Comparación empírica de métodos de diseño de enjambres de robots:

un estudio en simulación sobre enjambres que coordinan a otros enjambres

David Alfredo Garzón Ramos

Universidad Nacional de Colombia
Facultad de Ingeniería y Arquitectura
Departamento de Ingeniería Eléctrica, Electrónica y Computación
Manizales, Colombia
2022

Empirical comparison of methods for the design of robot swarms:

a simulation study of swarms that coordinate other swarms

David Alfredo Garzón Ramos

Universidad Nacional de Colombia
Facultad de Ingeniería y Arquitectura
Departamento de Ingeniería Eléctrica, Electrónica y Computación
Manizales, Colombia
2022

Comparación empírica de métodos de diseño de enjambres de robots:

un estudio en simulación sobre enjambres que coordinan a otros enjambres

David Alfredo Garzón Ramos

Tesis presentada como requisito parcial para optar al título de:
Magister en Ingeniería - Automatización Industrial

Director:
Ph.D., Carlos Vargas Hernández

Línea de Investigación:
Diseño de comportamientos colectivos para enjambres de robots

Universidad Nacional de Colombia
Facultad de Ingeniería y Arquitectura
Departamento de Ingeniería Eléctrica, Electrónica y Computación
Manizales, Colombia
2022

A mi familia.

In fact, the colony is the real organism, not the individual.

—Daniel Suarez, Kill Decision.

Agradecimientos

Los primeros agradecimientos los dedico a mi familia.

También, quisiera agradecer al Dr. Carlos Vargas Hernández, director de esta tesis, por su paciencia, apoyo, confianza y orientación en el desarrollo de la investigación. A la Facultad de Ingeniería y Arquitectura de la Universidad Nacional de Colombia, sede Manizales, por brindarme la oportunidad de concluir estos estudios de maestría. De forma especial, quisiera agradecer al Dr. Mauro Birattari, cuyo trabajo ha inspirado muchas de las ideas presentadas en esta tesis; y a IRIDIA, el laboratorio para el desarrollo de la Inteligencia Artificial de la Université libre de Bruxelles, por brindarme los recursos para llevar acabo la investigación. Así mismo, quisiera a gradecer a Jazmín, Juan Pablo y Gabriel, del grupo de estudio en robótica de enjambres de la Universidad de Nariño, por participar en los experimentos. Finalmente, sinceros agradecimientos todas esas personas que han contribuido de forma directa o indirecta en la realización de este trabajo.

Esta investigación se desarrolló con el apoyo de la Universidad Nacional de Colombia, la Université libre de Bruxelles e IRIDIA. El desarrollo de esta tesis ha contado con financiación del proyecto DEMIURGE (ERC grant agreement No 681872, PI Mauro Birattari) y también con financiación del Ministerio Colombiano de Ciencia, Tecnología e Innovación–Minciencias.

Resumen

Esta tesis presenta una comparación empírica de métodos de diseño de comportamientos colectivos para enjambres de robots que deben coordinar a otros enjambres. En este estudio, los métodos de diseño producen software de control sin ser provistos con información explícita sobre el comportamiento de los enjambres que deben ser coordinados—esta información debe ser obtenida durante el proceso de diseño. Los métodos considerados en la comparación son `Pistacchio`, `EvoCMY`, `C-Human`, `R-Walk`. `Pistacchio` es un método automático modular, perteneciente a la familia de métodos AutoMoDe. `EvoCMY` es un método automático basado en neuroevolución. La eficiencia de los métodos automáticos fue comparada con `C-Human`, un método de diseño manual en donde un grupo de diseñadores producen el software de control para los robots. `Pistacchio`, `EvoCMY` y `C-Human` fueron comparados con `R-Walk`, una implementación de movimiento aleatorio que sirve como línea base. `Pistacchio` y `EvoCMY` se presentan por primera vez en esta tesis, y son una contribución de la investigación. Los experimentos llevados a cabo con `Pistacchio` y `EvoCMY` permitieron determinar que los métodos de diseño automático son efectivos en (i) la identificación de posibles formas de interacción entre el enjambre que coordina y el enjambre que es coordinado; y en (ii) el aprovechamiento de estas formas de interacción para ejecutar misiones en donde los dos enjambres operan de forma coordinada. La comparación de los métodos se llevó a cabo empleando ARGoS3, un simulador especializado en robótica de enjambres, y tomando como plataforma de referencia al robot e-puck. El diseño mediante métodos automáticos resultó significativamente más eficiente que el diseño manual en las misiones propuestas.

Palabras clave: Robótica de enjambres, diseño automático, diseño manual, AutoMoDe, neuroevolución, coordinación entre enjambres.

Abstract

This thesis presents an empirical comparison of methods for the design of collective behaviors for a robot swarm that must coordinate a second robot swarm. In this study, the design methods produce control software without being provided with explicit information about the behavior of the swarm that must be coordinated—in all cases, this information must be obtained during the design process. The methods considered in the study are `Pistacchio`, `EvoCMY`, `C-Human`, `R-Walk`. `Pistacchio` is an automatic modular method that belongs to the family of methods AutoMoDe. `EvoCMY` is an automatic method based on neuro-evolution. The performance of the automatic methods was compared with the performance of `C-Human`, a manual design method in which a group of human designers produces the control software for the robots. `Pistacchio`, `EvoCMY` and `C-Human` were compared against `R-Walk`, an implementation of a random walk that serves as a baseline. `Pistacchio` and `EvoCMY` are presented for the first time in this thesis, and they are an original contribution. Experiments conducted with `Pistacchio` and `EvoCMY` showed that automatic design methods are effective in (i) identifying interaction dynamics between the swarm that coordinates and the swarm that is being coordinated; and in (ii) using these dynamics for addressing missions in which the two robot swarms must act in a coordinated manner. The design methods were compared in simulation, using the swarm-dedicated simulator ARGoS3 and a simulated model of the e-puck robot. The automatic design approach performed significantly better than the manual design in the proposed missions.

Keywords: Swarm robotics, automatic design, manual design, AutoMoDe, neuroevolution, coordination between robot swarms.

Contenido

Agradecimientos ix

Resumen xi

Lista de figuras xvii

Lista de tablas 1

1. Introducción 2
1.1. Justificación de la investigación . . . 5
1.2. Planteamiento del problema . . . 6
1.3. Objetivo general . . . 7
1.3.1. Objetivos específicos . . . 7
1.4. Productos de la investigación . . . 7

2. Marco teórico y trabajos relacionados 8
2.1. Enjambres de robots y robótica de enjambres . . . 8
2.1.1. Características de un enjambre de robots . . . 9
2.1.2. Propiedades de un enjambre de robots . . . 10
2.2. Diseño de enjambres de robots . . . 11
2.2.1. Diseño manual de comportamientos colectivos . . . 12
2.2.2. Métodos basados en optimización . . . 13
2.2.3. Métodos de diseño relevantes para esta tesis . . . 14
2.3. Coordinación entre grupos de robots . . . 14

3. Metodología 16
3.1. Robot y entorno de simulación . . . 16
3.1.1. Simulador . . . 19
3.2. Método I: AutoMoDe-`Pistacchio` . . . 19
3.2.1. Módulos de software . . . 20
3.2.2. Diseño automático de software de control . . . 23
3.3. Método II: `EvoCMY` . . . 24
3.4. Método III: `C-Human` . . . 25
3.5. Método IV: `R-Walk` . . . 29

4. Diseño experimental **31**
4.1. Tipos de enjambres en los experimentos: activo y pasivo 31
4.1.1. Comportamientos del enjambre pasivo 32
4.2. Misiones 35
4.2.1. Agrupamiento 36
4.2.2. Dispersión 36
4.2.3. Pastoreo 38
4.2.4. Segregación 38
4.3. Protocolo 39
4.3.1. Estadística 39

5. Resultados y discusión **41**
5.1. Misión I: Agrupamiento 41
5.1.1. `Pistacchio` 42
5.1.2. `EvoCMY` 43
5.1.3. `C-Human` 43
5.1.4. `R-Walk` 44
5.2. Misión II: Dispersión 44
5.2.1. `Pistacchio` 44
5.2.2. `EvoCMY` 44
5.2.3. `C-Human` 45
5.2.4. `R-Walk` 46
5.3. Misión III: Pastoreo 46
5.3.1. `Pistacchio` 46
5.3.2. `EvoCMY` 47
5.3.3. `C-Human` 48
5.3.4. `R-Walk` 48
5.4. Misión IV: Segregación 49
5.4.1. `Pistacchio` 50
5.4.2. `EvoCMY` 50
5.4.3. `C-Human` 51
5.4.4. `R-Walk` 51
5.5. Resultados agregados 51
5.5.1. Test de Friedman 52
5.6. Consideraciones finales 54

6. Conclusiones **58**
6.1. Trabajo futuro 60

A. Anexo: material suplementario **61**

B. Anexo: instrucciones y cuestionario empleado en los experimentos de C-Human 62

Bibliografía 72

Lista de Figuras

3-1. Robot e-puck: modelo real y simulación del robot en ARGoS3 17

4-1. Maquinas de estados de los comportamientos *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión* del enjambre pasivo. 33
4-2. Misiones consideradas en los experimentos: AGRUPAMIENTO, DISPERSIÓN, PASTOREO y SEGREGACIÓN. 37

5-1. Resultados obtenidos en AGRUPAMIENTO. 42
5-2. Resultados obtenidos en DISPERSIÓN. 45
5-3. Resultados obtenidos en PASTOREO. 47
5-4. Resultados obtenidos en SEGREGACIÓN. 49
5-5. Resultados agregados y test de Friedman. 53

Lista de Tablas

3-1. Especificaciones del robot - RM 3.3 18
3-2. Módulos de software integrados en `Pistacchio` 22
3-3. Topologia de la red neuronal artificial y parámetros de evolución de `EvoCMY` . 26
3-4. Módulos de software disponibles en `C-Human`, y protocolo de experimentación 28
3-5. Módulo de software integrado en `R-Walk` 30

4-1. Módulos de software empleados en los comportamientos colectivos del enjambre pasivo. 34

1. Introducción

¿Es posible diseñar a un grupo de robots que coordina a otro grupo de robots, sin conocer cómo estos últimos han sido programados? Esta tesis aborda este cuestionamiento desde el punto de vista de la robótica de enjambres; en donde la *coordinación* se define como la habilidad que tiene un grupo de robots para actuar en respuesta a las acciones de sus pares, o para influenciar dichas acciones. A partir de un estudio empírico, se evalúa si métodos existentes de diseño manual y automático de comportamientos colectivos son efectivos en la producción de software de control para un enjambre que coordina a otro. El desafío en este estudio es que, antes de empezar la producción del software de control, el método de diseño no es provisto con información de cómo opera el segundo enjambre. Entonces, los métodos de diseño bajo estudio deben: (i) descubrir el tipo de interacciones que pueden surgir entre los dos grupos de robots; y, (ii) emplear estas interacciones para ejecutar misiones de forma eficiente.

En este capítulo introductorio se resalta el contexto, aspectos principales de la metodología, y se resumen los resultados principales. Más adelante, se presenta la justificación del trabajo y el planteamiento del problema abordado en la investigación. Al finalizar, se formalizan los objetivos y se listan los productos resultantes de la tesis.

La robótica de enjambres [1, 2] es la disciplina que estudia las formaciones auto-organizadas de grandes grupos de robots que operan de forma colectiva para alcanzar un objetivo en común. Estos grupos de robots se denominan enjambres de robots, y se caracterizan por presentar comportamientos colectivos similares a aquellos observados en animales sociales [1]. Los enjambres de robots operan de forma distribuida y sin el control de entes externos. El comportamiento colectivo del enjambre emerge de las interacciones locales entre robots, y de los robots con el entorno. Estas características los dotan con propiedades de interés como escalabilidad, flexibilidad, y robustez [3].

El estudio de sistemas basados en robótica de enjambres es relativamente reciente: las investigaciones fundacionales no datan más allá de hace dos décadas [1, 2]. Los trabajos más relevantes del campo tienen un enfoque científico, más que ingenieril. Es decir, la mayor parte de la literatura se enfoca en buscar metodologías de diseño y validación de comportamientos de colectivos, y en estudiar sus propiedades [4]; aunque es posible tratar de esgrimir algunas aplicaciones futuras en entornos reales [5–7].

El diseño del software de control para robots que operan dentro de un enjambre es particularmente desafiante: la tarea que los robots deben ejecutar se define de forma global para todo

el grupo, pero la programación de los robots se debe hacer de forma individual. La dificultad radica en que no existe una metodología para determinar lo que el robot debe hacer de forma individual, para que un comportamiento colectivo determinado emerja [8]. Para solucionar esta problemática, estudios recientes se han enfocado en la producción automática de software de control para enjambres de robots mediante métodos de optimización [9–12]. En esta metodología, el diseño del software de control se traduce en un proceso de optimización que, a partir de la definición de una tarea específica para el enjambre, explora posibles soluciones de control y escoge la de mayor eficiencia de acuerdo con una métrica asociada a la tarea. Esta línea de investigación lleva a plantear los siguientes cuestionamientos: ¿qué relación existe entre la plataforma robótica, la tarea que debe cumplir el enjambre, y el método de diseño automático? [13–15] ¿qué algoritmos de optimización y qué arquitecturas de control pueden emplearse en estos procesos? [16–18] ¿qué tan eficientes son estos métodos en comparación a métodos de diseño manual? [19–21] ¿pueden estos métodos ser empleados para diseñar el comportamiento colectivo de enjambres que interactúan con otros robots?

Esta tesis aborda dichas preguntas de investigación a través de un estudio empírico desarrollado con simulaciones. En particular, la tesis se enfoca en aquellas preguntas relacionadas con la eficiencia que diferentes métodos de diseño pueden alcanzar cuando diseñan comportamientos colectivos para grupos de enjambres de robots que operan de forma conjunta. Para esto, se ha concebido un marco experimental en el que dos grupos de enjambres interactúan en un mismo escenario, y deben ejecutar una misión en común. En este marco, un primer *enjambre activo* coordina las acciones de un segundo *enjambre pasivo*. El enjambre activo es el objeto de diseño para el cual los diferentes métodos producen software de control. El enjambre pasivo tiene comportamientos predefinidos, que son desconocidos para los métodos de diseño. El marco experimental se concibe con la hipótesis de que los métodos de diseño de enjambres de robots, de enfoque automático o manual, son viables para determinar el comportamiento del enjambre pasivo y pueden ser empleados para diseñar enjambres de robots activos que puedan coordinarlo. En otras palabras, se espera que estos métodos puedan diseñar comportamientos colectivos en donde el enjambre activo aprovecha el comportamiento predefinido del enjambre pasivo para ejecutar una misión de forma eficiente.

Para poner a prueba esta hipótesis se ha llevado a cabo una campaña experimental que compara diferentes clases de métodos de diseño. Para empezar, se han implementado dos métodos de diseño automático de enjambres de robots: `Pistacchio` y `EvoCMY`. `Pistacchio` es un método de diseño modular de familia de métodos AutoMoDe [11, 22] que construye software de control mediante la creación de maquinas de estado. `EvoCMY` es un método de diseño que funciona *via* neuroevolución [23], es decir, que produce software de control en la forma de redes neuronales artificiales. Los resultados obtenidos en la experimentación con estos métodos son contrastados con resultados obtenidos mediante diseño manual. En el caso del diseño manual, se han llevado a cabo experimentos con diseñadores humanos que producen software de control siguiendo el método de diseño `C-Human`; propuesto anteriormente

como una opción para comparar la efectividad de algunos métodos de diseño automático [19]. La línea base sobre la cual se compara la eficiencia de los tres métodos es `R-Walk`; una implementación de movimiento aleatorio. La campaña experimental se desarrolla con cuatro misiones en donde la eficiencia del enjambre activo está condicionada por su habilidad para coordinar las acciones del enjambre pasivo. Las misiones se desarrollan en tres casos de estudio: (i) el enjambre pasivo puede ser estimulado y ser atraído mediante señales de colores; (ii) el enjambre pasivo puede ser estimulado y ser repelido mediante señales de colores; y (iii) una combinación de comportamientos de atracción y repulsión. *A priori*, se esperaba que todos los métodos de diseño, a excepción de `R-Walk`, pudieran producir software de control para que los robots activos emplearan estímulos adecuados para interactuar con los robots pasivos. Los experimentos se ejecutan en simulación empleando el simulador multi-robot ARGoS3 [24], y los robots de ambos enjambres corresponden a un modelo simulado del robot e-puck [25].

Los resultados muestran que los métodos automáticos son efectivos en la identificación de comportamientos colectivos que aprovechan los comportamientos predefinidos de los robots pasivos. Es decir, con diseño automático, los robots activos coordinan eficientemente las acciones del enjambre pasivo a través de la emisión de señales de colores. Por el contrario, el diseño manual se mostró ineficiente en la identificación de los comportamientos del enjambre pasivo, y por tanto, la coordinación entre los enjambres no es satisfactoria en la mayoría de los casos. El análisis cuantitativo de los resultados permitió determinar que los métodos de diseño automático son significativamente mejores que el diseño manual en los experimentos planteados, y que el diseño manual es a su vez, significativamente mejor que la línea base. A partir de esta tesis se logra establecer que el diseño automático es una herramienta viable para realizar enjambres de robots que deben coordinar a otros enjambres, aún cuando no se conozca la forma en que los segundos han sido programados.

En la siguiente parte de la introducción se presenta la justificación de la investigación, el problema abordado, los objetivos del trabajo y productos resultantes. Más adelante, al finalizar el capítulo, la tesis se desarrolla de la siguiente forma: en el Capítulo 2 se presenta el estado del arte; el Capítulo 3 detalla `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk`—los métodos bajo estudio; el protocolo experimental, los tipos de enjambres de robots considerados, y las misiones para las cuales se diseña software de control se describen en el Capítulo 4; el Capítulo 5 presenta los resultados y la discusión; y, para terminar, el Capítulo 6 cierra esta tesis con algunas consideraciones generales y líneas de trabajo futuro. La parte final de la tesis se dedica a los anexos y referencias bibliográficas.

1.1. Justificación de la investigación

La revista *Science Robotics* recientemente presentó diez grandes retos para la robótica de la siguiente década [26]. Estos retos esgrimen líneas de investigación que permitirían el desarrollo de sistemas y plataformas robóticas esenciales para la realización de nuevos descubrimientos científicos. Uno de estos grandes retos es el diseño y despliegue de enjambres de robots. La premisa es que robots simples, de bajo costo y con características modulares, pueden adaptarse y auto-organizarse en equipos para ejecutar diferentes tipos de misiones. La investigación en robótica de enjambres, en parte, busca desarrollar sistemas multi-robot que sean tan efectivos como robots clásicos de arquitectura monolítica, que tienen alta complejidad y costo, y que normalmente se construyen para ejecutar una única tarea—perdiendo así flexibilidad. Los enjambres de robots operan mediante comportamientos colectivos, en dónde la coordinación y la cooperación emergen de las interacciones individuales entre robots, y entre robots y el entorno [3]. A día de hoy, aún no existe una metodología general para diseñar el comportamiento colectivo para enjambres de robots. Igualmente, no existen métodos para programar individualmente a los robots de un enjambre para que, cuando estos actúen de forma conjunta, puedan ejecutar las misiones definidas por el diseñador. Esta razón ha motivado a la comunidad científica a desarrollar métodos de diseño automático que se alinean con conceptos de aprendizaje máquina—en donde algoritmos de optimización son la base para el desarrollo de sistemas inteligentes [27]. El objetivo fundamental de esta línea de investigación es que, en un futuro, usuarios de enjambres de robots puedan describir misiones que uno o varios enjambres deben ejecutar de forma simultánea. Y que luego, esta misión sea el punto de partida para que un sistema inteligente produzca de forma automática el software de control que permita a los robots operar de forma coordinada en el entorno de trabajo [9, 28]. Esta metodología tiene el potencial de reducir la necesidad de intervención manual y el alto grado de experiencia que se requiere hoy para diseñar y operar un enjambre de robots. De forma general, el uso de enjambres de robots se plantea una como una solución prometedora para abordar tareas difícilmente alcanzables para robots clásicos y monolíticos, como pueden ser, la atención y cosechado de cultivos extensivos, el despliegue de sistemas de información y la actuación en escenarios de desastre [5–7, 29].

1.2. Planteamiento del problema

El enfoque tradicional de diseño de comportamientos colectivos para enjambres de robots es el diseño manual. Un diseñador define el comportamiento colectivo que quiere alcanzar y luego sigue un proceso de iteración por ensayo y error para producir el software de control. Este trabajo es laborioso y demanda un conocimiento previo del funcionamiento de los robots y la intuición del diseñador para producir, modificar y evaluar el software de control hasta alcanzar el resultado deseado. Con suficiente tiempo y recursos, el diseñador obtiene el comportamiento colectivo que busca, pero difícilmente puede asegurarse de que es tan eficiente como podría ser. Como alternativa a este proceso, se han propuesto métodos basados en optimización, y en particular, métodos que operan de forma automática. Es decir, aquellos en los que una vez definida la misión del enjambre, el diseñador no interviene en ninguna de las etapas del proceso de diseño.

Actualmente existen dos enfoques principales para el diseño automático de software de control para enjambres de robots: el diseño modular [11, 22] y el diseño por neuroevolución [23]. Existe muy poca literatura que compara estos métodos bajo un mismo diseño experimental, y aún menos la que compara su eficiencia con métodos de diseño manual. En todos los casos, los estudios se han enfocado en el diseño de comportamientos colectivos para enjambres homogéneos—es decir, enjambres en donde todos los robots operan con el mismo software de control. Existe un vacío de conocimiento sobre cómo estos métodos se pueden desempeñar cuando los enjambres deben compartir su entorno con otros grupos de robots. Un problema que se torna importante al considerar un posible futuro en dónde varios grupos de robots con características distintas deben ejecutar tareas de forma simultánea. Por ejemplo, este podría ser el caso de grupos de robots para agricultura que deben operar en cultivos mixtos, pero en donde cada grupo de robots está especializado en la cosecha de un tipo vegetal. Típicamente, el diseño de software de control en investigaciones relacionadas [30–32] se lleva a cabo de forma manual. El diseño de enjambres de robots que pueden coordinar las acciones de otro enjambre, el cual pudo ser desplegado con interioridad en el entorno de trabajo, ha sido poco abortado. La falta de comparación de los métodos de diseño manual y automático en este tipo de escenarios no ha permitido establecer de forma clara si los métodos de diseño automático son viables para diseñar comportamientos colectivos para enjambres que se coordinan con otros grupos de robots. Esta tesis pretende contribuir a esclarecer esta cuestión.

1.3. Objetivo general

Concebir un marco experimental para estudiar el comportamiento de enjambres de robots que coordinan a otros enjambres, y emplearlo para comparar métodos de diseño manual y automático de comportamientos colectivos.

1.3.1. Objetivos específicos

1. Concebir un marco experimental para estudiar el diseño de comportamientos colectivos para grupos de enjambres de robots que operan de forma conjunta.
2. Implementar métodos para la producción automática de software de control para enjambres de robots que deben coordinar las acciones de otros enjambres.
3. Diseñar misiones y casos de estudio apropiados para evaluar la eficiencia de los métodos de producción de software de control.
4. Conducir experimentos en el diseño automático de comportamientos colectivos para enjambres de robots.
5. Comparar la eficiencia de métodos de diseño automático y métodos estándar de diseño manual de enjambres de robots.

1.4. Productos de la investigación

Los productos resultantes de esta investigación son:

1. Dos implementaciones de métodos de diseño de software de control para enjambres de robots—`Pistacchio` y `EvoCMY`. Las implementaciones se proveen a la comunidad académica en forma de software libre, gratuito y con código abierto.
2. Un marco experimental para el estudio de comportamientos colectivos en donde grupos diferentes de enjambres de robots deben ejecutar tareas de forma conjunta.
3. Un artículo científico (en preparación) para ser sometido a revisión.
4. El documento de tesis.

2. Marco teórico y trabajos relacionados

Este capítulo introduce conceptos importantes para el desarrollo de la tesis, y a su vez, resalta trabajos relevantes en cada caso. Para una revisión exhaustiva sobre robótica de enjambres, se referencia el libro sobre robótica de enjambres de Heiko Hamann [33] y otras revisiones de la literatura relevantes [4, 5, 7, 29, 34, 35]. A continuación se introducen nociones importantes sobre robótica de enjambres y las características y propiedades principales de los enjambres de robots. Luego se listan algunas metodologías de diseño de comportamientos colectivos para enjambres. Después, el capítulo centra su atención el diseño de enjambres de robots mediante métodos de optimización. Finalmente, la última parte del capítulo relaciona trabajos en donde enjambres de robots coordinan a otros enjambres.

2.1. Enjambres de robots y robótica de enjambres

Un enjambre de robots [2] es un grupo grande de robots relativamente simples que, cuando actúan juntos, pueden realizar tareas que escapan de las capacidades individuales de cada robot [3]. Algunos autores, como Gerardo Beni [1], los han definido también como grupos de robots no inteligentes, que agrupados, forman un robot inteligente. En esta definición, la noción de inteligencia se limita a referir la capacidad de procesamiento de información de los robots del enjambre—siendo estos en su mayoría de naturaleza reactiva y sin capacidad de inferencia o planificación. La característica particular de los enjambres de robots es que su comportamiento colectivo emerge de las interacciones locales entre individuos, y entre los individuos y su entorno.

A partir del interés científico en el diseño de estos sistemas se consolida la robótica de enjambres [35]: disciplina dedicada al estudio del diseño, producción, validación y aplicación de grupos de robots que operan con inteligencia de enjambres [36]. Originalmente, la robótica de enjambres nació de la necesidad de validar modelos sobre comportamientos de animales sociales [6]—trabajos propuestos desde investigaciones en biología. Sin embargo, una gran parte del trabajo reciente en robótica de enjambres se ha enfocado en dejar de lado la emulación de comportamientos de sistemas biológicos, y se ha enfocado en desarrollar herramientas ingenieriles que permitan crear enjambres de robots de forma sistemática, y asegurando ciertos niveles de eficiencia [4]. Nuevos trabajos en robótica de enjambres buscan definir reglas y metodologías para diseñar enjambres de robots que reduzcan el esfuerzo requerido para realizar y operar estos sistemas [37].

Existen dos perspectivas para analizar el comportamiento de un enjambre: a nivel microscópico y a nivel macroscópico [33]. El nivel microscópico se enfoca en estudiar y definir los comportamientos individuales de los robots. En este caso, el análisis se centra en las capacidades de cada robot para procesar y actuar ante la información local que obtiene del entorno. En estos análisis existe una incertidumbre sobre el posible comportamiento colectivo del enjambre—dado que se centra en el individuo. El nivel microscópico está limitado a estudiar las interacciones simples entre robots.
En contraposición existen los análisis macroscópicos. En este nivel se pone mayor interés en el análisis del comportamiento colectivo de los robots [38]. La información global del entorno toma relevancia para establecer cómo los estímulos generados por el espacio de operación de los robots influencian el comportamiento colectivo. El reto en este tipo de análisis está en la dificultad para señalar cuáles comportamientos individuales contribuyen (directamente o indirectamente) a la ejecución de una tarea. El enfoque macroscópico se centra en la operación del enjambre como una unidad que puede ser empleada para ejecutar misiones específicas.

