bajo, el efecto es débil y la trayectoria de la partícula sigue un camino amplio. A medida que φ aumenta, se acorta la longitud de la onda oscilatoria, pero los pasos individuales se alargan hasta que se supera el límite v_{max}. Tras esto v_{max} impone un tamaño de paso fijo. Generalmente se escoge $\varphi_1 = \varphi_2 \sim 2.0$. Para valores $\varphi \geqq 4.0$ la amplitud de la onda se expande hasta el infinito. Ozcan y Mohan(1999) [49, 48] determinaron que las trayectorias de las partículas siguen olas sinusoidales periódicas. Véase Figura C.4

Un enfoque para controlar la búsqueda es la implementación de un peso inercial. Otro método desarrollado por el matemático francés Maurice Clerc implica un sistema con "coeficientes de constricción" aplicados a varios términos de la formula.

Como ya hemos mencionado, el vecindario de un individuo viene determinado por la topología adoptada en cada algoritmo. Las más comunes en este ámbito son la topología en anillo ((a) en la Figura C.5), la topología en estrella ((b) en la Figura C.5) y la *Full-connected topology* ((c) en la Figura C.5).

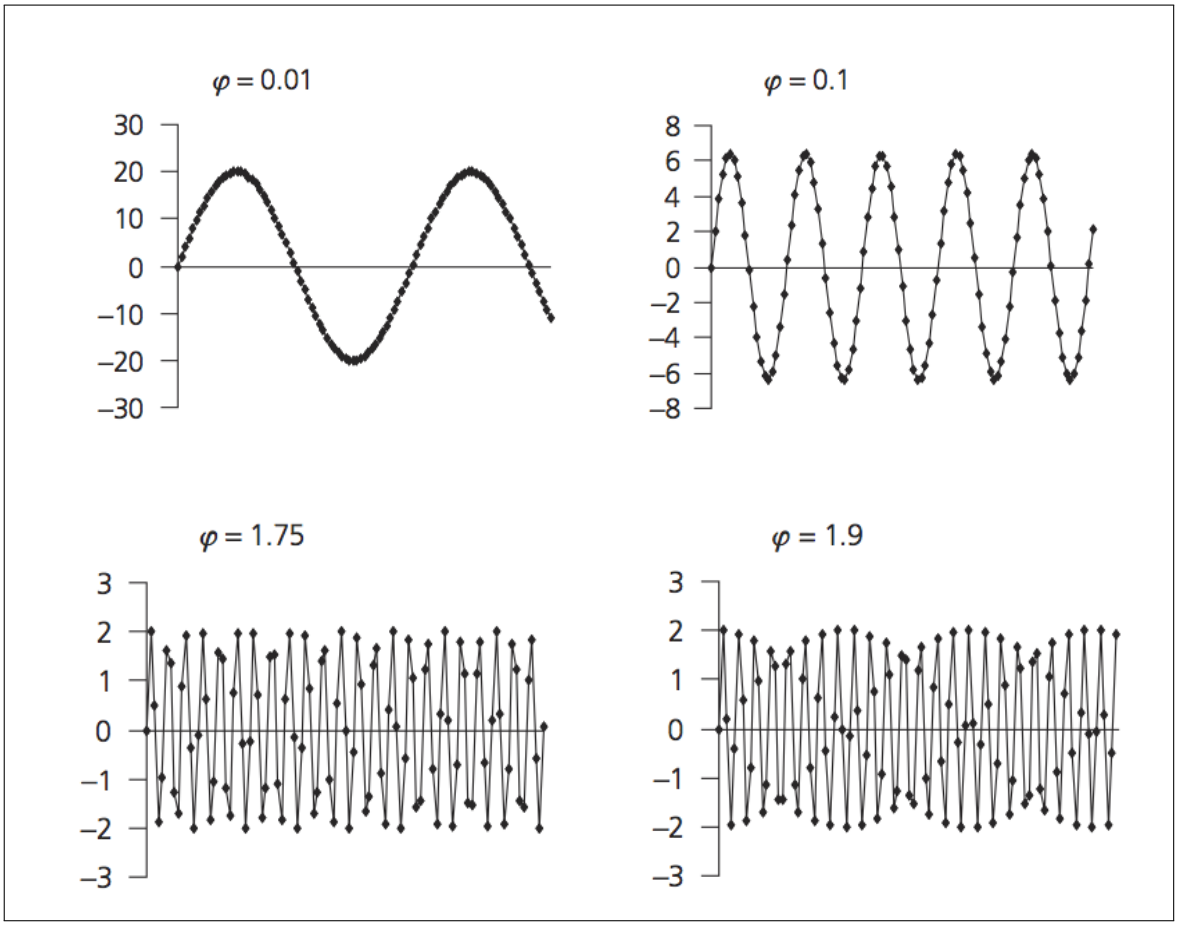


Figura C.4: Resultados de los experimentos de Ozcan Y mohan

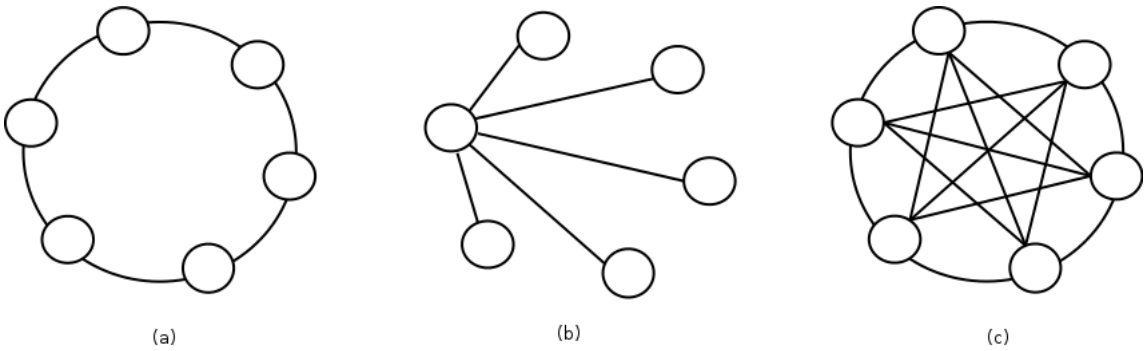


Figura C.5: Topologías

La mejor posición del vecindario $\hat{y}_i$ de una partícula i depende de la topología del vecindario. Si la topología es de tipo anillo, la posición $\hat{y}_i$ se refiere a la *best position* del vecindario de la partícula i. Luego, se trata de la posición *lbest* (local best) y el algoritmo se denomina lbest-PSO. Sin embargo, si la topología es de tipo *full-connected*, la posición $\hat{y}_i$ se refiere a la *best position* de todo el enjambre. En este caso tenemos la posición *gbest* (global best) y el algoritmo resultante se denomina gbest-PSO. En la topología de estrella un nodo central influye en toda la población y se ve afectado por ella, también. En casos menos habituales podemos encontrar la topología Von Neuman, que consiste en organizar las partículas del vecindario en una celosía o topologías de tipo pirámide donde se muestra la relación entre partículas mediante un triángulo tridimensional.

C.3.1.3. Aplicaciones (Aplications, Capítulo 9)

En este capítulo se muestran a modo de ejemplo algunas aplicaciones reales de la PSO.

- Redes neuronales con Particle Swarm: Desde la popularización del algoritmo de retropropagación a mediados de los años 1980 (Rumelhart y McClelland, 1986 [63]; Werbos, 1974 [69]), ha habido un aumento significativo en la investigación y desarrollo en el área de aplicación de las técnicas de la computación evolutiva (Evolutuonary Computation o EC) con el fin de evolucionar algunos aspectos de las redes neuronales artificiales. En este caso, se usa la PSO para evolucionar los pesos de una red y, de forma indirecta, evolucionar la estructura. La metodología tiene el beneficio adicional de hacer innecesario el preprocesamiento de los datos de entrada.

- Aplicaciones en el ámbito de la ingeniería biomédica: Se aplicó con éxito la PSO para desarrollar una red neuronal que clasificaba los temblores humanos (enfermedad de Parkinson o temblor esencial) en comparación con sujetos normales. El método es muy rápido y muy preciso.

- 4 aplicaciones de ingeniería en áreas muy diversas:

  - Optimización del fresado controlado numéricamente por ordenador.
  - Optimización de "mezcla de ingredientes" (fuentes de carbono, fuentes de nitrógeno, sales, minerales, etc.)
  - Control de potencia reactiva y voltaje.
  - Estimación del estado de carga de una batería de un vehículo eléctrico.

C.3.1.4. Implications and especulations (Capítulo 10)

Numerosos fenómenos científicos son inferidos matemática o estadísticamente en lugar de ser observados; lo cual no es razón para negarles la atención de la investigación científica. Es realmente difícil estudiar la mente pues, nuestras propias mentes se

protegen y en ocasiones no desean descubrir hechos que nos lleven a cambiar o hacernos cuestionar nuestra propia existencia. Además, los científicos, como cualquier otra persona, tienen esperanzas, normas y creencias éticas, y podrían no estar dispuestos a descubrir que sus creencias no son válidas.

Las siguientes dos declaraciones resumen los temas centrales del libro:

- La mente es social: la inteligencia humana es el resultado de la interacción social. La cultura y la cognición son consecuencias inseparables de la sociabilidad humana.
- Particle Swarm es una metodología útil para la inteligencia computacional, en concreto para el Soft Computing: la Swarm Intelligence ofrece un paradigma útil para la implementación de sistemas adaptativos. La PSO también es una extensión de los autómatas celulares[4].

Este capítulo se centra en Albert Bandura, que es el autor principal de la visión teorética del aprendizaje humano por medio de la observación:

"Los observadores pueden adquirir habilidades cognitivas y nuevas pautas de comportamiento mediante la observación del desempeño de otros. El aprendizaje puede adoptar formas variadas, incluyendo nuevos patrones de conducta, normas de juicio, las competencias cognitivas y reglas generativas para crear conductas." (Bandura, 1986, p. 49 en [15])

También discurre sobre el significado de las palabras "aprendizaje" y "adaptación" recalcando que, dependiendo de las personas y el ámbito de uso toman significados distintos. Finalmente, profundiza en el concepto conocido como Soft Computing[5].

C.3.1.5. Las conclusiones (In conclusion)

A menudo realizamos simulaciones mentales para probar las consecuencias de las acciones sin peligro, para entender las cosas que no podemos controlar, para ver dónde un argumento acabaría. Los individuos de estas simulaciones son los componentes de un sistema que se nutre de su participación y que a su vez nutre a estos también. El algoritmo de Particle Swarm es una corta secuencia de pasos algebraicos que pueden resolver difíciles problemas matemáticos fácilmente. Aunque surge de los algoritmos evolutivos, se diferencia de ellos en la explicación metafórica y en la forma en que funciona. Es, por tanto, un avance más allá de los métodos ya existentes. La simple característica de los individuos de imitar a sus superiores conduce a la población hacia formas mejores de hacer las cosas.

Otro tema que se ha planteado reiteradamente es el de la integración de la experiencia individual con el aprendizaje social. Leon Festinger presentó la siguiente hipótesis en uno de sus artículos que está incluido en el volumen de documentos de Eddie Harmon-Jones y Mills Judson en 1999 [32].

[4]los autómatas celulares (CA) son sistemas estructurados topológicamente en los cuales las posiciones de los miembros de la topología (células) no varían. En estos sistemas cada célula realiza cálculos muy simples.

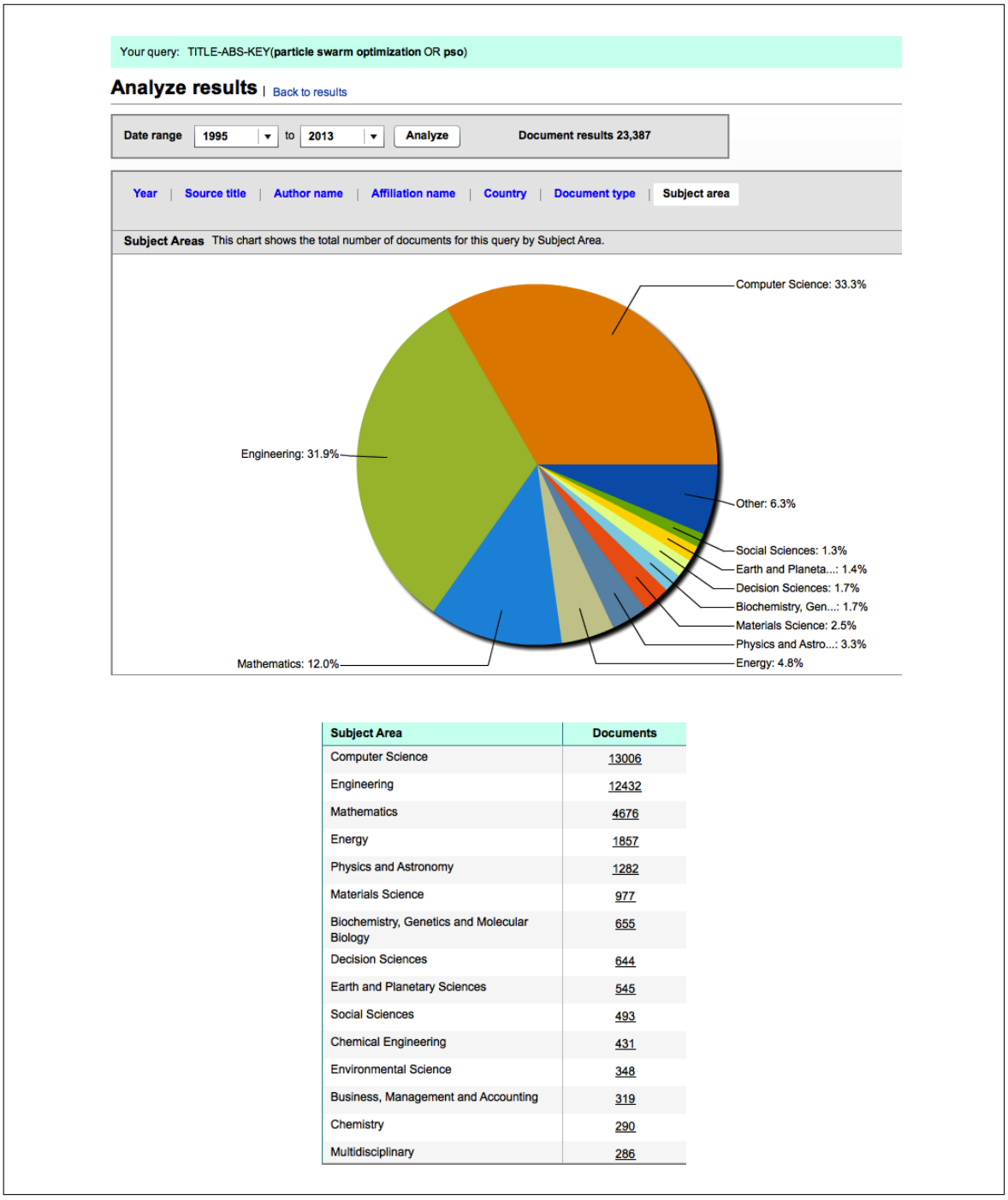
[5]Se trata de un término empleado en informática, que engloba diversas técnicas empleadas para solucionar problemas que manejan información incompleta, con incertidumbre e inexacta.

Hypothesis I: Hay dos fuentes principales de conocimiento, a saber, la experiencia propia y la comunicación con otros (p. 355).

