UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

ESTUDIO DE LA CONVERGENCIA PREMATURA EN LA CO-EVOLUCIÓN DE MORFOLOGÍA Y CONTROL

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:

LUIS ANDRÉS EGUIARTE MORETT

TUTORES:
DRA. WENDY ELIZABETH AGUILAR MARTÍNEZ
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y SISTEMAS

CIUDAD UNIVERSITARIA, CD. MX., JUNIO 2023

DEDICATORIA

A mi familia, en especial a mi madre, cuyo amor, apoyo incondicional y sabiduría, me han guiado a lo largo de mi vida.

Agradecimientos

Agradezco al CONACYT por la beca brindada durante la realización de mi maestría. A la UNAM por la excelente oferta académica brindada. A mi tutora, por su paciencia e indispensable guía. A la "International Summer School of AI and Games" en sus ediciones 2021 y 2022 por ayudarme con créditos de AWS (Amazon Web Services), los cuales fueron fundamentales para obtener los recursos de cómputo que me permitieron escalar mis experimentos.

Declaración de autenticidad

Por la presente declaro que, salvo cuando se haga referencia específica al trabajo de otras personas, el contenido de esta tesis es original y no se ha presentado total o parcialmente para su consideración para cualquier otro título o grado en esta o cualquier otra Universidad. Esta tesis es resultado de mi propio trabajo y no incluye nada que sea el resultado de algún trabajo realizado en colaboración, salvo que se indique específicamente en el texto.

LUIS ANDRÉS EGUIARTE MORETT. CIUDAD UNIVERSITARIA, CD. MX.,
JUNIO 2023

Índice general

Capítulo 1

Introducción

El diseño y construcción automatizado de máquinas adaptativas y autónomas, como lo son los robots autónomos, es un problema abierto. El único proceso conocido capaz de producir máquinas completamente adaptativas y autónomas es la evolución biológica [1].

En la robótica evolutiva se utilizan los algoritmos evolutivos (parte de una clase más grande de algoritmos llamados algoritmos metaheurísticos basados en poblaciones), para optimizar algunos aspectos de un robot autónomo [1]. Dichos algoritmos aplican los principios de selección, variación y herencia de la evolución natural al problema de diseño de robots [2], viéndolos como organismos autónomos artificiales, capaces de desarrollar sus propias habilidades al interactuar con el ambiente, sin ningún tipo de intervención humana [3].

Al diseñar un robot se deben considerar varios aspectos de manera simultánea, entre ellos su morfología, sensores, motores, arquitectura de control, políticas de control, etc., los cuales deben interactuar de manera conjunta para determinar su comportamiento. En este contexto, dos de los aspectos más importantes que pueden ser optimizados son la morfología (cuerpo) y el control (cerebro), la morfología se refiere a la configuración espacial del robot, mientras que el control o política de control se refiere a la estrategia para controlar el comportamiento que permita al robot realizar una cierta tarea. A la evolución de estos dos aspectos del robot que se influencian mutuamente, se le conoce como coevolución de morfología y control. Cabe mencionar que dado que modificar los controladores y morfología de robots físicos en este proceso de optimización sería no sólo impráctico sino costoso, la robótica evolutiva suele recurrir a simulaciones basadas en creaturas virtuales para realizar dicho proceso y posteriormente manufacturar los robots físicos con base en los diseños optimizados [4].

La morfología y control de un robot son profundamente interdependientes y cualquier cambio en uno puede influenciar al otro significativamente. Es por ello que se deben considerar ambos aspectos como parte de un todo y tratar directamente con ese todo,

para poder explotar las interdependencias existentes, en contraste con la aproximación reduccionista tradicional a la robótica, cuyo objetivo es mantener el proceso de diseño tan modular como sea posible, enfocándose en un aspecto del robot a la vez mientras se ignora el resto [2]. La idea aquí es que la dinámica acoplada entre el cuerpo, el cerebro y el entorno de un agente provocan que emerja comportamiento inteligente en éste, algo que se conoce como inteligencia (artificial) corporizada [5]. La distinción entre la robótica evolutiva y la robótica tradicional se da porque en esta última se suelen utilizar métodos de Machine Learning para optimizar las políticas de control del robot [1]. Una ventaja asociada al uso de algoritmos evolutivos es su capacidad de ser utilizados como optimizadores para cualquier aspecto del robot, incluidas las políticas de control e inclusive la morfología, mientras que una desventaja es la posibilidad de que el algoritmo converja a un óptimo local (convergencia prematura) o que no converja [4].

El bien conocido fenómeno de la convergencia prematura, prevalente en el cómputo evolutivo, se hereda a este dominio debido a que es un problema sumamente complejo que deriva en un paisaje accidentado de aptitud [4].

En particular cuando se intenta coevolucionar la morfología y control en creaturas o robots virtuales el problema prevalente que se ha observado [4] [6] [7] es que la evolución de la morfología se estanca muy pronto en un subconjunto muy similar. Lo anterior quiere decir que se pierde el potencial beneficio de coevolucionar ambos aspectos y la optimización únicamente sucede en los controladores. Se ha encontrado que esta pérdida de diversidad en la morfología es causada por el hecho de que los cambios morfológicos de una generación a otra causan grandes disrupciones en los controladores optimizados y esto a su vez causa que los individuos con cambios morfológicos recientes sean descartados por la presión de selección orientada a maximizar la aptitud en el desempeño de la tarea elegida (caminar, nadar, seguir un objetivo, evadir obstáculos, etc.). La teoría de la cognición corporizada que viene de la psicología provee una posible explicación para este fenómeno: la inteligencia no puede puede ser separada del cuerpo, sino que resulta de las interacciones complejas entre el cuerpo y la mente [4].

Algunas direcciones promisorias perseguidas previamente son la imposición de mecanismos de protección de la diversidad morfológica, por ejemplo: reducir temporalmente la presión de selección con recientes cambios morfológicos [8], utilizar algoritmos de QD (Quality-Diversity) tales como MAP-Elites, definiendo un conjunto de características morfológicas deseadas en las cuales se desee descubrir calidad dentro de la diversidad misma [9], y evolucionar control y morfología en conjunto en una primera fase y en una segunda únicamente el controlador para dejar que se adapte a los cambios recientes en la morfología [10].

Todas estas propuestas proveen contribuciones y resultados promisorios, sin embargo, muchas suelen medir el beneficio únicamente contra utilizar una función de evaluación de aptitud en un algoritmo evolutivo tradicional como referencia, y poco se ha hecho, para comparar varias aproximaciones que explícitamente busquen mitigar el problema de la convergencia prematura de morfología respecto a control, bajo un mismo marco de

trabajo, con distintos indicadores, y una metodología bien definida de benchmarking de algoritmos de optimización. Es por ello, que en este trabajo, se propone la comparación de varias aproximaciones propuestas en el estado del arte para mitigar el problema de la convergencia prematura en coevolución de morfología y control, y no sólo respecto a el algoritmo evolutivo base, bajo un mismo marco de trabajo para la realización y comparación de experimentos de este tipo.

1.1. Objetivos

En esta sección se presentan los objetivos del trabajo, comenzando por un objetivo general, que a su vez da lugar para el esbozo de los objetivos específicos.

1.1.1. Objetivo General

Comparar algunos de los algoritmos más representativos que se han utilizado en el estado del arte para mitigar el problema de la convergencia prematura en robótica evolutiva, con particular interés en el fenómeno de convergencia prematura de morfología respecto al control que se da en el ámbito de coevolución de morfología y control, bajo un mismo marco de trabajo estadístico y respecto a una serie de indicadores de interés.

1.1.2. Objetivos específicos

- Proponer un marco de trabajo general para realizar experimentos en creaturas virtuales y robótica evolutiva.
- Utilizar dicho marco de trabajo para proponer experimentos con algunos de los algoritmos más representativos de la literatura, buscando responder a las preguntas de investigación presentadas en la Sección 1.2.
- Identificar si el problema de la convergencia prematura afecta de la misma manera a los algoritmos utilizados, mediante el uso y proposición de distintos indicadores.
- Comparar el rendimiento de los distintos algoritmos, respecto a los distintos indicadores y declarar un ordenamiento de dichos algoritmos.

1.2. Preguntas de investigación

- ¿Cómo se compara un algoritmo evolutivo que no presenta ningún tipo de protección a la diversidad morfológica con otros algoritmos que si la presentan, en cuanto al fenómeno de la convergencia prematura de morfología respecto a control?

- ¿Cómo afecta la adición de la novedad morfológica como segundo objetivo a la aptitud de las soluciones y al fenómeno de la convergencia prematura de morfología respecto a control?
- ¿Cómo se compara una aproximación multiobjetivo que utiliza aptitud global, contra sus contrapartes QD que consideran aptitud local, en cuanto al fenómeno de convergencia prematura y a la aptitud de las soluciones?
- ¿Cómo afecta el uso de la competencia local en lugar de aptitud global a la aptitud de las soluciones y al fenómeno de la convergencia prematura de morfología respecto a control?
- ¿Cómo afecta el uso de la diversidad genética en lugar del crowding distance?
- ¿Puede agregar un objetivo de novedad en un BC (definido en el Capítulo 3 Sección 3.3.8) que se supone como alineado a la calidad, mitigar el problema del progreso lento de los algoritmos QD, cuando la diversidad de interés (morfológica) se encuentra desalineada a la calidad? ¿Se pierde alguna característica deseable de los algoritmos QD? En especial deseable para la convergencia prematura de morfología respecto a control.
- ¿Cómo afecta a un algoritmo QD la introducción de una presión de selección global? ¿Sigue manteniendo las características deseables de los algoritmos QD? ¿Cómo se compara con un algoritmo Multi-BC QD, en cuanto al fenómeno de convergencia prematura y a la aptitud de las soluciones?

1.3. Metodología propuesta

- Proponer un marco de trabajo general para realizar experimentos de creaturas virtuales y robótica evolutiva.
  - Identificación de las características en común que comparten los experimentos de robótica evolutiva.
  - Proposición de un método general para crear experimentos de robótica evolutiva que utilice la mayor abstracción posible, e implementarlo en un lenguaje de programación moderno orientado a objetos.
- Efectuar experimentos comparativos para explorar la convergencia prematura de la morfología respecto al control debida a la pérdida de diversidad morfológica.
  - Identificación de métricas para la convergencia prematura de la morfología respecto al control que nos ayuden a establecer criterios de comparación de diversos algoritmos.
  - Identificar y proponer un experimento base para la observación cualitativa y cuantitativa de la convergencia prematura de la morfología respecto al

control.

- Identificar y proponer varios experimentos, explorando en la literatura los algoritmos que atacan explícitamente o implícitamente la convergencia prematura de la morfología respecto al control.
- Efectuar los experimentos, recopilar resultados y efectuar análisis estadístico.
- Comparar cuantitativamente los algoritmos elegidos en los experimentos y establecer un ordenamiento respecto a uno o varios de los indicadores del paso uno.

1.4. Estructura de la tesis

Este trabajo se desarrolla en 6 capítulos. Como ya lo vimos, en el presente capítulo se introduce el tema, la motivación y los objetivos de la tesis. Posteriormente, en el segundo y tercer capítulo se presenta el estado del arte, en cómputo evolutivo y la convergencia prematura en la coevolución de morfología y control de creaturas virtuales, respectivamente. En el cuarto y quinto capítulo se presentan los experimentos, los resultados y se efectúa el análisis comparativo, utilizando el marco de trabajo mencionado en los objetivos. Finalmente, en el sexto capítulo se presentan las conclusiones a las que se pudo llegar con dichos resultados.

Capítulo 2

Algoritmos evolutivos

Los algoritmos evolutivos suelen ser utilizados en problemas de optimización, como una alternativa a los métodos numéricos, cuando la función a optimizar es costosa de evaluar y no se tienen sus derivadas disponibles. Dicho escenario es justo el que se tiene en el diseño automatizado de controladores y morfología de creaturas/robots virtuales. La función de evaluación suele estar basada en alguna métrica de desempeño del robot en la tarea de interés, tomada a partir de una simulación en un ambiente virtual sujeto a ciertas leyes físicas. Dichas simulaciones suelen ser costosas, complejas y requerir un gran poder de cómputo. En este capítulo se presentan en primera instancia y de manera general los principales conceptos y algoritmos pertenecientes al cómputo evolutivo, cuidando mencionar aquellos que se han aplicado de manera prevalente en el área de creaturas virtuales y robótica evolutiva, así como algunos cuantos que constituyen avances en el área de cómputo evolutivo en general.