2.1.1. Características de un enjambre de robots

Los enjambres de robots adoptan características [3] propias de los sistemas basados en inteligencia de enjambres [36]. Estos operan sin necesidad de un control centralizado o una infraestructura externa. Como se mencionó anteriormente, el comportamiento de un enjambre de robots nace de las interacciones locales de los individuos, y por tanto, no necesitan un líder o un sistema centralizado que guíe las acciones de los robots. Un enjambre de robots puede auto-organizarse, es decir, puede definir roles y distribuir tareas entre sus individuos de forma autónoma. Los robots de un enjambre solo cuentan con información local sobre su entorno, sin embargo, pueden contribuir al desarrollo de tareas complejas. Por ejemplo, Ferrante *et al.* [39] experimentaron con la aparición autónoma de roles en un enjambre de robots que divide una misión compleja en un conjunto de actividades más simples; división que aumenta la eficiencia del enjambre. Dado que los robots de un enjambre operan solo con información local, no requieren de una infraestructura externa u otros sistemas que provean de información a los robots—cómo por ejemplo, sistemas externos que asistan su localización y comunicaciones.
Típicamente, los robots de un enjambre son homogéneos—es decir, los robots comparten la misma morfología física, son construidos con el mismo hardware y operan con el mismo software de control. En pocos casos se han considerado sistemas cuasi-homogéneos, en donde los robots tienen capacidades físicas similares pero no idénticas, o en donde operan con software de control que puede variar entre los robots. Por ejemplo, Jones *et al.* [40] experimentaron con el diseño automático de comportamientos colectivos para un enjambre de robots en donde cada individuo producía su propio software de control. Al final de la misión, cada robot

del enjambre portaba un software de control que difería de el de sus pares. De forma similar, en muy pocos casos se han realizado experimentos con enjambres de robots con morfologías completamente distintas. El trabajo más relevante en este sentido es el realizado por Dorigo *et al.* [31] en el proyecto Swarmanoid. En este proyecto, los investigadores crearon un enjambre con tres tipos de robots: (i) aéreos, con (ii) ruedas, y (iii) con garras; que de forma auto-organizada podían cooperar para ejecutar tareas.

Diferencias con otros sistemas multi-robot

Un enjambre de robots es un caso particular de los sistemas que tradicionalmente se consideran como sistemas multi-robot [41, 42]. En el contexto de esta tesis, la diferenciación entre los dos tipos de sistemas es una forma conveniente de organizar la literatura para hacer comparaciones justas y fijar expectaciones adecuadas sobre el posible rendimiento puede presentar un sistema multi-robot tradicional o un ejambre de robots. Las características descritas en las secciones anteriores son las hipótesis de partida sobre las cuales se diseñan los enjambres de robots en esta investigación. Entonces, se deja de lado posibles comparaciones con sistemas multi-robot (o enjambres) que cuenten con: (i) información global sobre su entorno; (ii) reglas de interacción entre robots complejas; (iii) comunicación explicita y asegurada entre los individuos; (iii) asignación explícita de roles e identidades; (iv) información sobre el tamaño del enjambre al que hacen parte o del enjambre que deben coordinar; y (v) control o coordinación centralizada.

2.1.2. Propiedades de un enjambre de robots

Las características mencionadas anteriormente permiten asegurar (en cierto grado) que un enjambre de robots cuenta con tres propiedades deseables: robustez, escalabilidad y flexibilidad.

La robustez de un enjambre se asocia con su capacidad para tolerar fallos. Un enjambre de robots típicamente cuenta con un gran número de individuos y es un sistema altamente redundante. Si un robot en el enjambre falla, la operación del sistema no se ve severamente impactada. También, gracias a que el enjambre opera sin la necesidad de un robot líder o una infraestructura externa, no existe un punto único de fallo. Christensen *et al.* [43] condujeron experimentos en la detección y reparación autónoma de robots de un enjambre que fallan durante la ejecución de una misión.

La segunda propiedad deseable de un enjambre es la escalabilidad. Esto es, la capacidad de añadir y remover unidades del enjambre sin tener que definir las reglas de interacción entre los individuos. En otras palabras, que es posible modificar el número de robots en el enjambre sin que esto afecte considerablemente su capacidad de operación. Salman *et al.* [15] presentaron experimentos en dónde un método de diseño automático escogía el número adecuado de robots según la tarea a ejecutar y según el costo financiero para producir dicho

enjambre. La escalabilidad de un enjambre está estrechamente relacionada con la localidad de la información con la que cuentan los individuos. Dado que cada robot solo interactúa con pares que se encuentran en una vecindad cercana, un robot no se verá afectado por las acciones (o la inacción) de robots que no se encuentran en su rango de percepción. Es decir, este no se verá afectado por la aparición y desaparición de robots en lugares del enjambre que no percibe directamente. Sin embargo, la escalabilidad del enjambre no es absoluta; está condicionada por su densidad [33]. Si el espacio en que operan los robots es limitado, la operación del enjambre se afectará si el número de robots cambia drásticamente. Cuando hay una densidad alta, el número de robots que entran en la vecindad de cada individuo aumenta, y por tanto, también aumenta la probabilidad de interferencia entre ellos. Cuando hay una densidad baja, existe una disminución del número de robots en la vecindad, y consecuentemente, también existe una disminución del número de interacciones que los robots pueden establecer para trabajar de forma coordinada.
Finalmente, la tercera propiedad deseable de un enjambre es la flexibilidad. La flexibilidad se define como la capacidad del enjambre para adaptarse fácilmente a un número diverso de tareas y a posibles cambios en su entorno. De forma general, los robots de un enjambre son homogéneos, no están especializados, y se despliegan sin un rol predefinido. Entonces, a partir de su habilidad para auto-organizarse, estos pueden adaptarse a los requerimientos específicos de la tarea que desarrollan. Cómo prueba de esto, la mayoría de investigaciones en robótica de enjambres se han basado en robots genéricos [6], simples y de capacidades limitadas como el e-puck [25] y el foot-bot [31]. A pesar de no ser robots especializados, estos robots se han utilizado en investigaciones sobre tareas diversas y complejas como son la morfogénesis (generación de forma) [30] y la aparición de propiedades cognitivas a nivel del enjambre [32]. Investigaciones que se unen al gran número de experimentos desarrollados en misiones típicas [4, 29, 34] de enjambres como agregación, *foraging*, toma de decisiones colectiva, entre otras.

2.2. Diseño de enjambres de robots

De forma tradicional, la investigación en robótica de enjambres se ha enfocado en encontrar las reglas de interacción que permiten diseñar comportamientos colectivos, producir enjambres de robots que puedan ejecutarlas, y verificar si los comportamientos deseados emergen [4]. Sin embargo, discusiones recientes [5, 7, 26] predicen un cambio de enfoque gracias a avances recientes en diferentes líneas de investigación dentro de esta disciplina [28]: (i) la aparición de nuevos robots que pueden operar en entornos no estructurados y dinámicos [44]; (ii) el desarrollo de nuevas metodologías para el diseño de comportamientos colectivos [30]; (iii) nuevas oportunidades para aprovechar la inteligencia emergente en el enjambre [32]; y (iv) el cambio de enfoque hacia aplicaciones apropiadas para el despliegue de grandes grupos de robots—por ejemplo, agricultura de precisión [45] o tareas de logística [46, 47]. Aunque

hay un futuro prometedor para la robótica de enjambres, la realidad es que la mayoría de avances aún ocurren bajo condiciones controladas en el laboratorio [48].

El diseño y operación de un enjambre de robots es un proceso que enmarca diferentes fases de trabajo [4]: modelado, diseño, operación y verificación. En la fase de modelado se definen (con diferentes grados de precisión) las especificaciones y/o requerimientos del funcionamiento del enjambre y de la misión a desarrollar. Luego, en la siguiente fase, se diseñan las reglas de interacción para obtener un comportamiento colectivo específico, se produce el software de control, y este se valida con simulaciones y/o con robots físicos. La siguiente fase corresponde al despliegue y operación del enjambre, en la cuál se hace necesario obtener métricas sobre su eficiencia. Y, finalmente, en la última fase, se llevan a cabo procesos de verificación y mantenimiento del sistema. El trabajo de investigación desarrollado en esta tesis está centrado en la fase de diseño y verificación de comportamientos colectivos; las demás fases se discuten de forma parcial en revisiones de la literatura recientes [37, 46].

Diseñar comportamientos colectivos para enjambres de robots es una tarea difícil. Un diseñador humano define a nivel global la misión que los robots deben ejecutar, pero los robots deben programarse de forma individual. A día de hoy, no existe una metodología general para diseñar el comportamiento individual de los robots de tal forma que el comportamiento colectivo deseado emerge [8]. Sin embargo, pueden diferenciarse dos metodologías generales de diseñó: el diseño manual y el diseño por métodos de optimización. A continuación se describen las características principales de los dos enfoques.

2.2.1. Diseño manual de comportamientos colectivos

El diseño manual es la metodología típica de diseño de comportamientos colectivos para enjambres de robots. Los métodos de diseño manual se centran en la obtención de comportamientos con características específicas, más que en su rendimiento. Tradicionalmente, el proceso de diseño tiene una naturaleza iterativa y está basado en ensayo y error: un diseñador crea y refina manualmente el software de control de los robots hasta que el comportamiento colectivo emerge [49]. Este proceso es costoso en tiempo y recursos, y no asegura que los resultados sean reproducibles. El proceso de diseño manual termina cuando el diseñador está satisfecho con el comportamiento que observa en el enjambre. Algunos métodos estructurados de este tipo se han propuesto para abordar problemas particulares del diseño de enjambres de robots. Por ejemplo, se han propuesto métodos para la creación de comportamientos de toma de decisiones [50], o herramientas que facilitan la programación manual del software de control para los robots [51]. El diseño manual ha permitido establecer reglas para facilitar la realización de un gran número de comportamientos colectivos [13]: por ejemplo, auto-ensamblado y morfogénesis [30, 52–54], detección colectiva de fallos [30, 43], movimiento coordinado [30, 54, 55], distribución autónoma de tareas [32, 56–58], entre otros. Dada la inexistencia de una plataforma de desarrollo común para robótica de enjambres, los

diseñadores producen implementaciones *ad hoc* de los comportamientos colectivos. Estas, en su mayoría, son de difícil portabilidad hacia otros sistemas. Es decir, el software de control producido no puede portarse de un robot a otro, y por este motivo, las comparaciones entre métodos de diseño manual son poco comunes.

2.2.2. Métodos basados en optimización

Cómo alternativa al diseño manual, en años más recientes se han propuesto métodos de diseñó de comportamientos colectivos basados en optimización [8, 10]. En estos métodos, el proceso de diseño es asistido (total o parcialmente) por algoritmos de optimización. La ventaja del diseño por optimización es que permite diseñar y comparar la eficiencia de sinnúmero de posibles opciones de software de control. Este tipo de diseño está basado en la eficiencia del enjambre, a diferencia del diseño manual. El proceso de diseño no termina cuando emerge un comportamiento colectivo con ciertas características sino cuando el comportamiento diseñado alcanza una cierta eficiencia.

En el diseño por optimización, un algoritmo de optimización explora posibles instancias de software de control para los robots y selecciona la que maximiza su eficiencia—de acuerdo a una métrica de rendimiento definida para cada misión. Clasificaciones comunes [10] de estos métodos los dividen en diseño (i) *on-line* y *off-line*, y en diseño (ii) semi-automático y automático. Los métodos *on-line* diseñan el software de control directamente cuando los robots se encuentran en su entorno de trabajo—es decir, mientras estos ejecutan la misión [40]. Estos métodos están generalmente limitados a una exploración reducida del espacio de diseño. Por el contrario, los métodos de diseño *off-line* diseñan el software de control antes de que los robots se desplieguen en su entorno de trabajo—típicamente en simulación y empleando una gran capacidad de computo [22]. En los métodos de diseño semi-automático, un diseñador humano emplea un algoritmo de optimización como su herramienta principal. En este caso, el diseñador humano guía y manipula el algoritmo de optimización para alcanzar el comportamiento colectivo deseado [59]. En contraposición, los métodos de diseño automático no requieren ningún tipo de intervención durante el proceso de diseño [9]. Esta clasificación no es estricta y existen híbridos entre los diferentes enfoques. Sin embargo, al igual que en la diferenciación entre enjambres de robots y sistemas multi-robot, esta clasificación es conveniente para comparar métodos bajo las mismas hipótesis y, también, para realizar comparaciones justas sobre su eficiencia. En esta tesis solo se consideran métodos de diseño *off-line* y automáticos, dado que este tipo de diseño permite explorar un espacio de diseño mayor y no requiere la intervención de un diseñador humano con conocimiento experto.

Los métodos de diseño automático *off-line* tienen un gran número de ventajas, sin embargo, a día de hoy su rendimiento está fuertemente condicionado por un fenómeno en particular: el *reality-gap* [9]. El *reality-gap* [60] es la diferencia, a veces sutil pero existente, entre el modelo del robot empleado para producir el software de control y el robot final en el que

se despliega. Típicamente, el *reality-gap* se manifiesta como una perdida de rendimiento al trasladar el software de control desde las simulaciones a los robots físicos [61]. En esta tesis no se aborda esta problemática, dado que el diseño experimental se ha contemplado completamente en simulación. El alcance de esta tesis se ha limitado a evaluar la viabilidad de emplear métodos de diseño automático para diseñar enjambres de robots que pueden coordinar a otros enjambres. Sin embargo, determinar cual es el efecto del *reality-gap* es un problema que debe investigarse en profundidad en trabajos futuros.

2.2.3. Métodos de diseño relevantes para esta tesis

Para desarrollar la investigación de esta tesis se han concebido dos métodos de diseño automático *off-line*, `Pistacchio` y `EvoCMY`, y también se ha empleado un método de diseño manual, `C-Human`. `Pistacchio` es un método modular que pertenece a la familia de métodos AutoMoDe [11, 22]. `EvoCMY` es una implementación típica del método neuroevolutivo [23, 62]. Se ha decidido concebir una implementación de AutoMoDe y un método neuroevolutivo dado que ambos han sido consistentemente evaluados y comparados en simulación y con robots físicos. Métodos de ambas familias han sido empleados, por ejemplo, en el estudio de los efectos del *reality-gap* [20, 61] o el diseño de comportamientos colectivos basados en comunicación [13, 14, 63]. `C-Human` es un método diseño manual que ha sido empleado anteriormente para comparar la eficiencia de métodos de la familia AutoMoDe y métodos neuroevolutivos. `Pistacchio`, `EvoCMY` y `C-Human` son comparados con una línea base, `R-Walk`, la cual es una implementación manual de un comportamiento básico de movimiento aleatorio [64]. Los cuatro métodos son descritos en profundidad en el Capítulo 3.

2.3. Coordinación entre grupos de robots

El problema de coordinar un enjambre de robots con otro enjambre de robots ha sido abordado principalmente en el estudio comportamientos de movimiento coordinado [65–70]. En estos problemas se asume que existen dos tipos de enjambre que operan en conjunto: un enjambre activo que busca influenciar las acciones de otros enjambres; y un enjambre pasivo, cuyas acciones son influenciadas. Basados en diseño manual, un diseñador humano se encarga de modelar y producir software de control tanto para los robots del enjambre activo como para los robots del enjambre pasivo. En la mayoría de los casos, los enjambres pasivos actúan de acuerdo a modelos típicos de *flocking*—un comportamiento colectivo en donde los individuos se mueven en conjunto, emulando a las bandadas de aves. La contribución de estos trabajos está en el modelado y diseño de comportamientos colectivos para el enjambre activo, los cuales se ajustan al tipo de *flocking* con el cual opera el enjambre pasivo. Los resultados obtenidos en estas investigaciones son positivos y muestran que es posible coordinar un enjambre pasivo con un solo robot activo [65–67], o con un enjambre activo que actúa

de forma cooperativa [68–70]. Sin embargo, los métodos propuestos no son de aplicabilidad general. Es decir, los métodos no han sido evaluados por su habilidad para resolver diferentes tipos de misiones, ni tampoco por su habilidad para adaptarse a una gran variedad de comportamientos colectivos del enjambre pasivo. Normalmente, el comportamiento colectivo se crea a partir de la intuición del diseñador humano, el cual aprovecha su conocimiento previo sobre el comportamiento del enjambre pasivo para producir una solución acorde.
La coordinación entre enjambres, y en particular la interacción entre robots activos y pasivos, no ha sido abordada desde la perspectiva del diseño automático *off-line* de enjambres de robots. La idea original de esta tesis es que los métodos de diseño automático podrían emplearse para producir software de control adecuado para el enjambre activo. Y aún más importante, estos métodos podrían no necesitar ser provistos con información explícita sobre el funcionamiento del enjambre pasivo. La diferencia de esta tesis con trabajos anteriores en la coordinación entre enjambres es que, en esta tesis, el método de diseño descubre de forma automática las dinámicas de interacción entre robots pasivos y activos, y las aprovecha para ejecutar una variedad de misiones. Así, el método de diseño automático podría ser capaz de explorar las posibles configuraciones del software de control que se ajustan mejor al comportamiento del enjambre pasivo, con el fin de optimizar la eficiencia del enjambre activo.

3. Metodología

Este capítulo detalla los métodos empleados en el diseño de comportamientos colectivos para un enjambre de robots que debe coordinar las acciones de otro enjambre. Como se detalla en el Capítulo 2, este tipo de problema ha sido poco abordado en la literatura y no existe un método estándar para ejecutar misiones relacionadas. En la mayoría de los casos, los investigadores presentan estudios elaborados *ad hoc*, con software de control producido manualmente, y el cual está diseñado para ejecutar misiones específicas. En ningún caso se ha hecho una comparación con métodos de diseño basados en optimización y métodos de diseño manual sobre un conjunto variado de misiones.
Los métodos de diseño considerados en esta tesis son `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk`. Estos métodos difieren en su forma de operación y fueron seleccionados para estudiar la efectividad de diferentes enfoques de diseño y producción de software de control para un enjambre. `Pistacchio` y `EvoCMY` son métodos basados en procesos de optimización que permiten producir software de control de forma automática. `C-Human` es un método de diseño manual en donde el software de control es producido por un grupo de diseñadores humanos. `R-Walk` es la línea base que sirve de referencia para determinar la eficacia de los demás métodos. `Pistacchio` es el método principal presentado en esta tesis, y es el marco sobre el cual se ha establecido la metodología que permite experimentar con grupos de enjambres que actúan de forma coordinada. Los demás métodos se derivan de las ideas originales introducidas con `Pistacchio`.
A continuación se formalizan las especificaciones del robot e-puck, la plataforma para la cual `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` producen software de control. Luego se introduce ARGoS3, el simulador de robots utilizado en el desarrollo de la investigación. En la siguiente parte del capítulo se presentan y detallan los cuatro métodos de diseño. En el caso de `Pistacchio` y `EvoCMY`, se describe la arquitectura de control que estos emplean y el proceso de optimización que permite el diseño automático de comportamientos colectivos. En el caso de `C-Human`, se describen los procedimientos de selección de diseñadores humanos y de producción manual de software de control. El capitulo termina con la descripción de `R-Walk`.

3.1. Robot y entorno de simulación

`Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` producen software de control para una versión extendida del robot e-puck [25, 71]—ver Figura **3-1**. El e-puck es una plataforma robótica

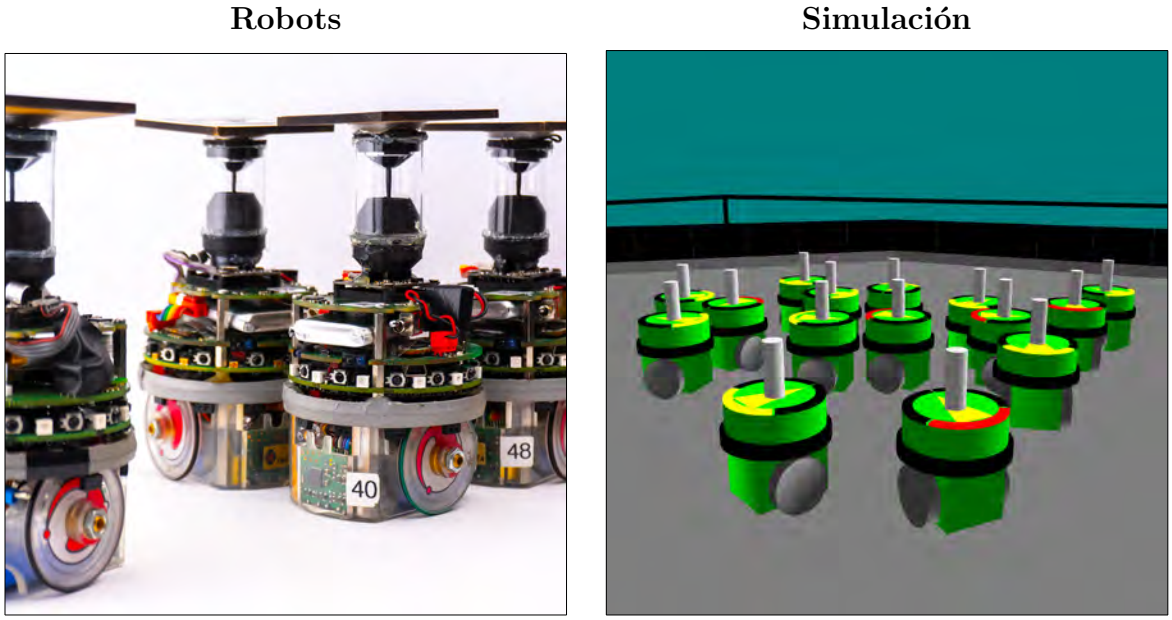


Figura 3-1.: Robot e-puck: modelo real (izquierda) y simulación del robot en ARGoS3 (derecha). De acuerdo con la especificación RM 3.3, el robot cuenta con sensores de proximidad y color del suelo, un sistema de visión omnidireccional, ruedas a cada lado del robot y LEDs RGB.

móvil originalmente diseñada para fines educativos. El robot tiene una altura de 13.5 cm y un diámetro de 7.5 cm, y se mueve mediante accionamiento diferencial en sus dos ruedas. Este robot ha sido ampliamente usado [6] en trabajos de investigación en robótica de enjambres [13, 22, 32, 40, 72] debido a su reducido factor de forma y a que es posible extender sus capacidades integrando módulos de hardware—por ejemplo, módulos de comunicación directa [73] o indirecta [72]. El trabajo presentado en esta tesis se ha basado únicamente un modelo simulado del e-puck. Sin embargo, las librerías de software [24, 71] empleadas en la simulación son las mismas librerías de software con las que se produce software de control para el robot físico. Esto permite asegurar que, de ser de interés en trabajo futuro, el software de control producido en esta tesis podría ser implementado en el robot sin requerir modificación alguna.

`Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` han sido concebidos para operar con un modelo de e-puck que integra los sensores y actuadores definidos por la especificación RM 3.3. La especificación basada en modelos únicos de referencia (*Reference Model - RM*, originalmente en inglés) [8] es un concepto que permite formalizar las características de la plataforma para la cuál es posible producir software de control con un método de diseño. Las especificaciones definidas mediante el modelo de referencia definen las características de funcionamiento del robot tanto en simulación como en la realidad. De forma particular, el modelo de referencia define las entradas y salidas del software de control, y la forma en que estas se relacionan

Tabla 3-1.: Especificaciones definidas en el modelo de referencia RM 3.3. Los robots pueden mantener sus LEDs apagados ($\varnothing$) o emitir el color cian (C), magenta (M) y amarillo (Y). Simultáneamente, los robots pueden percibir a otros robots que emiten estos colores. V_c agrega las posiciones de los robots percibidos en tres vectores de dirección distintos; uno para cada color en el conjunto $\{C, M, Y\}$.

Entrada	**Valores**	**Descripción**
$prox_{i\in\{1,\dots,8\}}$	$[0, 1]$	lectura del sensor de proximidad i
$gnd_{j\in\{1,\dots,3\}}$	$\{black, gray, white\}$	lectura del sensor de color del suelo j
$cam_{c\in\{C,M,Y\}}$	$\{yes, no\}$	color percibido
$V_{c\in\{C,M,Y\}}$	$(1.0; [0, 2]\,\pi\,\text{rad})$	vector dirección para cada color
Salida	**Valores**	**Descripción**
$v_{k\in\{l,r\}}$	$[-0.12, 0.12]\,\text{m/s}$	velocidad lineal deseada
$LEDs$	$\{\varnothing, C, M, Y\}$	color emitido por los LEDs

Periodo del ciclo de control: 0.1 s.

con el hardware del robot. La Tabla **3-1** lista las especificaciones definidas en el modelo de referencia RM 3.3.

El e-puck descrito por el modelo de referencia RM 3.3 cuenta con sensores de proximidad y color del suelo, un sistema de visión omnidireccional [74], ruedas a cada lado del robot y LEDs RGB. El robot puede detectar objetos cercanos empleando sus ocho sensores de proximidad ($prox_i$), los cuales están distribuidos al rededor de su cuerpo y aproximadamente tienen un rango de 3 cm. Tres transceptores de infrarrojo sirven de sensor de color del suelo (gnd_j) y le permiten al robot diferenciar regiones de color negro, gris y blanco. El sistema de visión omnidireccional (cam_c) del e-puck le permite percibir a otros robots que emiten los colores cian, magenta y amarillo con sus LEDs. El e-puck puede percibir a sus pares con un rango de visión de 360° y a una distancia máxima cercana a los 0.5 m. Por cada color percibido, un vector unitario (V_c) representa la dirección agregada de los robots que emiten dicho color. El software de control puede ajustar de forma independiente la velocidad (v_k) de cada rueda entre -0.12 y $0.12\,\text{m/s}$ y, también, puede emplear los LEDs ubicados en la parte superior del e-puck para emitir los colores cian, magenta, y amarillo.

El modelo de referencia RM 3.3 es una modificación de RM 3.0 [75]—modelo originalmente propuesto en la creación de AutoMoDe-`TuttiFrutti` [13] para la experimentación con e-pucks que interactuan mediante la emisión y percepción de señales de colores. Los resultados de estudios desarrollados con `TuttiFrutti` muestran que la interacción entre robots basada en colores permite desarrollar misiones complejas, en donde los robots pueden emplear estas interacciones para comunicarse y navegar su entorno. También, el uso de colores simplifi-

ca la visualización y monitorización de los enjambres de robots—una propiedad deseable para facilitar el entendimiento de los comportamientos colectivos [76]. Esta propiedad es particularmente relevante para los experimentos considerados en esta tesis: la supervisión de los robots requiere una clara diferenciación entre dos tipos de enjambres que operan en un mismo espacio. Es decir, es deseable que el enjambre que coordina y el enjambre que es coordinado sean visualmente diferentes. Por este motivo se decidió adoptar un sub-conjunto de especificaciones que están en concordancia con las propuestas en el modelo de referencia RM 3.0, las cuales han sido ya evaluadas en el contexto de `TuttiFrutti`. Una diferencia importante entre los modelos de referencia RM 3.3 y RM 3.0 es que en RM 3.3 se ha desactivado la habilidad del e-puck para comunicarse y localizar a sus pares mediante su módulo de *range-and-bearing* [73]. Este módulo ha sido desactivado con el fin de reducir el número de variables de diseño y, por tanto, la complejidad del diseño manual de software de control en el método `C-Human`.