El libro está totalmente enfocado en la idea del enjambre (swarm). Sin embargo, el algoritmo no presenta el comportamiento de un enjambre real, ya que los bichos no realizan alianzas para toda la vida, mientras que los boids interactúan durante toda su existencia con el mismo conjunto de individuos. Aun así, en ciencias de la computación los enjambres proporcionar una metáfora muy útil y detallada para el ámbito del Soft Computing. Al parecer, los enjambres de partículas (Particle Swarm) están siendo investigados y utilizados para propósitos de ingeniería en los Estados Unidos, Gran Bretaña, Japón, China, Turquía, Grecia, Francia, Alemania, Canadá y Australia. Se puede aplicar a muchos tipos de problemas con tan solo unos pequeños ajustes en el algoritmo. Este análisis concluye recalcando que el paradigma se encuentra es su primera etapa y que el trabajo no termina ahí. Para desarrollar este paradigma los lectores deben continuar aportando ideas, y es así como los individuos colaboran en la mejora del estado del conocimiento.

C.3.2. Impacto

En los últimos años, la PSO ha sido aplicada con éxito en investigación y muchas áreas de aplicación. Se demuestra que PSO obtiene mejores resultados en la optimización de funciones más rápido, mucho más barato en comparación con otros métodos. Otra razón por la que la PSO es atractiva es que hay pocos parámetros de ajuste. Una versión del algoritmo original, con ligeras variantes, funciona bien en una amplia variedad de aplicaciones. La PSO se ha enfocado para una amplia gama de aplicaciones, así como para otras específicas centradas en un requisito concreto. El trabajo de investigación sobre la aplicación de la PSO en algunos campos como la ingeniería eléctrica y las matemáticas están muy extendidas, pero otros campos como la industria química y la ingeniería civil no se han trabajado tanto. El algoritmo de PSO, como algoritmo importante de optimización, tendrá más aplicaciones en el futuro en diversas ciencias como la economía, finanzas, negocios, medicina, ingeniería, etc. Actualmente, los documentos publicados abarcan las diferentes áreas tal como se muestra en la Figura C.6



Subject Area	Documents
Computer Science	13006
Engineering	12432
Mathematics	4676
Energy	1857
Physics and Astronomy	1282
Materials Science	977
Biochemistry, Genetics and Molecular Biology	655
Decision Sciences	644
Earth and Planetary Sciences	545
Social Sciences	493
Chemical Engineering	431
Environmental Science	348
Business, Management and Accounting	319
Chemistry	290
Multidisciplinary	286

Figura C.6: Número de documentos sobre PSO en diferentes áreas. Fuente: SciVerse-Scopus

C.4. PSO MODELS:

La mayoría de estudios del PSO son empíricos y solo unos pocos se centran en comprender la trayectoria de las partículas. Estos estudios teóricos se concentran prin-

cipalmente en los modelos de PSO simplificados. Tomando como punto de partida un artículo de 2005 de Engelbrecht [67] que menciona varios estudios teoréticos, investiga en las trayectorias de las partículas y ofrece una demostración de la convergencia de las mismas, analizaremos las distintas simplificaciones, ampliaciones y modificaciones que han realizado diversos científicos a lo largo del tiempo centrándonos en particular en Kennedy, Eberhart y Shi[28, 29, 34, 35, 37, 36, 62].

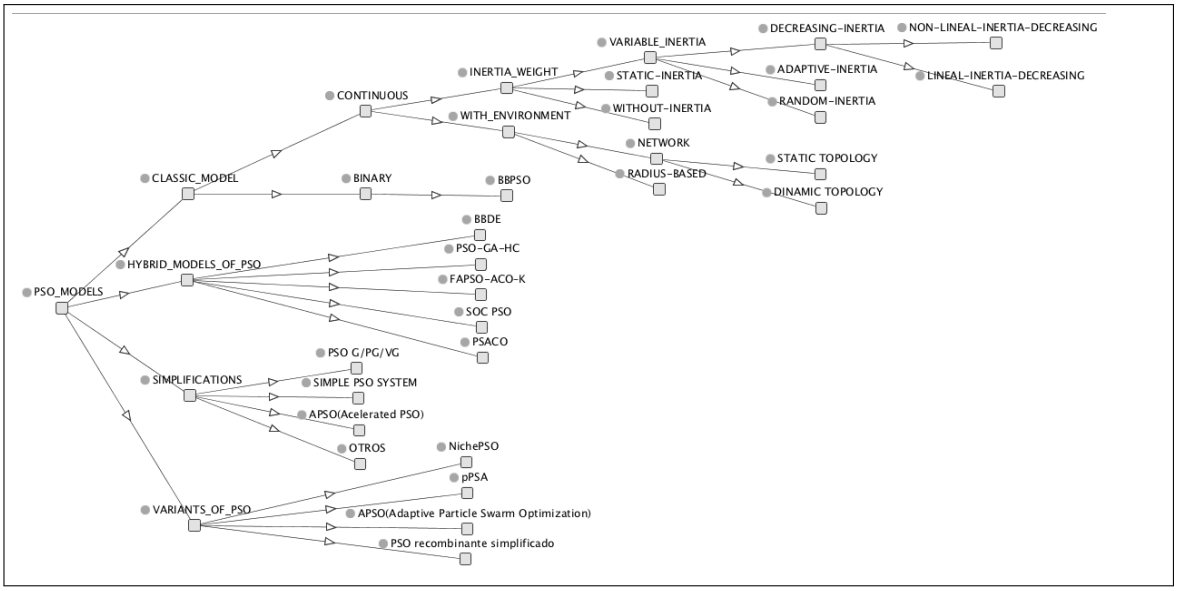


Figura C.7: Modelos PSO

C.4.1. CLASSIC MODEL

C.4.1.1. BINARY

- Bare-bone PSO

  Este modelo se ha explicado más arriba como parte del resumen del libro de James Kennedy, Russell C. Eberhart y Yuhui Shi [**?**]

C.4.1.2. CONTINOUS

4.1.2.1) INERTIA WEIGHT
Se incorpora el término inercia a la ecuación inicial de la PSO:

$$v_i(t+1) = wv_{i,}(t) + c_1 r_1(t)[y_{i,}(t) - x_{i,}(t)] + c_2 r_2(t)[\hat{y}_i(t) - x_i(t)]$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

y se proponen diferentes formas de inicializar y/o modificar el peso inercial en el algoritmo como se muestra en los siguientes modelos.

a) STATIC INERTIA

Se conoce como *Static-inertia PSO* aquel algoritmo básico de PSO que incluye el término inercia como una constante. En numerosos artículos anteriores a 1998

observamos el estudio que determina el valor de este peso inercial, donde cada autor analiza el comportamiento del algoritmo en función de parámetros como la inercia ω y las componentes c_1 y c_2. Así, Trelea[66] propuso los valores $\omega = 0{,}6$; $c_1 = 1{,}7$ y $c_2 = 1{,}7$; mientras que Clerc sugirió $\omega = 0{,}729$; $c_1 = 1{,}49445$; $c_2 = 1{,}49445$ y fueron probados en [29] por R.C. Eberhart y Y. Shi. Estos últimos autores recalcan que, el buen rendimiento del algoritmo consiste en la selección cuidadosa del peso inercial ω, c_1 y c_2.

b)VARIABLE INERTIA

b1) ADAPTIVE INERTIA

Desde que el parámetro inercia fue incorporado al modelo básico de PSO ha habido numerosas propuestas que planteaban diferentes estrategias para la modificación de este parámetro a lo largo del proceso de ejecución y la inercia adaptiva es uno de ellos. Esta estrategia consiste en analizar la situación de búsqueda y adaptar el valor del peso inercial en base a uno o más parámetros de feedback. Estos parámetros de feedback podrían ser: Best fitness[45] , Fitness de las iteraciones actual y anterior [45], Global best y el promedio del mejor fitness local [45], Particle rank[45], Distancia a la partícula y las mejores posiciones globales [54], Distancia a la mejor posición global [65].

De manera resumida:

actualización de ω	**parámetro de feedback**
Fuzzy Rules	*best fitness*
$\omega_i^t = \omega_{initial} - \alpha(1 - h_i^t) + \beta s$	*Fitness* de la iteración actual y anterior
$\omega = 1{,}1 - \frac{gbest}{(pbest_i)average}$	*Global best* y *local best fitness* medio
$\omega_i = \omega_{min} + (\omega_{max} - \omega_{min})\frac{Rank_i}{Total\,Population}$	*Rank* de las partículas
$\omega_{ij} = 1 - \alpha(\frac{1}{1+e^{ISA_{ij}}})$, $ISA_{ij} = \frac{\lvert x_{ij} - P_{ij}\rvert}{\lvert p_{ij} - p_{gj}\rvert + e}$	Distancia a la partícula y las mejores posiciones globales
$\omega_i = \omega_{initial}(1\frac{dist_i}{maxDist})$, $dist_i = (\sum_{d=1}^{D}(gbest_d - x_{i,d})^2)^{1/2}$	Distancia a la mejor posición global.

Cuadro C.1: Estrategias de modificación del peso inercial

b2) DECREASING INERTIA

En los modelos iniciales con inercia ω era constante. Sin embargo, análisis como los realizados en [21] indican que es mejor inicializar ω en un valor grande con el fin de promover la exploración global del espacio de búsqueda y e ir disminuyendo su valor para obtener soluciones más refinadas. Se proponen 2 estrategias: lineal y no lineal.

- **Lineal-Inertia-Decreasing**

El valor del peso inercial viene dado por la siguiente ecuación:

$$\omega = \omega_{end} + (\omega_{start} - \omega_{end})(1 - t/t_{max})$$

donde, ω_{start} es el valor3 inicial del coeficiente de ponderación; ω_{end} es el valor final del coeficiente de ponderación; t_{max} es el número máximo de iteraciones; t es la iteración actual. En el artículo de Yuhui Shi and Russell Eberhart [62] proponen los siguientes valores: $\omega_{start} = 1.4$, $\omega_{end} = 0$ y $t_{max} = 4000$.

- **Non-Lineal-Inertia-Decreasing**

  Para un mejor equilibrio entre la capacidad de búsqueda local y global se proponen 3 estrategias que tienen como base la siguiente ecuación:

  $$\omega_0 = \omega_{end} + (\omega_{start} - \omega_{end}) \times (1 - ((t/t_{max})^{k_1})^{k_2}$$

  donde k_1 y k_2 son dos números naturales y ω_{start}, ω_{end}, t_{max} y t son tal como se definieron en la ecuación lineal.

  Cuando k_1=k_2=1 se trata precisamente de la estrategia lineal mencionada en el apartado anterior. A esta primera estrategia se le da el nombre de *LDIW strategy*. A continuación tenemos otras 3 estrategias posibles en función de los valores de k_1 y k_2. La estrategia número 1 se da mientras $k_1 > 1$ y $k_2 = 1$ y aumenta la búsqueda gobal, sacrificando su búsqueda local; la número 2 se corresponde con los valores $k_1 = 1$ y $k_2 > 1$ y aumenta la búsqueda local sacrificando la búsqueda global; la tercera y última estrategia se corresponde con los valores $k_2 > k_1 > 1$ y aumenta tanto la búsqueda local como la global, sin embargo, el decrecimiento de ω es muy rápido en las etapas intermedias.

b3) RAMDOM INERTIA.

Podemos revisar algunos artículos **[45, 54]** donde se propone modificar de forma aleatoria el valor del peso inercial en entornos dinámicos de la siguiente manera :

$$\omega = 0{,}5 + \frac{rand()}{2}$$

donde *rand()* es un numero aleatorio en [0, 1]; entonces ω es una variable uniforme aleatoria en el rango [0.5, 1]. Esta estrategia se conoce como *PSO-RANDIW*.

c) WITHOUT INERTIA

Se trata del modelo básico de PSO, ya mencionado anteriormente:

$$v_i(t+1) = v_i(t) + \phi_1(t)(y_i(t) - x_i(t)) + \phi_2(t)(\hat{y}_i(t) - x_i(t))$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

para i=1, ..., s, donde

s es el número total de partículas que posee el enjambre,
$\phi_1(t) = c_1 r_1(t)$ y $\phi_2(t) = c_2 r_2(t)$,
c_1 y c_2 son coeficientes de aceleración,
$r_1(t)$,$r_2(t) \sim U(0, 1)$,

$x_i(t)$ es la posición de la partícula i en el instante t,
$v_i(t)$ es la velocidad de la partícula i en el instante t,
$y_i(t)$ es la mejor solución personal de la partícula i en el instante t,
$\hat{y}_i(t)$ es la mejor posición encontrada por el vecindario de la partícula i en el instante t.

4.1.2.1) WITH ENVIRONMENT

Aunque ya hemos mencionado algunos aspectos relacionados con los enjambres y sus topologías en apartados anteriores, explicaremos a continuación con más detalle los diferentes entornos que dan soporte a los algoritmos de la PSO.

a) NETWORK

a1) STATIC TOPOLOGY

Los métodos más comunes para definir la influencia de unas partículas sobre otras son los ya mencionados *lbest* y *gbest*, que se corresponden con las topologías *anillo* y *full-connected*, representadas anteriormente en este documento. Los primeros experimentos partían de grafos generados aleatoriamente y posteriormente optimizados para satisfacer algunos criterios deseados. Sin embargo, en el artículo de James Kennedy y Rui Mendes [36] se presenta el estudio de 6 sociometrías:

- gbest: que considera a toda la población dentro del vecindario de cada individuo.
- lbest: donde los miembros adyacentes en el array de población componen el vecindario.
- pyramid: una pirámide tridimensional formada por mallas triangulares.
- star: un nodo central influye y es influenciado por todos los demás miembros de la población.
- small: donde la mayoría de los nodos no son vecinos el uno del otro, pero se puede llegar de uno a otro con un pequeño número de saltos o pasos, es decir, el grafo tiene una topología "small world".[6]
- von Neumann: vecinos arriba, abajo y a cada lado en una cuadricula bidimensional (Norte, Sur, Este, Oeste).

a2) DYNAMIC TOPOLOGY

Este algoritmo, conocido como DMS-PSO [41], trabaja sobre una topología de vecindario dinámica y asignada aleatoriamente. Esta estructura de vecindario tiene dos características:

- 1) Enjambres de tamaño pequeño: en la mayoría de problemas de la PSO la población es extremadamente pequeña comparada con otros algoritmos evolutivos, por lo cual se opta por utilizar vecindarios de pequeño tamaño con el fin de ralentizar la velocidad de convergencia de la población y aumentar la diversidad.

[6]La distancia L (el número de pasos requeridos) entre dos nodos seleccionados al azar crece proporcionalmente al logaritmo de la cantidad de nodos(N) que tiene la red, que es: $L\ \alpha\ log\ N$.

- 2) Reagrupamiento aleatorio: si mantuviéramos las estructuras vecinas sin cambios, entonces no habría intercambio de información entre los enjambres y sería una PSO co-evolutiva con estos enjambres en una búsqueda en paralelo. Para evitar esto se introduce un cronograma de reagrupación aleatorio. Cada R generaciones la población se reagrupa de forma aleatoria y comienza su búsqueda en base a su nueva configuración. R recibe el nombre de *periodo de reagrupación.* De esta manera se consigue que la buena información recogida por los vecindarios fluya a través de todo el enjambre. Este tipo de estructura posee más libertad y ofrece un mejor comportamiento cuando se trata de problemas multimodales complejos.

El diagrama de flujo es dado en la Figura C.8.

m: tamaño de cada enjambre

n: número de enjambres

R: periodo de reagrupamiento

Max_gen: nº máximo de generaciones, criterio de parada.