2.1. Algoritmos evolutivos

La teoría de la evolución de Darwin es utilizada en biología para explicar la diversidad en los seres vivos y los mecanismos que la provocan. Nos dice que en un entorno con recursos limitados y una población de individuos en constante competencia entre ellos, existe un mecanismo de selección por el cual únicamente los individuos más aptos para desempeñarse en el ambiente puedan sobrevivir para reproducirse. A esta idea se le llama "selección del más apto" [11]. Aquellos individuos que sobreviven para reproducirse con otros propagan aquellas características que los hicieron capaces de sobrevivir, pero con variaciones fenotípicas debidas tanto a la mezcla de material genético, como a mutaciones aleatorias [11]. Tanto la supervivencia del más apto, como la variación de las características son el motor para la evolución, que permite no solo la diversidad de la población, sino que conforme pasen las generaciones se vayan produciendo individuos más y más aptos en función de su ambiente y nicho.

Es de esta idea de evolución de las especies que surgen los algoritmos evolutivos: un tipo especial de algoritmo de búsqueda metaheurístico basado en poblaciones cuyo funcionamiento se encuentra inspirado en los principios de **variación** y **selección** presentes en la evolución [12].

La idea general de dichos algoritmos es:

- Tener una **población** de posibles soluciones a un problema (**individuos**).
- Evaluar dicha población con base en alguna métrica que nos indique que tan bien se desempeña cada uno de los individuos (**aptitud**) para resolver algún problema o realizar una tarea (**función de evaluación**).
- Seleccionar de forma estocástica o determinista a aquellos que sean más aptos con base en dicha métrica (**selección de supervivientes**) y de dichos supervivientes seleccionarlos para producir otra población (**selección de padres**).
- Someter dicha población a los operadores de **variación** (**mutación** y **cruza**) y producir un conjunto de descendientes y agregarlos a la población.

Este proceso se repite iterativamente y cada iteración recibe el nombre de generación, manteniendo así un conjunto de soluciones potenciales e incrementalmente mejores para la resolución del problema [11].

La explicación anterior se ilustra en el algoritmo 1, donde la línea 9 corresponde a la obtención de la aptitud de los individuos al desempeñarse en el ambiente, la línea 12 aplica el principio de selección de los más aptos y la línea 7 aplica el principio de variación. Estos tres pasos básicos se aplican hasta llegar algún criterio de paro definido conveniente. Algunos criterios de paro comunes son el máximo número de generaciones o que cierto número de soluciones lleguen a un valor de aptitud aceptable.

Algorithm 1: Algoritmo evolutivo genérico (Adaptado de [11] y del framework referenciado en [13]).

Entrada: problem, algorithm, population_size
Salida : population

```
1  //Inicializar la población de manera aleatoria
2  initial_population = problem.initialize_population(population_size)
3  problem.evaluate(initial_population)
4  parent_population = algorithm.select(initial_population)
5  while !algorithm.stop_criteria(parent_population) do
6  |   //Aplicar los operadores de variación genética a la población de padres (cruza,
   |    mutación o ambos), para producir la población de hijos de la siguiente
   |    generación
7  |   child_population = algorithm.variate(parent_population)
8  |   //Evaluamos a los individuos para obtener su aptitud
9  |   problem.evaluate(child_population)
10 |   //Aplicamos supervivencia y algún esquema de selección para elegir a los
   |    //padres de la siguiente generación
11 |   survivors_pool = algorithm.survival(parent_population, child_population)
12 |   parent_population = algorithm.select(survivors_pool)
13 end
14 population = parent_population
```

Las distintas formas de implementar las líneas 7, 9, 11 y 12, junto con la forma de representación de las soluciones a nivel genotipo-fenotipo, y el mapeo entre ambas representaciones, constituyen las distintas variantes de algoritmos evolutivos y derivan en las 4 principales familias de estos: Algoritmos genéticos, Programación genética, Estrategias evolutivas y Programación evolutiva [11].

Cuando hablamos de representación de las soluciones, nos referimos a la elección de una estructura de datos adecuada para codificar las características expresadas como la solución a un problema. La codificación de las soluciones representa el genotipo del individuo, mientras que dicha codificación expresada o decodificada como la solución a un problema se denomina fenotipo.

La representación está fuertemente acoplada con el problema a resolver. Por ejemplo una estructura natural para resolver problemas de tipo combinatorio, como el problema de las n-reinas, es mediante una lista de números enteros. Para dicho caso, cada posible solución se representa como una lista de n números enteros sin repetición (permutaciones), donde cada elemento de la lista es la columna ocupada por la reina del renglón i-ésimo.

En resumen, existen 6 componentes principales en un algoritmo evolutivo [11]:

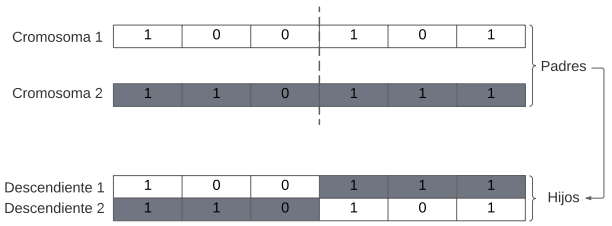
- Representación.

- Población de individuos.
- Función de aptitud o evaluación.
- Operadores de variación.
- Mecanismo de selección de supervivientes.
- Mecanismo de selección de padres.

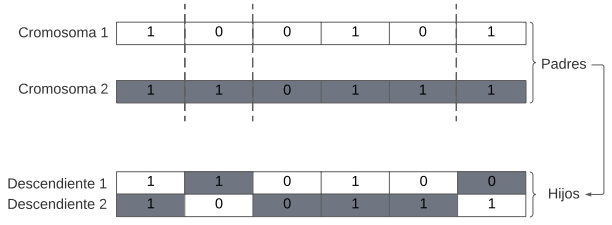
De dichos componentes, además de la representación, la población de individuos y la función de aptitud suelen ser fuertemente dependientes del problema.

Las cuatro familias de algoritmos evolutivos, son en realidad sólo puntos en un espacio de diseño de algoritmos potencialmente infinito por la cantidad de combinaciones y distintas formas de implementar cada uno de los componentes anteriormente expuestos. En realidad esta subdivisión únicamente existe por razones históricas, ya que en la práctica el problema puede ameritar un algoritmo evolutivo que no caiga en ninguna de las cuatro familias o que pertenezca a varias [12]. A continuación se presentan únicamente tres de las cuatro familias de algoritmos evolutivos, con base en la relevancia que presentan en la literatura de coevolución de morfología y control.

2.1.1. Algoritmos genéticos



(a) Cruza de un punto.



(b) Cruza multipunto.

Figura 2.1: Cruza de un solo punto y multipunto en algoritmos genéticos con representación binaria.

Los algoritmos genéticos fueron desarrollados por John H. Holland en la década de los sesenta, culminando con la publicación de su libro en 1975 [14], como un intento de replicar y estudiar el comportamiento de los sistemas adaptativos naturales por medio de la creación de sistemas adaptativos artificiales, la motivación original no fue en realidad la resolución de problemas de optimización.

2.1.1.1. Representación

Este tipo de algoritmos tienen la característica de no operar directamente sobre las posibles soluciones del problema, sino que utilizan codificaciones que abstraen el concepto de genotipo y fenotipo. Donde el genotipo o la representación codificada se encuentra en el espacio de búsqueda y el fenotipo o el genotipo expresado en características, se encuentra en el espacio de soluciones. La transformación entre genotipo y fenotipo se efectúa mediante una operación de decodificación y es crucial, dado que la evaluación de los individuos se suele efectuar sobre el fenotipo.

Gracias a la operación de decodificación, en teoría podemos utilizar varias representaciones para un mismo problema, siempre y cuando garanticemos que el fenotipo resultante sea el mismo con todas ellas. De tal suerte que podemos utilizar los mismos algoritmos de cruza y mutación específicos a ciertas codificaciones para cualquier problema y simplemente proveer una forma de decodificar del genotipo al fenotipo, para cada problema específico.

Las primeras implementaciones utilizaron una codificación binaria para el genotipo [14], y ya que ésta es favorecida por la teoría de los algoritmos genéticos en particular el teorema del esquema [15], se sigue utilizando por los teóricos, sin embargo, los algoritmos genéticos modernos suelen utilizar codificaciones de números reales.

2.1.1.2. Variación

El operador de cruza que se suele utilizar es la cruza de un punto o multipunto. Dicho operador, en su variante de un punto, consiste en elegir una posición en el cromosoma de manera aleatoria (punto de cruza) y producir dos hijos, uno con el material genético del primer padre hasta el punto de cruza y el del segundo padre del punto de cruza hasta el final. El segundo hijo se producirá similarmente pero invirtiendo el orden. En la variante multipunto, se tienen m-puntos de cruza, y se altera el material genético de manera similar. Tanto la cruza de un punto como la multipunto se ilustran en las Figuras 2.1a y 2.1b respectivamente. En cuanto a la mutación, para cada alelo del cromosoma se genera un número aleatorio entre 0 y 1, si es menor que la probabilidad de mutación P_m se invierte el bit. En estos algoritmos, el operador principal es la cruza, mientras que la mutación es un operador secundario. Esto significa que, dados dos padres, la probabilidad de cruza entre ellos es mucho más alta ($P_c \geq 0.7$) que la probabilidad de mutación ($P_m \leq \frac{1}{l|G|}$, donde l es la longitud del genoma y $|G|$ es el número total de padres a elegir).

2.1.1.3. Selección

En este tipo de algoritmos se suele observar un tipo de selección de supervivientes bajo el esquema (μ, λ), es decir, la población de supervivientes se obtendrá sustituyendo la población de padres μ, por la de hijos λ. Mientras que la selección de padres se puede dar con cualquier algoritmo de selección (e.g. torneo, ruleta, etc.).

2.1.2. Programación genética

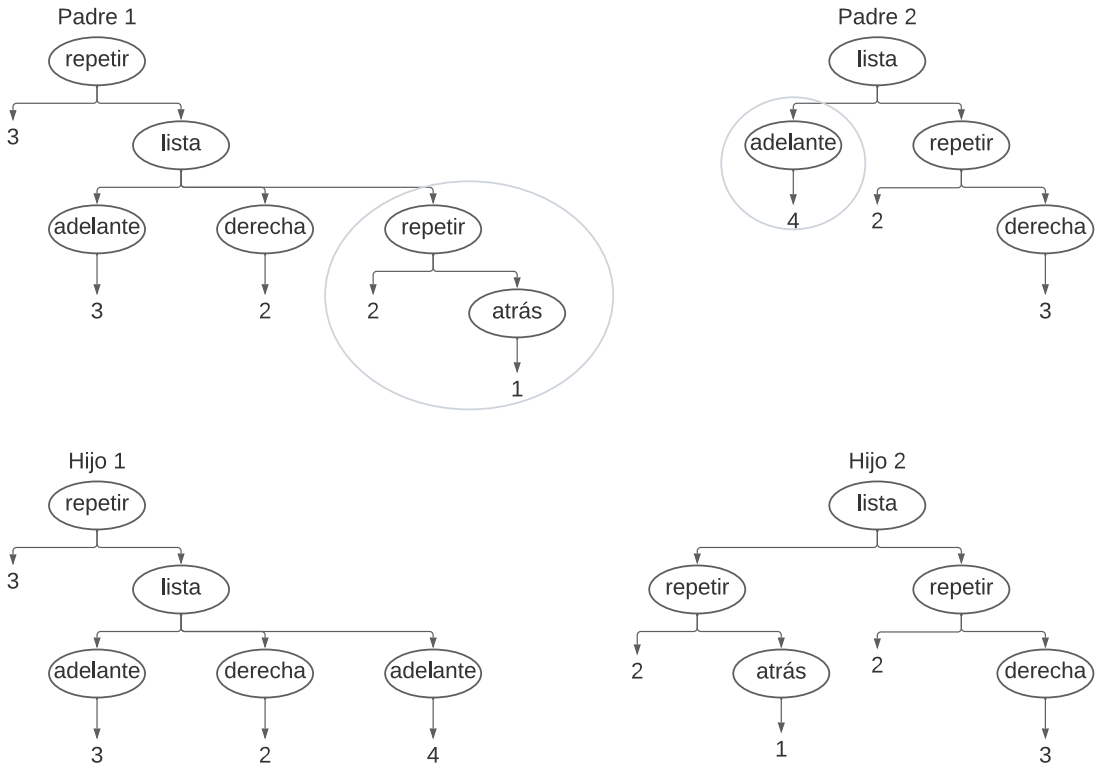


Figura 2.2: Cruza en programación genética (replicada de [12]).

La programación genética fue propuesta por Koza [16] en la década de los 90 como un tipo especializado de algoritmo genético aplicado a estructuras de árbol que representan programas o funciones y argumentos.

2.1.2.1. Representación

La representación es el enfoque principal y mayor innovación de la programación genética, como ya se mencionó, los genotipos son estructuras de tipo árbol que pueden representar árboles sintácticos y/o semánticos de programas, o funciones con argumentos.