3.1.1. Simulador

Las implementaciones de los métodos presentados en esta tesis han sido desarrolladas para ARGoS3 [24]—un simulador de robótica de software libre y con código abierto. De forma precisa, se ha empleado ARGoS3 versión beta 48 en conjunto con argos3-epuck [71]—una librería de software especialmente diseñada para la versión extendida del e-puck descrito al principio de esta sección. ARGoS3 se especializa en la simulación de robots móviles, y particularmente, en la simulación de sistemas multi-robot. En comparación a otros simuladores de robótica del estado del arte, como por ejemplo Gazebo [77] y V-Rep [78], ARGoS3 puede simular grandes grupos de robots con un costo computacional relativamente bajo [79]. Esta característica es fundamental para la investigación en el diseño automático de comportamientos colectivos. En estos métodos, el software de control se produce a partir de la ejecución de grandes cantidades de simulaciones—por ejemplo, en esta tesis, cada instancia de software de control se produce con 100 000 simulaciones. Este número elevado de simulaciones difícilmente podría ser ejecutado con Gazebo o V-Rep en un tiempo razonable para el desarrollo de la tesis. Por este motivo, ARGoS3 es *de facto* el simulador estándar en las comparaciones de métodos de diseño automático de comportamientos colectivos para enjambres de robots. Dichas investigaciones corresponden a trabajos enfocados tanto en el diseño modular como en el diseño basado en neuroevolución [12, 13, 20, 22, 39, 80].

3.2. Método I: AutoMoDe-`Pistacchio`

AutoMoDe-`Pistacchio` es un método automático de diseño de comportamientos colectivos para enjambres de robots que pertenece a la familia de métodos modulares AutoMoDe. La particularidad de `Pistacchio` es que permite la operación simultanea de dos tipos de

enjambres: un *enjambre activo*—para el cual se diseña el software de control de forma automática; y un *enjambre pasivo*—que actúa en el mismo espacio que el enjambre activo, pero que opera con un software de control diferente. Igual que otros métodos de AutoMoDe [11], `Pistacchio` se caracteriza por producir software de control de forma modular y con la asistencia de un proceso de optimización. Los métodos existentes en la familia AutoMoDe [13–16, 20–22, 81, 82] están limitados al diseño de enjambres en donde todos los robots operan con un mismo software de control. En estos casos, el proceso optimización tiene completa libertad para establecer las reglas de interacción entre los robots del enjambre. `Pistacchio` es un método creado en esta tesis para investigar si la metodología propuesta con AutoMoDe es efectiva también en el diseño de enjambres activos que deben ejecutar misiones en coordinación con enjambres pasivos. En este caso, el proceso de optimización está restringido por las reglas de interacción que rigen el comportamiento del enjambre pasivo; las cuales son desconocidas *a priori* y solo pueden ser determinadas durante el proceso de diseño. Esta idea original adoptada en `Pistacchio` es el fundamento sobre el cuál se desarrolla esta tesis, y sobre la cuál se conciben los demás métodos: `EvoCMY`, `C-Human` y `R-Walk`.

`Pistacchio` diseña software de control mediante la combinación y sintonización de módulos de software. Estos módulos de software se desarrollaron manualmente y se mantuvieron sin modificaciones una vez empezada la campaña experimental. Las siguientes sub-secciones describen los módulos de software integrados en `Pistacchio` y el proceso de optimización que combina y sintoniza estos módulos dentro de una arquitectura de control.

3.2.1. Módulos de software

La característica principal de los métodos de la familia AutoMoDe es que producen software de control mediante la selección, combinación y sintonización de módulos paramétricos de software. Estos módulos se conciben de forma agnóstica y no están destinados a ser empleados en una única misión. Por el contrario, el proceso de optimización puede combinarlos y parametrizarlos para establecer comportamientos colectivos que son efectivos en una variedad de misiones. El tipo de misiones que el enjambre puede ejecutar está caracterizado por las especificaciones del robot. Por ejemplo, es razonable pensar que robots móviles terrestres no pueden ejecutar tareas que requieren sobrevolar una área de interés. Por este motivo, los módulos de software en un método de diseño modular se seleccionan de acuerdo con las capacidades definidas dentro de las especificaciones del robot.

`Pistacchio` ha sido desarrollado tomando como base el modelo de referencia RM 3.3. Los módulos concebidos para `Pistacchio` han sido construidos sobre aquellos desarrollados anteriormente para `TuttiFrutti` [13]. Al igual que en `TuttiFrutti`, los módulos de `Pistacchio` permiten al robot interactuar con otros robots mediante el uso de señales de colores. `Pistacchio` integra un conjunto de 6 comportamientos de bajo nivel—EXPLORATION, STOP, COLOR-FOLLOWING, COLOR-ELUSION, CIRCLING y DRIVEN, y un conjunto de 5 condiciones de

transición—BLACK-FLOOR, GRAY-FLOOR, WHITE-FLOOR, FIXED-PROBABILITY y COLOR-DETECTION. Los comportamientos de bajo nivel corresponden a las acciones que el robot puede ejecutar, y las condiciones de transición corresponden a eventos que causan la transición entre comportamientos de bajo nivel. `Pistacchio` combina estos módulos dentro de máquinas de estados finitos probabilísticas—al igual que otros métodos de AutoMoDe [13–15, 20, 22, 81, 82]. La Tabla **3-2** lista los módulos de software integrados en `Pistacchio`, los cuales se describen en detalle a continuación.

Módulos de comportamiento

En EXPLORATION, el robot se mueve en línea recta siguiendo un comportamiento balístico hasta que detecta un obstáculo en frente con sus sensores de proximidad ($prox_i$). Cuando existe dicho obstáculo, el robot rota durante un periodo de tiempo determinado por el número de ciclos de control definidos con el parámetro τ, en un rango $\tau \in \{0, \ldots, 100\}$. STOP detiene el movimiento del robot. COLOR-FOLLOWING y COLOR-ELUSION mueven el robot con velocidad constante hacia o en contra—respectivamente—otros robots que emiten un color en específico (cam_c). El parámetro $\delta \in \{C, M, Y\}$ determina el color al que reacciona el robot. Si el robot no percibe robots emitiendo dicho color, este se mueve en línea recta. CIRCLING conduce al robot en trayectorias circulares. El parámetro θ define la dirección deseada de desplazamiento respecto del frente del robot, en el rango $[-3.13, 3.13]\,\pi$ rad. El movimiento circular que se ejecuta está condicionado tanto por la dirección deseada de desplazamiento como por las restricciones cinemáticas del robot. DRIVEN es un módulo creado especialmente para diseñar el comportamiento colectivo del enjambre pasivo. En este comportamiento, un robot solo se mueve ante la acción de estímulos externos. De forma precisa, un robot ejecutando DRIVEN sólo se moverá por la acción causada por otros robots—por ejemplo, por las acciones de los robots del enjambre activo. El parámetro $\psi \in \{0, 1, 2\}$ determina la acción de DRIVEN: el módulo puede ser configurado para que el robot mantenga un estado semi-estático constante $\{0\}$; o un movimiento semi-estático que cambia a un movimiento con velocidad constante hacia $\{1\}$ o en contra $\{2\}$ de otros robots que emiten un color $\delta \in \{C, M, Y\}$. El robot mantiene un comportamiento semi-estático cuando no percibe a otros robots emitiendo el color definido por δ. Este comportamiento semi-estático es similar a STOP, pero le permite al robot moverse si detecta objectos en sus cercanías mediante los sensores de proximidad. COLOR-DETECTION, COLOR-ELUSION, CIRCLING y DRIVEN incluyen un componente de evasión de obstáculos basado en fuerzas repulsivas virtuales [83]. Todos los módulos de comportamiento tienen un parámetro $\gamma \in \{\varnothing, C, M, Y\}$ que determina el color que el robot emite con sus LEDs. Los parámetros τ, δ, γ y θ son sintonizados durante el proceso de diseño del software de control.

EXPLORATION, STOP, COLOR-FOLLOWING y COLOR-ELUSION, son implementaciones de los módulos introducidos originalmente en `TuttiFrutti`. CIRCLING y DRIVEN son introducidos por primera vez con `Pistacchio`.

Tabla 3-2.: Conjunto de módulos de software integrados en `Pistacchio`. Las novedades con respecto a `TuttiFrutti` se resaltan en gris. Estas corresponden a módulos que han sido añadidos para extender las dinámicas de coordinación entre diferentes tipos de enjambres. Los módulos operan de a cuerdo a la especificación RM 3.3, ver Tabla **3-1**.

Comportamiento de bajo nivel *	**Parámetros**	**Descripción**
EXPLORATION	$\{\tau, \gamma\}$	movimiento aleatorio
STOP	$\{\gamma\}$	estado estático
COLOR-FOLLOWING	$\{\delta, \gamma\}$	movimiento hacia robots que emiten el color δ
COLOR-ELUSION	$\{\delta, \gamma\}$	movimiento en contra de robots que emiten el color δ
CIRCLING	$\{\theta, \gamma\}$	movimiento en círculos con un ángulo de apertura θ
DRIVEN **	$\{\psi, \delta, \gamma\}$	estado semi-estático que de acuerdo con ψ puede tornase en movimiento (i) hacia o (ii) en contra de robots que emiten el color δ
Condición de transición	**Parámetros**	**Descripción**
BLACK-FLOOR	$\{\beta\}$	detectado suelo de color negro
GRAY-FLOOR	$\{\beta\}$	detectado suelo de color gris
WHITE-FLOOR	$\{\beta\}$	detectado suelo de color blanco
FIXED-PROBABILITY	$\{\beta\}$	transición con una probabilidad fija
COLOR-DETECTION	$\{\delta, \beta\}$	detectado robot que emite el color δ

* Todos los comportamientos de bajo nivel pueden pueden emitir un color del conjunto $\gamma \in \{\varnothing, C, M, Y\}$ mientras ejecutan las acciones descritas. ** Todos los módulos, a excepción de DRIVEN, se emplean en el diseño de comportamientos colectivos para el enjambre activo. DRIVEN ha sido reservado para el diseño de comportamientos colectivos del enjambre pasivo.

Módulos de transición

BLACK-FLOOR, GRAY-FLOOR y WHITE-FLOOR causan una transición de comportamientos cuando el robot percibe que se encuentra un área en donde el color del suelo es negro, gris y blanco (gnd_j); respectivamente. El parámetro β determina la probabilidad de transición cuando dicha condición se cumple. FIXED-PROBABILITY causa una transición con una probabilidad fija determinada por $\beta \in [0, 1]$. Las transiciones causadas por COLOR-DETECTION se basan en la percepción que el robot tiene sobre el color emitido por sus pares (cam_c). El parámetro $\delta \in \{C, M, Y\}$ define el color que causa la transición, con probabilidad $\beta \in [0, 1]$. Los parámetros β y δ son sintonizados de forma automática durante el proceso de diseño del software de control.
Todos los módulos de transición son implementaciones de los módulos introducidos originalmente en `TuttiFrutti`.

3.2.2. Diseño automático de software de control

`Pistacchio` produce software de control siguiendo el procedimiento originalmente concebido por Francesca *et al.* para `Chocolate` [20], y que también ha sido empleado en la realización de enjambres de robots con otros métodos de AutoMoDe como son `Gianduja` [84], `Coconut` [82] y `TuttiFrutti`. `Pistacchio` es un método de diseño automático, por tanto, transforma el problema de diseñar software de control en un problema de optimización. A partir de la especificación de una misión que el enjambre debe desarrollar, un algoritmo de optimización selecciona una combinación apropiada de módulos y parámetros que se articulan dentro de una arquitectura de control predefinida. Cuando esta arquitectura de control se despliega en todos los robots, el enjambre opera bajo el comportamiento colectivo que ha sido diseñado. En este sentido, el software de control resulta de un proceso de optimización que maximiza la eficiencia del enjambre, medida a partir de una métrica específica asociada a la misión a ejecutar—la función objetivo. En `Pistacchio`, la arquitectura de control toma la forma de una maquina de estados finitos probabilística. La maquina de estados está restringida a un máximo de cuatro estados—los comportamientos de bajo nivel (Sección 3.2.1); y un máximo de cuatro transiciones salientes por estado—las condiciones de transición (Sección 3.2.1). En todos los casos, una transición se origina y termina en estados distintos. Dentro de este espacio de diseño, el algoritmo de optimización tiene libertad para escoger una configuración que se ajusta a la misión a ejecutar.
En `Pistacchio`, los módulos incluidos en la máquina de estados y los valores de sus parámetros son seleccionados *off-line*—ver Capítulo 2. En los métodos de diseño *off-line*, el software de control se genera antes de que el enjambre se despliegue en su entorno de trabajo. Cuando se experimenta con robots físicos, el software de control se genera típicamente en simulación y luego se despliega en los robots—en este caso, el entorno de trabajo es el escenario físico donde operan los robots. En esta tesis se ha considerado únicamente el uso de simulaciones

para realizar la comparación empírica de métodos. Por tanto, la noción de diseño *off-line* ha sido ajustada para el diseño experimental propuesto: el software de control se genera en simulación, y este es luego desplegado en nuevas simulaciones—en este caso, el entorno de trabajo es el nuevo escenario simulado donde operan los robots. Las condiciones iniciales de las simulaciones empleadas para evaluar la eficiencia de los robots difieren de aquellas empleadas durante el proceso de diseño—de forma precisa, en las posiciones iniciales de los robots. De esta forma, el análisis estadístico de la eficiencia del enjambre se realiza en escenarios con condiciones iniciales que no han sido empleadas durante el proceso de diseño automático.

El proceso de diseño en `Pistacchio` emplea Iterated F-race: un método de optimización multi-propósito, usado con frecuencia en la de configuración automática de algoritmos [85], y que se basa en el algoritmo F-race [86]. Iterated F-race explora el espacio de diseño en búsqueda de configuraciones efectivas de software de control. Durante el proceso de diseño, la eficiencia de las posibles configuraciones es evaluada mediante simulaciones en ARGoS3, ver Sección 3.1.1. La duración del proceso de optimización está determinada por el número máximo de simulaciones que Iterated F-race dispone para encontrar una configuración viable. Cuando el proceso de diseño alcanza un numero máximo (y predefinido) de simulaciones, Iterated F-race termina el proceso de optimización y `Pistacchio` retorna la mejor configuración encontrada. Esta configuración se pasa a todos los robots activos y se evalúa posteriormente sin ser modificada.

3.3. Método II: `EvoCMY`

La neuroevolución [23, 62] es la metodología típica de diseño de comportamientos colectivos para enjambres de robots mediante optimización. Sin embargo, no existe un procedimiento definido para seleccionar una arquitectura adecuada para la red neuronal o una configuración adecuada para el algoritmo evolutivo. Aunque se han propuesto una gran variedad métodos de evolución artificial sofisticados (`CMA-ES` [87], `xNES` [88] y `NEAT` [89]), estudios empíricos recientes [12] han demostrado que, cuando no se puede aplicar un trabajo extensivo de ajuste manual, estos métodos no son más eficientes que los métodos típicos más simples (`EvoStick` [22]). Es decir, la complejidad añadida en `CMA-ES`, `xNES` y `NEAT` no los hace más efectivos que `EvoStick` en contextos en donde se busca una aplicabilidad general de los métodos. Por este motivo, en esta investigación se ha optado por desarrollar `EvoCMY`: un método neuroevolutivo que se ciñe a las implementaciones típicas de la literatura, y que se especializa en el diseño de comportamientos colectivos para robots que interactúan mediante señáles de colores. `EvoCMY` es una adaptación de `EvoColor` [13]—un método neuroevolutivo inspirado en `EvoStick`. En estudios anteriores, `EvoColor` ha servido como línea base para el estudio del diseño de comportamientos colectivos para robots que pueden emitir y percibir señales de colores—definidos bajo la especificación RM 3.0. En esta tesis se modifica

EvoColor y se concibe EvoCMY para producir software de control para e-pucks que operan según la especificación RM 3.3. Al igual que en Pistacchio, la capacidad de los robots para interactuar mediante señales de colores faculta a EvoCMY para producir software de control que, en principio, permite al enjambre activo coordinar las acciones del enjambre pasivo.
La tabla **3-3** resume la topología de la red neuronal y los parámetros del proceso de evolución artificial de EvoCMY. El software de control producido por EvoCMY tienen la forma de una red neuronal artificial sin capas ocultas, totalmente conectada, con 24 nodos de entrada (in) y con 8 nodos de salida (out). En esta topología, la red neuronal transfiere información desde los nodos de entrada hacia los nodos de salida mediante conexiones sinápticas ($conn$) con pesos (ω) en el rango $[-5, 5]$. La activación de cada nodo de salida está determinada por la suma ponderada de las señales provenientes de los nodos de entrada, las cuales son filtradas mediante una función logística estándar. Las lecturas de los sensores de proximidad ($prox$) y de color del suelo (gnd) se pasan directamente a la red neuronal. Cada vector en $V_{c\in\{C,M,Y\}}$ se transforma en proyecciones escalares sobre cuatro vectores unitarios que apuntan a 45°, 135°, 225°, y 315° respecto del frente del robot. Luego, cada proyección se pasa a la red a través de un nodo de entrada independiente. El último nodo de entrada de la red corresponde a un *bias* de entrada. Cuatro nodos de salida codifican dos tuplas (v') que representan componentes negativos y positivos de la velocidad de las ruedas $v_{k\in\{l,r\}}$. Cada tupla se asocia a dos nodos de salida independientes y se define de la forma $v' = ([-12, 0], [0, 12])$. La velocidad de cada rueda (v) se calcula mediante la suma de los dos elementos de la tupla (v'). De forma similar, el color emitido por los LEDs RGB del robot se selecciona comparando el valor de los nodos de salida asociados a cada color en el conjunto $\{\varnothing, C, M, Y\}$. El color que se emite corresponde al máximo valor encontrado a través de los cuatro nodos de salida.
EvoCMY selecciona pesos apropiados para las conexiones sinápticas a través de un proceso de optimización basado en evolución artificial. De forma similar al proceso seguido en Pistacchio, EvoCMY busca y sintoniza configuraciones de la red neuronal que maximizan la eficiencia del enjambre de acuerdo con la métrica de rendimiento definida en cada misión—la función objetivo. Para esto, el algoritmo evolutivo emplea operaciones de elitismo y mutación que le permiten sintonizar los pesos sinápticos y, en consecuencia, diseñar un comportamiento colectivo adecuado para ejecutar la misión. El proceso de evolución artificial termina cuando se agota el número máximo de simulaciones de las que dispone EvoCMY para producir el software de control para los robots. La mejor red neuronal encontrada se pasa a todos los robots activos y se evalúa posteriormente sin ser modificada.

3.4. Método III: C-Human

C-Human es un método de diseño manual de enjambres de robots introducido por Francesca *et al.* [19] que ha servido como base para determinar la efectividad de métodos de diseño automático como Vanilla [19], Chocolate [20], y Cedrata [21]. De forma precisa, este

Tabla 3-3.: Topología de la red neuronal artificial y parámetros de evolución de `EvoCMY`. La red neuronal tiene conexión completa entre nodos de entrada y salida, sin capas ocultas entre ellos. El proceso de evolución emplea operaciones de elitismo y mutación para sintonizar los pesos de las conexiones sinápticas. La red neuronal opera de acuerdo con la la especificación RM 3.3, ver Tabla **3-1**.

Nodo de entrada	**Descripción**
$in_{a\in\{1,\ldots,8\}}$	lectura de los sensores de proximidad $prox_{i\in\{1,\ldots,8\}}$
$in_{a\in\{9,\ldots,11\}}$	lectura de los sensores de color del suelo $gnd_{j\in\{1,\ldots,3\}}$
$in_{a\in\{12,\ldots,23\}}$	proyecciones escalares para $V_{c\in\{C,M,Y\}}$
$in_{a\in\{24\}}$	*bias* de entrada
Nodo de salida	**Descripción**
$out_{b\in\{1,\ldots,4\}}$	tuplas v' para el cálculo de velocidades $v_{k\in\{l,r\}}$
$out_{b\in\{5,\ldots,8\}}$	activación y emisión de colores $\{\varnothing, C, M, Y\}$
Conexión	**Descripción**
$conn_{s\in\{1,\ldots,192\}}$	conexiones sinápticas con peso $\omega\in[-5, 5]$
Número de generaciones *	—
Tamaño de la población	100
Selección por elitismo	20
Individuos que mutan	80
Evaluaciones por individuo	10
Post-evaluaciones por individuo **	100

* El número de generaciones se calcula de acuerdo con el número máximo de disponibles para diseñar el software de control. ** La población obtenida en la última generación es post-evaluada para seleccionar al mejor individuo.

método fue concebido para evaluar la habilidad de diseñadores humanos en la producción de software de control para robots que se coordinan mediante comportamientos colectivos. La particularidad de C-Human es que no requiere que los diseñadores produzcan el software de control desde cero. En cambio, C-Human provee una serie de módulos de software paramétricos predefinidos que el diseñador puede combinar y sintonizar para producir el software de control—los módulos de AutoMoDe. En este método de diseño manual, el diseñador toma el rol que en métodos automáticos es dado al algoritmo de optimización. C-Human facilita la labor del diseñador y evita la necesidad de un conocimiento profundo sobre el modo de operación de los robots y el lenguaje de programación requerido para trabajar con ellos. *A priori*, sería razonable pensar que restringir al diseñador a usar los módulos de software de AutoMoDe limita su habilidad para desarrollar comportamientos colectivos complejos y (posiblemente) mejores. Sin embargo, estudios anteriores [19] han mostrado que diseñadores que tienen total libertad para producir el software de control crean comportamientos colectivos que son menos eficientes que C-Human. En esta tesis también se busca investigar cómo los dos métodos basados en optimización propuestos—Pistacchio y EvoCMY—se comparan con posibles métodos de diseño manual. Por este motivo, dentro de la investigación también se incluye a C-Human como una alternativa de diseño de comportamientos colectivos para enjambres de robots que deben coordinar a otros enjambres.

En el marco experimental de esta investigación, tres diseñadores humanos toman el rol de C-Human. Los diseñadores de C-Human deben producir software de control para el enjambre activo empleando los módulos definidos en Pistacchio, y están restringidos también a usar la misma arquitectura de control. Es decir, pueden crear máquinas de estado de hasta cuatro comportamientos de bajo nivel, con hasta cuatro transiciones salientes de cada uno de ellos. Los diseñadores que participan en los experimentos de esta tesis son estudiantes de Ingeniería Electrónica de octavo semestre en la Universidad de Nariño, Pasto, Colombia. Este grupo de diseñadores ha sido escogido porque son investigadores en formación que tienen un año de experiencia en el campo de la robótica de enjambres, también tienen familiaridad con ARGoS3, y han participado previamente en competiciones de programación de enjambres de robots [47]. Antes de participar en el estudio, los diseñadores tuvieron contacto con los experimentos desarrollados en TuttiFrutti—dándoles un conocimiento parcial sobre el funcionamiento de los métodos de diseño modular. Sin embargo, los diseñadores no fueron involucrados el desarrollo de Pistacchio o EvoCMY. Las especificaciones del robot y los módulos de software predefinidos les son presentadas al momento de empezar los experimentos. La tabla **3-4** lista los módulos de software empleados por los diseñadores humanos, y resume el protocolo experimental de C-Human. El Anexo B detalla las instrucciones que son entregadas a los diseñadores que participan en la producción manual de software de control para el enjambre activo. La implementación de los módulos de C-Human es idéntica a la realizada en Pistacchio. Sin embargo, en C-Human se han renombrado los módulos y sus parámetros para mantener la compatibilidad de los módulos con AutoMoDe Editor [90]—la

Tabla 3-4.: Conjunto de módulos de software disponibles en `C-Human` y protocolo de experimentación. Los módulos son idénticos a los concebidos para `Pistacchio` y operan de a cuerdo a la especificación RM 3.3, ver Tabla **3-1**.

Comportamiento de bajo nivel *	**Parámetros**	**Descripción**
EXPLORATION	$\{\tau, \gamma\}$	movimiento aleatorio
STOP	$\{\gamma\}$	estado estático
COLOR-FOLLOWING	$\{\delta, \gamma\}$	movimiento hacia robots que emiten el color δ
COLOR-ELUSION	$\{\delta, \gamma\}$	movimiento en contra de robots que emiten el color δ
CIRCLING	$\{\theta, \gamma\}$	movimiento en círculos con un ángulo de apertura θ
Condición de transición	**Parámetros**	**Descripción**
BLACK-FLOOR	$\{\beta\}$	detectado suelo de color negro
GRAY-FLOOR	$\{\beta\}$	detectado suelo de color gris
WHITE-FLOOR	$\{\beta\}$	detectado suelo de color blanco
FIXED-PROBABILITY	$\{\beta\}$	transición con una probabilidad fija
COLOR-DETECTION	$\{\delta, \beta\}$	detectado robot que emite el color δ
Diseñadores humanos		3
Misiones por diseñador		4
Fase de preparación		48 h
Fase de experimentación		4 h
Herramientas de diseño		AutoMoDe Editor y ARGoS3

* Todos los comportamientos de bajo nivel pueden emitir un color del conjunto $\gamma \in \{\varnothing, C, M, Y\}$ mientras ejecutan las acciones descritas.

herramienta para la construcción manual software de control con módulos de AutoMoDe.
Los experimentos de diseño de software de control con C-Human se ejecutan en dos fases: (i) preparación y (ii) experimentación. En la fase de preparación, los diseñadores son informados del experimento, la metodología, las misiones a desarrollar, las herramientas de simulación, y el protocolo experimental, pero no tienen acceso a las herramientas de diseño y simulación. Esta fase inicia 48 h antes de empezar la fase de experimentación, dando lugar espacios de preguntas y respuestas particulares sobre la metodología de diseño. Posteriormente, en la fase de experimentación, los diseñadores trabajan continuamente durante 4 h en el diseño de comportamientos colectivos para cuatro misiones diferentes. Durante esta fase, los diseñadores producen el software de control empleando AutoMoDe Editor. Esta herramienta les permite construir, visualizar y manipular las máquinas de estado, para luego simular directamente el software de control en ARGoS3. En las simulaciones, los diseñadores pueden inspeccionar visualmente el comportamiento del enjambre y también son informados sobre la eficiencia de su diseño de acuerdo con el puntaje obtenido en la métrica de rendimiento asociada a cada misión—la función objetivo. En C-Human, el tiempo para ejecutar el proceso de diseño no está restringido por un número específico de simulaciones, si no por el tiempo disponible en la fase de experimentación. Una vez agotado este tiempo, se recoge el mejor software de control producido para cada misión—de acuerdo con lo reportado por los diseñadores. Al terminar el experimento, se toma el tiempo que los diseñadores invierten en el proceso de diseño y se cuestiona por los comportamientos colectivos que han logrado identificar en el enjambre pasivo. El software de control producido con C-Human se evalúa posteriormente sin ser modificado.