Inicializar $m * n$ partículas (posición y velocidad)

Dividir la población en n enjambres aleatoriamente con m partículas en cada

enjambre.

For i=1: 0.9*Max_gen

Actualizar cada enjambre usando la versión local de PSO

If $mod(i, R)$==0,

Reagrupar los enjambres aleatoriamente,

End

For i=0.9*Max_gen : Max_gen

Actualizar todas las partículas empleando la versión global de la PSO

End

Figura C.8: DMS-PSO

b) RADIUS GEOMETRY

El vecindario de un individuo lo forman aquellos individuos que se encuentran dentro de un radio de distancia R, de manera que es influenciado por los individuos mas cercanos en cada momento tal como se muestra en [59].

C.4.2. HIBRID MODELS

C.4.2.1. BBDE

Se trata de un algoritmo híbrido entre barebone PSO(BBPSO) y Diferential Evolution (DE) al que Mahamed G.H. Omran, A. P. Engelbrecht y Ayed Salman bautizan como BBDE [47]. Desarrollaron unas estrategias DE eficientes que eliminaban la necesidad de optimizar los parámetros de control de la PSO. La DE es un algoritmo evolutivo propuesto por Storn y Price en 1995 [64] y utiliza las diferencias entre dos vectores seleccionados aleatoriamente (individuos) como fuente de las variaciones aleatorias de un tercer vector (individuo), que se refiere como el vector objetivo. Este proceso se conoce como el operador de mutación, donde se muta el vector objetivo. Entonces se aplica una recombinación o crossover para producir una descendencia que sólo se acepta si mejora en la condición física del individuo padre. Para el barebone DE, la actualización de las posiciones se realizan de la siguiente manera:

$$x_{ij}(t) = \begin{cases} p_{ij}(t) + r_{2j} \times (x_{i_1 j}(t) - x_{i_2 j}(t)) & if\, U(0,1) > p_r \\ y_{i_3 j}(t) & otherwise \end{cases}$$

donde $p_{ij}(t) = r_{1j}(t) y_{ij}(t) + (1 - r_{1j}(t))\hat{y}_{ij}(t)$

con $i_1, i_2, i_3 \sim U(1, \ldots, s)$, $i_1 \neq i_2 \neq i$, $r_{1,j}, r_{2.j} \sim U(0,1)$ y p_r es la probabilidad de recombinación.

En la ecuación, $p_i(t)$ representa el atractor de partículas como una media estocástica ponderada de la mejor posición *personal y* global. El operador de mutación de la DE se utiliza para explorar alrededor del atractor actual $p_i(t)$. El *crossover* se realiza con un *personal best* y_{i3} seleccionado al azar, ya que estos *personal best* representan una memoria de las mejores soluciones encontradas por los individuos desde el inicio del proceso de búsqueda. Empleando la ecuación de posición propuesta, para una proporción de $(1 - p_r)$ de las actualizaciones se utiliza la información de un *personal best* seleccionado de manera aleatoria (lo que facilita la explotación), mientras que para proporciónp_r de las actualizaciones los tamaños del *step* son mutaciones del punto de atracción p_i(facilitando la exploración).

C.4.2.2. FAPSO-ACO-K

Este modelo combina (FAPSO), ACO y K-means como algoritmo de clustering. En este algoritmo, el peso inercial y factores de aprendizaje de PSO se ajustan dinámicamente utilizando reglas Fuzzy del tipo IF / THEN. Incorpora la inteligente estructura del algoritmo de toma de decisiones ACO dentro del FAPSO original donde la mejor posición global es única para cada partícula. La salida del algoritmo híbrido FAPSO-ACO [46]se considera como estado inicial del K-means. El diagrama de flujo queda representado en la Figura C.9

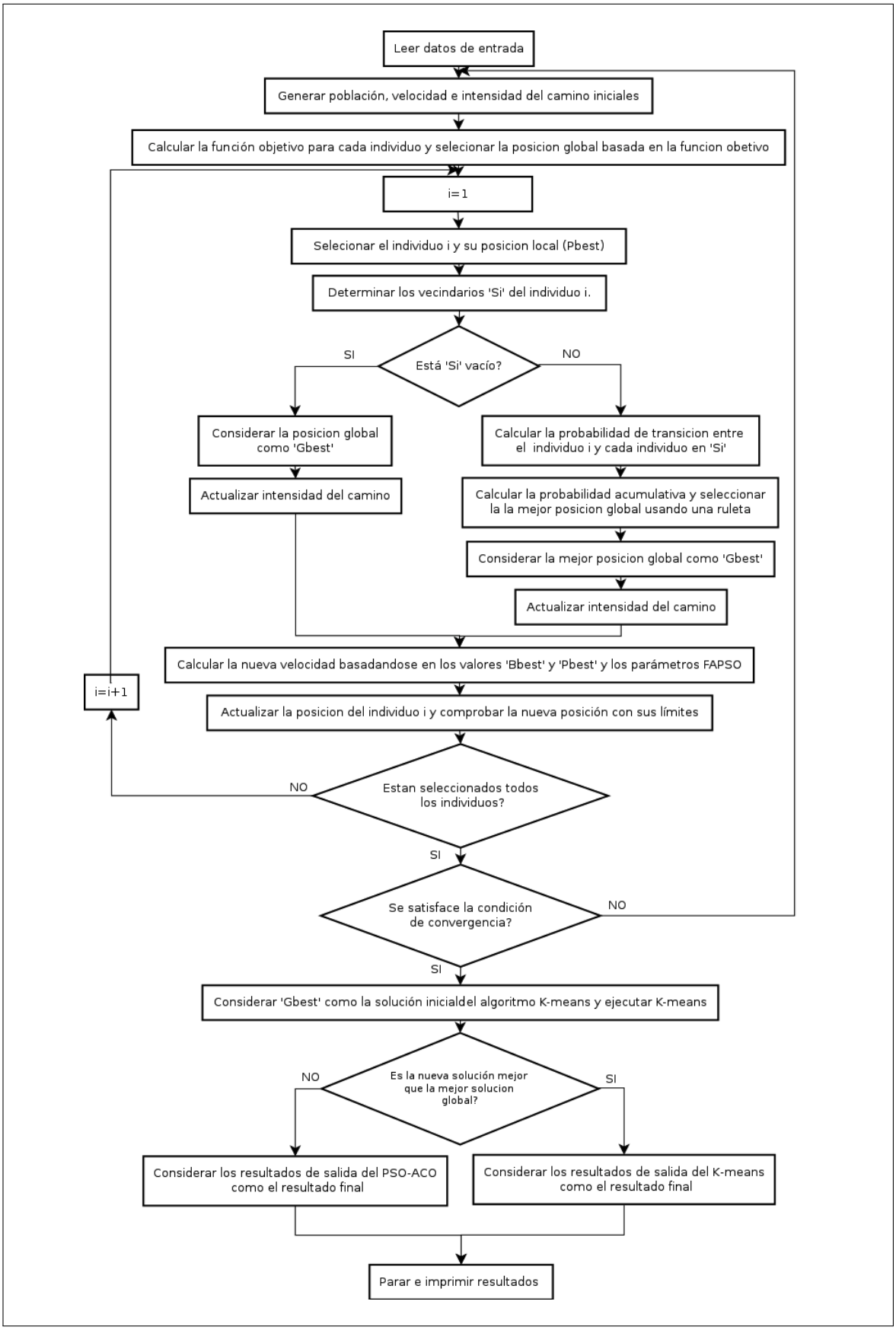


Figura C.9: Diagrama de flujo FAPSO-ACO-K

C.4.2.3. PSACO

P.S. Shelokar Patrick Siarry, V.K. Jayaraman y B.D. Kulkarni [61]proponen en su artículo un algoritmo híbrido entre Particle Swarm y ACO bajo el nombre de PSACO. Se trata de utilizar un mecanismo simple guiado por feromonas que permita mejorar el rendimiento de la PSO. El algoritmo, que se muestra en la Figura C.2consta de dos etapas. En la primera etapa, se aplica PSO, mientras que ACO se lleva a cabo en la segunda etapa. ACO funciona como una búsqueda local, en donde, las hormigas aplican mecanismos guiados por feromona para refinar las posiciones encontradas por las partículas en la fase de PSO.

C.4.2.4. PSO-GA-HC

Este modelo, conocido como PSO-GA-HC [38], combina PSO con algoritmos genéticos(GA) y hillclimbers(HC). El modelo LifeCycle, representado en la Figura C.10, trata de una búsqueda heurística auto-adaptativa donde los individuos comienzan siendo partículas PSO que pueden transformarse en individuos de un algoritmo genético, luego en Hill Climbers, después partículas de nuevo y así sucesivamente.

- Modelo PSO:

  En cada iteración velocidad y posición se actualizan de la siguiente manera:

  $$v_i = \chi(wv_i + \varphi_{1i}(p_i - x_i) + \varphi_{2i}(p_g - x_i)),$$

  $$x_i = x_i + v_i,$$

  siendo φ_{1i}=φ_{2i}=2.0, con w decreciendo linealmente de 0.7 a 0.4, el coeficiente de restricción χ=1 y v_{max} la mitad del espacio de búsqueda. El vecindario está formado por 20 partículas con una topología de tipo full-connected.

- Modelo GA:

  Un algoritmo genético clásico consiste en una población de individuos refinando sus soluciones candidatas a través de la interacción y adaptación. Cada individuo representa una solución candidata para el problema dado. Después de la inicialización el GA entra en un bucle en el cual la población es evaluada, se selecciona una nueva población y esta nueva población se altera. El modelo GA de este LifeCycle utiliza selección por torneo binario para generar una nueva población y el elitismo para asegurar la supervivencia del individuo con mejor fitness. El modelo GA LifeCycle altera la población mediante cruce y mutación. El operador de cruce utilizado en el modelo GA LifeCycle es el conocido como cruce aritmético. Este operador sustituye a dos individuos padres seleccionados para el cruce con dos individuos hijo de la siguiente manera:

  $$x_{child1} = w{*}x_{parent1} + (1-w){*}x_{parent2}$$

  $$x_{child2} = w{*}x_{parent2} + (1-w){*}x_{parent1}$$

- step 1. Inicializar Optimización.
  - step 1.1. Inicializar constantes P, t_{max}para los procesos PSO y ACO.
  - step 1.2. Inicializar aleatoriamente todas las posiciones x_t^i y velocidades v_t^i de las partículas.
  - step 1.3. Evaluar la función objetivo como $f(x_t^i)$ en la ecuación $(F) = \begin{cases} min & f(x) \\ s.t. & S \subset \mathbb{R}^n \end{cases}$
  - step 1.4. Asignar mejores posiciones $p_t^i = x_t^i$ con $f(p_t^i) = f(x_t^i)$, $i = 1, \ldots, P$.
  - step 1.5. Buscar $f_t^{best}(p_t^{best}) = min\{f(x_t^1), \ldots, f(x_t^P)\}$ e inicializar $p_t^g = p_t^{best}$ y $f(p_t^g) = f^{best}(p_t^{best})$.
- step2. Realizar Optimización.

  mientras $(t \leq t_{max})$
  - step 2.1. Actualizar velocidad v_t^i y posición x_t^i de todas las P partículas de acuerdo con las ecuaciones $v_{t+1}^i = w_t v_t^i + c_1 r_1 (p_t^i - x_t^i) + c_2 r_2 (p_t^g - x_t^i)$ y $x_{t+1}^i = x_t^i + v_{t+1}^i$
  - step 2.2. Evaluar la función objetivo como $f(x_t^i)$ en la ecuación $(F) = \begin{cases} min & f(x) \\ s.t. & S \subset \mathbb{R}^n \end{cases}$
  - step 2.3. Actualizar la mejor posición de la partícula si $f(p_t^i) = f(x_t^i)$ entonces $p_t^i = x_t^i$ con $f(p_t^i) = f(x_t^i)$.
  - step 2.4. Buscar $f_t^{best}(p_t^{best}) = min\{f(x_t^1), \ldots, f(x_t^P)\}$; si $f(p_t^g) > f(p_t^{best})$ entonces $p_t^g = p_t^{best}$ y $f(p_t^g) = f^{best}(p_t^{best})$.
  - step 2.5. Incrementar contador de iteraciones $t = t + 1$.

  fin mientras
- step3. Reportar la mejor solución p^g del enjambre con valor de la función objetivo $f(p^g)$.

Cuadro C.2: Algoritmo PSACO

```
program LifeCycle Model
begin
    inicializar
    while (not condicion-terminada) do
    begin
        for (todos los individuos)
            evaluar fitness
            cambiar el escenario del LifeCycle si
                no hay nuevas mejoras
        for (partículas PSO)
            calcular los nuevos vectores de velo-
                cidad
        for (individuos GA)
            seleccionar nueva población
            recombinar población
            mutar población
        for (HillClimbers)
            buscar una nueva posible solución
                vecina
            evaluar el fitness para la nueva solu-
                ción
            cambiar a la nueva solución con una
                probabilidad p
    end
end
```

Figura C.10: PSO-GA-HC

donde$w$ es un valor aleatorio entre cero y uno. La entrada j de un individuo es mutada de acuerdo con:

$$\triangle x_j \begin{cases} +(Max) - x_j)(1 - r^{(1-t/T)^b}) \\ -(x_j - Min)(1 - r^{(1-t/T)^b}) \end{cases}$$

teniendo cada una probabilidad del 50 %. Max es el máximo del espacio de búsqueda, Min el mínimo, r es un número aleatorio en [0..1], $t$es la iteración actual, T es el número total de iteraciones y b es un parámetro para determinar el grado de dependencia del número de iteración.

- Modelo Hill Climber:

  En el modelo LifeCycle cada HillClimber consiste en una solución de x_c. Para cada iteración, un nueva solución candidata x_n es seleccionada en el vecindario

de x_c. x_c es remplazada por x_n con una probabilidad p dada por:

$$p = 1/(1 + exp(\frac{eval(x_n) - eval(x_c)}{T}))$$

donde T toma el valor 10 en los experimentos realizados por Krink y Lovbjerg[38]. Para conocer los experimentos con detalle se pueden ver las tablas 1, 2, 3 y 4, a partir de la página 5.

C.4.2.5. SOC PSO

Este modelo extiende PSO con Self-Organized Criticality(SOC). La idea principal de SOC es que la mayoría de las transiciones de estado de un componente dentro de un sistema complejo sólo afecten a su vecindario, pero de vez en cuando una avalancha de propagación de transiciones de estado puede llevar a una importante reconfiguración del sistema. La estructura del SOC PSO se muestra en la Figura C.11, donde el umbral de distancia TD está inicializado a un 1 % del espacio de búsqueda en una dimensión y decrece linealmente hasta 0, $CL = tampoblación/5$ y $CR = 2/tampoblación$. La velocidad se actualiza de la siguiente manera:

$$v_i = \chi(wv_i + \varphi_{1i}(p_i - x_i) + \varphi_{2i}(p_g - x_i)),$$

donde $w = 0{,}2 + \frac{C}{10}$, $\varphi_1 = \varphi_2 = 2{,}0$ y v_{max} es la mitad del espacio de búsqueda. En sus experimentos, Lovbjerg y Krink[44] establecen una población de 20 individuos

C.4.3. VARIANTS

C.4.3.1. APSO(Adaptive PSO)

Adaptive Particle Swarm Optimization(APSO)[**?**] es una variante de PSO que puede realizar una búsqueda global sobre todo el espacio de búsqueda con una velocidad de convergencia más rápida.