2.1.2.2. Variación

Al igual que los algoritmos genéticos, también tienen como operador principal la cruza y como operador secundario la mutación. Para la cruza se eligen aleatoriamente dos padres, y un nodo de cada padre, dichos nodos serán los puntos de cruza, de manera tal que se intercambian los subárboles a partir de ambos nodos para producir a los dos hijos (Figura 2.2). En la implementación original de Koza [16] no existía tal cosa como mutación aplicada a programación genética, argumentando que con la cruza sería suficiente dado que produce por si sola la variabilidad correspondiente a la mutación, sin embargo, en la siguiente parte de su obra fundacional [17], la mutación fue incluida como la elección aleatoria de un nodo en el padre y la sustitución del subárbol a partir de dicho nodo, por un árbol producido de manera aleatoria.

2.1.2.3. Selección

Dado que son un tipo especializado de algoritmo genético, suelen presentar un tipo de selección de supervivientes (μ, λ), y la selección de padres se puede dar con cualquier algoritmo de selección.

Algunos problemas que surgen con la PG son:

- Redundancia: Los genotipos muchas veces pueden contener código redundante o basura. A este código genético se le conoce como intrones.
- Redundancia funcional: Muchos genotipos pueden mapearse al mismo fenotipo. Tal fenómeno se debe a código redundante que al final no aporta nueva información al traducirse en forma del fenotipo. Por lo anterior, la diversidad genotípica no necesariamente se corresponde con diversidad en los fenotipos.
- Bloat: Es el rápido crecimiento (exponencial en algunos casos) en el tamaño o complejidad de los individuos (en PG se traduce en programas compuestos de muchos nodos), este crecimiento además no se ve traducido en incrementos en fitness.

  Cuando se presenta, la evaluación de los programas se puede hacer muy lenta y requerir mucha memoria, en algunos casos, cuando el árbol llega a cierta profundidad se puede hacer incluso imposible poder evaluarlo, dado que la pila de llamadas en los lenguajes de programación tiene un límite, evaluar expresiones que excedan la pila de llamadas lleva a un desbordamiento de la pila.
- Tamaño del espacio de búsqueda: En programación genética, el espacio de búsqueda se define como el espacio de todas las estructuras posibles que se pueden formar a partir de las funciones (alfabeto de no terminales) y argumentos (alfabeto de terminales) definidos para el problema, esto da pie a espacios de búsqueda potencialmente infinitos.

Otros problemas que se dan no sólo en PG, sino en cómputo evolutivo en general:

- Epistasis: Es cuando un mismo fenotipo se puede trazar a varios genes, es decir, el fenotipo es el resultado de las interacciones complejas de varios genes.
- Convergencia: La población deja de progresar, la diversidad se reduce y los individuos son similares.

De los puntos anteriores, los primeros tres junto con epistasis, se pueden soportar en la teoría de la evolución neutral, la cual nos dice que la mayoría de la variación entre especies se debe a alelos mutantes que son neutrales en el proceso de selección, es decir, no representan ni una ventaja ni desventaja para la supervivencia y reproducción de los organismos.

Tradicionalmente la programación genética se ha aplicado a problemas de aprendizaje de máquina supervisado, tales como la regresión simbólica, en la cual se tiene un conjunto de datos $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, producido por alguna función $\mathbf{y}^{(i)} = F(\mathbf{x}^{(i)})$, la idea de la regresión simbólica es evolucionar una población de programas hasta producir uno o varios que aproximen con el menor error $\mathbf{y}^{(i)}$, es decir, que se ajusten mejor a la función que produjo los datos.

2.1.3. Estrategias evolutivas

Las estrategias evolutivas fueron propuestas por Ingo Rechenberg en 1964.

2.1.3.1. Representación

En las estrategias evolutivas el genotipo y fenotipo son iguales, es decir, los cromosomas son directamente los parámetros del problema y suelen estar representados por vectores de números reales.

2.1.3.2. Variación

En este tipo de algoritmo, el operador principal es la mutación y la cruza el secundario, sin considerarse la mayoría de las veces. Las mutación, suele consistir en efectuar pequeñas desviaciones a los genes. Dichas desviaciones son, en su forma más simple, obtenidas a partir de una distribución Gaussiana con media cero y desviación estándar uno.

El problema, es que no existe una adaptabilidad al progreso de la búsqueda. Es por ello que en implementaciones más avanzadas se introducen al cromosoma los llamados parámetros de estrategia. Dichos parámetros de estrategia representan las desviaciones estándar de cada parámetro del problema, y varían junto con los mismos, de tal manera que se tienen genotipos de la forma $(x_1, ..., x_i, ..., x_n, \sigma_1^2, ..., \sigma_i^2, ...\sigma_n^2) \in \mathbb{R}^{2n}$, donde n es el número de dimensiones que tiene el espacio de soluciones del problema en cuestión.

2.1.3.3. Selección

En su forma más simple se componen de un padre y un hijo, conocida como $(1+1)-EE$, donde los unos representan al número de padres y el número de hijos, y el signo más representa selección de supervivientes aplicada a ambos. Se aplica el operador de mutación al padre, produciéndose así un hijo, se evalúa al padre y al hijo, y se mantiene al que sea mejor para que sea el padre en la siguiente generación.

Las estrategias evolutivas modernas suelen ser poblacionales, y tienen el esquema de selección de supervivientes conocido como $(\mu+\lambda)-EE$, donde ya no se sustituye simplemente a los padres por los hijos, sino que se elige a los superviviente de entre las dos poblaciones, ya sea de manera determinista (elitismo) o no determinista.

2.2. Algoritmos evolutivos multiobjetivo

En algoritmos evolutivos de un solo objetivo se tienen funciones escalares de aptitud, es decir, funciones de la forma $\mathbb{R}^n \to \mathbb{R}$. Cuando las funciones a optimizar son vectoriales de variable vectorial, i.e. $\mathbb{R}^n \to \mathbb{R}^m$, se comienza a hablar de algoritmos evolutivos multiobjetivo.

Para un problema multiobjetivo de minimización, se desea encontrar un vector $x^* \in \mathcal{Q}$ (donde $\mathcal{Q} \subset \mathbb{R}^n$), tal que minimice la función $F : \mathcal{Q} \to \mathbb{R}^m$ sujeta a restricciones de desigualdad $G : \mathcal{Q} \to \mathbb{R}^J$ y de igualdad $H : \mathcal{Q} \to \mathbb{R}^K$.

Lo anterior se puede expresar matemáticamente como:

$$\begin{aligned} x^* = \arg\min_{x\in\mathcal{Q}} & F(x). \\ \text{s.t. } & G(x) \leq \hat{0}. \\ & H(x) = \hat{0}, \end{aligned} \tag{2.1}$$

$$\text{donde: } F(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_i(x) \\ \vdots \\ f_m(x) \end{bmatrix}. \; G(x) = \begin{bmatrix} g_1(x) \\ \vdots \\ g_j(x) \\ \vdots \\ g_J(x) \end{bmatrix}. \; H(x) = \begin{bmatrix} h_1(x) \\ \vdots \\ h_k(x) \\ \vdots \\ h_K(x) \end{bmatrix}.$$

$$\mathcal{Q} = \{x \in \mathbb{R}^n |\; g_j(x) \leq 0 \; j \in \{1, \cdots, J\} \;\&\; h_k(x) = 0 \; k \in \{1, \cdots, K\}\}.$$

Al extender la noción de aptitud a más de una dimensión ya no se habla de individuos mejores o peores que otros en un solo criterio u objetivo, sino que se tienen varios objetivos en los cuales los individuos pueden ser mejores o peores. Es por este conflicto entre objetivos que ya no se habla de una sola solución óptima, sino de una familia de soluciones que optimicen la función de aptitud. Existen varios métodos para resolver

problemas multiobjetivo, pero para efectos de esta tesis únicamente nos interesaremos en aquellos basados en optimalidad de Pareto, donde la dominancia de Pareto es el concepto central. Un individuo domina a otro o es mejor que otro en el sentido de Pareto, si lo supera en todos los objetivos. Es decir:

$$\begin{gathered}\text{Sea } F : \Omega \to \mathbb{R}^m.\\ \text{Sean } x \text{ y } y \text{ dos vectores en } \Omega.\\ \text{Se dice que } x \text{ domina o precede a } y \text{ (i.e. } x \preceq y\text{), o bien:}\\ x \preceq y \Longleftrightarrow f_i(x) \leq f_i(y)\ \forall i \in [1, m].\\ F(x) \neq F(y).\end{gathered} \tag{2.2}$$

Ahora bien, cabe recalcar que dicha relación no implica que si $x \not\preceq y$ inmediatamente $y \not\preceq x$. Cuando $x \not\preceq y$ y $y \not\preceq x$, se dice que x y y no se dominan. Un conjunto de vectores que no son dominados por ningún otro se conoce como un conjunto de Pareto en el espacio de búsqueda o de decisión Ω, mientras que a su imagen en el espacio de los objetivos $\mathbb{R}^m$, se le conoce como frente de Pareto. Las soluciones a problemas multiobjetivo son conjuntos de Pareto. Más aún, una solución S_1 a un problema multiobjetivo será óptima en el sentido de Pareto si no existe otra solución S_2 tal que mejore en un objetivo sin empeorar al menos uno de los otros.

En la Figura 2.3(izquierda) se muestran 3 frentes de Pareto aproximados de un problema multiobjetivo de minimización, donde los puntos en rojo representan en este caso el frente óptimo de Pareto, dado que todos ellos no son dominados por ningún otro. Podemos observar que tal agrupamiento en frentes nos da un ordenamiento, o ranking parcial de las soluciones. Los algoritmos evolutivos multiobjetivo basados en optimalidad de Pareto se basan en aproximaciones sucesivas del frente óptimo de Pareto y selección basada en los ordenamientos parciales derivados de los frentes resultantes al evaluar las soluciones en cada generación.

2.2.1. NSGA-II

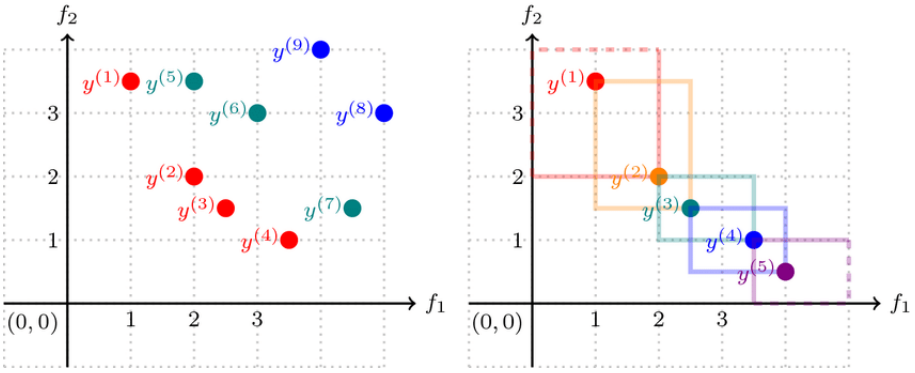


Figura 2.3: Ordenamiento parcial a través de frentes de Pareto (izquierda) y crowding distance (derecha) [18].

Algorithm 2: Selección de supervivientes en NSGA-II basada en el código de [13] y en [19], [20].

Entrada: parent_population, child_population
Salida : survivors_pool

$(S_1, S_2, \cdots, S_m) = non\_dominated\_sorting(parent\_population \cup child\_population)$
$survivors\_pool = \{\}$
$i = 1$
$N = |parent\_population|$
while $|survivors\_pool| + |S_i| < N$ **do**
    $S_i = crowding\_distance(S_i)$
    $survivors\_pool = survivors\_pool \cup S_i$
    $i = i + 1$
end
$S_L = S_i$
$S_L = crowding\_distance(S_L)$
if $|survivors\_pool| + |S_L| > N$ **then**
    $S_L = sort\_by\_cd(S_L)$
    $S_L = \{S_L^{(1)}, S_L^{(2)}, ..., S_L^{N-|S_L|}\}$
end
$survivors\_pool = survivors\_pool \cup S_L$

Se trata probablemente de uno de los algoritmos evolutivos multiobjetivo (MOEA) más conocidos y aplicados. Con un rendimiento que se acerca al estado del arte, pertenece a la familia de los algoritmos basados en optimalidad de Pareto. A grandes rasgos, NSGA-II (Non-dominated Sorting Genetic Algorithm - II) realiza en cada generación

aproximaciones sucesivas al frente óptimo de Pareto, ordenando a los individuos de acuerdo a dos criterios: el frente al que pertenezcan, y la diversidad que presenten en el espacio de los objetivos medida con respecto al crowding distance (Figura 2.3 derecha), logrando de esta forma un ordenamiento total.