3.5. Método IV: R-Walk

R-Walk es una implementación trivial de los posibles comportamientos colectivos de un enjambre de robots: los robots se mueven de forma aleatoria en su entorno de trabajo mientras evitan colisiones con sus pares o con otros objetos [64]. A diferencia de Pistacchio, EvoCMY y C-Human, este no es un método de diseño en si mismo. En R-Walk, ningún parámetro o característica del comportamiento de los robots es optimizada—bien sea por algoritmos de optimización o por diseño manual. Estudios anteriores en el diseño automático de enjambres de robots han incluido el movimiento aleatorio como línea base [12]. R-Walk se incluye en el estudio como la línea base para determinar un umbral de eficiencia inferior sobre el cuál comparar los demás métodos.
La tabla **3-5** define el módulo de software y parámetros que determinan el comportamiento integrado en R-Walk. R-Walk es una parametrización particular de EXPLORATION—un módulo de comportamiento definido en Pistacchio. Robots que ejecutan este comportamiento se desplazan mediante movimiento aleatorio, siguiendo trayectorias balísticas [64]. De forma precisa, los robots se mueven en línea recta hasta encontrar un obstáculo, y cuando esto

Tabla 3-5.: Módulo de software integrado en `R-Walk`. El módulo es una parametrización particular del módulo concebido para `Pistacchio`; operando también de acuerdo con la especificación RM 3.3, ver Tabla **3-1**.

Comportamiento de bajo nivel	**Parámetros**	**Descripción**
EXPLORATION *	$\{\tau, \gamma\}$	movimiento aleatorio con $\tau \in \{1\}$ y $\delta \in \{\varnothing\}$
Condición de transición	**Parámetros**	**Descripción**
—	—	—

* Los robots no emiten ningún color mientras ejecutan el movimiento aleatorio.

ocurre, rotan durante un número aleatorio de ciclos de control. El robot empieza un nuevo desplazamiento en línea recta en caso de que su trayectoria se encuentre despejada después de la rotación. En el caso contrario, inicia una nueva rotación. `R-Walk` es concebido una vez y para todo el experimento, definiendo los parámetros $\tau \in \{1\}$ y $\delta \in \{\varnothing\}$ en EXPLORATION. Es decir, la rotación se reinicia cada ciclo de control si el robot no encuentra el camino libre, y todas las trayectorias se ejecutan sin que el robot emita color con sus LEDs. De esta forma, enjambres activos que operan con el comportamiento definido en `R-Walk` tienen una interacción mínima con el enjambre pasivo. Esto debido a la inexistencia de señales de colores con las que los robots activos pueden interactuar con los robots pasivos.

4. Diseño experimental

Este capítulo describe los experimentos realizados para investigar la efectividad de `Pistacchio`, `EvoCMY` y `C-Human` en el diseño de comportamientos colectivos para enjambres de robots. Estos métodos se comparan con el método de referencia `R-Walk` en cuatro misiones diferentes: Agrupamiento, Dispersión, Pastoreo y Segregación. Los experimentos están concebidos para evaluar la efectividad de los cuatro métodos en el diseño de un enjambre de robots (activo) que debe coordinar las acciones de un segundo enjambre (pasivo). En cada misión, se consideran tres tipos de comportamiento con el que enjambre pasivo puede interactuar con el enjambre activo: *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*. El protocolo experimental seguido en este trabajo comparte características con otros estudios que evalúan diferentes aspectos del diseño automático de enjambres de robots [13–16, 20–22, 81, 82]. La diferencia principal entre los experimentos en esta tesis y estudios anteriores es que, en esta tesis, se consideran dos tipos de enjambres de robots que operan de forma simultánea en una misma área de trabajo.

En el resto del capítulo, primero se definen las características de los dos enjambres considerados en los experimentos—el enjambre activo y el enjambre pasivo. A continuación se describen los tres tipos de comportamiento embebidos en el enjambre pasivo. *A priori*, se espera que estos comportamientos sean aprovechados por el enjambre activo para coordinar las acciones de los robots pasivos. Más adelante se formalizan las especificaciones de las cuatro misiones que sirven como caso de estudio. Finalmente, el capitulo cierra con una definición detallada del protocolo experimental y de las herramientas estadísticas empleadas para el análisis de los resultados.

4.1. Tipos de enjambres en los experimentos: activo y pasivo

Esta tesis se centra en investigar si métodos de diseño automático, o diseñadores humanos, pueden producir software de control para un enjambre de robots que debe coordinar a otro enjambre. Por tanto, durante la fase experimental se consideran escenarios en donde existen dos tipos de enjambres que interactuan para ejecutar una misión de forma conjunta. El primer tipo de enjambre es el *enjambre activo*—el enjambre para el cuál se produce software de control con los métodos bajo estudio. El segundo enjambre es el *enjambre pasivo*—el enjam-

bre que debe ser coordinado, cuyo comportamiento está predefinido y no se construye en los procesos de diseño automático. Los dos enjambres operan bajo las mismas especificaciones—ver Tabla **3-1**. Es decir, los robots de los dos tipos de enjambre pueden percibir a sus pares e interactuar con ellos mediante sus sensores de proximidad, cámara omnidireccional, LEDs y por acción de su desplazamiento. El sensor de color del suelo es empleado para detectar regiones de interés en las misiones.

4.1.1. Comportamientos del enjambre pasivo

Con el fin de determinar la efectividad de los métodos bajo estudio en problemas generales de coordinación entre enjambres, y no solo en una instancia particular de estos, se consideran tres tipos de comportamiento para el enjambre pasivo: *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*. Estos tres comportamientos han sido diseñados manualmente para que los robots del enjambre pasivo tomen acción unicamente si sufren estímulos causados por los robots del enjambre activo. De esta manera, la efectividad del enjambre activo, para el cuál se diseña el software de control, estará directamente condicionada su habilidad para encontrar y emplear de forma adecuada dichos estímulos. Los robots activos pueden interactuar de dos formas con los robots pasivos: a través de la proximidad física o emitiendo señales de colores. El software de control de los enjambres pasivos ha sido producido empleando un subconjunto de los módulos de software introducidos con `Pistacchio`. De forma correspondiente, los módulos han sido combinados y sintonizados dentro de una máquina de estados. La Tabla **4-1** lista los módulos empleados y su posible parametrización. El diseño de estos comportamientos se ha llevado a cabo siguiendo un proceso iterativo de ensayo y error, y sin restricciones de tiempo. *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión* han sido concebidos de forma que el enjambre activo pueda coordinar de diferentes maneras, y con diferente eficiencia, al enjambre pasivo en las misiones propuestas. La Figura **4-1** muestra las máquinas de estado diseñadas para *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*. En el resto de la sección se describe su funcionamiento.

C1-Atracción

Robots ejecutando *C1-Atracción* empiezan su funcionamiento en un estado semi-estático, emitiendo el color amarillo con sus LEDs, siendo reactivos a dos tipos de estímulos: (i) son atraídos hacia otros robots que emiten el color magenta en un rango aproximado de 50 cm; o, (ii) son repelidos por robots que entran en el rango de percepción de sus sensores de proximidad; sin importar el color que emitan con sus LEDs. En caso de que no exista ninguno de estos estímulos, o que haya existido pero ya no sea perceptible, el robot detiene su movimiento y se mantiene su posición. En caso de que los dos estímulos ocurran de forma simultánea, la respuesta del robot va a ser la agregación de la respuesta para los dos estímulos. Es decir, el robot se va a mover de acuerdo a la fuerza resultante después de

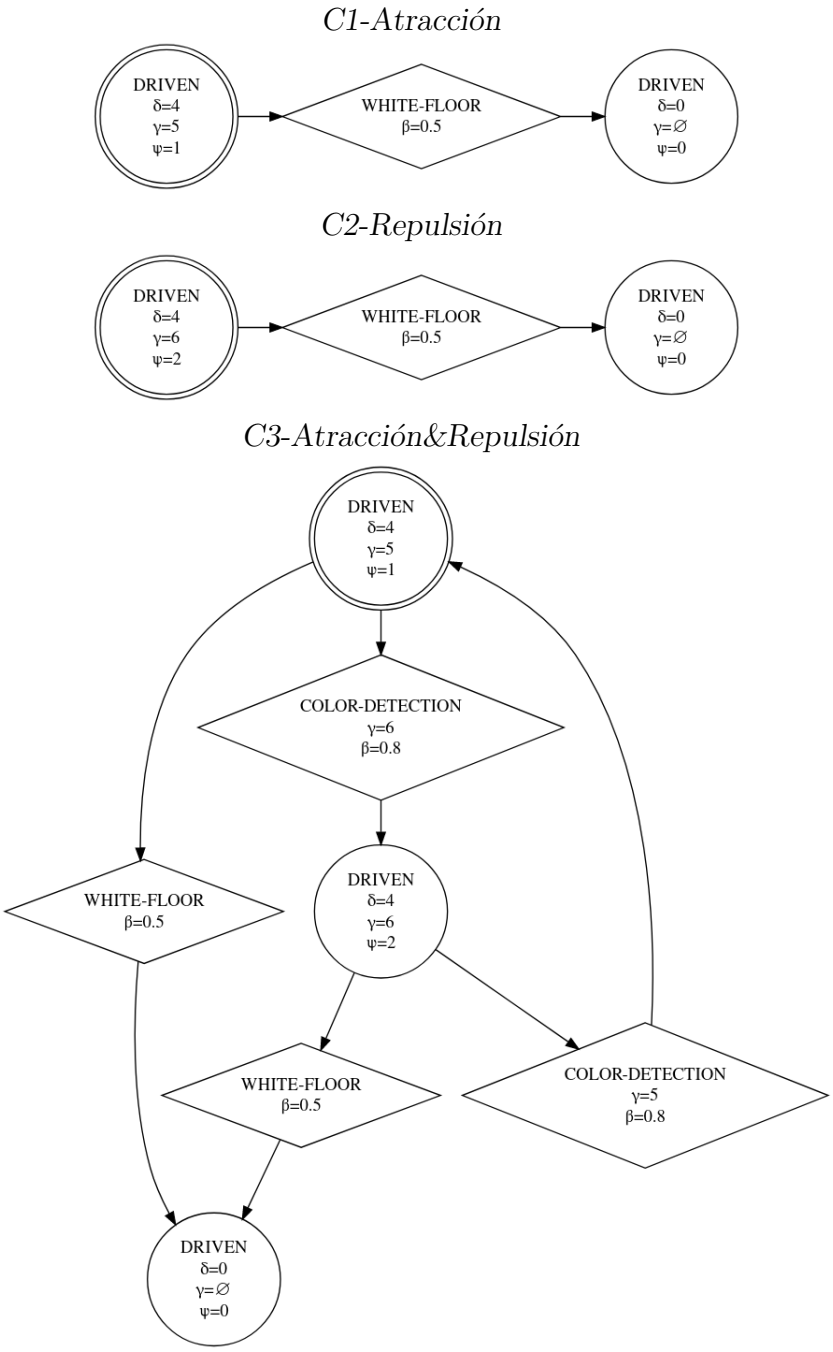


Figura 4-1.: Maquinas de estados de los comportamientos *C1-Atracción* (arriba), *C2-Repulsión* (centro) y *C3-Atracción&Repulsión* (abajo) del enjambre pasivo. Los estados (círculos) y transiciones (rombos) corresponden a los módulos de software definidos en `Pistacchio`, ver Sección 3.2.1. La máquina de estados inicia en el estado delineado por un circulo doble.

Tabla 4-1.: Módulos de software empleados en los comportamientos colectivos del enjambre pasivo. Los módulos son parametrízaciones particulares de los módulos definidos en `Pistacchio`; operando también de a cuerdo a la especificación RM 3.3, ver Tabla **3-1**.

Comportamiento de bajo nivel *	**Parámetros**	**Descripción**
DRIVEN	$\{\psi, \delta, \gamma\}$	estado semi-estático que de acuerdo a ψ puede tornase en movimiento (i) hacia o (ii) en contra de robots que emiten el color δ
Condición de transición **	**Parámetros**	**Descripción**
WHITE-FLOOR	$\{\beta\}$	detectado suelo de color blanco
COLOR-DETECTION	$\{\delta, \beta\}$	detectado robot que emite el color δ

* DRIVEN es parametrizado en el subconjunto $\gamma \in \{M, C\}$ y $\delta \in \{\varnothing, Y\}$. ** COLOR-DETECTION es parametrizado en el subconjunto $\gamma \in \{M, C\}$.

agregar las fuerzas de atracción y repulsión. En todo momento, y sin importar qué acción ejecuta el robot, este puede cambiar de estado si el sensor de color del suelo detecta una región de color blanco. Cuando ocurre esta transición, el robot apaga sus LEDs y únicamente se mantiene reactivo a través de sus sensores de proximidad. Este estado se mantiene hasta el final de la misión. Esta transición es relevante para la ejecución de misiones en donde existen regiones de interés marcadas con color blanco. La Figura **4-1** (arriba) muestra la maquina de estado que describe este comportamiento.

C2-Repulsión

Robots ejecutando *C2-Repulsión* actúan de forma similar. Los robots empiezan su funcionamiento en el estado semi-estático y emitiendo el color amarillo con sus LEDs. En este estado, también son repelidos por robots que entran en el rango de percepción de sus sensores de proximidad; sin importar el color que estos emitan con sus LEDs. A diferencia de *C1-Atracción*, en *C2-Repulsión* los robots reaccionan de forma repulsiva ante otros robots que emiten un color específico. De forma precisa, los robots se alejan de otros robots que emiten el color cian, cuando estos son percibidos en un rango aproximado de 50 cm. En caso de que no exista ninguno de estos estímulos, o que haya existido pero ya no sea perceptible, el robot detiene su movimiento y se mantiene estático en su posición. En este caso, el robot también cambia de estado si el sensor de color del suelo detecta una región de color blanco. Cuando ocurre esta transición, el robot apaga sus LEDs y solo se mantiene reactivo mediante

sus sensores de proximidad. Al igual que en *C1-Atracción*, este estado se mantiene hasta el final de la misión. La Figura **4-1** (centro) muestra la maquina de estado que describe este comportamiento.

C3-Atracción&Repulsión

C3-Atracción&Repulsión combina los dos comportamientos descritos anteriormente. Cada robot pasivo empieza su funcionamiento en el estado semi-estático y emitiendo el color amarillo con sus LEDs. De acuerdo a los estímulos percibidos por su cámara omnidireccional, el robot alterna entre comportamientos atractivos y repulsivos. El robot mantiene un comportamiento atractivo hacia robots que emiten el color magenta cuando estos son percibidos. Si la cámara percibe a robots que emiten el color cian, el robot cambia de estado e inicia un comportamiento repulsivo. En caso de que los dos estímulos ocurran de forma simultánea, el robot alterna su comportamiento de acuerdo a una probabilidad fija. Al igual que en *C1-Atracción* y *C2-Repulsión*, el robot cambia de estado si el sensor de color del suelo detecta una región de color blanco. Cuando ocurre esta transición, el robot apaga sus LEDs y solo se mantiene reactivo mediante sus sensores de proximidad. Este estado también se mantiene hasta el final de la misión. La Figura **4-1** (abajo) muestra la maquina de estado que describe este comportamiento.

4.2. Misiones

Los experimentos de esta tesis se desarrollan en cuatro misiones: Agrupamiento, Dispersión, Pastoreo y Segregación. Los métodos presentados en este trabajo no están restringidos unicamente a los escenarios estudiados en las misiones propuestas. Por el contrario, estas son una muestra representativa sobre la clase de posibles misiones que pueden abordarse dentro del objeto de estudio de esta investigación. Agrupamiento, Dispersión, Pastoreo y Segregación consideran aspectos típicos del comportamiento de los enjambres [4, 29, 34]; por ejemplo, la agregación de robots en un solo grupo, la dispersión de los robots, el desplazamiento de los robots hacia un área de interés y la separación del enjambre en grupos pequeños. En todos las misiones, la eficiencia del enjambre activo depende de su habilidad para coordinar las acciones del enjambre pasivo. De forma precisa, la eficiencia del enjambre activo es medida de acuerdo a una métrica específica asociada a cada misión—la función objetivo. La función objetivo se evalúa sobre las acciones observadas en el enjambre pasivo. Agrupamiento, Dispersión, Pastoreo y Segregación fueron seleccionadas bajo la hipótesis de que, para ser ejecutadas de forma efectiva, el enjambre activo debe emplear comportamientos colectivos que se ajustan a los comportamientos *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión* del enjambre pasivo. Y en concordancia, los métodos de diseño deben producir software de control que funciona de forma efectiva en cada

caso.
La Figura **4-2** muestra los escenarios experimentales de cada misión. En las cuatro misiones, los enjambres operan en una arena octagonal de $2.75\,\mathrm{m}^2$ y suelo gris. El suelo de la arena puede contener regiones de color blanco, las cuales son relevantes en la ejecución de la misión. Los robots ejecutan cada misión en un tiempo $T = 120\,\mathrm{s}$. A continuación se describe el escenario, la función objetivo, y el rol de los enjambres activo y pasivo en cada misión.

4.2.1. Agrupamiento

El enjambre activo debe agregar los robots del enjambre pasivo en un grupo compacto. La Figura **4-2** (arriba-izquierda) muestra el escenario experimental para AGRUPAMIENTO. El enjambre activo y el enjambre pasivo inician la misión estando distribuidos en todo el escenario. La eficiencia F_{agr} del enjambre activo se mide mediante la función objetivo descrita en la ecuación (4-1); cuyo valor debe minimizarse.

$$F_{agr} = \frac{\sum_{i=1}^{10} D_i(T)}{10}; \quad \text{con} \quad D_i(T) = (x_i, y_i) - (x_c, y_c). \tag{4-1}$$

F_{agr} mide la distancia promedio de cada robot al centro de masa del enjambre, calculado a partir de las posiciones de todos los robots. $D_i(T)$ es la distancia del robot i en la posición (x_i, y_i), al centro de masa del enjambre en la posición (x_c, y_c), al finalizar la misión en el tiempo T. *A priori*, se espera el enjambre activo sea eficiente agregando a los robots del enjambre pasivo cuando estos operan con el comportamiento de *C1-Atracción*. Por el contrario, se espera que el enjambre activo sea ineficiente agregando a los robots del enjambre pasivo cuando estos operan con el comportamiento *C2-Repulsión*.

4.2.2. Dispersión

El enjambre activo debe separar los robots del enjambre pasivo. La Figura **4-2** (arriba-derecha) muestra el escenario experimental para DISPERSIÓN. El enjambre activo y el enjambre pasivo inician la misión estando concentrados en el centro del escenario. La eficiencia F_{agr} del enjambre activo se mide mediante la función objetivo descrita en la ecuación (4-2); cuyo valor debe maximizarse.

$$F_{dis} = \frac{\sum_{i=1}^{10} D_i(T)}{10}; \quad \text{con} \quad D_i(T) = (x_i, y_i) - (x_c, y_c). \tag{4-2}$$

F_{dis} mide la distancia promedio de cada robot al centro de masa del enjambre, calculado a partir de las posiciones de todos los robots. $D_i(T)$ es la distancia del robot i en la posición (x_i, y_i), al centro de masa del enjambre en la posición (x_c, y_c), al finalizar la misión en el tiempo T. *A priori*, se espera el enjambre activo sea ineficiente separando a los robots del enjambre pasivo cuando estos operan con el comportamiento de *C1-Atracción*. Por el

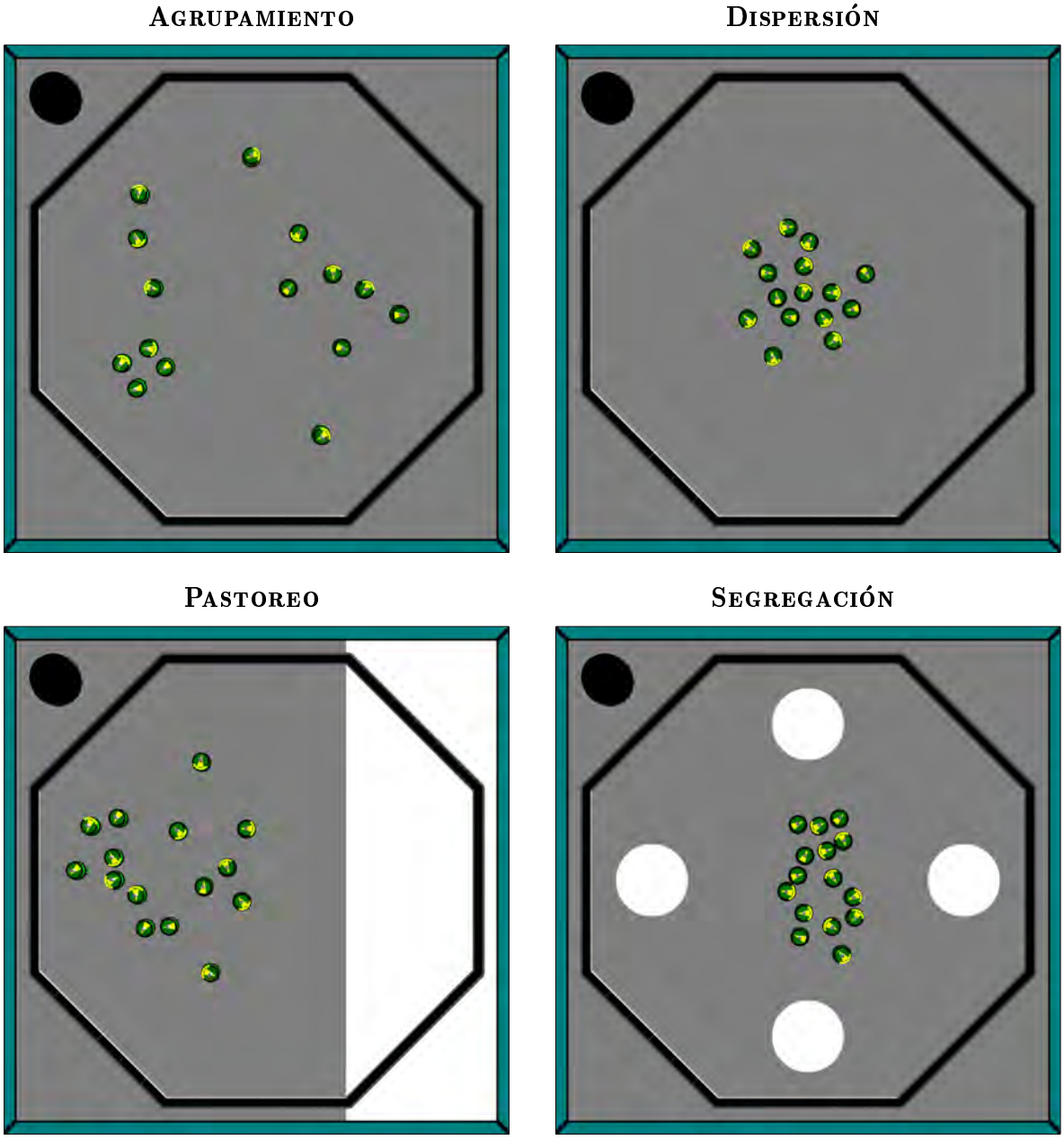


Figura 4-2.: Misiones consideradas en los experimentos: Agrupamiento (arriba-izquierda), Dispersión (arriba-derecha), Pastoreo (abajo-izquierda) y Segregación (arriba-derecha). Las figuras muestran un ejemplo de las posiciones iniciales de el enjambre de 5 robots activos y el enjambre de 10 robots pasivos. Los LEDs de los robots activos se encuentran apagados ($\delta \in \{\varnothing\}$). Los LEDs de los robots pasivos emiten el color amarillo ($\delta \in \{Y\}$). Las posiciones iniciales son seleccionadas de forma aleatoria en cada ejecución de la misión.

contrario, se espera que el enjambre activo sea eficiente separando a los robots del enjambre pasivo cuando estos operan con el comportamiento *C2-Repulsión*.

4.2.3. Pastoreo

El enjambre activo debe guiar a los robots del enjambre pasivo hacia una región de interés. La Figura **4-2** (abajo-izquierda) muestra el escenario experimental para PASTOREO. El enjambre activo y el enjambre pasivo inician la misión estando distribuidos a la izquierda del escenario. La región de interés en donde el enjambre pasivo debe agregarse está marcada con el suelo de color blanco, a la derecha del escenario. La eficiencia F_{pas} del enjambre activo se mide mediante la función objetivo descrita en la ecuación (4-3); cuyo valor debe minimizarse.

$$F_{pas} = K_{\mathrm{n}}(T); \tag{4-3}$$

en donde $K_{\mathrm{n}}(T)$ es el número de robots del enjambre pasivo que se encuentran fuera de la región de interés al finalizar la misión en el tiempo T. *A priori*, se espera el enjambre activo sea eficiente guiando a los robots del enjambre pasivo cuando estos operan con el comportamiento de *C1-Atracción*. Por el contrario, se espera que el enjambre activo sea ineficiente guiando a los robots del enjambre pasivo cuando estos operan con el comportamiento *C2-Repulsión*.

4.2.4. Segregación

El enjambre activo debe dispersar y guiar a los robots del enjambre pasivo hacia cuatro regiones de interés relativamente pequeñas. La Figura **4-2** (abajo-derecha) muestra el escenario experimental para SEGREGACIÓN. El enjambre activo y el enjambre pasivo inician la misión estando concentrados en el centro del escenario. Las regiones de interés en donde el enjambre pasivo debe agregarse son circulares y están marcadas con el suelo de color blanco, en cuatro puntos equidistantes al centro del escenario. La eficiencia F_{seg} del enjambre activo se mide mediante la función objetivo descrita en la ecuación (4-4); cuyo valor debe minimizarse.

$$F_{seg} = K_{\mathrm{n}}(T); \tag{4-4}$$

en donde $K_{\mathrm{n}}(T)$ es el número de robots del enjambre pasivo que se encuentran fuera de las regiones de interés al finalizar la misión en el tiempo T. *A priori*, se espera el enjambre activo sea eficiente guiando a los robots del enjambre pasivo cuando estos operan con el comportamiento de *C1-Atracción*. Por el contrario, se espera que el enjambre activo sea ineficiente guiando a los robots del enjambre pasivo cuando estos operan con el comportamiento *C2-Repulsión*.

4.3. Protocolo

Los cuatro métodos bajo análisis se evalúan bajo las mismas condiciones. Todos los métodos producen software de control para la versión del e-puck descrita en la sección 3.1. En todo el estudio, los métodos usan ARGoS3 como herramienta de simulación y de evaluación de la eficiencia del software de control producido. Todos los métodos son evaluados en las cuatro misiones—Agrupamiento, Dispersión, Pastoreo y Segregación, y en los tres casos de comportamiento del enjambre pasivo—*C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*. En este sentido, en esta investigación se consideran 12 escenarios diferentes para evaluar los métodos de diseño. Ninguno de los métodos bajo estudio es provisto con información sobre los comportamientos del enjambre pasivo. Es decir, los métodos de diseño automático y los diseñadores humanos deben descubrir estos comportamientos durante el proceso de diseño.