Consta de 2 partes principales:

- 1. Mediante la evaluación de la distribución de la población y el fitness de las partículas se realiza un procedimiento de estimación en tiempo real sobre los estados evolutivos (real-time evolutionary state estimation procedure) para identificar a uno de los siguientes cuatro estados evolutivos definidos: exploración, explotación, convergencia y "saltando" en cada generación. Esto permite el control automático de la masa de inercia, coeficientes de aceleración y otros parámetros algorítmicos en tiempo de ejecución, para mejorar la eficiencia de la búsqueda y la velocidad de convergencia.

- 2. Una estrategia de aprendizaje elitista cuando se encuentra en el estado evolutivo convergencia.

APSO únicamente incluye 2 parámetros adicionales al paradigma PSO, de tal manera que no introduce un diseño adicional ni complejidad de implementación. El algoritmo general viene dado por la Figura C.13.

```
begin
inicializar (C = 0; CL )
        while (not condición-terminada) do
        begin
                evaluar
                calcular nuevos vectores de velocidad
                mover
                calcular criticidad (C)
                reducir criticidad (C = C-CR)
                while (algún agente tenga valor critico> límite
                    de criticidad) do
                begin
                        dispersar criticidad
                        recolocar agente (2 formas de hacer-
                            lo)
                    end
            end
    end
```

Figura C.11: SOC PSO

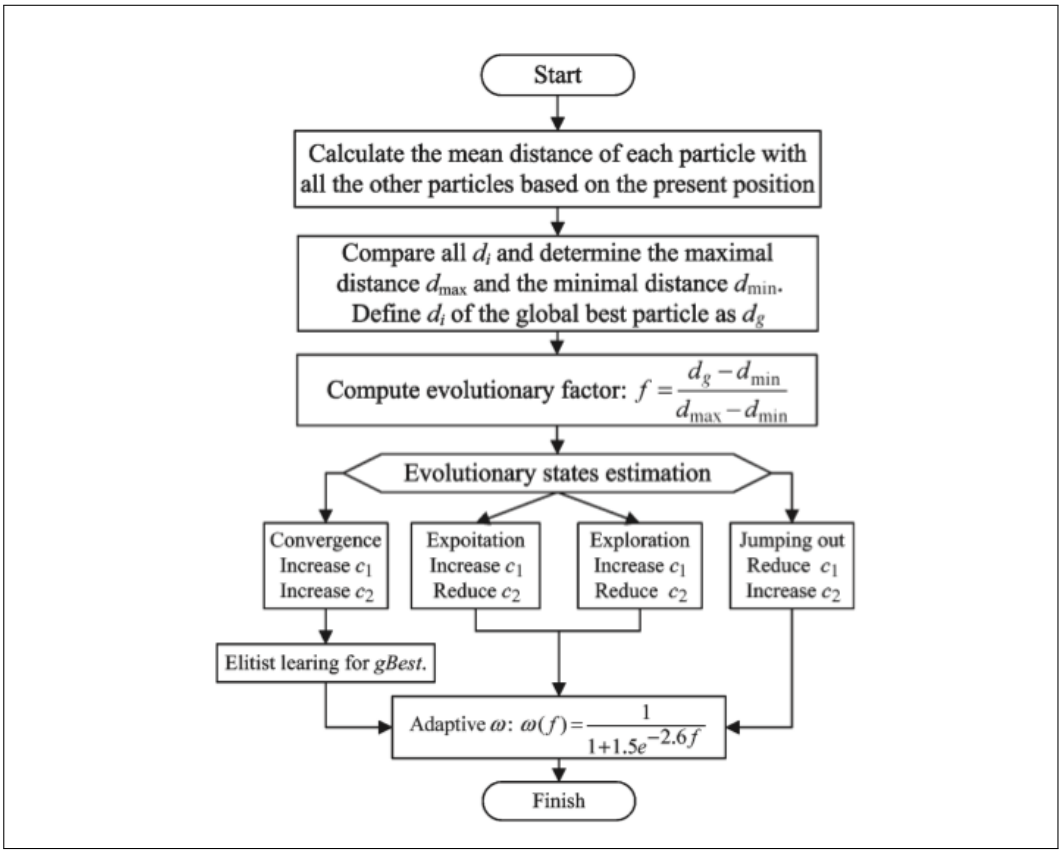


Figura C.12: Proceso de control de parámetros adaptivo

Se realiza un control adaptivo de los parámetros generales PSO, donde la adaptación del peso inercial es $\omega(f) = \frac{1}{1+1,5e^{-2,6f}} \in [0{,}4, 0{,}9]\ \forall f \in [0, 1]$ y el control de los coeficientes de aceleración y la limitación de los mismos es tal como se muestra en la Figura C.12.

En varios experimentos [75, 73] se inicializa inercia ω = 0.9, coeficientes c1= c2= 2.0 y una población de 20 individuos.

C.4.3.2. NichePSO

R. Brits, A.P. Engelbrecht y F. van den Bergh [19]presentan un nuevo método de PSO bautizado como NichePSO. El resultado del algoritmo de NichePSO es un conjunto de enjambres de partículas, cada uno representando una solución única. Los nichos son identificados mediante la monitorización del *fitness* de las partículas individuales y desarrollando los sub-swarms de la población inicial. Usando una estrategia de optimización global las subpoblaciones refinan luego la solución que representa el nicho. Los métodos de Niching[7] pueden categorizarse como *secuenciales* o *paralelos*. Independientemente de la forma en que se encuentran los nichos (es decir, en pa-

[7]J. Horn (1997) define el niching como *"a form of cooperation around finite, limited resources, resulting in the lack of competition between such areas, and causing the formation of species for each niche"*

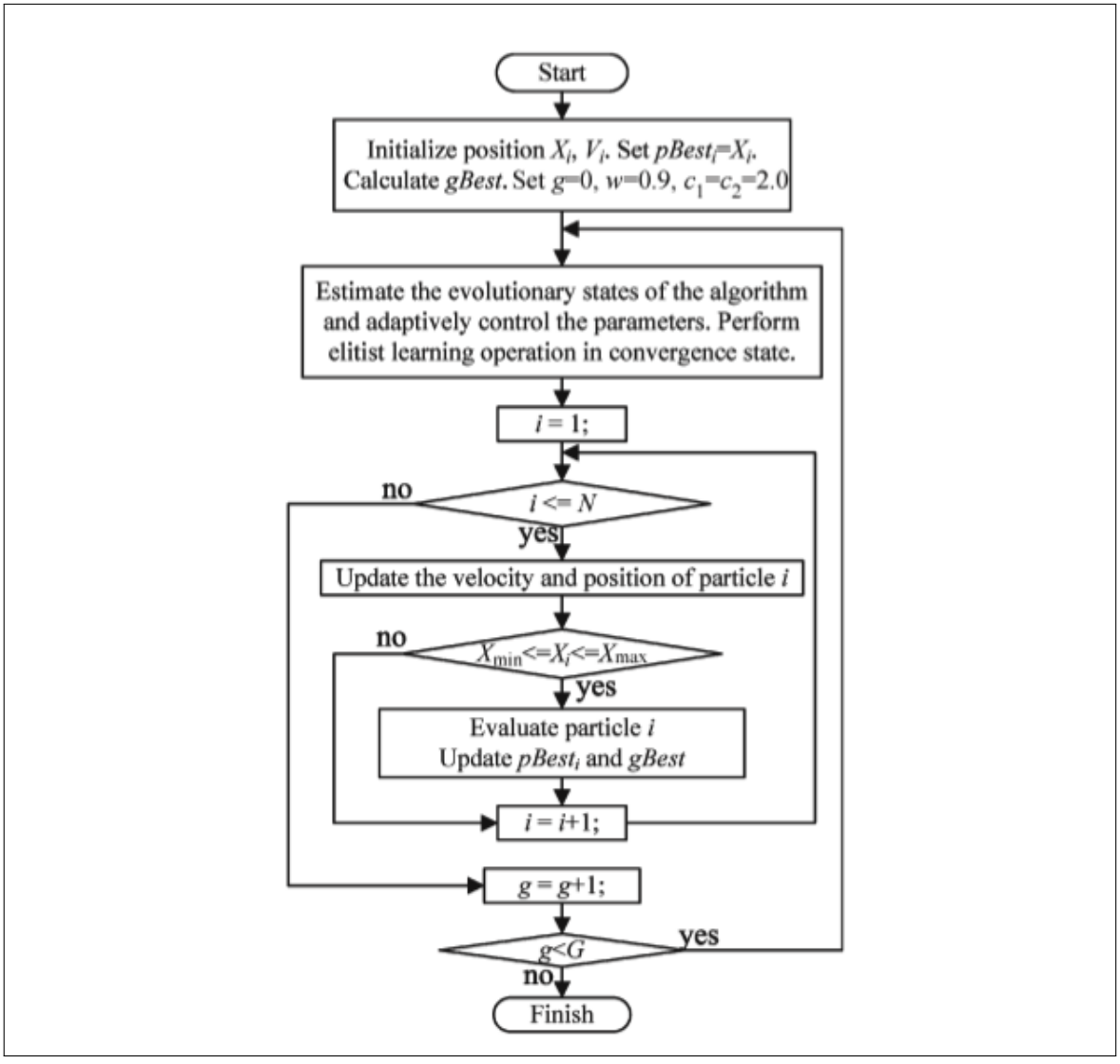


Figura C.13: Algoritmo APSO

ralelo o secuencialmente), la distribución de los individuos puede ser formalizada de diversas maneras de acuerdo con su comportamiento: *Sympatric speciation*, *Allopatric speciation* o *Parapatric speciation.*

El enjambre inicial se conoce como enjambre 'principal'. Cuando se encuentra una partícula que representa una solución candidata, se crea un sub-swarm con esta partícula y su vecino más cercano. Los sub-swarms se cultivan a partir del enjambre 'principal'. Cuando todas las partículas del enjambre 'principal' se hayan agotado, no se podrán crear sin más sub-swarms. Los sub-swarms que ocupan regiones en el espacio de búsqueda que representan el mismo nicho se fusionan.

Para garantizar la convergencia Van den Bergh et al. introducen un nuevo algoritmo llamado *Guaranteed Convergence PSO(GCPSO).* En él la velocidad y la posición se actualizan de la siguiente manera:

$$v_{\tau,j}(t+1) = -x_{\tau,j}(t) + \hat{y}_j(t) + wv_{\tau,j}(t) + p(t)(1 - 2r_{2,j})$$

$$x_{\tau,j}(t+1) = \hat{y}_j(t) + wv_{\tau,j}(t) + p(t)(1 - 2r_{2,j})$$

donde τ se refiere al índice de la mejor partícula global, el término $-x_\tau$ restablece la posición de la partícula a la mejor posición global $\hat{y}$, $wv_{\tau,j}$ significa la dirección de búsqueda y $p(t)(1 - 2r_{2,j})$ proporciona un término de búsqueda aleatoria a la ecuación. El parámetro $p(t)$ se adapta dinámicamente para controlar el tamaño del cuadro delimitador alrededor de $\hat{y}$ en el cual se lleva a cabo una búsqueda local para forzar un cambio en el valor de $\hat{y}$.

C.4.3.3. pPSA

Esta variante de PSO se conoce como *Perturbed particle swarm algorithm (pPSA)* y consiste en un algoritmo de swarm perturbado con el fin de escapar de la trampa del óptimo local. La estrategia de actualización de las partículas se basa, en la posibilidad de tratar el problema de mantener la diversidad dentro del enjambre, así como promover la exploración del espacio. En PSO básico, el enjambre converge rápidamente dentro de la zona intermedia del gbest. Esto lleva a:

1. perdida de la diversidad y
2. convergencia prematura si el gbest pertenece a un óptimo local.

Por ello, se propone pPSA [71], que está basado en la perturbación de la estrategia de actualización del gbest $(p-gbest) = (p_{g,1} \times p_{g,2} \times ... \times p_{g,D})$. El cálculo de la velocidad de una partícula se reescribe de la siguiente manera:

$$p_{g,d}^{k'} = N(p_{g,d}^k, \sigma)$$

$$\sigma = p(k)$$

$$v_{i,d}^{k+1} = \omega \times v_{i,d}^k + c_1 \times r_1^k \times (p_{i,d}^k - x_{i,d}^k) + c_2 \times r_2^k \times (p_{g,d}^{k'} - x_{i,d}^k)$$

1. Inicializar el Swarm de partículas 'principal'.
2. Entrenar las partículas del swarm 'principal' usando una iteración del modelo *'cognition only'*[a].
3. Actualizar el *fitness* para cada partícula del swarm 'principal'.
4. Para cada sub-swarm:
   a) Entrenar las partículas del sub-swarm usando una iteración del algoritmo GCPSO
   b) Actualizar el *fitness* de cada partícula.
   c) Actualizar el radio del sub-swarm.
5. Si es posible, fusionar sub-swarms
6. Permitir a los sub-swarms que absorban cualquier partícula del swarm 'principal' que se mueva dentro del suyo.
7. Buscar el swarm 'principal' cualquier partícula que cumpla con los criterios de partición. Si se encuentra alguna, crear un sub-swarm con ésta y su vecino mas cercano.
8. Repetir desde 2 hasta que se alcance el criterio de parada.

[a]se conoce como *Cognition only model* cuando una partícula sólo tiene en cuenta su propia "historia y experiencia" en forma de un *personal best* según estableció Kennedy

Cuadro C.3: Algoritmo NichePSO

donde $p_{i,d}^k$ es la dimensión número d del ciclo k del *p-gbest;* el *p-best* está caracterizado por una distribución normal $N(p_{g,d}^k, \sigma)$ donde σ representa el grado de incertidumbre acerca de la optimalidad de la posición del *gbest* y $p(k)$ está definido como:

$$p(k) = \begin{cases} \sigma_{max} & cycles < \alpha \times max\_cycles \\ \sigma_{min} & otherwise \end{cases}$$

donde σ_{max} , σ_{min} y α son parámetros manualmente establecidos.
El algoritmo se muestra a continuación:

1: Inicializar un array de partículas con posición y velocidad aleatorias en D dimensiones del espacio de búsqueda.
2: loop
3: para cada partícula, evaluar la función fitness de optimización deseada en las D dimensiones.
4: comparar la evaluación del fitness de las partículas con el $pbest_i$. *Si el valor actual es mejor que el* $pbest_i$*, entonces sustituir el valor del* $pbest_i$ *por el valor actual y definir* p_i *como la posición actual* x_i *en el espacio D-dimensional.*
5: Identificar la partícula del vecindario con mejor éxito hasta el momento y asignar su índice a la variable g.
6: Cambiar la velocidad y posición de la partícula de acuerdo con las siguientes ecuaciones:

$$p_{g,d}^{k'} = N(p_{g,d}^k, \sigma),$$

para $d = 1,..., D$, y donde σ está definido anteriormente.

$$\vec{v}_i^k \leftarrow \omega \times \vec{v}_i^k + c_1 \times \vec{U}(0,1) \otimes (\vec{p}_i^k - \vec{x}_i^k) + c_2 \times \vec{U}(0,1) \otimes (\vec{p}_g^k - \vec{x}_i^k)$$

$$\vec{x}_i^k \leftarrow \vec{x}_i^k + \vec{v}_i^k$$

7: si se alcanza el criterio de parada (fitness satisfecho o máximo número de iteraciones alcanzado), salir del loop.
8: End Loop.