Se trata de un algoritmo genético con un esquema de selección de supervivientes basado en $(\mu + \lambda) - EE$, fundamentalmente distinto a una EE principalmente por la distinción entre genotipo y fenotipo, y por la presencia de cruza como operador de variación principal, características que lo sitúan en el dominio de los algoritmos genéticos. La selección de supervivientes se basa en el ordenamiento de no dominados (Non-dominated Sorting) y el crowding distance, con lo cual se logra un ordenamiento absoluto de las soluciones.

El ordenamiento de no dominados, se encarga de agrupar a los individuos de la generación en frentes de Pareto $\{S_1, S_2, \cdots, S_m\}$, siendo S_1 el ranking 1 y así sucesivamente hasta llegar al frente S_m. Un ejemplo de tal ordenamiento se muestra en la Figura 2.3(izquierda). Posteriormente, para cada frente se calcula el crowding distance, una métrica que se encarga de cuantificar la diversidad de una solución en particular respecto a las demás dentro del mismo frente. Intuitivamente el crowding distance C de un vector x en un frente S, se puede ver como el perímetro del hipercubo formado por las soluciones adyacentes a x en el mismo frente, tal y como se puede visualizar en la Figura 2.3 (derecha). La idea anterior, se expresa formalmente como:

$$C(x) = \sum_{i=1}^{n} c_i(x), \ \ x \in S, \tag{2.3}$$

$$\text{donde } c_i(x) = u_i - l_i.$$

$$l_i(x) = \text{máx}(\{f_i(y) | y \in S \backslash \{x\} \& f_i(y) \leq f_i(x)\} \cup \{-\infty\}).$$

$$u_i(x) = \text{mín}(\{f_i(y) | y \in S \backslash \{x\} \& f_i(y) \geq f_i(x)\} \cup \{\infty\}).$$

De lo anterior, es evidente que para los extremos del frente, $C(x) = \infty$. Si observamos la Figura 2.3 (derecha), en el frente óptimo de Pareto $S_1 = \{y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)}\}$, $C(y^{(4)}), C(y^{(1)}) = \infty$.

Siguiendo el algoritmo 1, la implementación de NSGA-II únicamente depende de la selección de supervivientes y la selección de padres. La selección de supervivientes se basa en el non-dominated sorting de la población de padres e hijos, quedándose con los frentes con mejores rankings, hasta llenar la población de padres. Si el último frente en ser agregado a la población de padres provoca que esta última exceda el máximo, se utiliza el crowding distance como factor para decidir cuales individuos de este último frente van a sobrevivir al ordenarlos de manera descendente respecto al crowding distance y quedarse con la porción que llene a la población de padres. Dicho procedimiento de selección de padres se ilustra en el algoritmo 2. Posteriormente, a partir de los supervivientes, se selecciona a los padres para la siguiente generación por medio de torneo binario. Para ello, se utiliza el ordenamiento completo que se obtiene

al realizar el non-dominated sorting, y posteriormente ordenar cada frente de acuerdo al crowding distance.

Capítulo 3

Coevolución de morfología y control de creaturas virtuales

Ahondando un poco en el contexto histórico del campo, la coevolución de morfología y control surge como parte del trabajo de Sims [21] en el ámbito de la computación gráfica y animación. Sims reconoce la dificultad y complejidad de diseñar controladores para objetos en un ambiente virtual con dinámica simulada, la cual aumenta conforme la complejidad de los objetos también aumenta. Además, cada vez que se quiera cambiar el comportamiento deseado o la morfología se tiene que cambiar el controlador. Para abordar este problema, se propone la utilización de métodos de optimización, específicamente de algoritmos genéticos, para generar automáticamente la morfología y el control de las llamadas criaturas virtuales, las cuales son capaces de realizar tareas como: caminar, nadar, moverse hacia un punto en movimiento, competir por alimento, etc. [22].

A grandes rasgos el proceso que se sigue para realizar la coevolución de morfología y control en el trabajo de Sims [21], consiste en inicializar una población de criaturas de manera aleatoria y aplicar el ciclo del algoritmo evolutivo genérico descrito en el Algoritmo 1 hasta que se cumpla algún criterio de convergencia. La evaluación consiste en efectuar simulaciones físicas de las creaturas de cada generación y registrar el desempeño de cada una en la tarea de interés. Cada creatura, tiene un controlador (cerebro y sistema nervioso) y una correspondiente morfología (cuerpo), ambas codificadas como un grafo conexo anidado. Cada nodo morfológico tiene a su vez un subgrafo de control asociado y dicho subgrafo es parte de la topología del grafo de control de la criatura, que de manera similar a lo que sucede en los seres vivos, está embebida en la morfología. La codificación de cada criatura se expresa en el ambiente y se evalúa qué tan buena es realizando una cierta tarea al someterla a una simulación en un ambiente sujeto a ciertas leyes físicas. La variación de dicho ambiente y de las tareas a realizar guían las formas y comportamientos que se evolucionan. Al evaluar a las criaturas de esta

manera se está buscando tener una métrica para poder evaluar la aptitud para realizar la tarea solicitada y así aplicar un algoritmo evolutivo. Por razones de costo y practicidad, la evaluación en simuladores de física se aplica en la mayoría de los trabajos de coevolución de morfología y control, o donde se evoluciona alguno de los dos aspectos.

El proceso anterior se puede generalizar para servir como una plantilla de un experimento de coevolución de morfología y control, dejando abiertas distintas formas de aproximarse al problema al identificar los componentes fundamentales de un experimento de este dominio. Una forma de identificar dichos componentes es reconociendo los aspectos en común que tienen los experimentos al examinar el estado del arte. A continuación, se enumeran los aspectos a considerar cuando se quiere proponer un experimento en este dominio.

3.1. Representaciones

Una clasificación prevalente de las diversas metodologías y algoritmos presentes en el campo es aquella basada en la representación (genotipo y expresión en el fenotipo) de los controladores y la morfología.

En esta sección nos enfocaremos en los tipos distintos de representaciones para coevolución de morfología y control, cuidando hacer alguna mención de importancia histórica de las representaciones que se enfocan únicamente en alguno de los dos aspectos.

3.1.1. Genotipo y fenotipo

Una buena representación del espacio de búsqueda o de soluciones en algoritmos evolutivos y en general en inteligencia artificial, es crucial y puede marcar la diferencia entre éxito o fracaso al momento de su aplicación en algún problema específico [23]. La definición de un genotipo y su mapeo al fenotipo, de manera que considere todas las partes de un robot es una pregunta abierta en el área. La morfología y control de un robot (o una criatura virtual) son interdependientes y cualquier cambio en uno puede influenciar al otro, es por ello que en una representación adecuada se deben considerar ambos aspectos de manera simultánea [2]. La idea aquí es que la dinámica acoplada entre el cuerpo, el cerebro y el entorno de un agente provocan que emerja comportamiento inteligente en éste, algo que se conoce como inteligencia (artificial) corporizada [5].

Es desde el trabajo seminal de Karl Sims [21] donde se reconoce la dificultad y complejidad de diseñar controladores para objetos tridimensionales en un ambiente virtual con dinámica simulada (animación basada en simulaciones físicas [24]).

Sims toma la idea de representar al genotipo como expresiones de Lisp (tal y como sucede en programación genética) que codifican la morfología y control como un par de grafos dirigidos, posiblemente cíclicos, donde la morfología es el grafo principal y el control se encuentra anidado en el grafo de morfología (Figura 3.1).

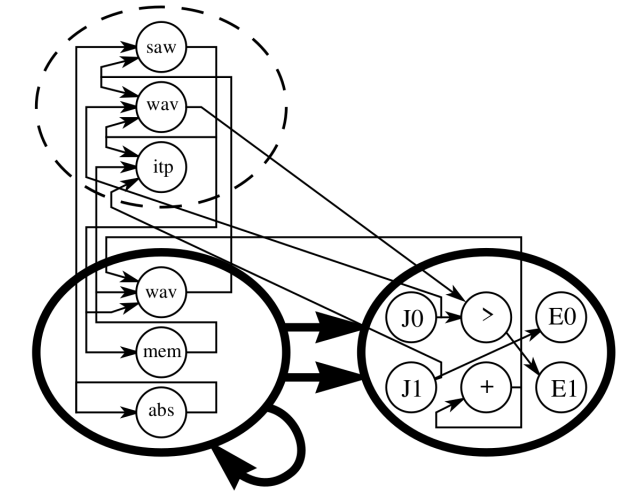


Figura 3.1: Ejemplo de genotipo de las creaturas de Sims [21].

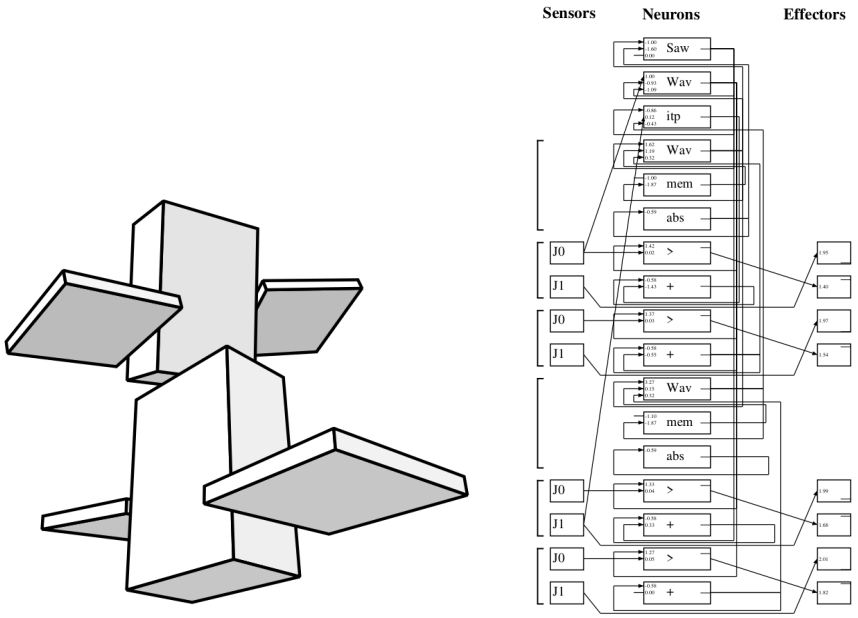
Para la morfología, los nodos del grafo representan cuerpos rígidos y las aristas representan articulaciones para unirlos. Se pueden tener varias aristas que lleven de un nodo a otro para crear simetrías, ya que representan conexiones a distintas instancias del tipo de nodo al que lleven, también se puede tener ciclos que permiten recursividad.

Con dicha codificación es posible producir estructuras recursivas, auto-similares y modulares (Figura 3.2a), como por ejemplo desde un brazo hasta un humanoide. En éste último, las extremidades se generan desde un nodo, definiendo el torso conectado con 4 aristas a un nodo de extremidades que a su vez definiría el módulo de extremidad recursivamente conectado (Figura 3.3).

En el grafo de control cada nodo representa neuronas de Mculloch y Pitts (perceptrones simples) [25] con distintas funciones de activación. Cada nodo puede ser ya sea un sensor, efector o unidad de procesamiento que efectúe alguna operación. Un sensor recibe directamente estímulos del ambiente o del ángulo de una articulación, una neurona de procesamiento puede tener como entrada la salida de un sensor, otra neurona, o su propia salida, un efector puede tener como entrada la salida de un sensor o la salida de una neurona de procesamiento. Los actuadores escalan la señal de entrada y la llevan a los rangos permitidos para incidir una fuerza lineal o una torca en una articulación.

Este sistema de neuronas interconectadas expresa un sistema de control que se asemeja a un programa del estilo de los que se evolucionan en programación genética y le permite a la creatura tener comportamientos arbitrarios (Figura 3.2b).

Se han dado algunas simplificaciones del modelo de Sims limitándose a grafos acíclicos, y a solo un subconjunto de las funciones de activación utilizadas por Sims originalmente, específicamente sigmoide y radial [26] [27]. La razón para utilizar solo estas dos funciones es para investigar si pueden emerger comportamientos complejos tan solo a partir del



(a) Ejemplo del fenotipo de morfología

(b) Ejemplo del fenotipo de control

Figura 3.2: Fenotipo ya expresado de las creaturas de Sims [21].

proceso evolutivo en sí, en particular los comportamientos oscilatorios que permiten los ciclos de caminado. En los experimentos originales de Sims, los comportamientos oscilatorios se atribuyen a neuronas con funciones de activación oscilatorias [27].