Todas las misiones son ejecutadas por un enjambre activo de 5 robots y un enjambre pasivo de 10 robots. `Pistacchio`, `EvoCMY` y `C-Human` son provistos de recursos equivalentes para diseñar el comportamiento colectivo de los robots del enjambre activo. `Pistacchio` y `EvoCMY` cuentan con 100 000 simulaciones para producir una instancia de software de control. El tiempo requerido para ejecutar las 100 000 simulaciones es de aproximadamente 4 h en un cluster computacional de 120 núcleos. Dado que los métodos de diseño automático pueden paralelizar el trabajo, y con el fin de obtener información estadística sobre estos métodos, `Pistacchio` y `EvoCMY` producen 10 instancias de software de control para cada una de los 12 escenarios. En el caso de `C-Human`, cada diseñador de `C-Human` produce 4 instancias de software de control—una por cada misión y al menos una por cada caso de comportamiento del enjambre pasivo. Entre las fases de preparación y experimentación, `C-Human` cuenta con un total de 52 h para producir una instancia de software de control para cada uno de los 12 escenarios. `C-Human` se basa en diseño manual, y como es común en este enfoque, el diseñador no requiere producir instancias diferentes de software de control para resolver una misma misión. `R-Walk` emplea el mismo software de control en las 12 misiones.

4.3.1. Estadística

Después de llevar a cabo los procesos de diseño, cada una de las instancias de software de control producidas por `Pistacchio` y `EvoCMY` se evalúa una vez. Esto permite tener 120 observaciones sobre su eficiencia—dados los 12 escenarios posibles. Para mantener un igual número de observaciones, cada una de las 12 instancias de software de control producidas por `C-Human` se evalúa 10 veces. De forma similar, `R-Walk` se evalúa 10 veces en cada uno de los escenarios para obtener 120 observaciones que sirven de línea base para comparar los demás métodos.

A partir de las 120 observaciones por método, se construyen diagramas de cajas con muescas que representan su eficiencia. De este modo, se reporta la eficiencia de los métodos de diseño

en cada uno de los 12 escenarios. En todos los casos, las comparaciones entre métodos se soportan con comparaciones estadísticas de las distribuciones de las observaciones, con intervalos de confianza del 95 % [91]. Esta comparaciones pueden visualizarse a través de las muescas de los diagramas de cajas: la diferencia de la eficiencia de dos métodos es significativa estadísticamente si las muescas no se solapan.

De forma adicional, se presentan resultados agregados de la eficiencia de los métodos a través de los 12 escenarios. Este análisis permite hacer un juicio general de la efectividad de un método al abordar la clase de misiones que es objeto de estudio. El análisis estadístico de los resultados agregados se soporta con un test de Friedman [91], realizado sobre todas las observaciones de cada método y con intervalos de confianza del 95 %. Dado que el test de Friedman es un test no paramétrico basado en rangos, no requiere que se escale la eficiencia medida a través de las diferentes funciones objetivo, ni que se planten hipótesis restrictivas sobre la distribución de las observaciones. Comparaciones entre métodos basadas en el test de Friedman pueden verificarse a través de los intervalos de confianza delineados en el gráfico correspondiente. La diferencia en la eficiencia de dos métodos es significativa estadísticamente si los intervalos de confianza no se solapan.

5. Resultados y discusión

Este capítulo presenta los resultados obtenidos en la campaña experimental. Estos resultados describen los comportamientos diseñados por `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` desde un punto de vista cuantitativo y cualitativo. Es decir, se analiza la eficiencia de cada método y se describen las características de los comportamientos colectivos diseñados. El capítulo empieza presentando los resultados obtenidos en cada misión, y para los comportamientos *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*. Luego se presentan los resultados agregados, obtenidos a través de los 12 escenarios, y se hace un análisis comparativo de la eficiencia de los métodos de diseño. Finalmente, el capítulo termina con una discusión sobre la relación de los resultados obtenidos con las preguntas de investigación planteadas en esta tesis.

En el contexto de estos resultados, referencias al color de los robots implican que los robots emiten el color especificado con sus LEDs—por ejemplo, *"robots magenta"* se refiere a robots que emiten el color magenta. De igual forma, referencias a *"robots activos"* y *"robots pasivos"* implican que los robots pertenecen al enjambre activo y al enjambre pasivo; respectivamente. El software de control producido, datos recogidos, y videos demonstrativos de los comportamientos de los enjambres se proveen como material suplementario, ver Anexo A.

5.1. Misión I: Agrupamiento

La Figura **5-1** muestra la eficiencia de `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` en Agrupamiento. Los resultados corresponden al diseño de comportamientos colectivos para un enjambre activos que debe coordinar las acciones de un enjambre pasivo; el cual opera con *C1-Atracción* (izquierda), *C2-Repulsión* (centro) y *C3-Atracción&Repulsión* (derecha). En esta misión, `Pistacchio` y `EvoCMY` son significativamente más eficientes que `C-Human` y `R-Walk`, cuando el enjambre pasivo opera con *C2-Repulsión* y *C3-Atracción&Repulsión*. Cuando se compara `Pistacchio` y `EvoCMY`, no existe una diferencia significativa de eficiencia entre estos dos métodos. Los experimentos ejecutados con *C1-Atracción* no arrojaron resultados que muestren una diferencia significativa en la eficiencia de algún método en particular. A continuación se describen los comportamientos colectivos diseñados por cada método. La descripción corresponde a las apreciaciones generales hechas mediante inspección visual, y sobre la ejecución de todas las instancias de software de control producidas.

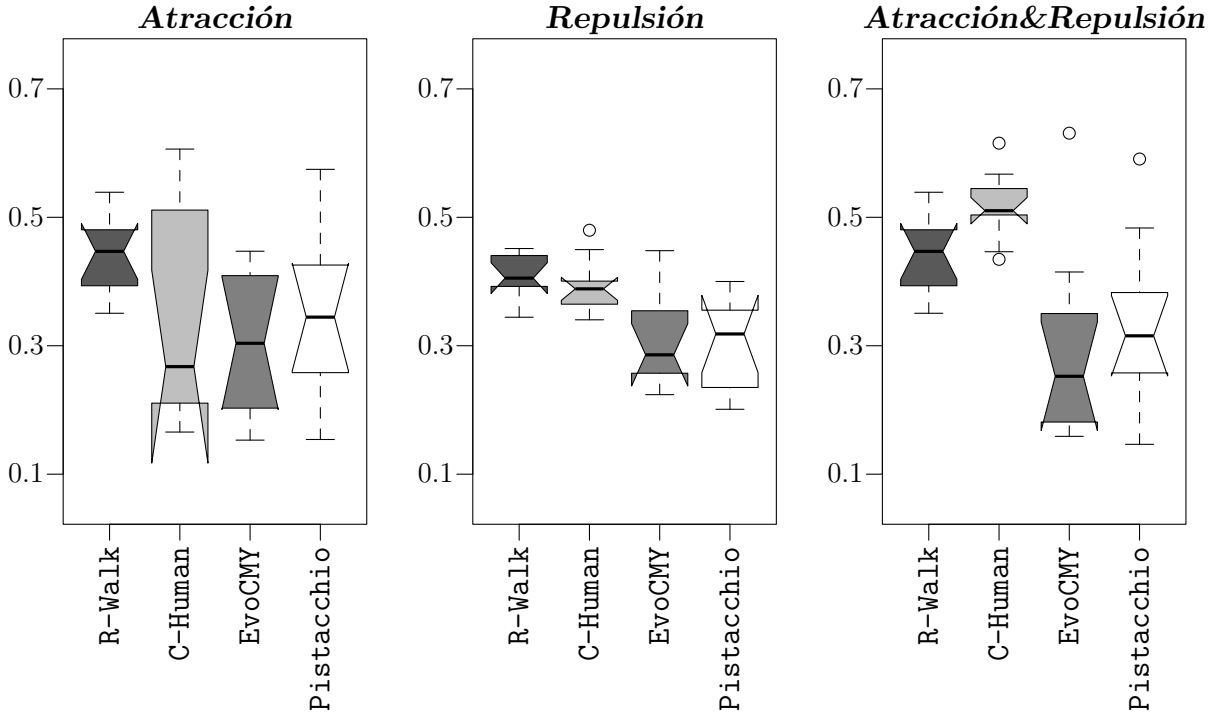


Figura 5-1.: Resultados obtenidos en AGRUPAMIENTO. Los diagramas de cajas muestran la eficiencia de los métodos `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` cuando el enjambre pasivo actúa con los comportamientos *C1-Atracción* (izquierda), *C2-Repulsión* (centro) y *C3-Atracción&Repulsión* (derecha).

5.1.1. `Pistacchio`

***C1-Atracción*:** los robots activos se tornan magenta para atraer a los robots pasivos, y formar así agrupaciones de robots. Para evitar la segmentación del enjambre en subgrupos, los robots activos también se ven atraídos hacia otros robots magenta—es decir, se atraen entre ellos. De esta forma los robots activos y pasivos se mantienen juntos. Los robots pasivos terminan la misión estando cerca del centro de masa del enjambre.

***C2-Repulsión*:** el comportamiento diseñado ocurre en dos fases. Primero, los robots activos se acercan a los robots pasivos siendo atraídos por el color amarillo que estos emiten. En este estado, los robots activos no emiten ningún color que pueda estimular a los robots pasivos. Después, los robots activos se tornan cían y se mueven de forma circular en el centro de la arena experimental. De esta forma logran desplazar a los robots pasivos hacia las paredes. Los robots pasivos terminan la misión estando cerca del centro de masa del enjambre.

***C3-Atracción&Repulsión*:** en la mayoría de ejecuciones, los robots activos operan bajo el comportamiento descrito en *C1-Atracción*.

5.1.2. `EvoCMY`

***C1-Atracción*:** los robots activos se tornan magenta y se desplazan en trayectorias circulares con una amplitud media, mientras son seguidos por los robots pasivos. Cuando los robots activos detectan la presencia de sus pares, se atraen entre ellos y las trayectorias circulares se cierran. Esto les permite acercarse y agrupar a los robots pasivos, manteniendolos cerca del centro de masa del enjambre.

***C2-Repulsión*:** los robots activos se tornan cían y se mueven siguiendo una trayectoria circular cerca de las paredes de la arena experimental. De esta forma, rodean a los robots pasivos, quienes poco a poco se desplazan hacia el centro al ser repelidos por los robots cían. En algunos casos, `EvoCMY` no diseñó comportamientos colectivos que le permitieran a los dos enjambres interactuar. El software de control mantiene estáticos a los robots activos, en su posición inicial. La falla en el proceso de diseño puede estar asociada a una dificultad para encontrar una configuración de la red neuronal que mejore el puntaje cuando el enjambre pasivo mantiene las posiciones iniciales. Aunque existen estos comportamientos en donde el enjambre activo falla en coordinar el enjambre pasivo, la eficiencia general de `EvoCMY` parece ser afectada. Los robots pasivos se mantienen cerca al centro de masa del enjambre.

***C3-Atracción&Repulsión*:** en la mayoría de los diseños, los robots activos operan según el comportamiento descrito en *C1-Atracción*.

5.1.3. `C-Human`

***C1-Atracción*:** los robots activos se tornan magenta y son atraídos por el color amarillo emitido por los robots pasivos. De este modo, se forman agrupaciones de robots activos y pasivos. Dado que los robots activos no interactuan entre ellos, como en `Pistacchio`, las agrupaciones de robots pueden establecerse en lugares diferente. Por este motivo, los robots pasivos no siempre terminan agregados en un solo grupo. Dependiendo de las posiciones iniciales de los robots, estos pueden terminar cerca o lejos del centro de masa del enjambre.

***C2-Repulsión*:** los robots activos se mueven en el entorno sin emitir ningún color, siendo repelidos por el color amarillo de los robots pasivos. El enjambre activo falla en la coordinación del enjambre pasivo, el cuál mantiene sus posiciones iniciales.

***C3-Atracción&Repulsión*:** los robots activos son atraídos hacia los robots pasivos por el color amarillo que estos emiten. En este comportamiento, los robots activos emiten el color amarillo, el cuál no tienen una influencia en el comportamiento de los robots pasivos. Los robots activos mueven a los robots pasivos mediante el comportamiento de evasión de obstáculos embebido en estos últimos—ver el módulo DRIVEN, Tabla **3-2**. Con este comportamiento los robots pasivos tienden a dispersarse, y el enjambre activo falla en agrupar al enjambre pasivo.

5.1.4. `R-Walk`

Los robots activos interactúan de forma mínima con los robots pasivos, y no causan que se desplacen considerablemente. El puntaje obtenido en la ejecución de `R-Walk` en *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión* es únicamente el resultado de las posiciones iniciales de los robots pasivos.

5.2. Misión II: Dispersión

La Figura **5-2** muestra la eficiencia de `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` en DISPERSIÓN. Los resultados corresponden a el diseño de comportamientos colectivos para un enjambre activo que debe coordinar las acciones de un enjambre pasivo; el cual opera con *C1-Atracción* (izquierda), *C2-Repulsión* (centro) y *C3-Atracción&Repulsión* (derecha). En esta misión, `Pistacchio` y `EvoCMY` son significativamente más eficientes que `C-Human` y `R-Walk` en todos los casos. Cuando se compara `C-Human` y `R-Walk`, `C-Human` es significativamente más eficiente que `R-Walk` en *C1-Atracción* y *C2-Repulsión*, pero no existe una diferencia significativa entre ellos en *C3-Atracción&Repulsión*. Al igual que en AGRUPAMIENTO, `Pistacchio` y `EvoCMY` no muestran una diferencia significativa de eficiencia. A continuación se describen los comportamientos colectivos diseñados por cada método.

5.2.1. `Pistacchio`

***C1-Atracción*:** los robots activos emiten el color magenta para atraer a los robots pasivos, y al mismo tiempo son repelidos por el color amarillo que estos emiten. De esta forma, el enjambre pasivo se segmenta en subgrupos que los robots activos dirigen hacia diferentes ubicaciones de la arena experimental. Para evitar que los robots pasivos se queden rezagados, los robots activos alternan este comportamiento. El movimiento se interrumpe en cada transición, y se reduce la velocidad efectiva de desplazamiento. Al estar divididos en subgrupos, los robots pasivos se alejan del centro de masa del enjambre.
***C2-Repulsión*:** los robots activos se tornan cían y se mueven siguiendo trayectorias circulares amplias. De esta forma logran desplazar a los robots pasivos hacia las paredes. Este comportamiento causa que los robots pasivos se alejen del centro de masa del enjambre.
***C3-Atracción&Repulsión*:** los robots operan con el comportamiento descrito en *C2-Repulsión*.

5.2.2. `EvoCMY`

***C1-Atracción*:** los robots activos se tornan magenta y se mueven siguiendo una trayectoria delineada por las paredes de la arena—similar al comportamiento que podría observarse en robots seguidores de línea. Los robots pasivos son atraídos por los robots magenta, y se

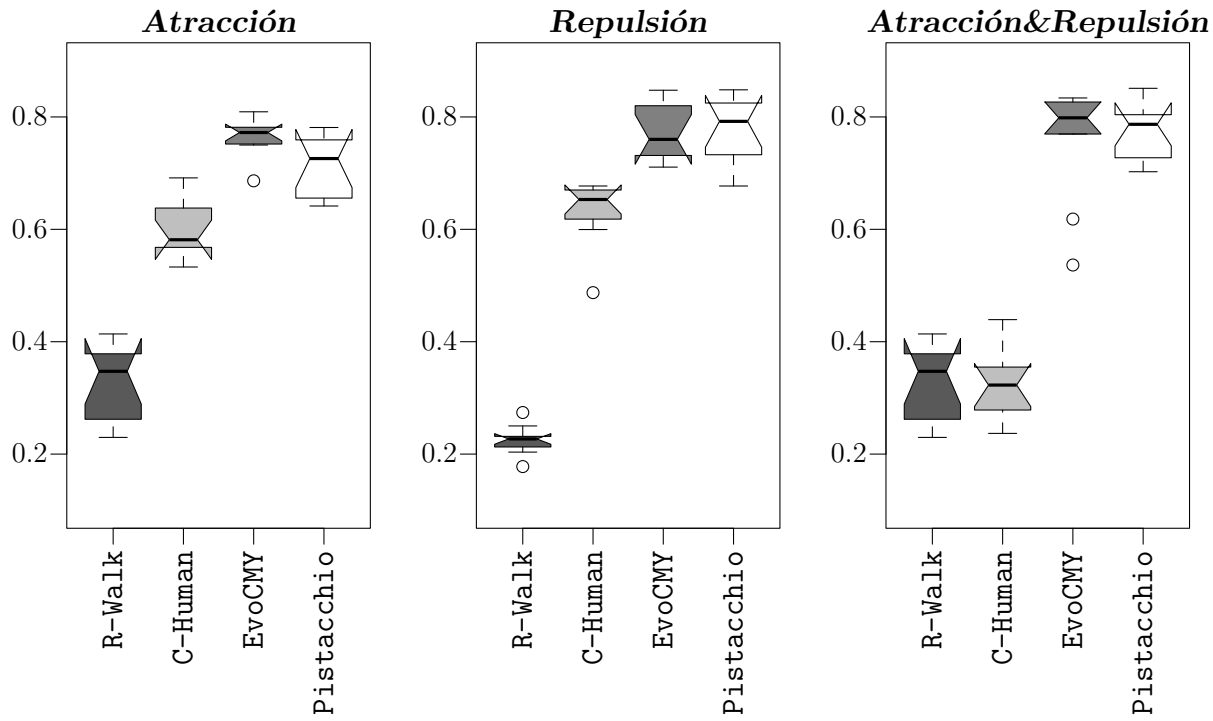


Figura 5-2.: Resultados obtenidos en Dispersión. Los diagramas de cajas muestran la eficiencia de los métodos `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` cuando el enjambre pasivo actúa con los comportamientos *C1-Atracción* (izquierda), *C2-Repulsión* (centro) y *C3-Atracción&Repulsión* (derecha).

ubican también en zonas cercanas a las paredes. De este modo, los robots pasivos se alejan del centro de masa del enjambre.

***C2-Repulsión*:** los robots activos se tornan cían y se mueven siguiendo trayectoria circulares en el centro de la arena experimental. Así, los robots pasivos son repelidos hacia los extremos, ubicándose cerca a las paredes. Este comportamiento es efectivo para separar a los robots pasivos y los mantiene lejos del centro de masa del enjambre.

***C3-Atracción&Repulsión*:** el comportamiento más común de los robots activos es el observado y descrito en *C1-Atracción*.

5.2.3. `C-Human`

***C1-Atracción*:** los robots activos se acercan a los robots pasivos al ser atraídos por el color amarillo que estos emiten. En este comportamiento, los robots activos no emiten ningún color. Por este motivo, los robots pasivos son unicamente desplazados por la acción de evasión de obstáculos. Esta estrategia es ineficiente y desplaza en poca cantidad a los robots pasivos. Los robots pasivos se alejan poco del centro de masa del enjambre.

***C2-Repulsión*:** los robots activos operan con el comportamiento descrito en *C1-Atracción*. ***C3-Atracción&Repulsión*:** los robots activos se tornan cían, y son atraídos hacia los robots pasivos por el color amarillo que estos emiten. De esta forma los robots activos pueden dispersar al enjambre pasivo, aunque no de manera eficiente. A través de estados adicionales, los robots activos cambian de dirección cuando perciben a otros robots cían. Esto les permite alejarse de sus pares y cubrir zonas separadas entre sí. Los robots pasivos se alejan poco del centro de masa del enjambre.

5.2.4. `R-Walk`

Al igual que en AGRUPAMIENTO, los robots activos interactúan de forma mínima con los robots pasivos. En este caso, los robots pasivos tampoco se desplazan considerablemente, y el puntaje obtenido en *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión* depende de las posiciones iniciales de los robots pasivos.

5.3. Misión III: Pastoreo

La Figura **5-3** muestra la eficiencia de `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` en PASTOREO. Los resultados corresponden a el diseño de comportamientos colectivos para un ejambre activo que debe coordinar las acciones de un enjambre pasivo; el cual opera con *C1-Atracción* (izquierda), *C2-Repulsión* (centro) y *C3-Atracción&Repulsión* (derecha). En esta misión, al igual que en las dos misiones anteriores, `Pistacchio` y `EvoCMY` son significativamente más eficientes que `C-Human` y `R-Walk` en todos los casos. La eficiencia de `R-Walk` en esta misión es nula, es decir, los puntajes medidos con la función objetivo son consistentemente los peores posibles. Cuando se compara `C-Human` y `R-Walk`, `C-Human` es significativamente más eficiente que `R-Walk` en *C1-Atracción* y *C3-Atracción&Repulsión*, pero no existe una diferencia significativa entre ellos en *C2-Repulsión*. En este sentido, la eficiencia de `C-Human` es igualmente nula en *C2-Repulsión*. En todos los casos, y al igual que en las misiones anteriores, `Pistacchio` y `EvoCMY` no muestran una diferencia significativa de eficiencia. A continuación se describen los comportamientos colectivos diseñados por cada método.

5.3.1. `Pistacchio`

***C1-Atracción*:** los robots activos se mueven de forma aleatoria y se tornan magenta para atraer a los robots pasivos. Igual que el comportamiento observado en DISPERSIÓN, los robots activos alternan este comportamiento con otros estados para reducir la velocidad efectiva de desplazamiento, y de esta forma evitar que los robots pasivos se queden rezagados. Los robots pasivos siguen a los robots activos hasta la zona blanca, luego apagan sus LEDs y detienen su movimiento. Los robots activos también se detienen poco después de entrar a

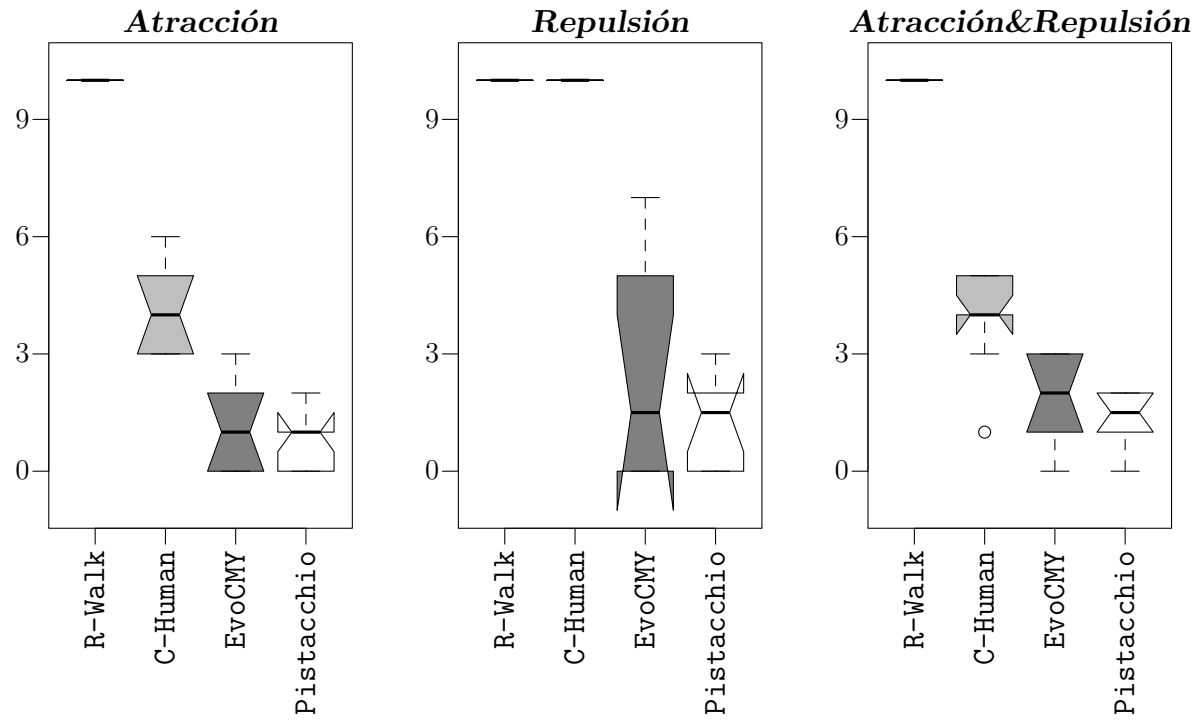


Figura 5-3.: Resultados obtenidos en Pastoreo. Los diagramas de cajas muestran la eficiencia de los métodos `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` cuando el enjambre pasivo actúa con los comportamientos *C1-Atracción* (izquierda), *C2-Repulsión* (centro) y *C3-Atracción&Repulsión* (derecha).

la zona blanca. Esto evita que los robots activos empujen a los robots pasivos fuera de la región de interés; dado que estos se acumulan cerca de la interfaz entre la zona blanca y gris. Los robots pasivos son guiados efectivamente hacia la zona blanca.

***C2-Repulsión*:** los robots activos se tornan cían y se acercan a los robots pasivos, siendo atraídos por el color amarillo que estos emiten. Los robots pasivos son repelidos por el color cían, y poco a poco entran en la región blanca. Una vez ahí se detienen y apagan sus LEDs. Robots activos que entran en la zona blanca salen de ella rápidamente a través de rotaciones que los devuelven a la zona gris. Los robots pasivos son repelidos efectivamente hasta la zona blanca.

***C3-Atracción&Repulsión*:** los robots activos operan con el comportamiento descrito en *C2-Repulsión*.

5.3.2. `EvoCMY`

***C1-Atracción*:** los comportamientos de los robots activos son variados. Se identificaron tres estrategias principales, todas basadas en la emisión del color magenta, pero con diferentes

características en el comportamiento de los robots. En la primera estrategia, los robots activos siguen trayectorias circulares en la arena. Los robots pasivos los siguen y eventualmente alcanzan la zona blanca. Una segunda estrategia se basa en el comportamiento observado en DISPERSIÓN, en el cuál los robots activos usan la pared para moverse en la arena experimental. Un ciclo completo al rededor de la arena asegura que estos alcance la zona blanca, y por tanto, que los robots pasivos los sigan hacia ella. Finalmente, en la tercera estrategia, los robots activos se acercan a los robots pasivos de tal forma que la evasión de obstáculos de los últimos se activa y los desplaza por la arena hasta la región de interés. Con cualquiera de estas tres estrategias, los robots pasivos son guiados efectivamente hacia la zona blanca.
***C2-Repulsión*:** los robots activos también muestran estrategias diferentes para operar en esta misión. al igual que en *C1-Atracción*, los dos comportamientos también aprovechan la reacción de los robots pasivos a robots de un color específico; en este caso cían. En la primer estrategia, los robots activos son atraídos por el color amarillo emiten los robots pasivos. Dado que estos son repelidos por los robots cían, poco a poco son empujados hacia la zona blanca. En la segunda estrategia, los robots activos se mueven de forma aleatoria, pero llevan a los robots pasivos hasta la zona blanca de la misma forma. Al finalizar la misión, en enjambre pasivo está ubicado en la región de interés.
***C3-Atracción&Repulsión*:** los robots activos operan con el comportamiento descrito en *C1-Atracción*.