Cuadro C.4: Algoritmo pPSA

Consideraciones:

- $\vec{U}(0,1)$ representa un vector de números aleatorios distribuidos uniformemente en [0,1], que ha sido generado aleatoriamente en cada iteración de cada partícula.
- $\otimes$ es una multiplicación 'component-wise'

Para realizar los experimentos Zhao Xinchao [71] propone los siguientes valores: número de dimensiones (N)=30; número de partículas ($nPART$)=30; número máximo de generaciones($nGENE$)= 2000; peso inercial (ω)= 0.9; c1 = 0.5; c2 = 0.3; σ_{max} = 0.15; σ_{min} =0.001 y α = 0.5

C.4.3.4. PSO recombinante simplificado

De acuerdo con diferentes experimentos donde se eliminan algunos de los componentes del algoritmo y se comprueba cómo afectan estos cambios al rendimiento, Dan Bratton y Tim Blackwell [18] proponen un algoritmo de PSO muy sencillo (PSO-DR3), pero que ofrece un rendimiento competitivo a nivel PSO, eliminando la aleatoriedad multiplicativa, la inercia y el término p_i que pertenece a la memoria personal en la actualización de la posición.

$$x_{id}^{t+1} = x_{id}^{t} + \phi(r_{id} - x_{id}^{t}),$$

donde $\phi = 2$. Además se define un vector recombiante de posiciones $\vec{r}$:

$$r_{id} = \eta_d p_{ld} + (1 - \eta_d) p_{rd},$$

donde $\eta_d = U\{0, 1\}$ y $\vec{p_l}$ y $\vec{p_r}$ son los vecinos inmediatos de i a izquierda y derecha en una topología de anillo.

C.4.4. SIMPLIFICATIONS

Con el fin de comprender mejor las trayectorias de las partículas se realizaron diversos estudios teoréticos. Estos quedan reflejados en los modelos de PSO Simplificados.

C.4.4.1. aPSO(Acelerated PSO)

Se trata de una versión simplificada del algoritmo PSO básico, empleando únicamente el gbest para acelerar la convergencia del algoritmo.

$$v_i^{t+1} = v_i^t + \alpha\epsilon_n + \beta(g * -x_i^t)$$

donde ϵ_n perteneciente a $N(0, 1)$ sustituye el segundo término. La actualización de la posición es simplemente:

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

con el fin de aumentar la convergencia aún más, también podemos escribir la actualización de la ubicación en una sola etapa:

$$v_i^{t+1} = (1 - \beta)x_i^t + \beta g * + \alpha\epsilon_n$$

Xin-She Yang, Suash Deb y Simon Fong [72] proponen $\alpha = 0{,}7^t$, donde $t \in [0, t_{max}]$ yt_{max} es el número máximo de iteraciones.

C.4.4.2. PSO-G

Las partículas son atraídas hacia la mejor posición conocida (G) por el enjambre [52].

$$\vec{x} \leftarrow \vec{x} + \phi_g r_g(\vec{g} - \vec{x})$$

C.4.4.3. PSO-PG

Esta simplificación no tiene velocidad recurrente, tan solo la atracción de las partículas hacia su propia mejor posición conocida (P) [52].

$$\vec{x} \leftarrow \vec{x} + \phi_p r_p(\vec{p} - \vec{x}) + \phi_g r_g(\vec{g} - \vec{x})$$

C.4.4.4. PSO-VG

Esta es la simplificación que tan solo tiene la atracción hacia la mejor posición (G) conocida por el enjambre [52].

$$\vec{v} \leftarrow \omega\vec{v} + \phi_g r_g(\vec{g} - \vec{x})$$

C.4.4.5. Simple PSO System

Este modelo de Ozcan y Mohan [49, 48] se centra en el estudio de la trayectoria de una sola partícula (s=1). Su primera simplificación consta de partículas unidimensionales (n=1) con un kept constante (y(t) = $\hat{y}$(t)= p) . Posteriormente amplían su modelo a n>= 1 con y_i(t)=y_i & $\hat{y}$(t)= $\hat{y}$. De este estudio se concluyó que las trayectorias de la partícula siguen olas sinusoidales periódicas.

C.4.4.6. Otros estudios con modelos simplificados

El modelo simplificado que Clerc y Kennedy presentan en 2002 [22] está definido como:

$$\begin{cases} v_{t+1} = v_t + \varphi y_t \\ y_{t+1} = -v_t + (1 - \varphi) y_t \end{cases}$$

Se trata de un modelo unidimensional (1-D) al que posteriormente se le añade el término conocido como "coeficiente de constricción". El objetivo de este coeficiente es evitar que la velocidad crezca fuera de los límites. Se estudiaron 2 modelos de constricción:

- A) Constricción para el Modelo Tipo1

$$\begin{cases} v(t+1) = \chi(v(t) + \varphi y(t)) \\ y(t+1) = -\chi(v(t) + (1-\varphi)y(t)) \end{cases}$$

el criterio de convergencia se satisface cuando$\chi < min(\frac{1}{|e_1|}, \frac{1}{e_2})$ donde

$$\begin{cases} e_1 = 1 - \frac{\varphi}{2} + \frac{\sqrt{\varphi^2 - 4\varphi}}{2} \\ e_2 = 1 - \frac{\varphi}{2} - \frac{\sqrt{\varphi^2 - 4\varphi}}{2} \end{cases}$$

con $\varphi < 2$. Cuando $|e_1| \leq |e_2|$ se produce el siguiente coeficiente de constricción

$$\chi = \frac{\kappa}{|e_2|},\ \kappa \in]0, 1[.$$

- B) Constricción para el Modelo Tipo 1'

$$\begin{cases} v(t+1) = \chi(v(t) + \varphi y(t)) \\ y(t+1) = -\chi(v(t) + (1-\varphi)y(t)) \end{cases}$$

con el siguiente coeficiente

$$\chi = \frac{\kappa}{|e_2|},\ \kappa \in]0, 1[,\ para\ \varphi \in]0, 2[.$$

Cuadro C.5: Modelos de constricción

Tanto Ozcan y Mohan como Clerc y Kennedy estudiaron modelos simplificados sin el término inercia, sin embargo, Frans van der Bergh propone en su tesis [68] una simplificación que estudia las trayectorias incluyendo el término inercia. En esta tesis, posteriormente revisada por Engelbrecht [67] se demuestra que cada partícula i de un *gbest* PSO converge a un punto estable. El término convergencia se usa para referirse a la propiedad de que el límite

$$\lim_{t \to +\infty} x(t) = p$$

existe, donde p es una posición arbitraria en el espacio de búsqueda, y $x(t)$ es la posición de una partícula en el momento t. La ecuación que actualiza la posición y la velocidad incluyendo el término inercia es de la siguiente manera:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2,j}(t)[\hat{y}_j(t) - x_{i,j}(t)]$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

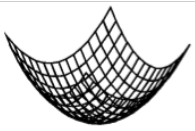
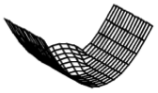
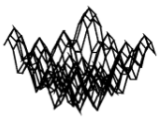
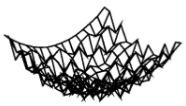
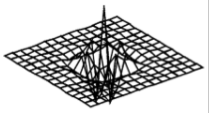
Considerando la velocidad y posición de una partícula en pasos de tiempo discretos se da la siguiente recursión:

$$\lim_{t \to +\infty} x_t = k_1 = \frac{\phi_1 y + \phi_2 \hat{y}}{\phi_1 + \phi_2}$$

donde ϕ_1 y ϕ_2 son $c_1 r_1$ y $c_2 r_2$

C.5. PSO benchmark

Podemos hacer uso de las siguientes funciones objetivo, tal como figura en [66]:

Name	Formula	Dim (n)	Optimal f	Goal for f	Sketch in 2D
Sphere	$f_0(\vec{x}) = \sum_{i=1}^{n} x_i^2$	30	0	0.01	
Rosenbrock	$f_1(\vec{x}) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	30	0	100	
Rastrigin	$f_2(\vec{x}) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	30	0	100	
Griewank	$f_3(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	0	0.1	
Schaffer's f6	$f_6(\vec{x}) = 0{,}5 - \frac{(\sin\sqrt{x_1^2+x_2^2})^2 - 0{,}5}{(1+0{,}001(x_1^2+x_2^2))^2}$	2	0	0.0001	

Anexo D

Anexo IV Manual del Usuario

D.1. Introducción

Este manual tiene como objetivo facilitar la experiencia del usuario en el trabajo con el modelo PSOmodels. Está enfocado a utilizar aplicaciones específicas de este modelo a nivel usuario. Por tanto, si desea ampliar este modelo o desarrollar nuevos modelos consulte el manual propio de Netlogo.

D.2. Aplicaciones del programa

Este modelo de Netlogo permite experimentar con el algoritmo conocido como Particle Swarm Optimization(PSO). A diferencia de otros programas escritos en otros lenguajes, este permite simular y ver en tiempo real el vuelo de las partículas de PSO. Además, permite experimentar con algunas variantes y simplificaciones que diversos científicos realizaron partiendo del algoritmo original. Posee una gran cantidad de parámetros que incluyen valores por defecto recomendados por algunos científicos en sus artículos, pero también permite que el usuario modifique estos parámetros para poder experimentar y comprender mejor el funcionamiento del algoritmo.

D.3. Proceso de instalación

Disponemos de 2 versiones del programa: (a) una que se ejecuta desde la interfaz de NetLogo que requiere tener en nuestro equipo la carpeta de trabajo de Netlogo 5.0 o superior y (b) otra como Applet que se visualizará en el navegador y requiere al menos Java 5.0.

- a1) **Necesario:**
  - NetLogo 5.0.4 que se puede descargar desde: http://ccl.northwestern.edu/netlogo/download.shtml
  - Extensión 'goo'[1] que se puede descargar desde: https://github.com/downloads/NetLogo/Goo-Extension/goo-20120119.zip
  - Archivo PSOmodels.nlogo en cualquier directorio de nuestro ordenador.
- a2) **Ejecutar aplicación desde:** PSOmodels.nlogo
- b1) **Necesario:** Dentro de una misma carpeta deberemos tener lo siguiente:
  - carpetas que contienen las extensiones 'array', 'table' y 'goo'
  - el archivo 'NetLogoLite.jar' (se puede copiar del directorio donde está instalado NetLogo 5.0)

[1] *La carpeta que contiene la extensión goo deberá ser añadida a la subcarpeta 'extensions' de la carpeta NetLogo 5.0.*

    - los archivos PSOmodels.nlogo y PSOmodels.html

- b2) **Ejecutar aplicación desde:** PSOmodels.html

D.4. Elementos de la interfaz gráfica

Cuando arrancamos Netlogo, en la parte superior de la pantalla observamos tres pestañas[2] (ejecutar, información y código) tal como se muestra en la Figura D.1. En la primera de las pestañas (ejecutar) se encuentra este modelo de PSO y será por tanto la pestaña en la que nos centraremos en este manual. En la segunda pestaña (información) podremos ver información relativa a nuestro modelo y en la última pestaña (código) encontraremos los procedimientos que se encargan de llevar a cabo la ejecución de nuestro modelo, es decir, el código de programación.

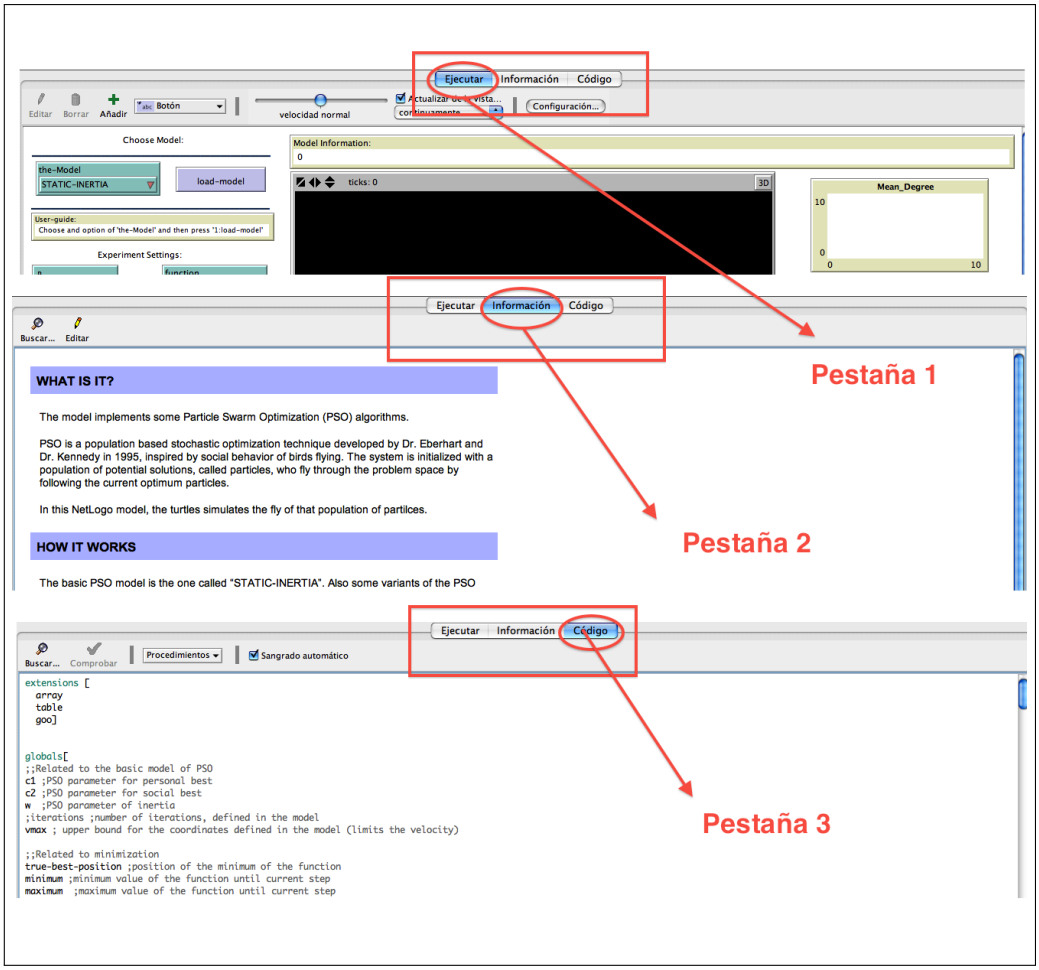


Figura D.1: Pestañas de la Interfaz Netlogo

La pestaña que nos interesa en este manual, a saber, la pestaña 'ejecutar' muestra el siguiente aspecto:

[2]En la versión Applet no existen las pestañas ejecutar, información y código; simplemente desde el navegador visualizaremos primero la interfaz que tenemos en 'ejecutar' y a continuación la información relativa a este modelo NetLogo.