Otras aproximaciones similares a aquellas basadas en programación genética se han llevado a cabo a través de sistemas-L (Gramáticas Lindenmayer) [28] [29], un tipo especial de gramática libre de contexto, donde las reglas de producción se aplican en paralelo a los símbolos no terminales, en una forma inspirada por como se dan las divisiones celulares en organismos multicelulares (Figura 3.4). En estos trabajos tanto morfología como control se representan en un sólo sistema-L paramétrico, un tipo especial de sistema-L donde se incorporan parámetros que pueden recibir los no terminales a la hora de ser aplicados, estos parámetros a su vez pueden ser manipulados algebraicamente para generar nuevos valores que se pasen como parámetros a los estados sucesores y que sean utilizados en expresiones condicionales (Figura 3.5).

Los ejemplos presentados, son instancias de un tipo de representación conocido en la literatura como codificación generativa, donde como indica el nombre, el genotipo no tiene un mapeo directo al fenotipo, sino que posee instrucciones y especificaciones que permiten regularidad [23], y ejemplifican una forma compacta de representar estructuras que pueden llegar a ser arbitrariamente complejas (escalabilidad del genotipo con respecto al tamaño del fenotipo).

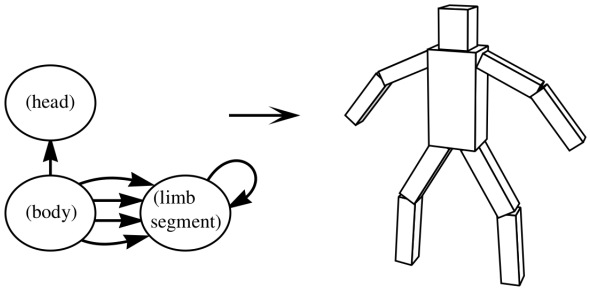


Figura 3.3: Ejemplo de genotipo y fenotipo de morfología para un humanoide [21].

$$\begin{array}{lll} & & a \\ & & ab \\ a: & \rightarrow a\ b & abba \\ b: & \rightarrow b\ a & abbabaab \end{array}$$

Figura 3.4: Ejemplo de un sistema-L y una aplicación de sus reglas de producción a una cadena inicial [28].

3.1.2. Codificaciones directas

Existen dos tipos principales de codificaciones para la representación de la morfología y control de creaturas virtuales: directas y generativas (indirectas). Las primeras, constituyen un mapeo uno a uno de los componentes del genotipo a los componentes del fenotipo. Con base en las estructuras de datos utilizadas para representarlas, se puede encontrar principalmente las **representaciones paramétricas** y **representaciones directas basadas en grafos**.

3.1.2.1. Representaciones paramétricas

El genotipo puede ser una lista de longitud fija o variable, e.g. un vector de números reales, donde cada variable representa un parámetro de la morfología (e.g. dimensiones espaciales de una parte o partes específicas, número de partes, peso, etc.) o del control (e.g. amplitud, fase, frecuencia, etc.). Este tipo de representaciones suelen prestarse a técnicas de cruza y mutación de codificación binaria, real o entera, e incorporar esquemas de selección de algoritmos genéticos, estrategias evolutivas o de alguna de las variantes multiobjetivo.

Algunos ejemplos de trabajos que han utilizado este tipo de representación son:

- En el trabajo de Nygaard et al. [10], se utilizó una representación de este tipo para optimizar el control y morfología de robots de seis piernas (Figura 3.6), simétricos verticalmente. El genotipo de longitud fija consistió de 11 parámetros para la morfología (Tabla 3.1) y 36 para el controlador (Tabla 3.2). Buscando optimizar

			$a(4)$
			$a(3)b(4)$
			$a(2)b(3)b(2)a(3)$
$a(n):$	$(n > 1)$	$\rightarrow a(n-1)\ b(n)$	$a(1)b(2)b(1.5)a(2)b(0)a(2)b(3)$
$a(n):$	$(n \leq 1)$	$\rightarrow a(0)$	$a(0)b(0)b(0)a(1)b(2)b(0)a(1)b(2)b(1.5)a(2)$
$b(n):$	$(n > 2)$	$\rightarrow b(n/2)\ a(n-1)$	$a(0)b(0)b(0)a(0)b(0)b(0)a(0)b(0)b(0)a(1)b(2)$
$b(n):$	$(n \leq 2)$	$\rightarrow b(0)$	$a(0)b(0)b(0)a(0)b(0)b(0)a(0)b(0)b(0)a(0)b(0)$

Figura 3.5: Ejemplo de un sistema-L paramétrico y una aplicación de sus reglas de producción a una cadena inicial [28].

2 objetivos: distancia recorrida y peso (maximizar y minimizar respectivamente), utilizando el algoritmo NSGA-II.

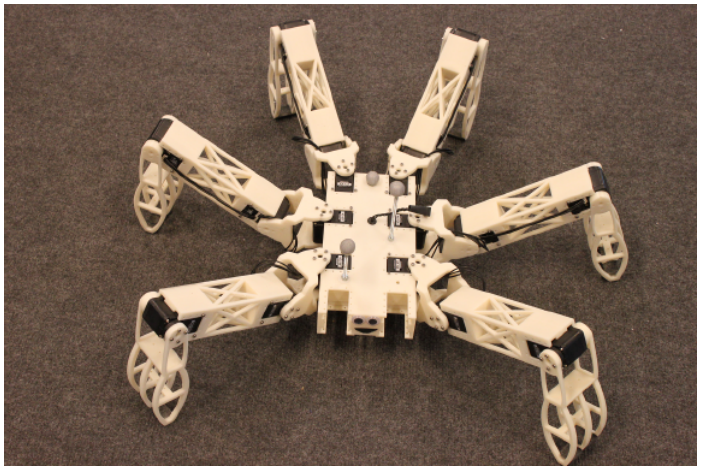


Figura 3.6: Robot evolucionado en [10]. Uno de los pocos trabajos que se preocupa por realizar evaluaciones de prototipos reales.

- Lipson y Pollack [30], utilizaron una representación de longitud variable, donde un robot se encuentra especificado por un conjunto de vértices, barras, neuronas y actuadores (Ecuación 3.1). En el proceso evolutivo, que es un híbrido entre estrategia evolutiva y programación evolutiva, los pesos de las sinapsis entre neuronas y las longitudes de las barras, ambos representados por parámetros reales, pueden ser mutados con cierta probabilidad.

$$\begin{aligned} \text{robot:} &= \langle\text{vértices}\rangle\langle\text{barras}\rangle\langle\text{neuronas}\rangle\langle\text{actuadores}\rangle. \\ \text{vértice:} &= \langle x, y, z\rangle. \\ \text{barra:} &= \langle\text{índice del vértice 1}, \text{índice del vértice 2}, \text{longitud}, \text{rigidez}\rangle. \\ \text{neurona:} &= \langle\text{umbral}, \text{pesos de sinapsis}\rangle. \\ \text{actuador:} &= \langle\text{índice de barra}, \text{índice de neurona}, \text{rango de barra}\rangle. \end{aligned} \tag{3.1}$$

Parámetro	# de parámetros	Intervalo
Longitud de Tibia	3	$[80, 254]mm$
Longitud de fémur	3	$[80, 254]mm$
B1	1	$[52.64, 94]mm$
B2	1	$[83, 284]mm$
B3	1	$[61.5, 254]mm$
B4	1	$[52.65, 254]mm$
B5	1	$[61.5, 254]mm$

Tabla 3.1: Genoma de la morfología en [10].

Parámetro	# de parámetros	Intervalo
Movimiento coxal 1	4	$[-0.81, 1.64]$
Movimiento coxal 2	2	$[-1.64, 1.64]$
Movimiento del fémur	6	$[-2.49, 2.49]$
Movimiento de tibia	6	$[-2.49, 2.49]$
Fases	18	$[-\pi, \pi]$

Tabla 3.2: Genoma del control en [10].

- Cheney et al. [31] utilizan una codificación directa de esta índole para compararla contra una codificación generativa basada en CPPN-NEAT (del cual se hablará en la Sección 3.1.3). En este trabajo se evolucionan robots con morfologías blandas, compuestos de voxeles hechos de distintos materiales (actuador, actuador con desfase relativo de 90 grados, pasivo y rígido). En cada voxel actuador, el control está embebido en forma de actuadores volumétricos que se expanden o contraen de manera periódica a una frecuencia específica. La codificación directa en estos experimentos es un genoma donde cada voxel tiene un valor binario representando su presencia o ausencia, y cuatro valores reales representando la probabilidad de que se encuentre hecho de algún material específico (al mayor de estos será el material del voxel). El tamaño de este genoma será de $2 \cdot n_{vox}$, donde n_{vox} estará especificado por la discretización de un espacio en 3D de tamaño fijo. e.g. Una discretización del espacio de 10 nos daría un tamaño del genoma de: $n_{gen} = 2 \cdot 10^3 = 2000$. De lo anterior es visible como en general, una codificación directa

puede explotar en tamaño conforme la dimensionalidad del problema aumenta.

3.1.2.2. Representaciones basadas en grafos

Este tipo de representaciones se suelen utilizar con técnicas de programación genética y programación evolutiva para la cruza y mutación, así como los esquemas de selección propios de las mismas o sus variantes multiobjetivo.

- En el mismo trabajo de Lipson y Pollack [30] mencionado anteriormente, la parte de programación evolutiva se da en los operadores de mutación de quitar o agregar barras o neuronas no conectadas, dividir un vértice en dos y agregar una barra o dividir una barra en dos y agregar un vértice y agregar o remover una neurona a una barra.
- Komosinsky y Rotaru-Varga [32], comparan codificaciones generativas contra directas. En la variante de sus creaturas correspondiente a la codificación directa, únicamente permiten que se puedan evolucionar estructuras arborescentes que son evolucionadas de manera similar a como sucede en programación genética
- Lipson y Pollack [28], comparan su codificación generativa basada en sistemas-L con una variante directa basada en únicamente permitir utilizar hasta 10000 funciones de construcción en secuencia para crear redes neuronales y morfologías.
- Algunos ejemplos más recientes se encuentran en los trabajos de Jelisavcic et al. [33] y Lan et al. [34], quienes utilizan una codificación directa de esta índole para representar las morfologías de sus robots modulares como árboles, donde la raíz representa el componente central desde donde se comienza a construir el robot, y desde ahí al hacer un recorrido primero en profundidad, se pueden construir cada una de las extremidades.

  Se muestra un ejemplo de este sistema en la Figura 3.7, donde se puede apreciar los distintos tipos de nodo de la morfología: C - Componente central (contiene al controlador en sí), H - Articulación con actuador, B - Ladrillo, o componente pasivo.

3.1.2.3. NEAT

El algoritmo de NEAT (Neuroevolution of Augmenting Topologies) [35] tiene como propósito la evolución tanto de los pesos como la arquitectura de redes neuronales.

Las ideas principales sobre las cuales parte este algoritmo son:

- Complejificación: Comenzar con poblaciones de redes neuronales con arquitecturas simples (i.e. redes de dos capas) e ir agregándoles complejidad por medio de mutaciones, garantizando así que las redes evolucionadas al final mantengan un mínimo de complejidad.

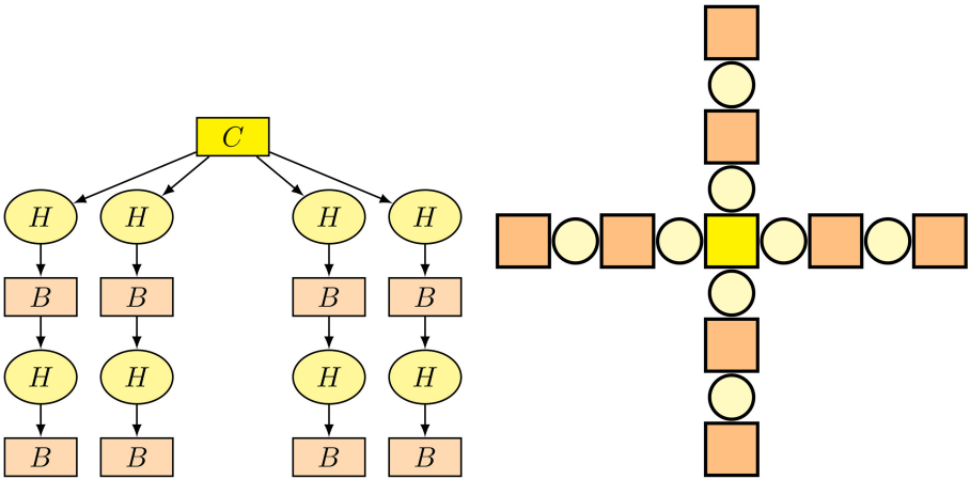


Figura 3.7: Ejemplo esquemático de genotipo y fenotipo en [33].