5.3.3. `C-Human`

***C1-Atracción*:** los robots activos operan de acuerdo a comportamientos ya observados en AGRUPAMIENTO y DISPERSIÓN. Los robots activos no emiten ningún color y se acercan a los robots pasivos al ser atraídos por el color amarillo que estos emiten. Esto causa que los robots pasivos sean desplazados solo su comportamiento de evasión de obstáculos. Esta estrategia es ineficiente y no guía de forma efectiva al enjambre pasivo a la zona blanca.
***C2-Repulsión*:** los robots activos operan con un comportamiento similar al descrito en *C1-Atracción*. Sin embargo, en este caso, el enjambre activo no logra guiar ningún robot del enjambre pasivo hacia la zona blanca. Esta ejecución se considera una falla total en la misión.
***C3-Atracción&Repulsión*:** los robots activos operan con el comportamiento descrito en *C1-Atracción*.

5.3.4. `R-Walk`

De forma similar que en AGRUPAMIENTO y DISPERSIÓN, los robots activos interactúan de forma mínima con los robots pasivos. Dado que no causan un desplazamiento en los robots pasivos, ninguno de estos alcanza la zona blanca y esta misión resulta en un fallo total en *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*.

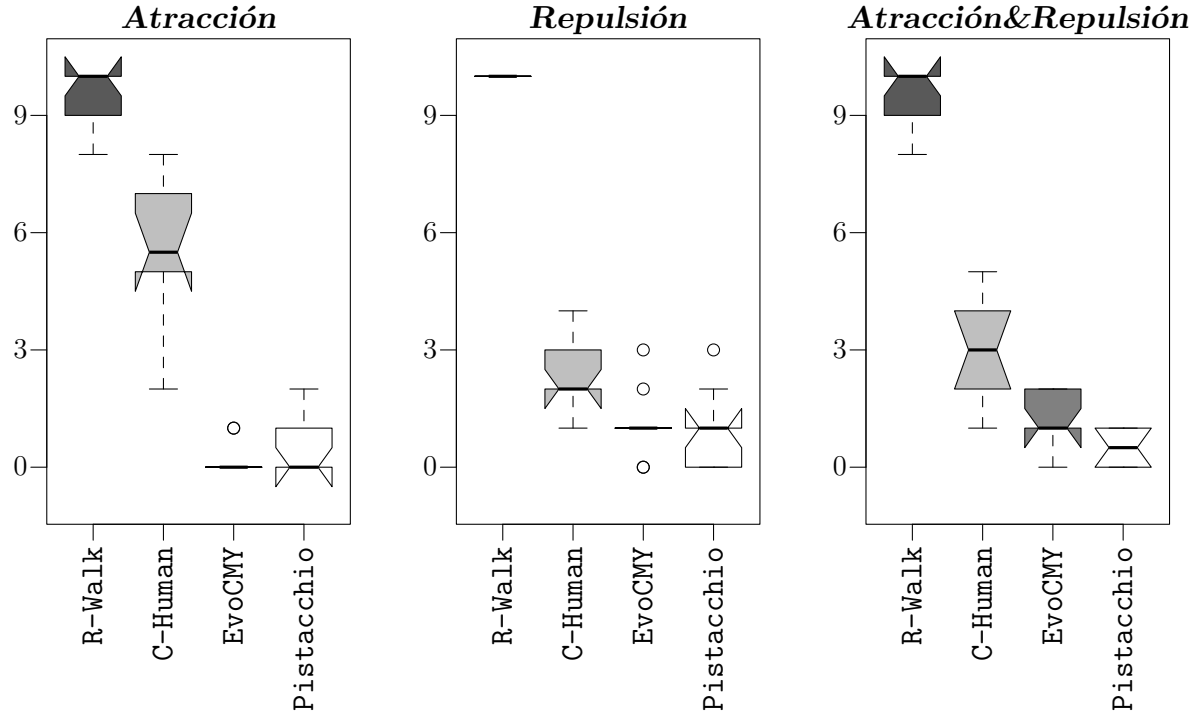


Figura 5-4.: Resultados obtenidos en Segregación. Los diagramas de cajas muestran la eficiencia de los métodos `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` cuando el enjambre pasivo actúa con los comportamientos *C1-Atracción* (izquierda), *C2-Repulsión* (centro) y *C3-Atracción&Repulsión* (derecha).

5.4. Misión IV: Segregación

La Figura **5-4** muestra la eficiencia de `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk` en Pastoreo. Los resultados corresponden a el diseño de comportamientos colectivos para un ejambre activo que debe coordinar las acciones de un enjambre pasivo; el cual opera con *C1-Atracción* (izquierda), *C2-Repulsión* (centro) y *C3-Atracción&Repulsión* (derecha). En esta misión, al igual que en las demás, `Pistacchio` y `EvoCMY` son significativamente más eficientes que `C-Human` y `R-Walk` en todos los casos. `R-Walk` tiene una eficiencia nula cuando los robots del enjambre pasivo operan con *C2-Repulsión*. En todos los casos, `C-Human` es significativamente más eficiente que `R-Walk`. Finalmente, siendo consistente con las demás misiones, `Pistacchio` y `EvoCMY` no muestran una diferencia significativa de eficiencia. A continuación se describen los comportamientos colectivos diseñados por cada método.

5.4.1. `Pistacchio`

***C1-Atracción*:** al igual que en comportamientos observados en otras misiones, los robots activos se mueven de forma aleatoria y se tornan magenta para atraer a los robots pasivos. La particularidad en esta misión es que los robots activos también son repelidos por el color magenta, y esto les permite moverse en diferentes direcciones al iniciar el experimento—alcanzando así las cuatro regiones de interés. En este caso también existe un cambio de estados que les permite reducir la velocidad de movimiento y, de este modo, no dejar atrás a los robots pasivos. Los robots activos detienen su movimiento cuando entran en alguna de las zonas blancas. Esto podría entenderse como una estrategia para evitar sacar de la zona blanca a los robots pasivos que los siguen y que también se han detenido al entrar en la región de interés. Los robots pasivos se ubican en las zonas blancas al finalizar la misión.

***C2-Repulsión*:** los robots activos se tornan de color cían y muestran dos comportamientos diferentes: se acercan a los robots pasivos siguiendo el color amarillo que estos emiten, o de forma alternativa, se mueven con trayectorias circulares en el centro de la arena experimental. En los dos casos, esto causa que los robots pasivos se desplacen por la arena hasta encontrar una de las zonas blancas, en donde luego se detienen. Los robots activos mantienen este comportamiento durante la ejecución de la misión y logran llevar a los robots pasivos hasta las regiones de interés.

***C3-Atracción&Repulsión*:** los robots activos operan con el comportamiento descrito en *C1-Atracción*.

5.4.2. `EvoCMY`

***C1-Atracción*:** los robots activos se tornan magenta y se alejan de otros robots emitiendo el mismo color. Esto les permite alejarse rápidamente del centro de la arena experimental, dirigiéndose hacia las cuatro zonas blancas. Los robots pasivos los siguen en esta trayectoria hasta entrar en ellas, y luego detienen su movimiento. Los robots activos salen de las regiones blancas y siguen operando. Al finalizar la misión, el enjambre pasivo está en las regiones de interés.

***C2-Repulsión*:** los robots activos emiten el color cían para repeler a los robots pasivos, al mismo tiempo que son atraídos por el color amarillo que estos emiten. Esto causa que los robots pasivos se muevan por la arena experimental hasta encontrar una de las zonas blancas, en donde apagan sus LEDs y se detienen. Los robots activos no son atraídos más y siguen en movimiento, y en caso de percibirlos, son nuevamente atraídos por otros robots amarillos. Con este comportamiento, el en enjambre pasivo termina la misión estando en las cuatro regiones de interés.

***C3-Atracción&Repulsión*:** los robots activos operan con el comportamiento descrito en *C1-Atracción*.

5.4.3. C-Human

***C1-Atracción*:** los robots activos emplean una combinación de comportamientos en los que son atraídos hacia los robots pasivos, mediante el color amarillo que estos emiten, de acuerdo a comportamientos ya observados en AGRUPAMIENTO y DISPERSIÓN. Los robots activos no emiten ningún color y se acercan a los robots pasivos al ser atraídos por el color amarillo que estos emiten. Esto causa que los robots pasivos sean desplazados solo por la acción de evasión de obstáculos. Esta estrategia es ineficiente no permite guiar a efectivamente al enjambre pasivo a las zonas blanca.
***C2-Repulsión*:** los robots activos operan con un comportamiento similar al observado en `Pistacchio`. los robots activos se tornan cían para repeler a los robots pasivos, pero al tiempo son atraídos por el color amarillo que estos emiten. Así, el enjambre activo desplaza al enjambre pasivo a través de la arena experimental. Eventualmente, algunos robots del enjambre pasivo entran en las zonas blancas y se quedan ahí. A pesar de que este comportamiento comparte características con el observado en `Pistacchio`, se muestra menos eficiente.
***C3-Atracción&Repulsión*:** los robots activos operan con el comportamiento ya descrito en las otras tres misiones. Los robots activos se acercan a los robots pasivos al ser atraídos por el color amarillo que estos emiten. Mientras ejecutan este comportamiento, los robots activos no emiten ningún color. De este modo solo interactúan con los pasivos mediante la acción de evasión de obstáculos que estos integran. Esta estrategia es ineficiente y no permite guiar efectivamente al enjambre pasivo a las cuatro zonas blancas.

5.4.4. R-Walk

Cómo ha ocurrido en las tres misiones anteriores, los robots activos interactúan de forma mínima con los robots pasivos. No se causan desplazamientos considerables a los robots pasivos, y por tanto, ninguno de estos alcanza la zona blanca. Esto resulta en un fallo total en *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*.

5.5. Resultados agregados

`Pistacchio` y `EvoCMY` diseñaron comportamientos colectivos en donde el enjambre activo coordina de forma efectiva el comportamiento del enjambre pasivo. Los robots activos perciben a los robots pasivos mediante el color amarillo que estos emiten. También, los robots activos emiten los colores específicos que estimulan a los robots pasivos cuando estos operan con *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*. Esto indica que los métodos de diseño automático exploran correctamente el espacio de diseño, y encuentran soluciones que son efectivas para resolver problemas dentro de la clase de misiones planteadas. A pesar de que los comportamientos de *C1-Atracción* y *C2-Repulsión* son completamente diferentes,

`Pistacchio` y `EvoCMY` encontraron soluciones para usarlos en todas las misiones propuestas. Este resultado difiere de las expectativas originales: *a priori*, se esperaba que el enjambre pasivo no podría ser agrupado cuando operara con *C2-Repulsión*, o que no podría ser dispersado eficientemente cuando operara con *C1-Atracción*. Sin embargo, los métodos de diseño automático produciendo software de control que se ajusta a los dos casos, y que no es menos eficiente que agrupar robots pasivos que operan con *C1-Atracción*, o dispersarlos cuando estos operan con *C2-Repulsión*.

En contraste con estos resultados, `C-Human` sólo diseñó software de control adecuado (operando con las señales de colores correctas) en cuatro de los doce escenarios. En este caso, es posible argumentar que el diseño manual por ensayo y error se tornó en un proceso engorroso, repetitivo y difícil. Esto pudo haber llevado a los diseñadores a no explorar de forma exhaustiva el espacio de diseño, y consecuentemente, a no encontrar los comportamientos colectivos adecuados para que los robots activos interactuen con los robots pasivos. Para corroborar esta hipótesis, se preguntó a los diseñadores por los comportamientos que pudieron identificar en los robots pasivos. Los tres diseñadores sólo reconocieron el comportamiento de evasión de obstáculos y la transición causada cuando los robots pasivos entran en una zona blanca. En algunos casos, los diseñadores usaron los LEDs de los robots como medio de visualización y, de forma inadvertida, seleccionaron los colores adecuados para estimular a los robots pasivos. A diferencia de `Pistacchio` y `EvoCMY`, el no tener información previa sobre el comportamiento de los enjambres pasivos fue un problema mayor para `C-Human`. A pesar de que los diseñadores tenían experiencia en la programación de enjambres, y habían estado en contacto con los experimentos realizados con `TuttiFrutti` [13], su experiencia previa no fue suficiente para descubrir la dinámica de comportamiento del enjambre pasivo. Dado que el comportamiento del enjambre pasivo cambiaba de una misión a otra, el conocimiento adquirido no era directamente transferible. Finalmente, podría argumentarse que el factor tiempo fue un limitante para `C-Human`. Sin embargo, esta conclusión queda parcialmente descartada dado que los diseñadores contaron con 48 h durante la fase de preparación; y en la fase de experimentación terminaron los diseños aproximadamente 1 h antes del tiempo provisto. Es decir, es probable que más tiempo de diseño no hubiera tenido un impacto notable.

5.5.1. Test de Friedman

La Figura **5-5** muestra el rango medio de los métodos cuando estos son evaluados agregando todas observaciones obtenidas en la campaña experimental. `Pistacchio` y `EvoCMY` tienen una eficiencia similar, desempeñándose significativamente mejor que `C-Human` y `R-Walk`. A su vez, `C-Human` diseñó software de control que es significativamente mejor que emplear un comportamiento trivial como `R-Walk`. A pesar de las limitaciones observadas en el proceso de diseño manual, los diseñadores fueron capaces de producir software de control que es viable, aunque poco eficiente.

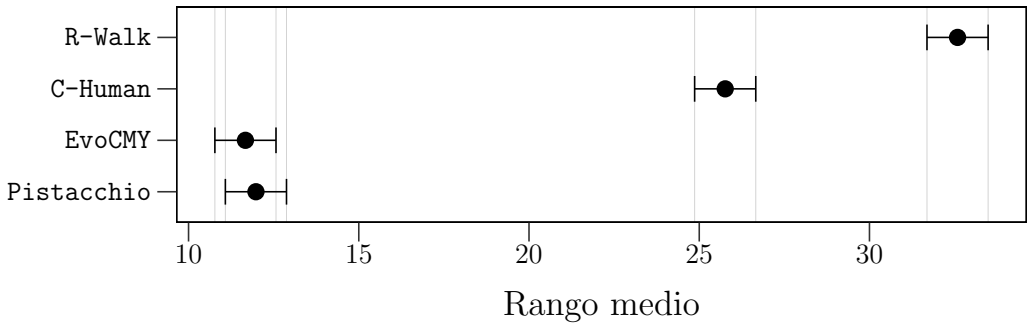


Figura 5-5.: Test de Friedman que agrega los resultados obtenidos en las misiones Agrupamiento, Dispersión, Pastoreo y Segregación, para los comportamientos del enjambre pasivo *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*. El gráfico muestra el rango medio de cada método y su intervalo de confianza. Un método es significativamente mejor que otro si su rango es menor, y si sus intervalos de confianza no se solapan.

El análisis cuantitativo permite afirmar que, en la clase de misiones considerada, los métodos de diseño automático son más eficientes que los métodos de diseño manual. La comparación mediante simulaciones entre el diseño modular, del cuál se desprende `Pistacchio`, y el diseño por neuroevolución, al cual pertenece `EvoCMY`, no arrojó ninguna diferencia significativa respecto a su eficiencia. Esto contrasta con estudios anteriores que hacen comparaciones similares [20, 61], en dónde los métodos neuroevolutivos han resultado significativamente mejores que los modulares cuando los experimentos se realizan en simulación. Estas observaciones están presentes en la comparación entre `TuttiFrutti` y `EvoColor`, los métodos que sirvieron de inspiración para crear `Pistacchio` y `EvoCMY`.

En la mayoría de las misiones, `Pistacchio` y `EvoCMY` alcanzan una eficiencia cercana a la esperada por un comportamiento óptimo. Es decir, los robots alcanzan la eficiencia que minimiza la función objetivo. Esta condición solo puede observarse de forma empírica, dado que en ambos casos los métodos de optimización son heurísticos y no aseguran alcanzar un valor óptimo. A partir del análisis de los resultados, es posible argumentar que los dos algoritmos de optimización han convergido a una región del espacio de diseño en donde no es posible mejorar el comportamiento colectivo a través de la sintonización de los parámetros del software de control. Por este motivo, aunque típicamente los métodos neuroevolutivos suelen explotar mejor las características del simulador para aumentar la eficiencia, en este caso no pueden mejorar mucho más la eficiencia alcanzada. Una pregunta que aún queda abierta es si al desplegar el software de control en robots físicos, también se apreciaría la caída de rendimiento observada en los estudios que comparan el diseño modular y la neuroevolución—el cual tiende a afectar significativamente más a los métodos neuroevolutivos [13, 20, 61].

5.6. Consideraciones finales

En esta última sección se abordan las preguntas de investigación planteadas en capítulo introductorio de la tesis (ver Capítulo 1). Estas consideraciones finales son presentadas en forma de una discusión abierta.

¿Qué relación existe entre la plataforma robótica, la tarea que debe cumplir el enjambre, y el método de diseño automático?

Este problema ha sido abordado también en otros estudios sobre diseño automático—por ejemplo, con la introducción de `TuttiFrutti` [13] para estudiar comportamientos colectivos basados en señales de colores y con `Gianduja` [14] para estudiar la emergencia de comunicación entre robots. Sin embargo, la pregunta había sido abordada unicamente para el caso del enjambre homogéneo, es decir, para el caso en que todos los robots son morfológicamente idénticos y operan con el mismo software de control. En esta tesis se logra demostrar que la filosofía detrás de estos métodos de diseño puede emplearse también para el caso de enjambres cuasi-homogéneos, en los que los robots son morfológicamente idénticos, pero que operan con software de control que difiere en algunos individuos. Durante el proceso de diseño, `Pistacchio` y `EvoCMY` identificaron los estímulos a los que puede responder el e-puck—según las especificaciones del RM 3.3, y diseñaron software de control adecuado para influir en el comportamiento de los robots pasivos en cada misión. Una siguiente etapa de esta investigación podría considerar el uso robots con diferentes capacidades sensoriales y/o de actuación—por ejemplo, e-pucks que dotados con módulos de feromonas artificiales [72].

¿Qué algoritmos de optimización y qué arquitecturas de control pueden emplearse en estos procesos?:

En esta tesis se implementaron dos métodos de diseño: `Pistacchio`, perteneciente a AutoMoDe [11, 22], y `EvoCMY`, de tipo neuroevolutivo [23, 62]. `Pistacchio` produce software de control con la forma de máquinas probabilísticas de estados finitos que se ensamblan a partir de módulos de software preexistentes. Al igual que otros métodos en AutoMoDe, `Pistacchio` emplea Iterated F-race como método de optimización para ensamblar, evaluar, y refinar las máquinas de estado. Por el contrario, `EvoCMY` produce software de control en forma de redes neuronales artificiales, en dónde los parámetros de la red son determinados por un proceso de optimización basado en evolución artificial. Métodos en AutoMoDe y métodos neuroevolutivos habían sido ya empleados con éxito para diseñar comportamientos colectivos para enjambres de robots. Los resultados de esta investigación indican que sus arquitecturas de control también son una opción viable para diseñar comportamientos colectivos para enjambres de robots que coordinan a otros enjambres. Dado que esta tesis provee un marco experimental para evaluar este problema, líneas futuras de investigación pueden

dedicarse a evaluar la viabilidad de otras arquitecturas de control y algoritmos de optimización empleadas anteriormente en diseño automático. Por ejemplo, software de control que toma la forma de arboles de comportamiento [16, 63], o algoritmos de optimización basados en otras heurísticas [17, 18].

¿Qué tan eficientes son estos métodos en comparación a métodos de diseño manual?:

Los resultados obtenidos muestran que, dentro del diseño experimental planteado, los métodos de diseño automático son significativamente más eficientes que el diseñó manual. Este resultado concuerda con lo observado en estudios anteriores que han considerado métodos de diseño y protocolos experimentales similares [20]. A partir de el análisis cualitativo de los resultados, es posible inferir que la dificultad para identificar el comportamiento colectivo del enjambre pasivo fue la causa fundamental de la diferencia en eficiencia entre `C-Human` y los métodos automáticos `Pistacchio` y `EvoCMY`. `C-Human` y `Pistacchio` operan con los mismos módulos de software y diseñan software de control bajo la misma arquitectura, entonces, potencialmente podrían producir software de control con la misma eficiencia. Sin embargo, al comparar los dos métodos, se encontró una diferencia en la habilidad de ambos para descubrir el funcionamiento del enjambre pasivo en cada misión, y consecuentemente, para diseñar un comportamiento colectivo acorde. Dado que los algoritmos de optimización pueden explorar un mayor espacio de diseño, estos pueden encontrar los comportamientos colectivos adecuados con mayor efectividad.
Cabe resaltar que en el diseño experimental de esta tesis, la comparación entre `Pistacchio`, `EvoCMY` y `C-Human` no está completamente balanceada. En cada escenario, `Pistacchio` y `EvoCMY` fueron evaluados con 10 instancias independientes de software de control que se ejecutan 1 vez. En cambio, `C-Human` fue evaluado en cada escenario con 1 instancia de software de control que se ejecuta 10 veces. Esta decisión fue tomada dadas las limitaciones existentes para realizar experimentos con diseño manual, contando tres participantes—experimentos anteriores han contado con 5 participantes en el mejor de los casos [20, 63]. Para consolidar las resultados experimentales de la comparación entre métodos manuales y automáticos, se requeriría de 10 instancias de diseño manual por cada escenario—en total, 120 diseñadores humanos que produzcan 120 instancias independientes de software de control. Una campaña experimental de esa dimensión está fuera del alcance de esta tesis y se deja abierta como trabajo futuro.

¿Pueden estos métodos ser empleados para diseñar el comportamiento colectivo de enjambres que interactúan con otros robots?:

A través de esta investigación se desarrolló un marco experimental que permite investigar en el diseño de comportamientos colectivos para grupos de enjambres que operan en conjunto. El marco experimental fue evaluado con tres métodos de diseño (`Pistacchio`, `EvoCMY` y

`C-Human`), cada uno concebido bajo una metodología de diseño distinta. Los resultados de los experimentos mostraron que, dependiendo de las especificaciones del robot, es posible encontrar diferentes tipos de interacción entre dos enjambres de robots que operan en un mismo entorno. En esta tesis se plantearon escenarios en donde un enjambre activo coordina las acciones de un enjambre pasivo. El comportamiento del enjambre pasivo estaba predefinido. Los tres tipos de comportamiento pasivos permitían dinámicas de interacción completamente opuestas, como *C1-Atracción* y *C2-Repulsión*, o una combinación de ambas, como es *C3-Atracción&Repulsión*. En todos los casos, los dos enjambres debían interactuar para poder ejecutar de forma efectiva la misión propuesta: las métricas de rendimiento se medían sobre las acciones del enjambre pasivo, pero los robots pasivos no tomaban acción a menos que fueran estimulados por robots del enjambre activo. Entonces, la coordinación entre los enjambres era indispensable para que pudieran ejecutar las misiones de forma eficiente. Este tipo de misiones no se habían abordado antes desde el punto de vista del diseño automático *off-line* de enjambres de robots [9]. En esta tesis, las misiones y comportamientos del enjambre pasivo fueron construidos manualmente. Trabajo futuro podría centrarse en reducir la posible varianza de los resultados y conclusiones, la cual está asociada al diseño experimental. Para esto podrían emplearse generadores de misiones aleatorias y de comportamientos aleatorios para el enjambre pasivo; que hagan un muestreo mayor sobre las posibles combinaciones de misión/comportamiento [92]. Este tipo de análisis ayudaría cuantificar el grado de generalización de las conclusiones de esta tesis.

¿Es posible diseñar a un grupo de robots que coordina a otro grupo de robots, sin conocer cómo estos últimos han sido programados?:

En esta tesis, `Pistacchio`, `EvoCMY` y `C-Human` diseñaron comportamientos colectivos para el enjambre activo sin haber contado con información explícita sobre el funcionamiento del enjambre pasivo. Aunque `Pistacchio` opera con módulos de software similares a los que emplea el enjambre pasivo, este no es un factor clave para diseñar comportamientos efectivos. Prueba de esto es que `EvoCMY` alcanza la misma eficiencia que `Pistacchio`, a pesar de que diseña software de control con una arquitectura de control diferente. Los tres métodos diseñaron software de control que, aunque menos eficiente en `C-Human`, le permite al enjambre activo coordinar las acciones del enjambre pasivo. Entonces, desde este punto de vista, se puede afirmar que si es posible diseñar un grupo de robots que puede coordinar a otro, sin conocer como este segundo ha sido programado.
Sin embargo, esta pregunta aún no puede darse por resuelta completamente. Los experimentos planteados en esta tesis parten de la hipótesis de que (i) existe un modelo del comportamiento del enjambre pasivo y (ii) dicho modelo puede emplearse para ejecutar simulaciones en los métodos de diseño. En esta tesis, el modelo del comportamiento del enjambre pasivo fue creado manualmente. Para resolver esta pregunta de forma más general, se requiere una nueva metodología que pueda modelar el comportamiento colectivo del enjambre pasivo sin

necesidad de la intervención manual—por ejemplo, a partir del análisis de vídeos preexistentes [93]. Esta nueva metodología permitiría concebir métodos de diseño que tengan un grado mayor de de automatización al momento de producir el software de control.

6. Conclusiones

La investigación documentada en esta tesis presenta una comparación empírica de métodos de diseño de comportamientos colectivos para enjambres de robots. Estos métodos son evaluados en la producción de software de control para un enjambre de robots que debe coordinar las acciones de otro enjambre. Este tipo de problema ha sido abordado con anterioridad en la literatura, sin embargo, no había sido abordado desde la perspectiva del diseño automático. Los métodos de diseño automático considerados diseñaron comportamientos colectivos en los cuales un enjambre interactua de forma efectiva con un grupo de robots que está presente en su entorno de trabajo. El diseño de los comportamientos colectivos se basó en la identificación de dinámicas relevantes para establecer interacciones entre los dos grupos de robots. Así, los métodos de diseño permitieron resolver misiones en los que un enjambre activo coordina a un enjambre pasivo, aún cuando el método de diseño no había sido provisto explícitamente con información sobre el funcionamiento del enjambre pasivo. Algunos trabajos relevantes sobre este problema se presentaron en el Capítulo 2, en donde también se hace una introducción al diseño de enjambres de robots y a los problemas de coordinación entre enjambres.

En esta tesis se planteó un marco experimental para estudiar el diseño de comportamientos colectivos para grupos de enjambres de robots que operan de forma conjunta. Estudios anteriores en el diseño automático de enjambres de robots sólo habían abordado la producción de software de control para enjambres homogéneos. Esta tesis provee los medios para desarrollar nuevos estudios que involucren: (i) enjambres activos y pasivos operando en el mismo entorno; (ii) la producción independiente de software de control para los dos tipos de enjambres; (iii) experimentación con diferentes comportamientos colectivos en los enjambres pasivos; (iv) un conjunto de misiones que, para ser ejecutadas, requieren la coordinación de los dos tipos de enjambre. Este diseño experimental se presentó en el Capítulo 4, en donde se introdujeron los tres comportamientos para el enjambre pasivo—*C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*; y las cuatro misiones que sirvieron de caso de estudio—Agrupamiento, Dispersión, Pastoreo y Segregación.