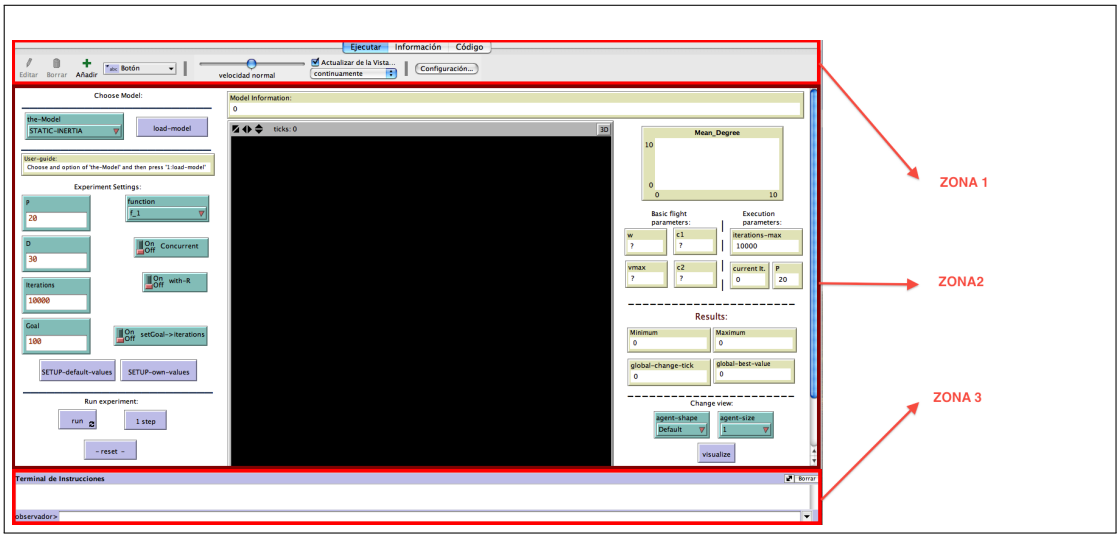


Figura D.2: Pestaña de ejecución de Netlogo

Tal como indica la Figura D.2 en esta pestaña de ejecución podemos diferenciar 3 zonas[3]. La primera, situada en la parte superior de la pantalla, permite configurar la vista que más adelante tendremos en el recuadro negro central, que es conocido como 'el mundo' y también definir la velocidad y el modo de actualizar esta vista. Si la casilla *'Actualizar de la vista'*[4] está activada los cambios en el modelo se nos mostrarán paso a paso. En cambio, si está desactivada, el mundo sólo se actualizará al finalizar la simulación. Si se selecciona '*continuamente'*, el mundo, variables y gráficas que se presentan en pantalla se actualizan en tiempo real. En cambio, si seleccionamos '*manualmente(ticks)*', la pantalla se actualizará al finalizar cada iteración (cada vez que aparezca la palabra tick en la ejecución del programa). Además, tal como muestra la Figura D.3 permite añadir nuevos elementos a la interfaz del modelo NetLogo.

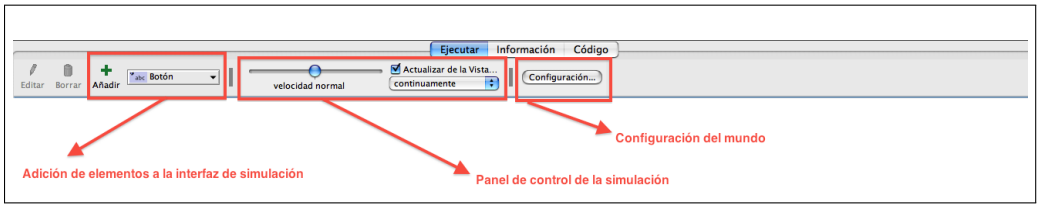


Figura D.3: Zona 1 en la pestaña de ejecución

Esta primera zona, así como la tercera zona que se encuentra en la parte inferior de la pantalla, es común para cualquier modelo de NetLogo. En esta tercera zona[5],

[3]En la versión Applet no podemos modificar el modelo NetLogo, por tanto no veremos la zona 1, pero sí podremos modificar la velocidad de la simulación mediante una barra que tendremos en la parte superior de 'el mundo'. Por defecto la velocidad figura como *velocidad normal*. Puede variarse desplazando la barra hacia la derecha (más deprisa) o hacia la izquierda (más despacio). El ordenador procesa los datos a la misma velocidad independientemente de lo elegido en esta barra, pero varía la velocidad con la que estos datos se muestran en la pantalla.

[4]Para nuestro modelo nos interesa que la vista se actualice continuamente

[5]Esta tercera zona tampoco existe en la versión Applet del modelo

que podemos ver en la Figura D.4, disponemos de un terminal donde podemos dar todo tipo de instrucciones de NetLogo desde el *observador*. Este terminal mostraría también cualquier mensaje de salida del programa si lo hubiera.

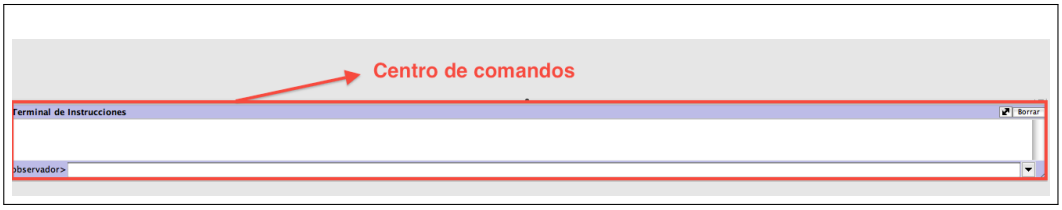


Figura D.4: Zona 3 en la pestaña de ejecución

La segunda, y más importante de todas las zonas, es la zona central que abarca todo el espacio existente entre la primera y la tercera zona y contiene todos los elementos específicos del programa, a saber, este nuestro modelo *PSOmodels*. Tal como se muestra en la Figura D.5, hemos agrupado los elementos en las siguientes secciones:

1- Choose model
2- Experiment settings
3- Run experiment
4- Parameters
5- Results
6- Change view
7- User guide
8- Model Information
9- El mundo
10- Mean Degree

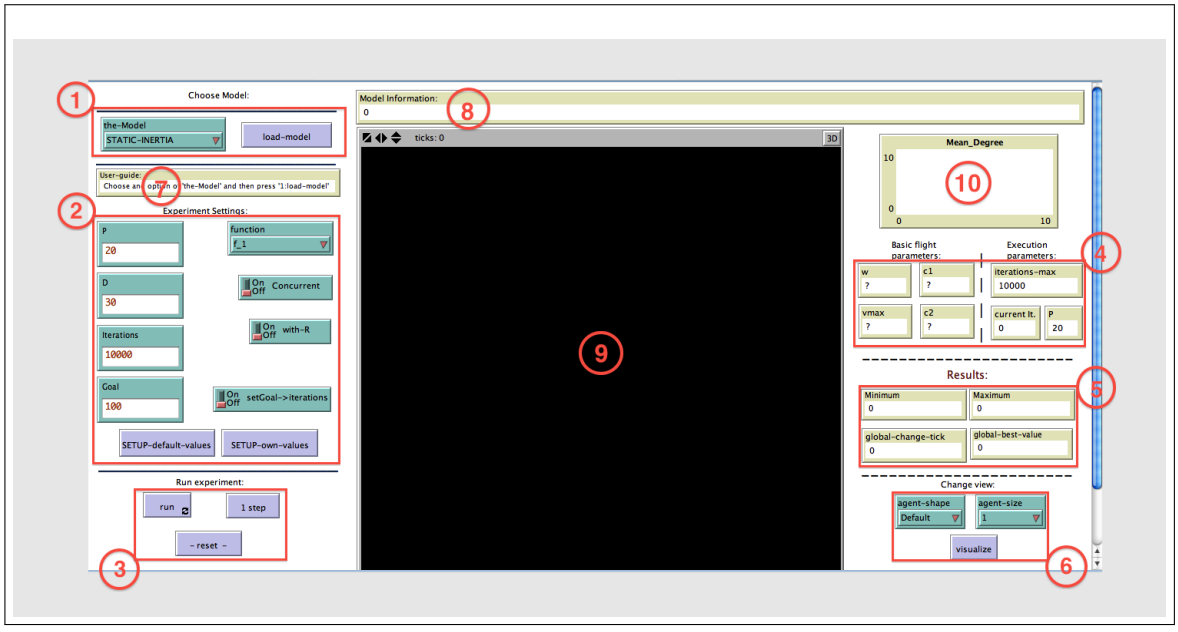


Figura D.5: Zona 2 en la pestaña de ejecución

1- Choose Model

Permite al usuario seleccionar y cargar uno de los modelos PSO implementados con la configuración de los parámetros de vuelo y ejecución que a este le correspondan, de acuerdo con numerosos artículos científicos sobre PSO.

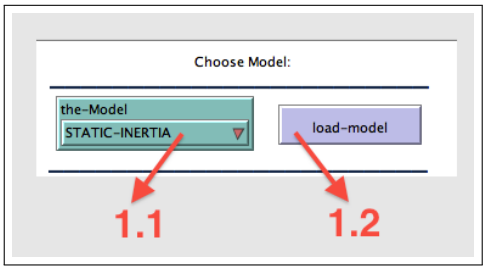


Figura D.6: Sección 1 de la zona 2: choose model

1.1- the-model:

Se trata de un *seleccionador* que despliega varias opciones (Static-Inertia, Without-Inertia, Randiw, Ldiw, PSO-G, PSO-VG y PSO-PG) para que el usuario seleccione una de ellas. Cada opción se corresponde con un modelo diferente de PSO. Su valor por defecto es el modelo: Static-Inertia.

1.2- load-model

Pulsando este *botón* el programa carga la configuración correspondiente al modelo seleccionado en el elemento 1.1.

2- Experiment Settings

Permite al usuario configurar algunos parámetros del experimento que desea realizar. Encontramos los siguientes elementos:

2.1- P:

Se trata de un elemento de *entrada* que muestra una variable de entrada del programa y permite al usuario modificarla. Esta variable P representa el tamaño de la población, es decir, la cantidad de partículas de nuestro enjambre. Posee un valor por defecto P=20.

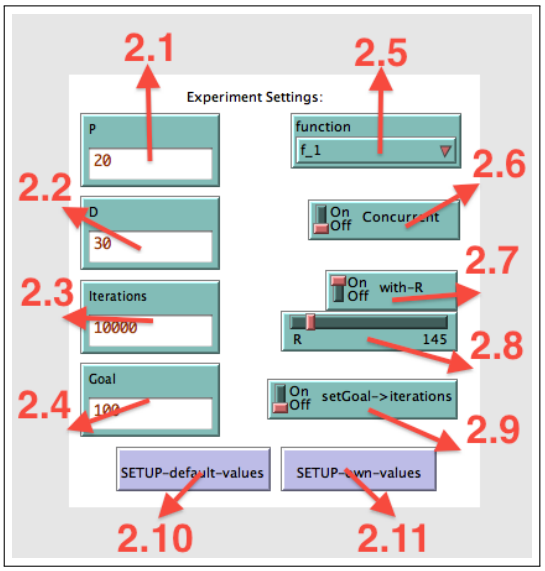


Figura D.7: Sección 2 de la zona 2: run experiment

2.2- D:

Se trata de un elemento de *entrada* que muestra una variable de entrada del programa y permite al usuario modificarla. Esta variable D representa el número de dimensiones con las que queremos realizar nuestro experimento, si bien gráficamente solo veremos representadas 2 dimensiones. Su valor por defecto depende de la selección del elemento 2.5.

2.3- Iterations:

Elemento de entrada que muestra la variable "Iterations" y permite modificarla. Esta variable define el número de iteraciones máximas permitidas en el experimento. Posee un valor por defecto de 10000.

2.4- Goal:

Se trata de un elemento de *entrada*

que muestra una variable de entrada del programa y permite al usuario modificarla. Esta variable *Goal* determina el objetivo, es decir, el criterio de parada del experimento. En este caso el valor por defecto dependerá de la selección que hagamos en el elemento 2.5, pero en todos los casos estará relacionado con el valor que debe alcanzar el '*global-best-value*', a menos que le indiquemos lo contrario mediante el elemento 2.9.

2.5- function:

Se trata de un *seleccionador* que despliega las opciones: f_0, f_1, f_2, f_3 y f_6. Estas opciones se corresponden con las funciones objetivo que usa la aplicación para realizar los experimentos. Por defecto utiliza la función f_0 si el usuario no selecciona ninguna.

2.6- Concurrent:

Mediante este *interruptor* podemos solicitar que los agentes que representan las partículas ejecuten el código de manera "concurrente". En versiones muy antiguas de NetLogo este tipo de concurrencia venía por defecto, pero desde NetLogo 4.0 (2007) las instrucciones de los agentes se ejecutan en serie por defecto, es decir, los agentes ejecutan sus comandos de uno en uno teniendo que esperar cada agente a que el anterior finalice todas sus instrucciones. Mediante el interruptor que activa la "concurrencia" se produce una concurrencia simulada a través de un mecanismo de turnos. En nuestro modelo, por defecto este interruptor se mostrará encendido para que la ejecución sea "concurrente".

2.7- with-R

Este interruptor activa la topología espacial que define el vecindario de una partícula mediante un radio.

2.8- R:

Este *deslizador* permite dar un valor a la variable de entrada R, que representa el radio dentro de unos límites establecidos.

2.9- setGoal->iterations:

Este interruptor activa una opción mediante la cual el *Goal* pasa a ser el número máximo de iteraciones. Por defecto este interruptor esta en modo *off.*

2.10- Setup-default-values

Este botón inicializa el experimento para su posterior ejecución. Los parámetros de inicialización serán los que la función objetivo seleccionada en el elemento 2.5 tenga definidas como tal.

2.11- Setup-own-values

Este botón inicializa el experimento para su posterior ejecución con los parámetros propios del modelo y los que el usuario haya definido en los elementos del 2.1 al 2.9.

3- Run Experiment

3.1- run:

Botón que ejecuta de manera continua todas las iteraciones del experimento configurado en las secciones *Choose Model* y *Experiment Settings* hasta que se satisfaga el criterio de parada.

3.2- 1 step:

Análogo al botón 3.1 pero ejecutando tan solo 1 iteración cada vez que se pulsa el botón.

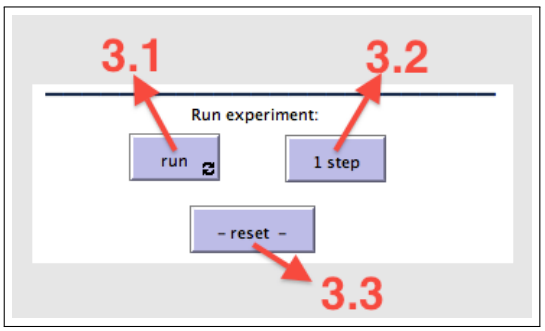


Figura D.8: Sección 3 de la zona 2: run experiment

3.3- reset:

Botón que deja el programa en su estado inicial, eliminando la configuración de los modelos y parámetros anteriores. Es importante pulsar siempre este botón antes de seleccionar un nuevo modelo de PSO.

4- Parameters

Los elementos del 4.1 al 4.4 muestran los parámetros de control que afectan a la ecuación de vuelo mientras que los del 4.5 al 4.7 muestran los parámetros relacionados con la configuración del experimento.

4.1- w:

Monitor que va mostrando el valor de la variable *w* en tiempo real. Esta variable representa el peso inercial en la ecuación de vuelo. Su valor por defecto depende del modelo PSO.

4.2- c1:

Monitor que va mostrando el valor de la variable *c1* en tiempo real. Esta variable representa el coeficiente de la componente cognitiva. Su valor por defecto depende del modelo PSO.

4.3- c2:

Monitor que va mostrando el valor de la variable *c2* en tiempo real. Esta variable representa el coeficiente de la componente social. Su valor por defecto depende del modelo PSO.

4.4- vmax:

Monitor que va mostrando el valor de la variable *vmax* en tiempo real. Esta variable limita la velocidad a un valor máximo definido por el modelo PSO.