- Protección de la diversidad: Utiliza una técnica llamada fitness sharing para darle diversidad a la población y permitir que nuevas innovaciones tengan tiempo de evolucionar antes de hacerlas competir contra individuos que han tenido más tiempo de madurar.
- Protección de la información: Utiliza información histórica para efectuar cruzas de manera efectiva y así mantener en un mínimo la pérdida de información.

Estas tres ideas le permiten al algoritmo optimizar tanto la arquitectura como los pesos de una red neuronal, de manera muy efectiva para tareas de aprendizaje por reforzamiento dada una forma adecuada de evaluar su rendimiento.

En este caso la representación de las redes neuronales sigue siendo directa, puesto que el genoma se representa como dos conjuntos de genes (diploide), uno para nodos, y otro para conexiones entre los nodos y la red neuronal resulta de un mapeo uno a uno de estos genes, esto puede ser observado en la Figura 3.8. A grandes rasgos, los operadores genéticos de NEAT funcionan de la siguiente manera:

- Mutación:
  - Se pueden mutar los pesos de las conexiones existentes.
  - Se pueden agregar nuevas conexiones con pesos aleatorios.
  - Se pueden agregar nuevos nodos, introduciéndolos entre dos nodos existentes: La conexión entrante al nodo nuevo recibe un peso de 1, la conexión saliente recibe el peso de la conexión anterior.
- Cruza:

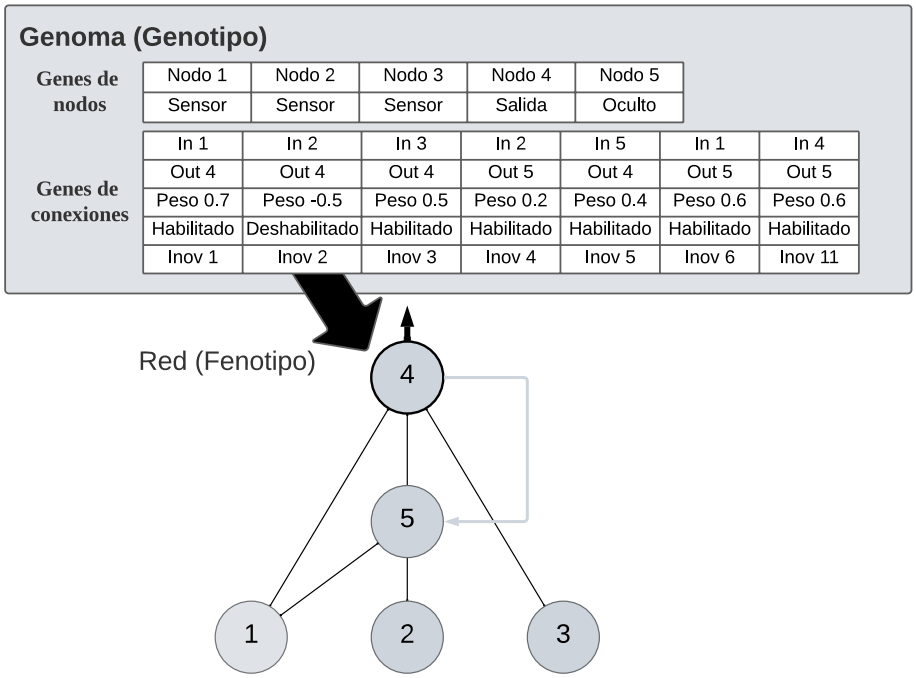


Figura 3.8: Ejemplo de genotipo y fenotipo en NEAT [35].

- El problema de "Competing conventions":
  - Se tiene más de una forma de expresar un mismo fenotipo (una misma red) a través de distintos genotipos. Esto lleva a la pérdida de información (Figura 3.9).
- Solución: Homología:
  - Se inspira en como funciona la alineación genética en la reproducción sexual natural.
  - Dos genes son homólogos si son alelos de la misma característica.
  - Homología basada en marcadores históricos: Dos genes son homólogos si comparten el mismo origen. Este es el mecanismo utilizado en NEAT.
- El mecanismo de cruza de NEAT (Figura 3.10.):
  - Los genes en común de ambos padres se heredan aleatoriamente.
  - Los genes disjuntos y excesos se heredan del padre más apto.
  - Si ambos padres tienen la misma aptitud, los genes disjuntos y excesos se eligen aleatoriamente.
- Especiación

- Se utiliza para proteger la diversidad.
- Es muy probable que aquellas redes neuronales sujetas a mutaciones recientes se hagan más complejas y sean menos aptas, sin embargo, le dan diversidad a la población. Vale la pena tener algún mecanismo que las proteja si no se quiere converger muy rápido a óptimos locales.
- También se tiene una presión de selección para mantener redes más "pequeñas", dado que estas se optimizan más rápido.

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \bar{W}.$$

Pesos: c_1, c_2 y c_3.
Número de genes en exceso: E.
Número de genes disjuntos: D.
Diferencia promedio de pesos entre genes similares: $\bar{W}$.
Número de genes en el genoma más grande: N.

- Cada especie tiene un genoma representativo que viene de la generación previa.
- Si $\delta < \delta_t$, el individuo pertenece a esa especie.
- Se agrega a la primera especie con la que se cumpla lo anterior.
- Si no se cumple que $\delta < \delta_t$ para ninguna especie, se crea una nueva con el individuo en cuestión.

Además de ser el algoritmo base para las variantes más utilizadas de neuroevolución (CPPN-NEAT y Hyper-NEAT) en el contexto de robótica evolutiva, y de haber sido utilizado para evolucionar las GRN artificiales en los trabajos de Joachimczak et al. [7] (tal y como se menciona más adelante en la Sección 3.1.3.1), NEAT también se utilizó en una implementación más moderna de los experimentos de Sims para evolucionar una variante del genotipo de la morfología y control de nuevo en forma de un grafo anidado [36].

3.1.3. Codificaciones generativas

Como ya se comentó al finalizar la Sección 3.1.1, en el caso de este tipo de codificaciones, el genotipo no tiene un mapeo directo con el fenotipo, el genotipo posee una especificación que permite producir regularidad (compresibilidad de la descripción en la estructura [37]) en el fenotipo. Algunos ejemplos de patrones regulares son:

- Repetición (reutilización o auto-similitud) - Presencia de múltiples instancias de la misma subestructura. Por ejemplo: Los distintos tipos de células que componen cada tipo distinto de tejido, desde las células de las distintas partes del cuerpo, hasta las neuronas del cerebro [23].

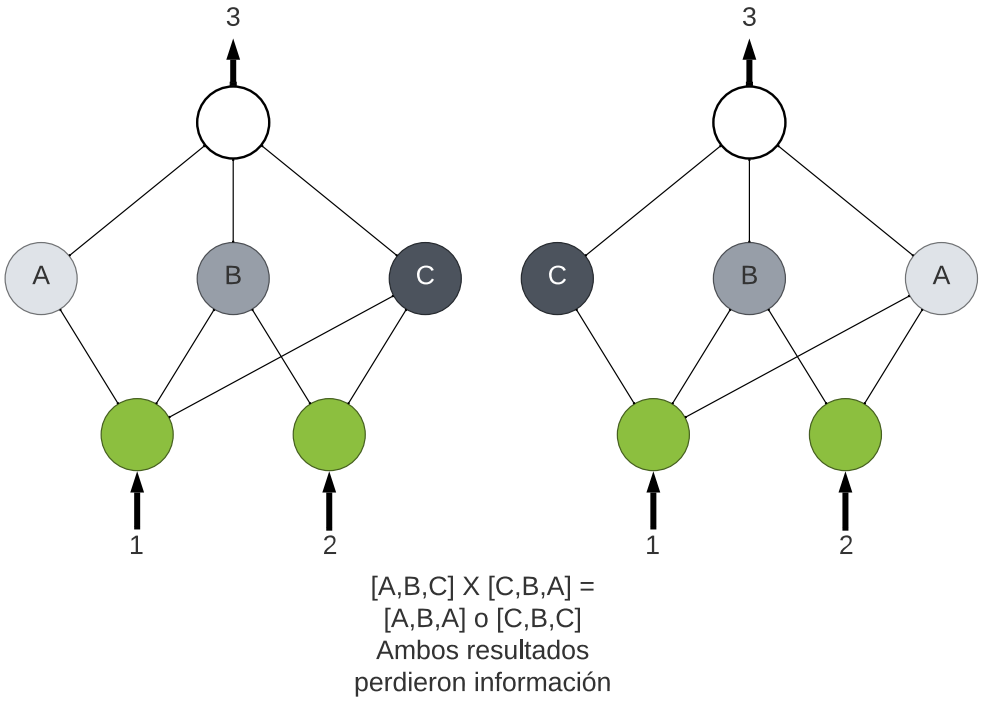


Figura 3.9: Problema de "competing conventions" en NEAT [35].

- Repetición con variación - Pueden darse diversas instancias de una misma subestructura donde cada instancia es ligeramente distinta a las demás. Por ejemplo: Cada vértebra de la espina dorsal es similar en estructura, sin embargo cada una tiene distintas proporciones y formas ligeramente distintas a las demás [23].
- Simetría - La repetición suele darse a través de simetría: quiralidad o efecto espejo. Por ejemplo: Las extremidades en el cuerpo humano [23].

Este tipo de representaciones buscan una abstracción de cómo el genoma de los seres vivos codifica estructuras astronómicamente complejas, e.g. el cerebro humano, que tiene billones de neuronas interconectadas, es codificado por un genoma que es comparativamente mucho más pequeño: 30 mil de genes.

Respecto a los trabajos presentados en la Sección 3.1.1:

- La codificación de Sims [21] permite regularidad en términos de repetición y simetría.
- La codificación de Channon [26] [27] remueve la repetición al simplificar el modelo de Sims a grafos acíclicos.
- La codificación de Hornby y Pollack [28] [29], permite además producir estructuras jerárquicas con regularidad.

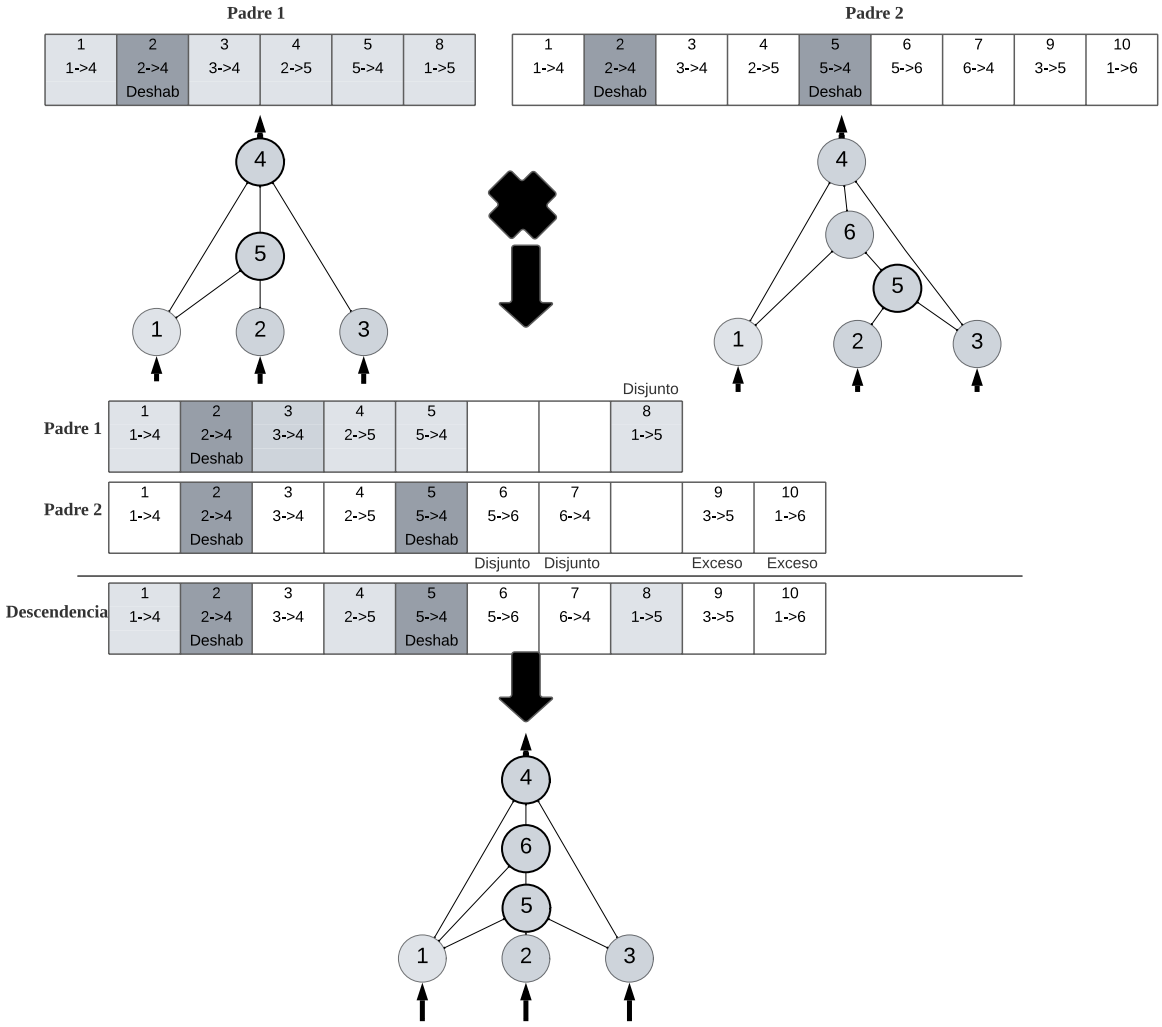


Figura 3.10: Cruza ejemplificada en NEAT [35].