La comparación empírica entre métodos se desarrolló en torno a un eje principal: determinar la habilidad de los métodos para establecer interacciones adecuadas entre los grupos de enjambres. Para esto se incluyeron dos métodos de diseño automático—`Pistacchio` y `EvoCMY`, un método de diseño manual—`C-Human`, y una línea base—`R-Walk`. Los métodos bajo estudio diseñaron comportamientos colectivos para un enjambre de robots 5 e-pucks

que debía coordinar las acciones de otro enjambre de 15 e-pucks. `Pistacchio` es introducido por primera vez en esta tesis, siendo un método creado bajo la metodología de diseño modular AutoMoDe y estando inspirado en `TuttiFrutti`—un método concebido para producir software de control para robots que pueden interactuar a través de señales de colores. `EvoCMY` es un método neuroevolutlivo también presentado por primera vez en esta tesis, y está inspirado en `EvoColor`—la contraparte de `TuttiFrutti` en el diseño mediante neuroevolución. `C-Human` es un método de diseño manual, en el que diseñadores humanos usan los módulos de software incluidos en `Pistacchio` para producir software de control—este método se toma de comparaciones anteriores entre métodos automáticos y manuales. Cada método fue descrito en profundidad en el Capítulo 3, y sus implementaciones se han puesto a disposición de la comunidad académica en forma de software libre, gratuito y con código abierto. `Pistacchio` es el método sobre el cual se construyó el marco experimental, y sus módulos pueden ser empleados para diseñar nuevos comportamientos colectivos para el enjambre pasivo y el enjambre activo.

La campaña experimental se desarrolló unicamente con simulaciones, y los resultados se presentaron acompañados de un análisis cuantitativo y cualitativo. El análisis cuantitativo permitió determinar que en los resultados agregados, `Pistacchio` y `EvoCMY` son significativamente mejores que `C-Human`, y que `C-Human` es a su vez significativamente mejor que `R-Walk`. No se encontró una diferencia significativa entre la eficiencia de `Pistacchio` y `EvoCMY`. El análisis cualitativo permitió determinar que `Pistacchio` y `EvoCMY` identificaron de forma correcta los estímulos necesarios para coordinar las acciones del enjambre pasivo, y las aprovecharon para ejecutar las misiones de forma eficiente. Las misiones propuestas son estrictamente cooperativas: las métricas de eficiencia se miden sobre las acciones de los robots pasivos, pero estos robots no toman acción a menos que sean estimulados por los robots activos. Es decir, la interacción entre los dos grupos de robots se torna fundamental para ejecutar la misión de forma eficiente. Por esto es posible afirmar que los métodos automáticos fueron efectivos en el diseño de comportamientos colectivos donde un enjambre coordina a otro. Tres diseñadores humanos participaron en los experimentos de diseño manual de comportamientos colectivos con `C-Human`. A partir de la definición de la tarea, los diseñadores produjeron software de control para los mismos escenarios que `Pistacchio` y `EvoCMY`. Sin embargo, no lograron identificar los comportamientos *C1-Atracción*, *C2-Repulsión* y *C3-Atracción&Repulsión*. Sus soluciones mostraron una interacción limitada entre los dos enjambres de robots, y esto se reflejó en la baja eficiencia de los resultados obtenidos con `C-Human`. Los resultados y una discusión alrededor de las preguntas de investigación planteadas fueron presentados en el Capítulo 5.

6.1. Trabajo futuro

Con esta investigación se espera contribuir a la investigación en el diseño de comportamientos colectivos para enjambres de robots. La automatización de los procesos de producción del software de control tiene el potencial para ayudar en la estandarización de los procedimientos, mitigar posible sesgo introducido por los desarrolladores del software, y reducir el tiempo de realización de estos sistemas. Esta tesis provee a la comunidad científica con métodos de diseño de software de control para grupos de enjambres de robots que deben operar de forma coordinada en un mismo entorno. El marco experimental, los métodos de diseño, y las ideas documentadas en esta tesis podrían ser extendidas mediante (i) el diseño de software de control con nuevas arquitecturas u algoritmos de optimización, (ii) la extensión del número de experimentos que comparan los métodos manuales y automáticos, (iii) el muestreo y evaluación de un mayor número de misiones y clases de comportamiento del enjambre pasivo, (iv) la automatización del modelado del comportamiento del enjambre pasivo, y (v) el estudio de los efectos del *reality-gap* al portar el software de control a los robots físicos. Una discusión más detallada sobre estas líneas futuras de trabajo se presentó en las consideraciones finales del Capítulo 5.

A. Anexo: material suplementario

Este anexo es una referencia al material suplementario de la tesis, el cual se encuentra disponible en el siguiente enlace:

`https://dgarzonramos.github.io/un-enjambres`

El material suplementario disponible es:

- Implementaciones de `Pistacchio` y `EvoCMY`.
- Software de control producido por `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk`.
- Datos recogidos durante los experimentos.
- Videos demostrativos de las ejecuciones experimentales realizadas con `Pistacchio`, `EvoCMY`, `C-Human` y `R-Walk`.

El software producido se distribuye de forma libre, gratuita y con código abierto.

B. Anexo: instrucciones y cuestionario empleado en los experimentos de C-Human

Este anexo presenta la guía instructiva provista a los tres diseñadores humanos para producir el software de control evaluado en C-Human. Durante los experimentos, los diseñadores tuvieron acceso remoto a un ordenador preparado con el software necesario para diseñar los comportamientos colectivos del enjambre de robots activos.

Manual Design of Control Software for Robot Swarms

These are instructions and relevant information to conduct an experiment on the manual design of control software for robot swarms. The particularity of this experiment is that the robot swarm you will operate, hereinafter referred to as *the active swarm*, must coordinate the actions of a second robot swarm that is also present on its workspace, hereinafter referred to as *the passive swarm*.

In the experiment, you are asked to design control software in four experimental scenarios. The scenarios differ in the mission that the robot swarm must perform. For each mission, you are given a performance metric that indicates how good the active swarm has performed in an individual experimental run. All missions are described by a performance metric that is related to the ability of the active swarm to coordinate the passive swarm. The behavior of the passive swarm is not disclosed during the experiment.

In the experiment, you are given up to four hours to design the control software for all the missions. This means that you have up to one hour for each experiment, however, you are free to distribute the time in the way you prefer.

In the following, you are given information of the software architecture you can use to design the control software of the robots, the pre-existing low-level behaviors that you can combine to produce it, the experimental scenarios, and the experimental protocol.

Software Architecture

You can design control software in the form of a probabilistic finite-state machine. In this architecture, states (nodes) represent low-level behaviors that the robots execute, and transition conditions (edges) represent events that trigger the change from one behavior to another. Each low-level behavior and transition condition is a parametric software module. You can design finite-state machines that have up to 4 states, and up to 4 outgoing transitions from each state.

The set of software modules that are available during the experiment are 5 low-level behaviors (nodes) and 5 transitions (edges). The control software for the robots can be produced by assembling and configuring these software modules into a finite-state machine.

Pre-existing software modules

The following table describes the low-level behaviors:

Behaviors (nodes)	**Parameters**	**Description**
Exploration	*rwm, cle*	The robot moves by random walk.
Stop	*cle*	Robot stops its movement and remains standing still.
Circling	*cle, ang*	The robot moves in circles at a steady velocity.
GoToColor	*clr, cle*	The robot steadily moves toward objects displaying a specific color.
GoAwayFromColor	*clr, cle*	The robot steadily moves away from objects displaying a specific color.

- The parameter *rwm* $\in$ *{1, 100}* determines the maximum number of time steps that a robot rotates when it faces an obstacle. This parameter is used when the robot is executing the behavior Exploration. In a simulation, the actual number of time steps is selected at random between 1 and the value of *rwm*.
- The parameter *ang* $\in$ *{-3.13, 3.13}* determines the angle at which robots aim to rotate in when executing Circling. More precisely, it determines the direction of desired displacement vector. However, the real displacement is constrained by the kinematics of the robot.
- The parameter *clr* $\in$ *{C, M, Y},* determines the color that the robot follows when executing GoToColor and the color that the robot avoids when executing GoAwayFromColor.
    - *clr* = 4 -> Yellow
    - *clr* = 5 -> Magenta
    - *clr* = 6 -> Cyan
- All low-level behaviors enable the robot for displaying a color with its LEDs. The parameter *cle* $\in$ *{0, C, M, Y},* determines the color that the robot displays.
    - *cle* = 0 -> No color displayed
    - *cle* = 4 -> Yellow
    - *cle* = 5 -> Magenta
    - *cle* = 6 -> Cyan

The following table shows the description of the condition transitions:

Transitions (Edges)	**Parameters**	**Description**
WhiteFloor	β	The robot detects white floor.
GrayFloor	β	The robot detects gray floor.
BlackFloor	β	The robot detects black floor detected.
ColorDetected	β, *clr*	The robot detects objects of a specific color.
FixedProbability	β	The robot transitions its behavior with a fixed probability.

In all the transitions, the parameter $\beta \in [0, 1]$ determines the probability of transitioning when the transition condition is fulfilled.

- The parameter $clr \in \{C, M, Y\}$, determines if the robot perceives objects of a certain color.
  - *clr* = 4 -> Yellow
  - *clr* = 5 -> Magenta
  - *clr* = 6 -> Cyan

Example of a finite-state machine.

In this example we show a typical finite-state machine for a robot swarm that explores the environment, stops moving when it steps into a region in which the floor is white, and returns to explore with a fixed probability.

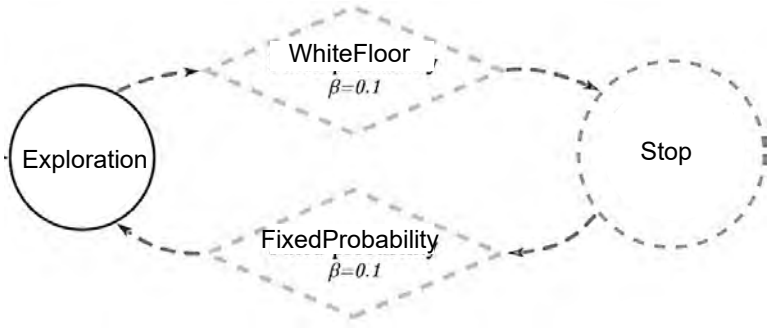
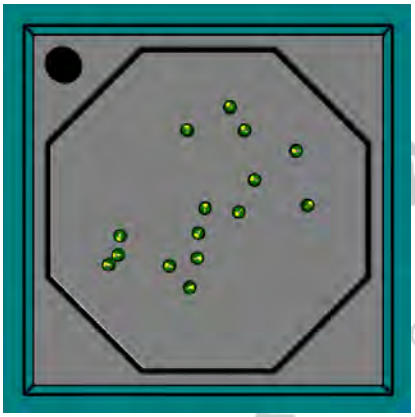


Figure 1: An example of a typical finite-state machine.

Experimental scenarios

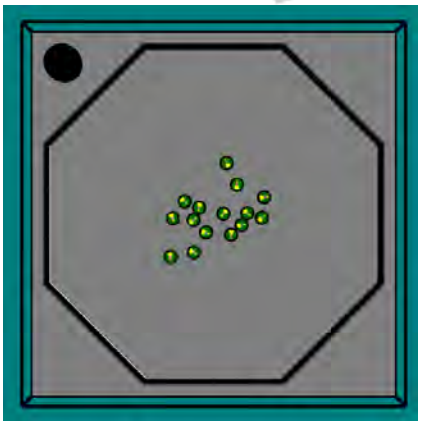
This section introduces the four missions that the active swarm must perform. You must design a suitable control software for each of them. In each mission, 5 active robots must coordinate the behavior of 15 passive robots. The robots operate in an octagonal arena. The duration of an experimental of any mission is 120 seconds. The experiments are performed using the ARGoS multi-robot simulator. The simulation is conducted with a time step of 0.01s.

1. Scenario 1: Clustering



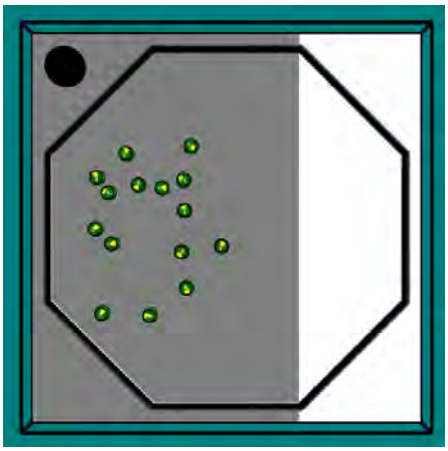
The goal of this mission is to cluster the passive robots so that they remain as close as possible from each other. The score is computed with the average distance from each passive robot to the center of mass of the swarm formed by all passive robots. The performance is measured only once at the end of the experimental run. In this mission, you are asked to **minimize** the score. At the beginning of each experimental run, the robots are positioned randomly in the arena.

2. Scenario 2: Dispersion



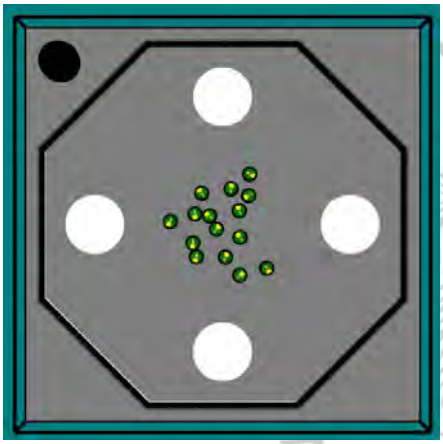
The goal of this mission is to disperse the passive robots so that they remain as far as possible from each other. The score is computed with the average distance from each passive robot to the center of mass of the swarm formed by all passive robots. The performance is measured only once at the end of the experimental run. In this mission, you are asked to **maximize** the score. At the beginning of each experimental run, the robots are positioned randomly in the center arena.

3. Scenario 3: Herding



The goal of this mission is to aggregate the passive robots in the white region. The score is determined by the number of passive robots that have not reached the aggregation region by the end of the experiment. The performance is measured only once at the end of the experimental run. In this mission, you are asked to **minimize** the score. At the beginning of each experimental run, the robots are positioned randomly in the left/gray side of the arena.

4. Scenario 4: Stop



The goal of this mission is to scatter the passive robots in the four white regions. The score is determined by the number of passive robots that have not reached any of the four white regions by the end of the experiment. The performance is measured only once at the end of the experimental run. In this mission, you are asked to **minimize** the score. At the beginning of each experimental run, the robots are positioned randomly in the center of the arena.

Protocol

- You are asked to design control software for each of the four experimental scenarios; that is, for each mission mentioned above.
- The control software must be produced using the architecture and modules described in this instructive document.
- You are given a total of up to 4 hours to design the control software for the four missions, you are free to decide the order and the time you spend on each.
- There is no limit in the minimum time to spend on each mission. You are free to finish the experiment and deliver the control software once you are satisfied with the control software you have designed.
- Communicating and sharing information during the execution of the experiments is strictly forbidden.

Instructions

I have setup a computer where you will conduct your experiments. The computer has been prepared with a visualization tool that you can use to design the control software and test it.

The visualization tool will allow you to easily drag and drop nodes and edges to create a finite-state machine and execute it in the ARGoS simulator. You can modify/fine-tune your designed as much as you wish.

Launching the Visualization Tool

You have been given an identifier for these experiments: Designer 1, 2, or 3. Once you are logged into the computer, open the terminal and enter the directory of the experiments that corresponds to your identifier.

The following commands illustrate how to enter the directory of the Designer 1 and to start the backend of the Visualization Tool for the Scenario 1. The same set of commands works for the other designers and scenarios, you only must change the number at the end of each word.

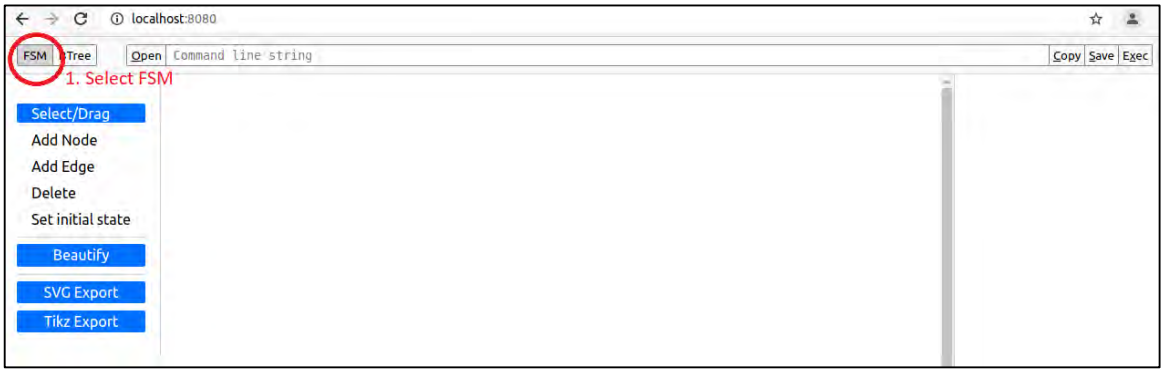
```
cd ~/hdesigner/designer_1/scenario_1

npm start
```

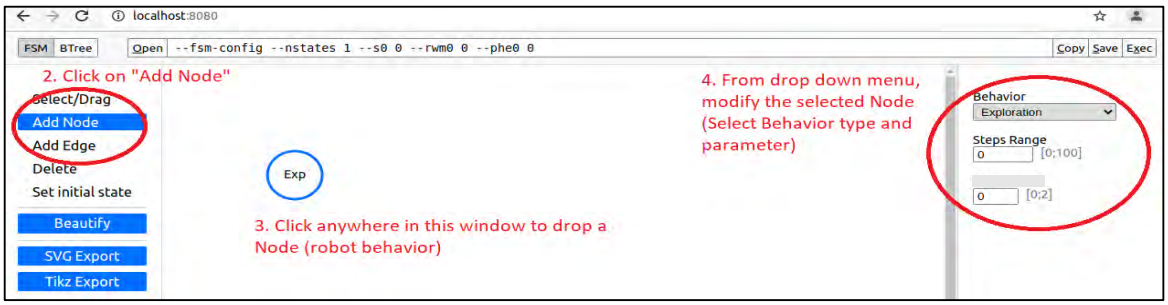
Once the backend is running, you can launch the graphical user interface by opening Firefox and entering the following URL.

```
localhost:8080
```

The visualization tool will appear in Firefox browser. First, select the finite-state machine architecture (FSM).



Then, start adding nodes (low-level behaviors) to your finite-state machine by following the instructions below. Nodes are represented with circular shapes.

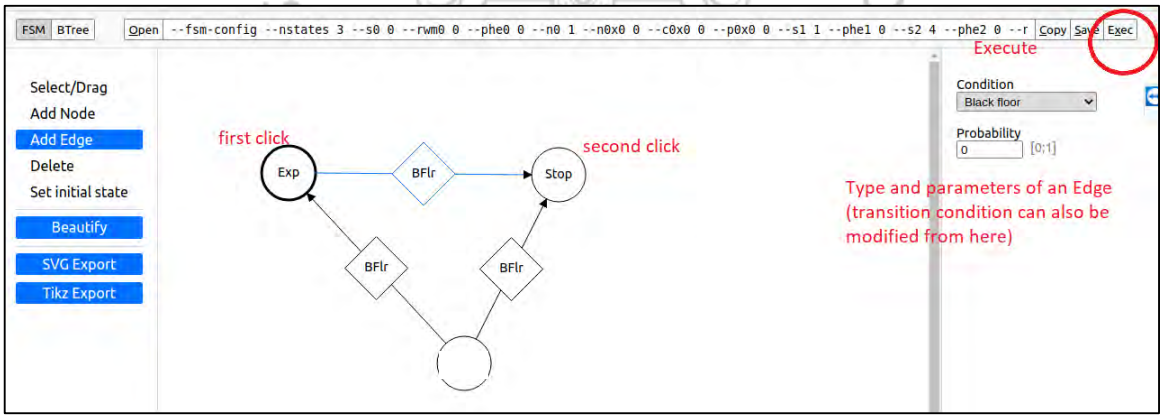


Once you have placed your desired nodes, you can click on “Add Edge” to add connections to configure the transitions between them. Edges are represented with rhomboids.

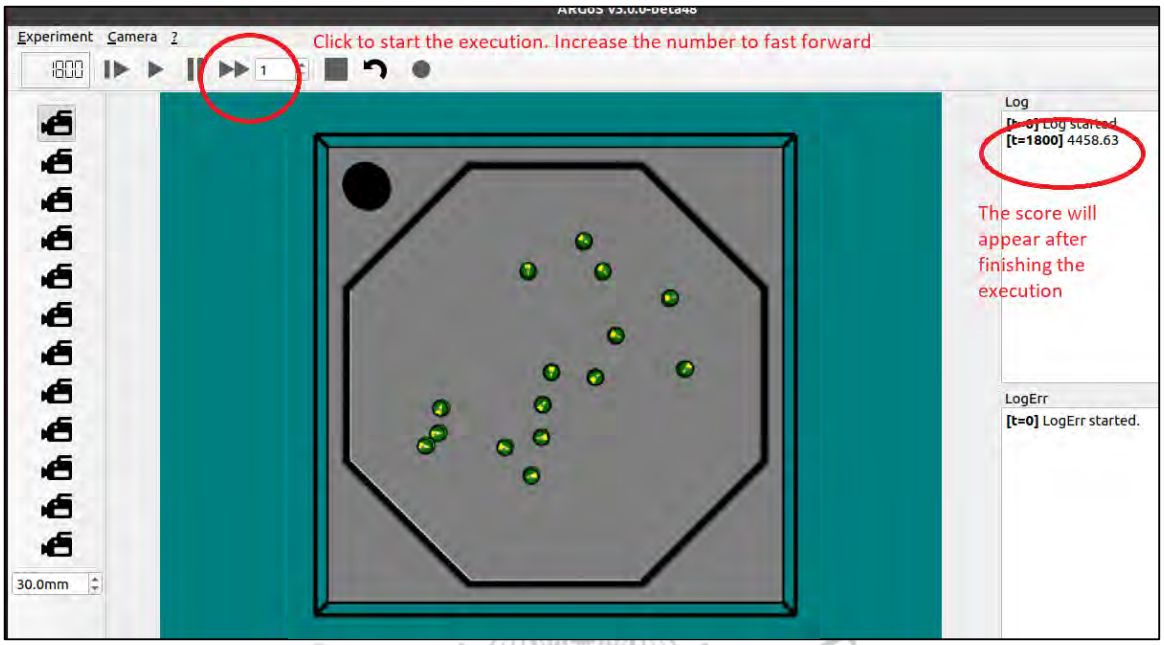
You can connect two nodes with an edge by clicking first on the node from where the edge is starting and clicking after on the node where the edge is terminating.

For example, to place a BlackFloor transition (Edge) between Exploration (Exp) and Stop, you must first click on the Exp and then on the Stop.

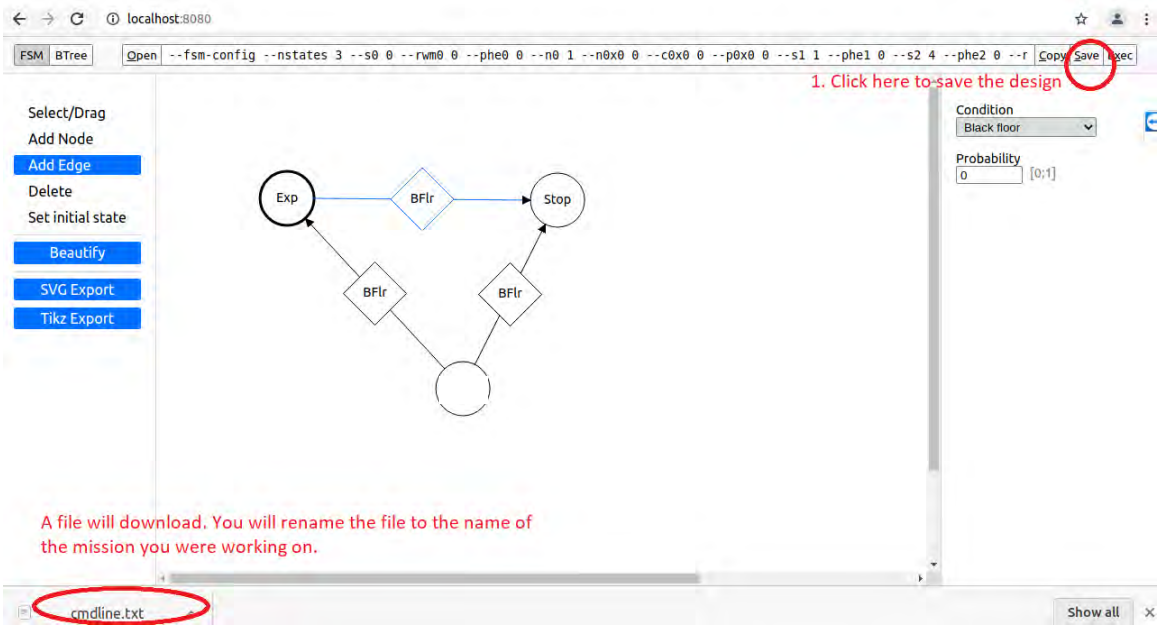
Once you are ready to test your finite-sate machine, click on “Exec” to execute the simulation.



A simulation window will appear. In this window, you can observe the behavior of the robots and the score of the simulation of the control software you have designed.



After executing the simulation, you must close the simulation window and return to the visualization tool. You can modify your design as many times as you want. When you are satisfied with the result, save the design by clicking the button “Save”. This command will download a file with your control software.



Rename the file according to the naming scheme indicated below. Only remember to chhange the numbers according to your identifier and the current scenario.

```
designer_1_scenario_1.txt
```

After collecting the four files you will generate, please send them by email to David Garzón Ramos (dgarzonr@ulb.ac.be).

Once you finish a mission and have saved your finite-state machine, close first Firefox, then return to the terminal where you executed `npm start` and press `Ctrl + c`.

You can start a new mission by following the same steps described for the first mission. For example, for Designer 1 and scenario 2, you can enter the following commands.

```
cd ~/hdesigner/designer_1/scenario_2

npm start
```

The backend will start running again, and now you can again launch the graphical user interface by opening Firefox and entering the URL.

```
localhost:8080
```

These steps must be repeated until you complete all four missions.

If you have any issues during the experiments, feel free to contact me at any time.

David Garzón Ramos

dagarzor@unal.edu.co
dgarzonr@ulb.ac.be

Bibliografía

[1] Gerardo Beni. From swarm intelligence to swarm robotics. In Erol Şahin y William M. Spears, editors, *Swarm Robotics: SAB 2004 International Workshop*, volume 3342 of *Lecture Notes in Computer Science*, pages 1–9, Berlin, Germany, 2005. Springer. doi: 10.1007/978-3-540-30552-1_1.

[2] Erol Şahin. Swarm robotics: from sources of inspiration to domains of application. In Erol Şahin y William M. Spears, editors, *Swarm Robotics: SAB 2004 International Workshop*, volume 3342 of *Lecture Notes in Computer Science*, pages 10–20, Berlin, Germany, 2005. Springer. doi: 10.1007/978-3-540-30552-1_2.