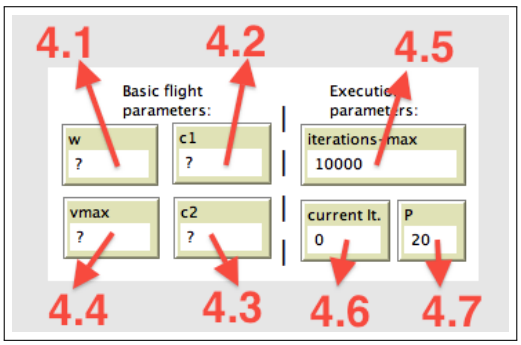


Figura D.9: Sección 4 de la zona 2: parameters

4.5- iterations max:

Monitor que muestra el valor de la variable *Iterations*. Esta variable indica el número máximo de iteraciones que ejecutará el experimento. Su valor por defecto es 10000 iteraciones.

4.6- current It.:

Monitor que va mostrando el valor de la variable T en tiempo real. Indica la iteración actual. Su valor puede incrementar uno a uno desde 0 hasta iteration max.

4.7- P:

Monitor que muestra el valor de la variable *P*. Esta variable representa el tamaño de la población y tiene asignado un valor por defecto de 20 individuos.

5- Results

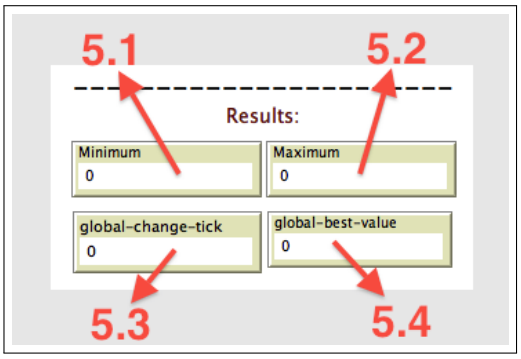


Figura D.10: Sección 5 de la zona 2: parameters

5.1- Minimum:

Monitor que va mostrando en tiempo real el valor de la variable *Minimum*. Esta variable representa el mínimo valor de la función hallado hasta el momento.

5.2- Maximum:

Monitor que va mostrando en tiempo real el valor de la variable *Maximum*. Esta variable representa el máximo valor de la función hallado hasta el momento.

5.3- global-change-tick:

Monitor que va mostrando el valor de la variable *global-change-tick* en tiempo real. Esta variable indica el número de cambios que se realizan en el gbest position y gbest value.

5.4- global-best-value:

Monitor que va mostrando en tiempo real el valor de la variable global-best-value, es decir, el óptimo global hasta el momento.

6- Change View

En esta sección se puede cambiar la vista de las partículas que se encuentran o se verán en *el mundo* (sección 9)

6.1- agent-shape:

Es un seleccionador que permite escoger la aparien-

cia de las partículas dentro del mundo, pudiendo elegir entre *default* (triángulos) y *pájaros*.

6.2- agent-size:

Es un seleccionador que permite escoger el tamaño de las partículas dentro del mundo, pudiendo elegir entre *1* (por defecto), *3* y *5*.

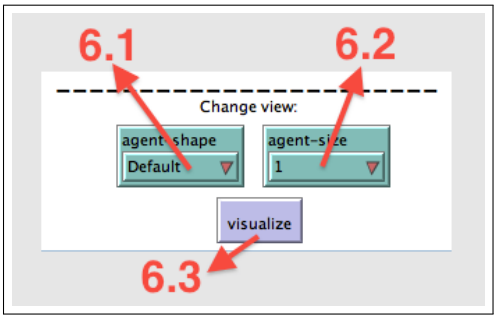


Figura D.11: Sección 6 de la zona 2: change view

6.3- visualize:

Este botón cambia el aspecto de la vista de las partículas en *el mundo* de acuerdo con lo que el usuario elige en los seleccionadores 6.1 y 6.2.

7- User guide

Este monitor va mostrando al usuario una pequeña guía de cómo proceder.

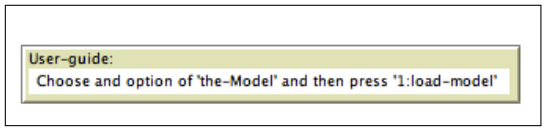


Figura D.12: Sección 7 de la zona 2: user guide

8- Model Information

Este monitor ofrece en una línea información sobre el modelo seleccionado en el elemento 1.1 y cargado mediante el elemento 1.2

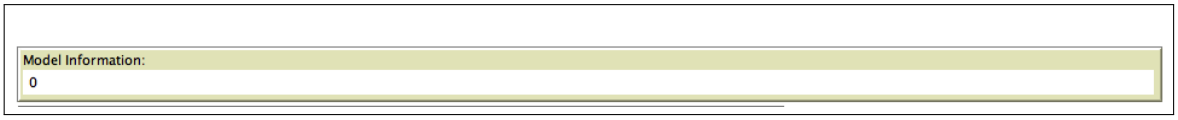


Figura D.13: Sección 8 de la zona 2: model information

9- El mundo

El "mundo" es el lugar donde se visualiza el comportamiento de los agentes que creamos en nuestro modelo. Está compuesto por una serie de patches o celdas. Se puede elegir el tamaño de cada celda y el número de celdas que hay en el mundo desde el botón "settings" de la zona 1.

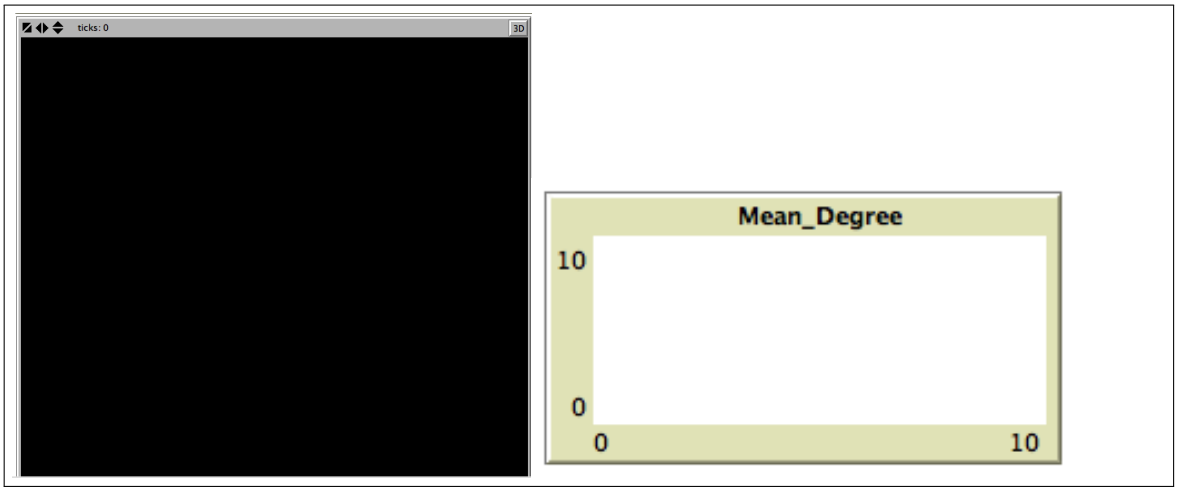


Figura D.14: Seccion 9(i) y 10 (d) de la zona 2

10- Mean Degree

Este gráfico representa el valor de una variable respecto el tiempo (ticks). En nuestro caso, la variable degree que indica el número medio de vecinos que tienen las tortugas (partículas de PSO). Como por defecto el elemento 2.7 (with-R) está apagado, el degree será constante a lo largo de toda la ejecución. Si, por el contrario, activamos el interruptor 2.7 la variable degree cambiará a lo largo de los ticks.

D.5. ¿Cómo usar PSOmodels?

Para realizar un experimento con uno de los modelos de la PSO debemos seguir los siguientes pasos: 1– Reset, 2- Load, 3- Setup, 4- Run y 5- Ver resultados.

5.1 Reset

Lo único que debemos hacer es pulsar el botón "reset" que se encuentra en la sección "Run experiment"

5.2 Load

Primero, desde el seleccionador llamado "the model" escogemos el modelo que queremos ejecutar y después pulsamos el botón "load-model". Inmediatamente podremos ver los nuevos valores que toman algunos parámetros en la sección "Fly parameters"

- Para el caso del modelo PSO, conocido como STATIC PSO, al pulsar el botón "load-model" se mostrarán por pantalla algunas preguntas que nos permiten configurar los parámetros básicos de la ecuación de vuelo. En este caso, cada vez que queremos modificar esos parámetros, podremos hacerlo volviendo a pulsar "load-model".

- Si queremos cambiar de modelo debemos pulsar Reset, antes de seleccionar y cargar un nuevo modelo.

5.3 Setup

Podemos elegir entre 2 tipos de Setup: 1 - "Setup-default-valve" y 2- "Setup-cwn-valve".

1 – Si utilizamos el "Setup-default-value" se cargaría la configuración por defecto de los parámetros del experimento. Esta configuración por defecto es distinta para cada función objetivo. Por defecto se carga la función f_0 y sus parámetros correspondientes. Si deseamos otra función debemos seleccionarla desde "function" y después pulsar "Setup-default-value".

2 – Si utilizamos "Setup-own-value" se cargará el experimento con los parámetros que hayamos definido en la sección "Experiment Settings".

Nota: Los parámetros definidos por defecto provienen del estudio de diversos científicos. Si deseamos un buen funcionamiento del algoritmo no es recomendable introducir valores que se alejen demasiado de la configuración por defecto ya que la PSO es muy sensible a la inicialización de éstos. Es aconsejable pulsar siempre "Setup-default-values" para ver los valores recomendados, después hacer ligeras modificaciones y, entonces, pulsar "setup-own-values". A continuación, se muestra la tabla con todos los valores que toma el programa por defecto:

		W					C1					C2					VMAX					P				
		F_0	F_1	F_2	F_3	F_6	F_0	F_1	F_2	F_3	F_6	F_0	F_1	F_2	F_3	F_6	F_0	F_1	F_2	F_3	F_6	F_0	F_1	F_2	F_3	F_6
STATIC INERTIA	user	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	max-pxcor	max-pxcor	max-pxcor	max-pxcor	max-pxcor	20	20	20	20	20
	basic	0,9	0,9	0,9	0,9	0,9	2	2	2	2	2	2	2	2	2	2	max-pxcor	max-pxcor	max-pxcor	max-pxcor	max-pxcor	20	20	20	20	20
	clerc	0,729	0,729	0,729	0,729	0,729	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	max-pxcor	max-pxcor	max-pxcor	max-pxcor	max-pxcor	20	20	20	20	20
	telea	0,6	0,6	0,6	0,6	0,6	1,7	1,7	1,7	1,7	1,7	1,7	1,7	1,7	1,7	1,7	10000	10000	10000	10000	10000	20	20	20	20	20
WUTHOUT INERTIA		0	0	0	0	0	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	max-pxcor	max-pxcor	max-pxcor	max-pxcor	max-pxcor	20	20	20	20	20
RANDIW		eq1*	eq1*	eq1*	eq1*	eq1*	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	max-pxcor	max-pxcor	max-pxcor	max-pxcor	max-pxcor	20	20	20	20	20
LDIW		eq2*	eq2*	eq2*	eq2*	eq2*	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	2	2	2	2	2	20	20	20	20	20
PSO-G/PSO-VG/PSO-PG		0,729	0,729	0,729	0,729	0,729	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	1,494	max-pxcor	max-pxcor	max-pxcor	max-pxcor	max-pxcor	20	20	20	20	20

eq1* --> w= 0.5 + rand()/2 eq2* --> w=w_end+(w_start - w_end)(1 - T/T_max)

		ITERATIONS					D					GOAL					CONCURRENT					WITH-R				
		F_0	F_1	F_2	F_3	F_6	F_0	F_1	F_2	F_3	F_6	F_0	F_1	F_2	F_3	F_6	F_0	F_1	F_2	F_3	F_6	F_0	F_1	F_2	F_3	F_6
STATIC PSO	user	10000	10000	10000	10000	10000	30	30	30	30	2	0,01	100	100	0,1	0,0001	on	on	on	on	on	off	off	off	off	off
	basic	10000	10000	10000	10000	10000	30	30	30	30	2	0,01	100	100	0,1	0,0001	on	on	on	on	on	off	off	off	off	off
	clerc	10000	10000	10000	10000	10000	30	30	30	30	2	0,01	100	100	0,1	0,0001	on	on	on	on	on	off	off	off	off	off
	telea	10000	10000	10000	10000	10000	30	30	30	30	2	0,01	100	100	0,1	0,0001	on	on	on	on	on	off	off	off	off	off
WUTHOUT INERTIA		10000	10000	10000	10000	10000	30	30	30	30	2	0,01	100	100	0,1	0,0001	on	on	on	on	on	off	off	off	off	off
RANDIW		10000	10000	10000	10000	10000	30	30	30	30	2	0,01	100	100	0,1	0,0001	on	on	on	on	on	off	off	off	off	off
LDIW		10000	10000	10000	10000	10000	30	30	30	30	2	0,01	100	100	0,1	0,0001	on	on	on	on	on	off	off	off	off	off
PSO-V/PSO...		10000	10000	10000	10000	10000	30	30	30	30	2	0,01	100	100	0,1	0,0001	on	on	on	on	on	off	off	off	off	off

Cuadro D.1: Valores por defecto de la aplicación

NOTA: En la versión Applet debemos pulsar "intro" cuando introducimos valores en los elementos de entrada de la sección "Experiment Settings"

5.4 Run

Para ejecutar el experimento podemos usar 2 botones: "Run" y "1 Step".

Si pulsamos el botón "Run" se ejecutarán todas las iteraciones del experimento hasta que el criterio de parada sea alcanzado y ponga fin a la ejecución.

Si pulsamos "1 Step" solo se ejecutará una iteración del experimento cada vez que pulsemos este botón. Podemos volver a repetir el mismo experimento pulsando setup y de nuevo Run o también podemos cambiar la configuración del experimento en "Experiment Setting" y después pulsar Setup y run.

Nota: La sección Change View permite cambiar la forma y tamaño de las partículas PSO que visualizamos en el mundo en cualquier momento

5.5 Ver resultados

En el elemento "el mundo" veremos durante la ejecución el vuelo de las partículas PSO en tiempo real. En la sección "Result" veremos, también en tiempo real, como cambian de valor los parámetros de interés para el experimento.

Anexo E

Conclusiones

E.1. Conclusiones Generales

Resulta fascinante cómo, observando la naturaleza y el comportamiento cooperativo de algunos animales, se pueden llegar a diseñar modelos que se aplican en las matemáticas y la informática para lograr algoritmos mas eficientes.

Las principales razones que nos han impulsado a enfocar este proyecto a la PSO son: por un lado, que la PSO es una técnica relativamente "nueva", por otro lado que es un algoritmo que en muy poco tiempo ha suscitado el interés de diversos científicos y que, a día de hoy, continua en proceso de investigación de manera muy activa y finalmente, que la PSO resulta fácil de implementar y a pesar de su "sencillez" ofrece una excelente y rápida convergencia.

Las estimaciones de tiempo que se realizaron para el diseño, el desarrollo y la investigación resultaron ser algo escasas pero gracias al margen que se planificó para posibles desviaciones de tiempo el proyecto ha sido finalizado en plazo.