3.1.3.1. Codificaciones de desarrollo artificial

Las representaciones generativas tienen una fuerte inspiración en el proceso de desarrollo basado en morfogénesis que se observa en los sistemas vivos. La idea es "crecer" el fenotipo desde el genotipo a partir de una serie de pasos de ensamble que permiten incrementar gradualmente la complejidad hasta llegar a una forma madura [23]. Dicho proceso nos permite tener una genotipo compacto y alcanzar un fenotipo mucho más complejo, aprovechando simetría y modularidad [38].

El proceso de desarrollo artificial se caracteriza, de manera similar a su homólogo del mundo natural, por un proceso de construcción de una estructura cada vez más compleja a partir de componentes simples [7].

Joachimczak et al. [7] con sus llamados Animats, combinaron un proceso de desarrollo embrionario (en el tiempo de desarrollo) y un proceso de metamorfosis (a lo largo de la vida) para coevolucionar la morfología y control de creaturas virtuales en 2D capaces de desempeñarse exitosamente en varios ambientes, logrando adaptación morfológica y de locomoción dependiendo del entorno (caminado, arrastrado, nado).

En la etapa de desarrollo embrionario, se da un proceso de auto-ensamble morfológico a través de células artificiales que permite una gran variedad de cuerpos, dicho proceso comienza con una sola célula y subsecuentes subdivisiones y muertes (apoptosis). El comportamiento de cada célula se rige por una misma GRN artificial que no es más que una red neuronal feedforward simple (cada célula se controla con una copia de la misma red), en donde las señales de entrada son la posición de la célula en el embrión y la posición de las otras células, logrando la emergencia de estructuras no triviales (una de las características buscadas por el desarrollo artificial). La evolución de la GRN está regida por el algoritmo NEAT.

La simulación física en el desarrollo embrionario carece de gravedad y se toma una fricción viscosa para simular la presencia de un fluido.

Una vez que se termina la etapa de desarrollo embrionario y emerge la morfología final, se pasa a la etapa de locomoción. En esta etapa la morfología se crea a partir de una triangulación de Delaunay, las células que conformaban al embrión se toman como masas puntuales (vértices) interconectadas por resortes (aristas), creando un sistema masa resorte.

El control se logra al modificar la longitud de los resortes de acuerdo a un patrón oscilatorio asociado a cada célula, donde la frecuencia y la fase de dicha oscilación se determinan con las salidas correspondientes de la GRN, al finalizar la etapa de desarrollo embrionario.

3.1.3.2. CPPN-NEAT

Representa una variante del algoritmo NEAT presentado anteriormente, pero aplicado a un tipo especial de red neuronal llamada CPPN (Compositional Pattern Producing Network).

Una red neuronal $CPPN$, es un grafo conexo, acíclico, ponderado. Cada nodo o neurona v corresponde a un perceptrón [25], con una función de activación $a^{(v)} \in A$, cuya evaluación es $\lambda(v)(z^{(v)}) = a^{(v)}(z^{(v)}) = a^{(v)}((\mathbf{w}^{(v)})^T\mathbf{x}^{(v)})$.

Es decir:

$$
\begin{aligned}
&CPPN = (V, E, A, \lambda, w).\\
&\quad \text{Donde:}\\
&\quad V \text{ es el conjunto de nodos o neuronas.}\\
&\quad E \text{ es el conjunto de aristas o conexiones}\\
&\qquad \text{compuesto de pares ordenados de vértices:}\\
&\qquad E \subseteq \{(x,y)|(x,y) \in V^2 \ \&\ x \neq y\}.\\
&\quad A \text{ es el conjunto de funciones de activación}\\
&\qquad \text{propio de las CPPN.}\\
&\quad \lambda \text{ es el mapeo } V \to A \text{ que indica la función}\\
&\qquad \text{de activación } a \text{ para cada nodo } v \in V.\\
&\quad w \text{ es el mapeo } E \to \mathbb{R} \text{ que indica el peso}\\
&\qquad w_{i,j} = w(i,j) \text{ para cada } (i,j) \in E.
\end{aligned}
\tag{3.2}
$$

En cuanto a la evaluación $\lambda(v)(z^{(v)})$ de cada neurona $v \in V$.

El vector $\mathbf{x}^{(v)} \in \mathbb{R}^{n+1}$ corresponde a las n entradas numéricas de v, las cuales a su vez pueden ser la salida de otras neuronas, o variables de entrada a la red neuronal, resultando de las evaluaciones $x_i^{(v)} = \lambda(u_i)(z^{(u_i)})\ \forall(u_i, v) \in E\ \&\ i \in \{1, \cdots, n\}$, más una entrada de bias $x_0 = 1$.

El vector $\mathbf{w}^{(v)} \in \mathbb{R}^{n+1}$ corresponde a los pesos de todas las conexiones de entrada a la neurona $(u_i, v) \in E$. Cada elemento $w_i^{(v)}$ del vector $\mathbf{w}^{(v)}$ se obtiene como: $w_i^{(v)} = w(u_i, v)\ \forall(u_i, v) \in E\ \&\ i \in \{1, \cdots, n\}$, más un peso de bias w_0.

El bias es $b = w_0 \cdot x_0$ y se encuentra implícito en la expresión $z^{(v)} = (\mathbf{w}^{(v)})^T \mathbf{x}^{(v)} = \sum_{i=0}^{n} w_i^{(v)} \cdot x_i^{(v)}$.

Para un conjunto de neuronas de salida $O = \{o_1, \cdots, o_n\}$, donde $O \subset V$, se tiene que la salida $\mathbf{y}$ de la red CPPN, dada una entrada $\mathbf{x} \in \mathbb{R}^m$, es un vector de la forma $\mathbf{y} \in \mathbb{R}^n$, donde $n = ||O||$ y m se refiere al número de variables de entrada. Con lo cual se tiene que la evaluación de una CPPN es:

$$
\mathbf{y} = CPPN(\mathbf{x}) = \begin{bmatrix} a^{(o_1)}(z^{(o_1)}) \\ \vdots \\ a^{(o_j)}(z^{(o_j)}) \\ \vdots \\ a^{(o_n)}(z^{(o_n)}) \end{bmatrix}. \tag{3.3}
$$

Cada evaluación de cada neurona de salida se realiza de manera recursiva para cada entrada, hasta llegar al caso base en el que alguna de las entradas sea una entrada a la red neuronal, o ya se haya evaluado.

Las CPPNs fueron ideadas por Stanley [23], y representan una codificación generativa que es capaz de prescindir de un proceso de desarrollo como el descrito arriba gracias a que abstrae dos partes importantes de éste y las incorpora en la representación misma: interacciones locales y desdoblado temporal, partiendo de la observación de que ambas son necesarias únicamente en un universo restringido por las leyes de la física y no son fundamentales en un proceso computacional.

- Prescindiendo del desdoblado temporal: Un fenotipo se puede describir como una función $F : \mathbb{R}^n \rightarrow \mathbb{R}$, donde n son las dimensiones del mundo físico [23], es decir, podemos describirlo tan sólo como una función en términos de coordenadas espaciales.

  Dado que una CPPN al final del día es una red neuronal, y como tal, sabemos que estas son aproximadores universales de funciones [39], cualquier morfología puede ser vista como una distribución de partículas en el espacio y obtenida como una evaluación instantánea de una función espacial (es decir, un sistema estacionario). Por lo tanto, codificar un proceso dependiente del tiempo podría ser innecesario, dado que siempre se puede sustituir (al menos teóricamente) por una descripción funcional [23]. Este concepto se ejemplifica en la Figura 3.11.

- Prescindiendo de las interacciones locales: Las cadenas causales derivadas de incorporar el tiempo dan lugar a que para formar el fenotipo se tengan que dar a su vez interacciones locales. Esto es similar al concepto de una línea de ensamble, donde existe una secuencia necesaria para la creación de un producto final. Si hacemos la analogía al armado de un juguete antropomórfico, tal vez primero tendríamos que partir de un torso, situar su centro y posteriormente hacer una transformación para situarnos en alguna de las articulaciones, ya sea de los brazos o piernas, para poder insertar las extremidades, donde cada una de las partes, posee un marco de referencia local que se debe tomar en cuenta. En este caso, el marco de referencia local del torso se toma como base para el armado de todo el cuerpo, y a partir de ahí se pueden definir simetrías útiles, en particular: las piernas y los brazos son simétricos respecto al eje y.

  Volviendo al trabajo de Joachimczak et al. [7], justamente se tenía un periodo de desarrollo embrionario en donde se tenía una GRN que describía las interacciones locales que se daban a lo largo del tiempo para que al final emergiera una morfología.

  Estas mismas interacciones locales que permiten patrones regulares pueden ser descritas a partir de una composición de funciones de activación particulares, específicamente se deben elegir funciones de activación simétricas (tales que permitan simetría) y periódicas (tales que permitan repetición). En la Figura 3.12 se muestra un ejemplo.

Otra ventaja asociada con una representación del fenotipo basada en CPPNs es su capacidad de trabajar con resoluciones arbitrarias. Esto se debe a que estamos traba-

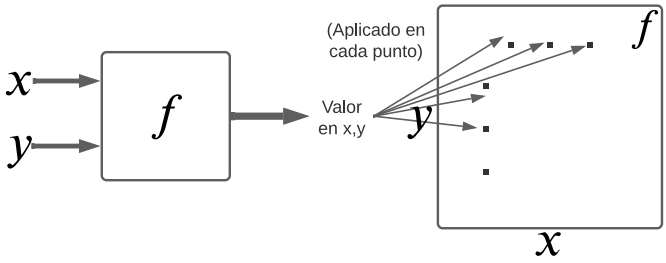


Figura 3.11: Representando un fenotipo como una función [23].

jando con funciones en el espacio y un muestreo de éste. Al igual que en cualquier otra función continua discretizada, a mayor granularidad de la discretización, es mayor la cantidad de puntos se toman en cuenta, aproximando mejor la función, al precio de un mayor tiempo de cómputo que escala de manera lineal con el número de muestras.

3.1.3.3. Hyper-NEAT

Partiendo de nuevo de la idea del uso de codificaciones generativas para representar estructuras arbitrariamente complejas: En el cerebro humano 100 billones de conexiones se representan por 30,000 genes. Inspirándose por esta idea, se encuentra HyperNEAT [40], el cual se trata de un método de neuroevolución que aprovecha las capacidades de representación de patrones regulares de las CPPN para obtener estructuras de redes neuronales mucho más complejas, al representar patrones de conectividad como funciones en el espacio. En resumen: se representan redes neuronales a través de redes neuronales.