[3] Marco Dorigo, Mauro Birattari, y Manuele Brambilla. Swarm robotics. *Scholarpedia*, 9 (1):1463, 2014. doi: 10.4249/scholarpedia.1463.

[4] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, y Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013. doi: 10.1007/s11721-012-0075-2.

[5] Marco Dorigo, Guy Theraulaz, y Vito Trianni. Reflections on the future of swarm robotics. *Science Robotics*, 5:eabe4385, 2020. doi: 10.1126/scirobotics.abe4385.

[6] Marco Dorigo, Guy Theraulaz, y Vito Trianni. Swarm robotics: past, present, and future. *Proceedings of the IEEE*, 109(7):1152–1165, 2021. doi: 10.1109/JPROC.2021.3072740.

[7] Marco Dorigo, Guy Theraulaz, y Vito Trianni. Swarm robotics: past, present, and future [point of view]. *Proceedings of the IEEE*, 109(7):1152–1165, 2021. doi: 10.1109/JPROC.2021.3072740.

[8] Gianpiero Francesca y Mauro Birattari. Automatic design of robot swarms: achievements and challenges. *Frontiers in Robotics and AI*, 3(29):1–9, 2016. doi: 10.3389/frobt.2016.00029.

[9] Mauro Birattari, Antoine Ligot, Darko Bozhinoski, Manuele Brambilla, Gianpiero Francesca, Lorenzo Garattoni, David Garzón Ramos, Ken Hasselmann, Miquel Kegeleirs,

Jonas Kuckling, Federico Pagnozzi, Andrea Roli, Muhammad Salman, y Thomas Stützle. Automatic off-line design of robot swarms: a manifesto. *Frontiers in Robotics and AI*, 6:59, 2019. doi: 10.3389/frobt.2019.00059.

[10] Mauro Birattari, Antoine Ligot, y Ken Hasselmann. Disentangling automatic and semi-automatic approaches to the optimization-based design of control software for robot swarms. *Nature Machine Intelligence*, 2(9):494–499, 2020. doi: 10.1038/s42256-020-0215-0.

[11] Mauro Birattari, Antoine Ligot, y Gianpiero Francesca. AutoMoDe: a modular approach to the automatic off-line design and fine-tuning of control software for robot swarms. In Nelishia Pillay y Rong Qu, editors, *Automated Design of Machine Learning and Search Algorithms*, Natural Computing Series, pages 73–90. Springer, Cham, Switzerland, 2021. doi: 10.1007/978-3-030-72069-8_5.

[12] Ken Hasselmann, Antoine Ligot, Julian Ruddick, y Mauro Birattari. Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms. *Nature Communications*, 12:4345, 2021. doi: 10.1038/s41467-021-24642-3.

[13] David Garzón Ramos y Mauro Birattari. Automatic design of collective behaviors for robots that can display and perceive colors. *Applied Sciences*, 10(13):4654, 2020. doi: 10.3390/app10134654.

[14] Ken Hasselmann y Mauro Birattari. Modular automatic design of collective behaviors for robots endowed with local communication capabilities. *PeerJ Computer Science*, 6: e291, 2020. doi: 10.7717/peerj-cs.291.

[15] Muhammad Salman, Antoine Ligot, y Mauro Birattari. Concurrent design of control software and configuration of hardware for robot swarms under economic constraints. *PeerJ Computer Science*, 5:e221, 2019. doi: 10.7717/peerj-cs.221.

[16] Antoine Ligot, Jonas Kuckling, Darko Bozhinoski, y Mauro Birattari. Automatic modular design of robot swarms using behavior trees as a control architecture. *PeerJ Computer Science*, 6:e314, 2020. doi: 10.7717/peerj-cs.314.

[17] Jonas Kuckling, Thomas Stützle, y Mauro Birattari. Iterative improvement in the automatic modular design of robot swarms. *PeerJ Computer Science*, 6:e322, 2020. doi: 10.7717/peerj-cs.322.

[18] Jonas Kuckling, Keneth Ubeda Arriaza, y Mauro Birattari. AutoMoDe-IcePop: automatic modular design of control software for robot swarms using simulated annealing. In Bart Bogaerts, Gianluca Bontempi, Pierre Geurts, Nick Harley, Bertrand Lebichot,

Tom Lenaerts, y Gilles Louppe, editors, *Artificial Intelligence and Machine Learning: BNAIC 2019, BENELEARN 2019*, volume 1196 of *Communications in Computer and Information Science*, pages 3–17. Springer, Cham, Switzerland, 2020.

[19] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Lorenzo Garattoni, Roman Miletitch, Gaëtan Podevijn, Andreagiovanni Reina, Touraj Soleymani, Mattia Salvaro, Carlo Pinciroli, Vito Trianni, y Mauro Birattari. An experiment in automatic design of robot swarms: AutoMoDe-Vanilla, EvoStick, and human experts. In Marco Dorigo, Mauro Birattari, Simon Garnier, Heiko Hamann, Marco Montes de Oca, Christine Solnon, y Thomas Stützle, editors, *Swarm Intelligence: 9th International Conference, ANTS 2014*, volume 8667 of *Lecture Notes in Computer Science*, pages 25–37, Cham, Switzerland, 2014. Springer International Publishing. doi: 10.1007/978-3-319-09952-1_3.

[20] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Lorenzo Garattoni, Roman Miletitch, Gaëtan Podevijn, Andreagiovanni Reina, Touraj Soleymani, Mattia Salvaro, Carlo Pinciroli, Franco Mascia, Vito Trianni, y Mauro Birattari. AutoMoDe-Chocolate: automatic design of control software for robot swarms. *Swarm Intelligence*, 9(2–3): 125–152, 2015. doi: 10.1007/s11721-015-0107-9.

[21] Jonas Kuckling, Vincent Van Pelt, y Mauro Birattari. Automatic modular design of behavior trees for robot swarms with communication capabilities. In Pedro A. Castillo y Juan Luis Jiménez Laredo, editors, *Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021*, volume 12694 of *Lecture Notes in Computer Science*, pages 130–145, Cham, Switzerland, 2021. Springer. doi: 10.1007/978-3-030-72699-7_9.

[22] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, y Mauro Birattari. AutoMoDe: a novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112, 2014. doi: 10.1007/s11721-014-0092-4.

[23] Vito Trianni. *Evolutionary Swarm Robotics*. Springer, Berlin, Germany, 2008. doi: 10.1007/978-3-540-77612-3.

[24] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni A. Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, y Marco Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012. doi: 10.1007/s11721-012-0072-5.

[25] Francesco Mondada, Michael Bonani, Xavier Raemy, Jim Pugh, Christopher Cianci, Adam Klaptocz, Stéphane Magnenat, Jean-Christophe Zufferey, Dario Floreano, y Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In Paulo

Gonçalves, Paulo Torres, y Carlos Alves, editors, *ROBOTICA 2009: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, Castelo Branco, Portugal, 2009. Instituto Politécnico de Castelo Branco.

[26] Guang-Zhong Yang, Jim Bellingham, Pierre E. Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, Bradley J. Nelson, Brian Scassellati, Mariarosaria Taddeo, Russell Taylor, Manuela Veloso, Zhong Lin Wang, y Robert Wood. The grand challenges of Science Robotics. *Science Robotics*, 3(14):eaar7650, 2018. doi: 10.1126/scirobotics.aar7650.

[27] Mauro Birattari. *Tuning Metaheuristics: A Machine Learning Perspective*. Springer, Berlin, Germany, 2009. doi: 10.1007/978-3-642-00483-4.

[28] David Garzón Ramos, Darko Bozhinoski, Gianpiero Francesca, Lorenzo Garattoni, Ken Hasselmann, Miquel Kegeleirs, Jonas Kuckling, Antoine Ligot, Fernando J. Mendiburu, Federico Pagnozzi, Muhammad Salman, Thomas Stützle, y Mauro Birattari. The automatic off-line design of robot swarms: recent advances and perspectives. In Giulia De Masi, Eliseo Ferrante, y Paolo Dario, editors, *R2T2: Robotics Research for Tomorrow's Technology*, 2021.

[29] Nadia Nedjah y Luneque Silva Junior. Review of methodologies and tasks in swarm robotics towards standardization. *Swarm and Evolutionary Computation*, 50:100565, 2019. doi: 10.1016/j.swevo.2019.100565.

[30] Nithin Mathews, Anders Lyhne Christensen, Rehan O'Grady, Francesco Mondada, y Marco Dorigo. Mergeable nervous systems for robots. *Nature Communications*, 8(1): 439, 2017. doi: 10.1038/s41467-017-00109-2.

[31] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, Daniel Burnier, Alexandre Campo, Anders Lyhne Christensen, Antal Decugnière, Gianni A. Di Caro, Frederick Ducatelle, Eliseo Ferrante, Alexander Förster, Javier Martinez Gonzales, Jérôme Guzzi, Valentin Longchamp, Stéphane Magnenat, Nithin Mathews, Marco Montes de Oca, Rehan O'Grady, Carlo Pinciroli, Giovanni Pini, Philippe Retornaz, James Roberts, Valerio Sperati, Timothy Stirling, Alessandro Stranieri, Thomas Stützle, Vito Trianni, Elio Tuci, Ali Emre Turgut, y Florian Vaussard. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71, 2013. doi: 10.1109/MRA.2013.2252996.

[32] Lorenzo Garattoni y Mauro Birattari. Autonomous task sequencing in a robot swarm. *Science Robotics*, 3(20):eaat0430, 2018. doi: 10.1126/scirobotics.aat0430.

[33] Heiko Hamann. *Swarm robotics: a formal approach*. Springer, Cham, Switzerland, 2018. ISBN 978-3-319-74526-8. doi: 10.1007/978-3-319-74528-2.

[34] Melanie Schranz, Martina Umlauft, Micha Sende, y Wilfried Elmenreich. Swarm robotic behaviors and current applications. *Frontiers in Robotics and AI*, 7:36, 2020. doi: 10.3389/frobt.2020.00036.

[35] Lorenzo Garattoni y Mauro Birattari. Swarm robotics. In John G. Webster, editor, *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 1–19. John Wiley & Sons, Hoboken, NJ, USA, 2016. doi: 10.1002/047134608X.W8312.

[36] Marco Dorigo y Mauro Birattari. Swarm intelligence. *Scholarpedia*, 2(9):1462, 2007. doi: 10.4249/scholarpedia.1462.

[37] Edmund R. Hunt y Sabine Hauert. A checklist for safe robot swarms. *Nature Machine Intelligence*, 2:420–422, 2020. doi: 10.1038/s42256-020-0213-2.

[38] Heiko Hamann. Towards swarm calculus: universal properties of swarm performance and collective decisions. In Mauro Birattari, Christian Blum, Anders Lyhne Christensen, Andries P. Engelbrecht, Roderich Groß, Marco Dorigo, y Thomas Stützle, editors, *Swarm Intelligence: 8th International Conference, ANTS 2012*, volume 7461 of *Lecture Notes in Computer Science*, pages 168–179, Berlin, Germany, 2012. Springer. doi: 10.1007/978-3-642-32650-9_15.

[39] Eliseo Ferrante, Ali Emre Turgut, Edgar A. Duéñez-Guzmán, Marco Dorigo, y Tom Wenseleers. Evolution of self-organized task specialization in robot swarms. *PLOS Computational Biology*, 11(8):e1004273, 2015. doi: 10.1371/journal.pcbi.1004273.

[40] Simon Jones, Alan Winfield, Sabine Hauert, y Matthew Studley. Onboard evolution of understandable swarm behaviors. *Advanced Intelligent Systems*, 1(6):1900031, 2019. doi: 10.1002/aisy.201900031.

[41] Juan Jesús Roldán, Elena Peña-Tapia, David Garzón Ramos, Jorge de León, Mario Garzón, Jaime del Cerro, y Antonio Barrientos. Multi-robot systems, virtual reality and ros: developing a new generation of operator interfaces. In Anis Koubaa, editor, *Robot Operating System (ROS): The Complete Reference*, volume 778 of *SCI*, pages 29–64. Springer, Cham, Switzerland, 2018. doi: 10.1007/978-3-319-91590-6_2.

[42] Mario Garzón, João Valente, Juan Jesús Roldán, David Garzón Ramos, Jorge de León, Antonio Barrientos, y Jaime del Cerro. Using ros in multi-robot systems: experiences and lessons learned from real-world field tests. In Anis Koubaa, editor, *Robot Operating System (ROS): The Complete Reference*, volume 707 of *SCI*, pages 449–483. Springer, Cham, Switzerland, 2017. doi: 10.1007/978-3-319-54927-9_14.

[43] Anders Lyhne Christensen, Rehan O'Grady, y Marco Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation*, 13(4): 754–766, 2009. doi: 10.1109/TEVC.2009.2017516.

[44] Florian Berlinger, Melvin Gauci, y Radhika Nagpal. Implicit coordination for 3D underwater collective behaviors in a fish-inspired robot swarm. *Science Robotics*, 6(50): eabd8668, 2021. doi: 10.1126/scirobotics.abd8668.

[45] Juan Jesús Roldán, Jaime del Cerro, David Garzón Ramos, Pablo Garcia-Aunon, Mario Garzón, Jorge de León, y Antonio Barrientos. Robots in agriculture: state of art and practical experiences. In Antonio J. R. Neves, editor, *Service Robots*. IntechOpen, London, United Kingdom, 2017. doi: 10.5772/intechopen.69874.

[46] Simon Jones, Emma Milner, Mahesh Sooriyabandara, y Sabine Hauert. Distributed situational awareness in robot swarms. *Advanced Intelligent Systems*, 2(11):2000110, 2020. doi: 10.1002/aisy.202000110.

[47] Suet Lee y Sabine Hauert. Simulations, real robots, and bloopers from the DOTS competition: powering emergency food distribution using swarms. ROBOTS Association, Préverenges, Switzerland, 2021. Robohub.

[48] Heiko Hamann, Melanie Schranz, Wilfried Elmenreich, Vito Trianni, Carlo Pinciroli, Nicolas Bredeche, y Eliseo Ferrante. Editorial: designing self-organization in the physical realm. *Frontiers in Robotics and AI*, 7:164, 2020. doi: 10.3389/frobt.2020.597859.

[49] David St-Onge, Vivek Shankar Varadharajan, Švogor Ivan, y Giovanni Beltrame. From design to deployment: decentralized coordination of heterogeneous robotic teams. *Frontiers in Robotics and AI*, 7:51, 2020. doi: 10.3389/frobt.2020.00051.

[50] Gabriele Valentini, Eliseo Ferrante, y Marco Dorigo. The best-of-n problem in robot swarms: formalization, state of the art, and novel perspectives. *Frontiers in Robotics and AI*, 4:9, 2017. doi: 10.3389/frobt.2017.00009.

[51] Carlo Pinciroli y Giovanni Beltrame. Buzz: a programming language for robot swarms. *IEEE Software*, 33(4):97–100, 2016. doi: 10.1109/MS.2016.95.

[52] Rehan O'Grady, Anders Lyhne Christensen, y Marco Dorigo. SWARMORPH: multirobot morphogenesis using directional self-assembly. *IEEE Transactions on Robotics*, 25 (3):738–743, 2009. doi: 10.1109/TRO.2008.2012341.

[53] Rehan O'Grady, Roderich Groß, Anders Lyhne Christensen, y Marco Dorigo. Self-assembly strategies in a group of autonomous mobile robots. *Autonomous Robots*, 28 (4):439–455, 2010. doi: 10.1007/s10514-010-9177-0.

[54] Nithin Mathews, Anders Lyhne Christensen, Alessandro Stranieri, Alexander Scheidler, y Marco Dorigo. Supervised morphogenesis: exploiting morphological flexibility of self-assembling multirobot systems through cooperation with aerial robots. *Robotics and Autonomous Systems*, 112:154–167, 2019. doi: 10.1016/j.robot.2018.11.007.

[55] Eliseo Ferrante, Ali Emre Turgut, Nithin Mathews, Mauro Birattari, y Marco Dorigo. Flocking in stationary and non-stationary environments: a novel communication strategy for heading alignment. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, y Günter Rudolph, editors, *Parallel Problem Solving from Nature – PPSN XI: 11th International Conference*, volume 6239 of *Lecture Notes in Computer Science*, pages 331–340, Berlin, Germany, 2010. Springer. doi: 10.1007/978-3-642-15871-1_34.

[56] Giovanni Pini, Arne Brutschy, Marco Frison, Andrea Roli, Marco Dorigo, y Mauro Birattari. Task partitioning in swarms of robots: an adaptive method for strategy selection. *Swarm Intelligence*, 5(3–4):283–304, 2011. doi: 10.1007/s11721-011-0060-1.

[57] Arne Brutschy, Lorenzo Garattoni, Manuele Brambilla, Gianpiero Francesca, Giovanni Pini, Marco Dorigo, y Mauro Birattari. The TAM: abstracting complex tasks in swarm robotics research. *Swarm Intelligence*, 9(1):1–22, 2015. doi: 10.1007/s11721-014-0102-6.

[58] Giovanni Pini, Arne Brutschy, Alexander Scheidler, Marco Dorigo, y Mauro Birattari. Task partitioning in a robot swarm: object retrieval as a sequence of subtasks with direct object transfer. *Artificial Life*, 20(3):291–317, 2014. doi: 10.1162/ARTL_a_00132.

[59] Anders Lyhne Christensen y Marco Dorigo. Evolving an integrated phototaxis and hole-avoidance behavior for a swarm-bot. In Luis M. Rocha, Larry S. Yaeger, Mark A. Bedau, Dario Floreano, Robert L. Goldstone, y Alessandro Vespignani, editors, *Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, pages 248–254, Cambridge, MA, USA, 2006. MIT Press. A Bradford Book.

[60] Nick Jakobi, Phil Husbands, y Inman Harvey. Noise and the reality gap: the use of simulation in evolutionary robotics. In F. Morán, A. Moreno, Juan J. Merelo, y P. Chacón, editors, *Advances in Artificial Life: Third European Conference on Artificial Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pages 704–720, Berlin, Germany, 1995. Springer. doi: 10.1007/3-540-59496-5_337.

[61] Antoine Ligot y Mauro Birattari. Simulation-only experiments to mimic the effects of the reality gap in the automatic design of robot swarms. *Swarm Intelligence*, 14:1–24, 2020. doi: 10.1007/s11721-019-00175-w.

[62] Stefano Nolfi y Dario Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA, USA, first edition, 2000. A Bradford Book.

[63] Jonas Kuckling, Vincent Van Pelt, y Mauro Birattari. AutoMoDe-Cedrata: automatic design of behavior trees for controlling a swarm of robots with communication capabilities. *SN Computer Science*, 2021. Submitted.

[64] Miquel Kegeleirs, David Garzón Ramos, y Mauro Birattari. Random walk exploration for swarm mapping. In Kaspar Althoefer, Jelizaveta Konstantinova, y Ketao Zhang, editors, *Towards Autonomous Robotic Systems: 20th Annual Conference, TAROS 2019*, volume 11650 of *Lecture Notes in Computer Science*, pages 211–222, Cham, Switzerland, 2019. Springer. doi: 10.1007/978-3-030-25332-5_19.

[65] Ryan A. Licitra, Zachary I Bell, y Warren E. Dixon. Single-agent indirect herding of multiple targets with uncertain dynamics. *IEEE Transactions on Robotics*, 35(4): 847–860, 2019. doi: 10.1109/TRO.2019.2911799.

[66] Katie Genter y Peter Stone. Influencing a flock via ad hoc teamwork. In Marco Dorigo, Mauro Birattari, Simon Garnier, Heiko Hamann, Marco Montes de Oca, Christine Solnon, y Thomas Stützle, editors, *Swarm Intelligence: 9th International Conference, ANTS 2014*, volume 8667 of *Lecture Notes in Computer Science*, pages 110–121, Cham, Switzerland, 2014. Springer International Publishing. doi: 10.1007/978-3-319-09952-1_10.

[67] Katie Genter y Peter Stone. Adding influencing agents to a flock. In *AAMAS '16: Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*, pages 615–623, Richland, SC, USA, 2016. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

[68] Anil Özdemir, Melvin Gauci, y Roderich Groß. Shepherding with robots that do not compute. In *ECAL 2017, the Fourteenth European Conference on Artificial Life*, Cambridge, MA, USA, 2017. MIT Press. doi: 10.7551/ecal_a_056.

[69] Alyssa Pierson y Mac Schwager. Controlling noncooperative herds with robotic herders. *IEEE Transactions on Robotics*, 34(2):517–525, 2018. doi: 10.1109/TRO.2017.2776308.

[70] Junyan Hu, Ali Emre Turgut, Tomáš Krajník, Barry Lennox, y Farshad Arvin. Occlusion-based coordination protocol design for autonomous robotic shepherding tasks. *IEEE Transactions on Cognitive and Developmental Systems*, page 1, 2020. doi: 10.1109/TCDS.2020.3018549.

[71] Lorenzo Garattoni, Gianpiero Francesca, Arne Brutschy, Carlo Pinciroli, y Mauro Birattari. Software infrastructure for e-puck (and TAM). Technical Report TR/IRIDIA/2015-004, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2015.

[72] Muhammad Salman, David Garzón Ramos, Ken Hasselmann, y Mauro Birattari. Phormica: photochromic pheromone release and detection system for stigmergic coordination in robot swarms. *Frontiers in Robotics and AI*, 7:195, 2020. doi: 10.3389/frobt.2020.591402.

[73] Álvaro Gutiérrez, Alexandre Campo, Marco Dorigo, Jesus Donate, Félix Monasterio-Huelin, y Luis Magdalena. Open e-puck range & bearing miniaturized board for local communication in swarm robotics. In Kinugawa Kosuge, editor, *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3111–3116, Piscataway, NJ, USA, 2009. IEEE. doi: 10.1109/ROBOT.2009.5152456.

[74] École polytechnique fédérale de Lausanne. Omnidirectional vision turret for the e-puck. `http://www.e-puck.org/index.php?option=com_content&view=article&id=26&Itemid=21`, 2010.

[75] Ken Hasselmann, Antoine Ligot, Gianpiero Francesca, David Garzón Ramos, Muhammad Salman, Jonas Kuckling, Fernando J. Mendiburu, y Mauro Birattari. Reference models for AutoMoDe. Technical Report TR/IRIDIA/2018-002, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2018.

[76] Andreas Kolling, Phillip Walker, Nilanjan Chakraborty, Katia Sycara, y Michael Lewis. Human interaction with robot swarms: a survey. *IEEE Transactions on Human-Machine Systems*, 46(1):9–26, 2016. doi: 10.1109/THMS.2015.2480801.

[77] Nathan P. Koenig y Andrew Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS)*, volume 3, pages 2149–2154, Piscataway, NJ, USA, 2004. IEEE. doi: 10.1109/IROS.2004.1389727.

[78] Eric Rohmer, Surya P. N. Singh, y Marc Freese. V-REP: a versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS)*, volume 3, pages 1321–1326, Piscataway, NJ, USA, 2013. IEEE. doi: 10.1109/IROS.2013.6696520.

[79] Lenka Pitonakova, Manuel Giuliani, Anthony Pipe, y Alan Winfield. Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators. In Manuel Giuliani, Tareq Assaf, y Maria Elena Giannaccini, editors, *Towards Autonomous Robotic Systems: 19th Annual Conference, TAROS 2018*, Lecture Notes in Computer Science, pages 357–368, Cham, Switzerland, 2018. Springer. doi: 10.1007/978-3-319-96728-8_30.

[80] Vito Trianni y Manuel López-Ibáñez. Advantages of task-specific multi-objective optimisation in evolutionary robotics. *PLOS ONE*, 10(8):e0136406, 2015. doi: 10.1371/journal.pone.0136406.

[81] Antoine Ligot, Ken Hasselmann, y Mauro Birattari. AutoMoDe-Arlequin: neural networks as behavioral modules for the automatic design of probabilistic finite state machines. In Marco Dorigo, Thomas Stützle, María J. Blesa, Christian Blum, Heiko Hamann, Mary Katherine Heinrich, y Volker Strobel, editors, *Swarm Intelligence: 12th International Conference, ANTS 2020*, volume 12421 of *Lecture Notes in Computer Science*, pages 109–122, Cham, Switzerland, 2020. Springer. doi: 10.1007/978-3-030-60376-2_21.

[82] Gaëtan Spaey, Miquel Kegeleirs, David Garzón Ramos, y Mauro Birattari. Evaluation of alternative exploration schemes in the automatic modular design of robot swarms. In Bart Bogaerts, Gianluca Bontempi, Pierre Geurts, Nick Harley, Bertrand Lebichot, Tom Lenaerts, y Gilles Louppe, editors, *Artificial Intelligence and Machine Learning: BNAIC 2019, BENELEARN 2019*, volume 1196 of *Communications in Computer and Information Science*, pages 18–33. Springer, Cham, Switzerland, 2020. doi: 10.1007/978-3-030-65154-1_2.

[83] Johann Borenstein y Yorem Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989. doi: 10.1109/21.44033.

[84] Ken Hasselmann, Frédéric Robert, y Mauro Birattari. Automatic design of communication-based behaviors for robot swarms. In Marco Dorigo, Mauro Birattari, Simon Garnier, Heiko Hamann, Marco Montes de Oca, Christine Solnon, y Thomas Stützle, editors, *Swarm Intelligence: 11th International Conference, ANTS 2018*, volume 11172 of *Lecture Notes in Computer Science*, pages 16–29, Cham, Switzerland, 2018. Springer. doi: 10.1007/978-3-030-00533-7_2.

[85] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, y Thomas Stützle. The irace package: iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. doi: 10.1016/j.orp.2016.09.002.

[86] Mauro Birattari, Thomas Stützle, Luis Paquete, y Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In William B. Langdon, Erick Cantú-Paz, Keith Mathias, Rajkumar Roy, David Davis, Riccardo Poli, Karthik Balakrishnan, Vasant Honavar, Günter Rudolph, Joachim Wegener, Lary Bull, Mitchell A. Potter, Alan C. Schultz, Julian F. Miller, Edmund K. Burke, y Natasha Jonoska, editors, *GECCO'02: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 11–18, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers.

[87] Nikolaus Hansen y Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

[88] Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, y Jürgen Schmidhuber. Exponential natural evolution strategies. In *GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 393–400, New York, NY, USA, 2010. ACM. doi: 10.1145/1830483.1830557.

[89] Kenneth O. Stanley y Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002. doi: 10.1162/106365602320169811.

[90] Jonas Kuckling, Ken Hasselmann, Vincent Van Pelt, Cédric Kiere, y Mauro Birattari. AutoMoDe Editor: a visualization tool for AutoMoDe. Technical Report TR/IRIDIA/2021-009, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2021.

[91] W. J. Conover. *Practical Nonparametric Statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons, New York, NY, USA, third edition, 1999.

[92] Mauro Birattari. Notes on the estimation of the expected performance of automatic methods for the design of control software for robot swarms. Technical Report TR/IRIDIA/2020-10, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2020.

[93] Haoxiang Zhang y Lei Liu. Intelligent control of swarm robotics employing biomimetic deep learning. 9(10):236, 2021. doi: 10.3390/machines9100236.