E.2. Líneas de Mejora Futuras

Si bien la aplicación cumple con los requisitos que fueron definidos al inicio del proyecto, podría ser mejorado y ampliado en el futuro a partir de las siguientes propuestas:

Propuestas referentes a la aplicación en NetLogo:

1. - Implementar nuevos modelos de PSO e incorporarlos a la aplicación.
2. - Proporcionar una función que permita cambiar el tamaño del espacio de búsqueda, es decir, que cambie el tamaño del mundo según las preferencias del usuario.
3. - Añadir nuevas funciones objetivo.
4. - Programar mas topologías para los vecindarios.
5. - Mejorar la asignación de valores de los parámetros *inercia* , *c1*, *c2* y *vmax* por el usuario usando, por ejemplo, deslizadores que ya tengan un rango de valores limitado y un valor por defecto, en lugar de obtener dichos valores sin límite alguno por medio de preguntas al usuario.
6. - Programar la aplicación para la versión 3D de NetLogo.
7. - Almacenar en ficheros de texto de manera automática los resultados de los experimentos que se ejecutan .
8. - Traducir la interfaz a castellano y euskera y proporcionar una copia de la aplicación en cada uno de los 3 idiomas.

Propuestas referentes a la ontología en protégé:

1. Incorporar nuevos modelos y bibliografía.

Parte III

ESPECIFICACIONES DEL SISTEMA

Dadas las características y la envergadura del proyecto se considera que la información aportada en la Memoria es suficiente para la comprensión del mismo y que, por tanto, no es necesario incluir este documento.

Parte IV

PRESUPUESTO

Dada la ausencia de costes asumidos por parte del cliente no se considera necesario incluir este documento.

Parte V

ESTUDIOS CON ENTIDAD PROPIA

Dadas las características del proyecto no se considera necesario incluir este documento.

Bibliografía

[1] URL http://www.scholarpedia.org/article/Ant_colony_optimization.

[2] . URL http://ccl.northwestern.edu/netlogo/.

[3] . URL http://www.turtlezero.com/wiki/doku.php/.

[4] URL http://www.scholarpedia.org/article/Particle_swarm_optimization.

[5] URL http://www.particleswarm.info/Programs.html.

[6] . URL http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

[7] . URL http://protege.stanford.edu/doc/users.html#tutorials.

[8] URL http://www.red3d.com/cwr/steer/.

[9] URL http://www.swarmintelligence.org/.

[10] . URL http://librosweb.es/css/.

[11] . URL http://librosweb.es/css_avanzado/.

[12] URL http://www.omgsysml.org/.

[13] URL http://www.eclipse.org/papyrus/usersTutorials/resources/TutorialOnPapyrusUSE_d20101001.pdf.

[14] P.J. Angeline. Using selection to improve particle swarm optimization. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 84–89, 1998. doi: 10.1109/ICEC.1998.699327.

[15] A. Bandura. *Social foundations of thought and action: a social cognitive theory*. Prentice-Hall series in social learning theory. Prentice-Hall, 1986. URL http://books.google.es/books?id=HJhqAAAAMAAJ.

[16] Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In Paolo Dario, Giulio Sandini, and Patrick Aebischer, editors, *Robots and Biological Systems: Towards a New Bionics?*, volume 102 of *NATO ASI Series*, pages 703–712. Springer Berlin Heidelberg, 1993. ISBN 978-3-642-63461-1. doi: 10.1007/978-3-642-58069-7_38. URL http://dx.doi.org/10.1007/978-3-642-58069-7_38.

[17] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353 – 373, 2005. ISSN 1571-0645. doi: 10.1016/j.plrev.2005.10.001. URL http://www.sciencedirect.com/science/article/pii/S1571064505000333.

[18] Dan Bratton and Tim Blackwell. A simplified recombinant pso. *J. Artif. Evol. App.*, 2008:14:1–14:10, January 2008. ISSN 1687-6229. doi: 10.1155/2008/654184. URL http://dx.doi.org/10.1155/2008/654184.

[19] R. Brits, A.P. Engelbrecht, and F. van den Bergh. Locating multiple optima using particle swarm optimization. *Applied Mathematics and Computation*, 189(2):1859 – 1883, 2007. ISSN 0096-3003. doi: 10.1016/j.amc.2006.12.066. URL http://www.sciencedirect.com/science/article/pii/S0096300306017826.

[20] A. Carlisle and G. Dozier. An off-the-shelf PSO. In *PSO Workshop*, 2001.

[21] Huang Chongpeng, Zhang Yuling, Jiang Dingguo, and Xu Baoguo. On some non-linear decreasing inertia weight strategies in particle swarm optimization. In *Control Conference, 2007. CCC 2007. Chinese*, pages 750–753, 2007. doi: 10.1109/CHICC.2006.4347175.

[22] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1):58–73, 2002. ISSN 1089-778X. doi: 10.1109/4235.985692.

[23] Maurice Clerc. Discrete particle swarm optimization illustrated by the traveling salesman problem. *New optimization techniques in engineering*, 141:219–239, 2004.

[24] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4):28–39, 2006. ISSN 1556-603X. doi: 10.1109/MCI.2006.329691.

[25] Marco Dorigo. *Optimization, Learning and Natural Algorithms (in Italian)*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

[26] Marco Dorigo and Socha Krzysztof. An introduction to ant colony optimization. *IRIDIA Technical Report Series*, 2006.

[27] Marco Dorigo, Vittorio Maniezzo, Alberto Colorni, M Dorigo, V Maniezzo, A Colorni, et al. Positive feedback as a search strategy. 1991.

[28] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, pages 39–43, 1995. doi: 10.1109/MHS.1995.494215.

[29] R.C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 84–88 vol.1, 2000. doi: 10.1109/CEC.2000.870279.

[30] George I Evers. *An automatic regrouping mechanism to deal with stagnation in particle swarm optimization*. PhD thesis, University of Texas-Pan American, 2009.

[31] S. Goss, S. Aron, J.L. Deneubourg, and J.M. Pasteels. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12):579–581, 1989. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0024827650&partnerID=40&md5=343e32f88f6b5dc7f2bd8c3af138b341. cited By (since 1996) 257.

[32] E. Harmon-Jones and J. Mills. Cognitive dissonance: progress on a pivotal theory in social psychology. edited by eddie harmon-jones and judson mills. american psychological association distributed by the eurospan group, 1999, isbn 1-55798-565-0. *Criminal Behaviour and Mental Health*, 11(S1):S34–S37, 2001. ISSN 1471-2857. doi: 10.1002/cbm.420. URL http://dx.doi.org/10.1002/cbm.420.

[33] F. Heppner and U. Grenander. A stochastic nonlinear model for coordinated bird flocks. In E. Krasner, editor, *The ubiquity of chaos*, pages 233–238. AAAS Publications, 1990.

[34] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, 1995. doi: 10.1109/ICNN.1995.488968.

[35] J. Kennedy and R.C. Eberhart. A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 5, pages 4104–4108 vol.5, 1997. doi: 10.1109/ICSMC.1997.637339.

[36] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1671–1676, 2002. doi: 10.1109/CEC.2002.1004493.

[37] James Kennedy and Russell C. Eberhart. *Swarm intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001. ISBN 1-55860-595-9.

[38] Thiemo Krink and Morten LÃÂžvbjerg. The lifecycle model: Combining particle swarm optimisation, genetic algorithms and hillclimbers. In *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, pages 621–630, 2002.

[39] Xiaodong Li. Particle swarm optimization: an introduction and its recent developments. In *School of Computer Science and IT(2006), RMIT University*, SEAL'06, 2006.

[40] Xiaodong Li and Andries P. Engelbrecht. Particle swarm optimization: an introduction and its recent developments. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, GECCO '07, pages 3391–3414, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-698-1. doi: 10.1145/1274000.1274118. URL http://doi.acm.org/10.1145/1274000.1274118.

[41] J. J. Liang and P.N. Suganthan. Dynamic multi-swarm particle swarm optimizer. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 124–129, 2005. doi: 10.1109/SIS.2005.1501611.

[42] J. J. Liang and P.N. Suganthan. Dynamic multi-swarm particle swarm optimizer with local search. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 522–528 Vol.1, 2005. doi: 10.1109/CEC.2005.1554727.

[43] Yong ling Zheng, Long-Hua Ma, Li yan Zhang, and Ji xin Qian. On the convergence analysis and parameter selection in particle swarm optimization. In *Machine Learning and Cybernetics, 2003 International Conference on*, volume 3, pages 1802–1807 Vol.3, 2003. doi: 10.1109/ICMLC.2003.1259789.

[44] M. Lovbjerg and T. Krink. Extending particle swarm optimisers with self-organized criticality. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1588–1593, 2002. doi: 10.1109/CEC.2002.1004479.

[45] Ahmad Nickabadi, Mohammad Mehdi Ebadzadeh, and Reza Safabakhsh. A novel particle swarm optimization algorithm with adaptive inertia weight. *Applied Soft Computing*, 11(4):3658 – 3670, 2011. ISSN 1568-4946. doi: 10.1016/j.asoc.2011.01.037. URL http://www.sciencedirect.com/science/article/pii/S156849461100055X.

[46] Taher Niknam and Babak Amiri. An efficient hybrid approach based on pso, aco and k-means for cluster analysis. *Applied Soft Computing*, 10(1):183 – 197, 2010. ISSN 1568-4946. doi: 10.1016/j.asoc.2009.07.001. URL http://www.sciencedirect.com/science/article/pii/S1568494609000854.

[47] Mahamed G.H. Omran, Andries P. Engelbrecht, and Ayed Salman. Bare bones differential evolution. *European Journal of Operational Research*, 196(1):128 – 139, 2009. ISSN 0377-2217. doi: 10.1016/j.ejor.2008.02.035. URL http://www.sciencedirect.com/science/article/pii/S0377221708002440.

[48] E. Ozcan and C.K. Mohan. Particle swarm optimization: surfing the waves. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages –1944 Vol. 3, 1999. doi: 10.1109/CEC.1999.785510.

[49] Ender Ozcan and Chilukuri K Mohan. Analysis of a simple particle swarm optimization system. *Intelligent Engineering Systems Through Artificial Neural Networks*, 8:253–258, 1998.

[50] D. Parrott and Xiaodong Li. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *Evolutionary Computation, IEEE Transactions on*, 10(4):440–458, 2006. ISSN 1089-778X. doi: 10.1109/TEVC.2005.859468.

[51] K.E. Parsopoulos and M.N. Vrahatis. On the computation of all global minimizers through particle swarm optimization. *Evolutionary Computation, IEEE Transactions on*, 8(3):211–224, 2004. ISSN 1089-778X. doi: 10.1109/TEVC.2004.826076.

[52] Magnus Erik Hvass Pedersen. *Tuning & simplifying heuristical optimization*. PhD thesis, PhD thesis, University of Southampton, 2010.

[53] M.E.H. Pedersen and A.J. Chipperfield. Simplifying particle swarm optimization. *Applied Soft Computing*, 10(2):618 – 628, 2010. ISSN 1568-4946. doi: 10.1016/j.asoc.2009.08.029. URL http://www.sciencedirect.com/science/article/pii/S1568494609001549.

[54] Z.a b Qin, F.a Yu, Z.a Shi, and Y.b Wang. Adaptive inertia weight particle swarm optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4029 LNAI: 450–459, 2006. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-33746239398&partnerID=40&md5=735e5e95ea1f009867a929942dfdba3e. cited By (since 1996) 11.

[55] Juan Rada-Vilela, Mengjie Zhang, and Winston Seah. A performance study on synchronous and asynchronous updates in particle swarm optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 21–28, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0557-0. doi: 10.1145/2001576.2001581. URL http://doi.acm.org/10.1145/2001576.2001581.

[56] A. Ratnaweera, S. Halgamuge, and H.C. Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *Evolutionary Computation, IEEE Transactions on*, 8(3):240–255, 2004. ISSN 1089-778X. doi: 10.1109/TEVC.2004.826071.

[57] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, August 1987. ISSN 0097-8930. doi: 10.1145/37402.37406. URL http://doi.acm.org/10.1145/37402.37406.

[58] Rahul Roy, Satchidananda Dehuri, and Sung Bae Cho. A novel particle swarm optimization algorithm for multi-objective combinatorial optimization problem.

International Journal of Applied Metaheuristic Computing (IJAMC), 2(4):41–57, 2011.

[59] E. Safavieh, A. Gheibi, M. Abolghasemi, and A. Mohades. Particle swarm optimization with voronoi neighborhood. In *Computer Conference, 2009. CSICC 2009. 14th International CSI*, pages 397–402, 2009. doi: 10.1109/CSICC.2009.5349613.

[60] Davoud Sedighizadeh and Ellips Masehian. Particle swarm optimization methods, taxonomy and applications. *International Journal of Computer Theory and Engineering*, 1(5):1793–8201, 2009.

[61] PS Shelokar, Patrick Siarry, VK Jayaraman, and BD Kulkarni. Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied mathematics and computation*, 188(1):129–142, 2007.

[62] Yuhui Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73, 1998. doi: 10.1109/ICEC.1998.699146.

[63] P. Smolensky. Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. chapter Information processing in dynamical systems: foundations of harmony theory, pages 194–281. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL http://dl.acm.org/citation.cfm?id=104279.104290.

[64] Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, 1995.

[65] K.a Suresh, S.a Ghosh, D.a Kundu, A.a Sen, S.a Das, and A.b Abraham. Inertia-adaptive particle swarm optimizer for improved global search. volume 2, pages 253–258, 2008. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-67449124383&partnerID=40&md5=8310e199951ac9d0931f7160d1c2c53f. cited By (since 1996) 7.

[66] Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317 – 325, 2003. ISSN 0020-0190. doi: 10.1016/S0020-0190(02)00447-7. URL http://www.sciencedirect.com/science/article/pii/S0020019002004477.

[67] F. van den Bergh and A.P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8):937 – 971, 2006. ISSN 0020-0255. doi: 10.1016/j.ins.2005.02.003. URL http://www.sciencedirect.com/science/article/pii/S0020025505000630.

[68] Frans Van Den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, Pretoria, South Africa, South Africa, 2002. AAI0804353.

[69] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, 1974.

[70] Xiao-Feng Xie, Wen-Jun Zhang, and Zhi-Lian Yang. Dissipative particle swarm optimization. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1456–1461, 2002. doi: 10.1109/CEC.2002.1004457.

[71] Z. Xinchao. A perturbed particle swarm algorithm for numerical optimization. *Applied Soft Computing Journal*, 10(1):119–124, 2010. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-70350103089&partnerID=40&md5=df4fe5c85b3a3ae2191348ce78f84b31. cited By (since 1996) 47.

[72] Xin-She Yang, Suash Deb, and Simon Fong. Accelerated particle swarm optimization and support vector machine for business optimization and applications. *Networked Digital Technologies*, pages 53–66, 2011.

[73] X.S. Yang. *Nature-Inspired Metaheuristic Algorithms: Second Edition*. Luniver Press, 2011. ISBN 9781905986286. URL http://books.google.es/books?id=iVB_ETlh4ogC.

[74] K. Yasuda, A. Ide, and N. Iwasaki. Adaptive particle swarm optimization. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 2, pages 1554–1559 vol.2, 2003. doi: 10.1109/ICSMC.2003.1244633.

[75] Zhi-Hui Zhan, Jun Zhang, Yun Li, and H.S.-H. Chung. Adaptive particle swarm optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(6):1362–1381, 2009. ISSN 1083-4419. doi: 10.1109/TSMCB.2009.2015956.

[76] Wen-Jun Zhang and Xiao-Feng Xie. Depso: hybrid particle swarm with differential evolution operator. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 4, pages 3816–3821 vol.4, 2003. doi: 10.1109/ICSMC.2003.1244483.