Representar patrones de conectividad de redes neuronales mediante funciones espaciales no es una tarea trivial, el problema se encuentra en cómo interpretar la salida de una función de esta índole para poder describir una red neuronal, es decir: ¿Cómo se pueden describir patrones de conectividad por medio de patrones espaciales? [40]. La manera en la que HyperNEAT resuelve dicha cuestión es que en lugar de pasarle a la CPPN la posición espacial de un sólo punto, se le pasan las coordenadas de 2 puntos que definen una conexión, y la salida nos daría el peso de dicha conexión, en lugar de un escalar representando la intensidad en un solo punto. Por ejemplo, una CPPN que represente patrones de conectividad en un espacio de dos dimensiones es una función escalar de variable vectorial en $\mathbb{R}^4$:

$$w = CPPN(x_1, y_1, x_2, y_2).$$

De esta manera, si discretizamos un espacio bidimensional, podemos describir cualquier conexión en dicho grid, incluyendo las conexiones recurrentes de la forma $(x_1, y_1) \hookrightarrow (x_1, y_1)$. Una conexión no será expresada si $w < w_t$, mientras que aquella que si lo esté, será normalizada a un valor máximo de w_{max}. Al espacio dominio dicretizado de la CPPN se le conoce como sustrato. Lo anterior se ejemplifica en la Figura 3.13, donde

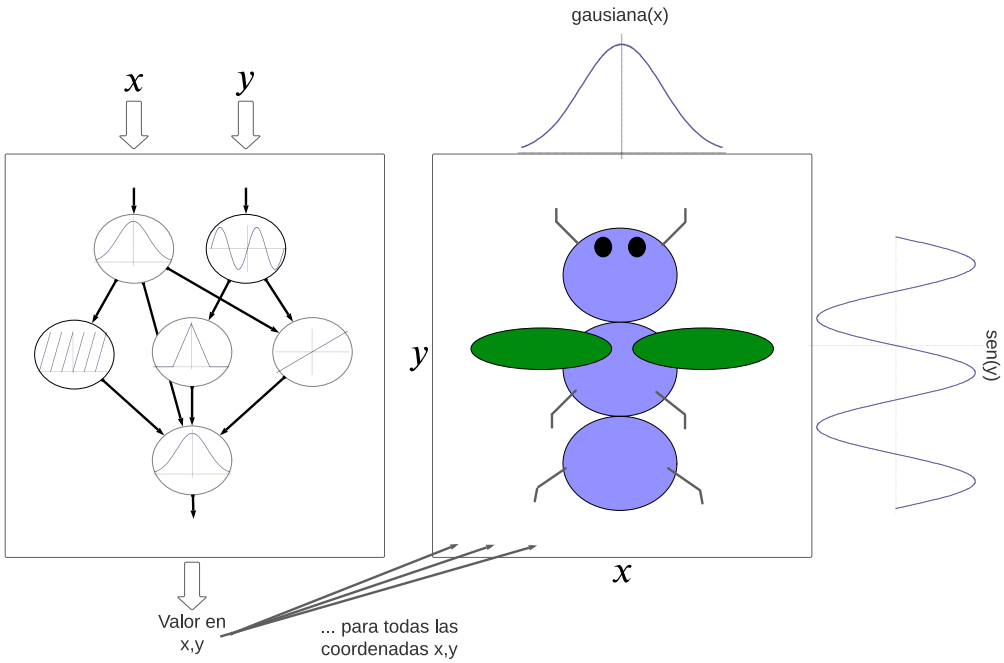


Figura 3.12: Obteniendo regularidad con una composición de funciones [23].

el sustrato está dado por un grid de 5 x 5 con (0, 0) en el centro, asumiendo que los nodos y sus posiciones en dicho sustrato se encuentran dadas.

El ejemplo anterior de sustrato se encontraba dado por un sistema de coordenadas cartesianas, sin embargo, se podrían tener sustratos en distintos sistemas coordenados, de tal forma que la distribución de los pesos en las conexiones exhiban un patrón producto de la geometría del sistema coordenado elegido [40].

Recordando las capacidades de las CPPNs para producir patrones espaciales con regularidad, éstas también pueden exhibir patrones de conexión regulares.

- Simetría: Dado que los patrones de conectividad (en un sustrato bidimensional), se encuentran descritos por dos valores de x y dos de y, se pueden producir patrones simétricos en un eje aplicando una función simétrica (e.g. una Gaussiana) a alguno de los dos valores asociados a un eje. Un ejemplo es la Figura 3.14a.
- Simetría con variación: Similarmente a la simetría simple, la simetría con variación se puede obtener aplicando un a función simétrica, pero esta vez a los dos valores asociados a un eje. Un ejemplo es la Figura 3.14b.
- Repetición: Se puede producir aplicando una función periódica (e.g. coseno) a alguno de los dos valores asociados a un eje, como por ejemplo la Figura 3.14c.
- Repetición con variación: Se produce aplicando una función periódica a los dos

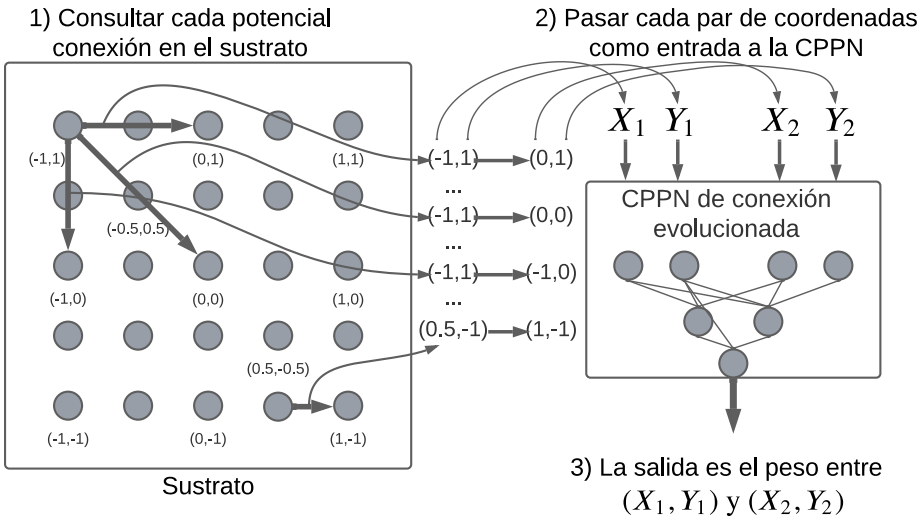
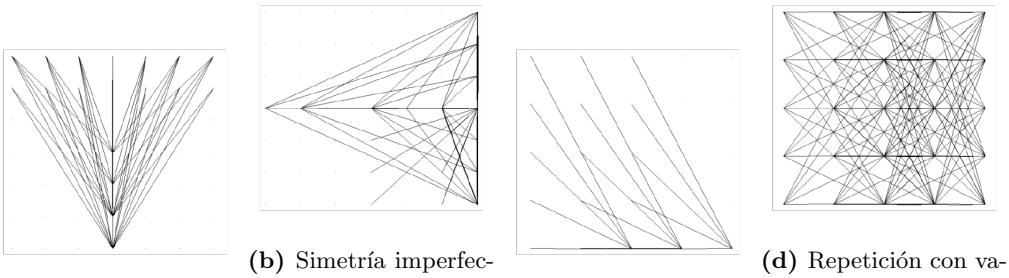


Figura 3.13: Ejemplo de un espacio sustrato y su CPPN de interpretación correspondiente [40].

valores asociados a un eje, tal y como se muestra en la Figura 3.14d.



(a) Simetría. **(b)** Simetría imperfecta. **(c)** Repetición. **(d)** Repetición con variación.

Figura 3.14: Algunos ejemplos de patrones de regularidad posibles en un sustrato bidimensional [40].

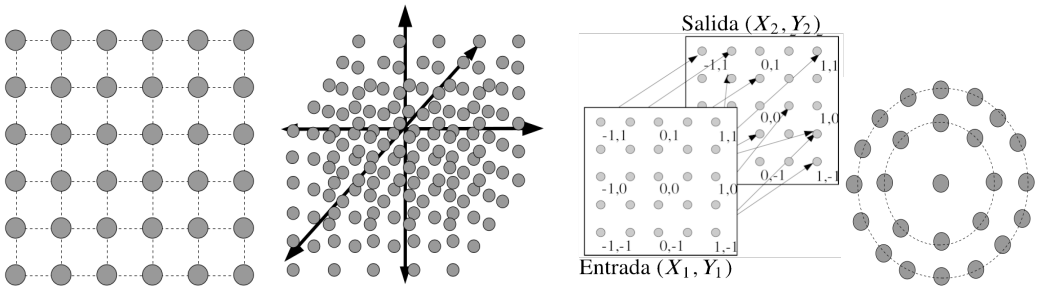
Como ya se mencionó, se pueden elegir distintos sistemas coordenados para el sustrato, de manera que cada configuración de sustrato es más o menos adecuada para un problema en particular. Algunos ejemplos de configuraciones de sustrato se muestran en la Figura 3.15. En la Figura 3.15b se muestra un sustrato para representar patrones de conectividad tridimensionales, a través de un hipercubo cuya CPPN tendría la forma $CPPN(x_1, y_1, z_1, x_2, y_2, z_2)$.

En la Figura 3.15c se muestra un patrón de conectividad denominado como “sandwich”

del espacio de estados. Este patrón tridimensional restringe las conexiones de un plano de entrada a otro de salida, de manera tal que el plano de entrada corresponde a las coordenadas (x_1, y_1) y el plano de salida corresponde a las coordenadas (x_2, y_2). Esto puede ser útil si se quieren representar redes neuronales con un número de capas fijo.

En la Figura 3.15a se muestra un patrón de conectividad de tipo grid, tal como el que se ejemplifica en la Figura 3.13.

En la Figura 3.15d se muestra un patrón de conectividad de tipo circular. Es probable que distintos tipos de configuraciones sean más apropiadas que otras con base en las propiedades geométricas subyacentes del problema [40].



(a) Grid. **(b)** Tridimensional. **(c)** Sandwich. **(d)** Circular.

Figura 3.15: Algunas posibles configuraciones de sustratos [40].

El patrón de conectividad de la Figura 3.15c, es de especial interés, puesto que tiene aplicación para representar el espacio de estados de un controlador. En [41] se utilizó un sustrato de dichas características con éxito (Figura 3.16) en la evolución de controladores de robots cuadrúpedos articulados. En dicho trabajo, se logró evolucionar controladores modulares capaces de producir patrones de caminado coordinados, sin necesidad de efectuar una descomposición manual del problema (es decir, evolucionar cada extremidad por separado), lo cual implica que HyperNEAT fue capaz de "comprender" la regularidad de las cuatro piernas del robot, al utilizar un mismo módulo para cada pierna, pero variando la fase. Para contrastar los resultados de HyperNEAT, se realizaron experimentos evolucionando una red de control con morfología fija y con codificación directa a través de FT-NEAT (Fixed Topology - NEAT). Los experimentos con FT-NEAT sólo lograron locomociones erráticas y poco coordinadas, llevando al robot frecuentemente a caer.

3.1.4. Clasificación basada en la estructura de datos

Ahora que se ha introducido cada uno de los conceptos y algoritmos más relevantes para la comprensión de las codificaciones generativas, se procede a realizar una clasificación de las distintas variantes de este tipo de codificación, con base en la estructura de datos

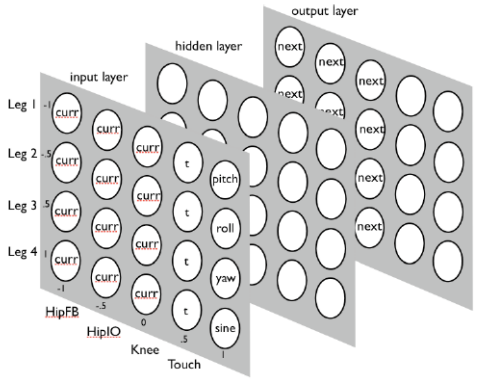


Figura 3.16: Sustrato utilizado para representar las conexiones la red neuronal de control [41]. En el sustrato de entrada, en cada renglón del grid se tienen las entradas de los sensores de cada pierna, estas entradas se llevan en un tiempo t a las neuronas de la capa oculta, que se encuentran especificadas en el sustrato de la capa oculta, y a su vez esta capa oculta lleva a una capa de salida, donde se dan las señales necesarias a incidirse en los actuadores para llevar al sistema a un estado siguiente en el tiempo $t + 1$, dando así lugar a la relación característica en sistemas dinámicos: $\vec{x}^{(t+1)} = G(\vec{x}^{(t)})$.

que se utiliza para representarlas, de manera similar a lo que se realizó al principio de la Sección 3.1.2

3.1.4.1. Basadas en grafos

Al límite de las codificaciones directas y generativas, se encuentran, con la menor capacidad expresiva, este tipo de codificaciones. Las codificaciones de Sims y Channon, anteriormente mencionadas sirven como ejemplo de este tipo de codificación.

3.1.4.2. Basadas en gramáticas

Este tipo de codificación, basada en gramáticas libres de contexto, se encuentra ejemplificada por los sistemas de Hornby y Pollack [28] [29] que fueron mencionados y explicados anteriormente. Un trabajo más reciente que utiliza este tipo de genotipo es el presentado por Matteo et al. [42] donde se utiliza una gramática como forma de representación para la morfología de robots modulares.

3.1.4.3. Basadas en redes neuronales

Las redes neuronales evolucionadas, han demostrado ser, gracias a su gran capacidad de generalización, una manera muy efectiva de representación indirecta. En la literatura,

se han encontrado varios trabajos que evolucionan topologías y pesos de redes neuronales como representación de parámetros de morfología y control de robots basados en voxeles utilizando CPPN-NEAT [31] y programación genética [4] [6]. También se han utilizado de manera efectiva como representación de los pesos de una red neuronal en el control de un cuadrúpedo [40] utilizando Hyper-NEAT. El trabajo de Cheney et al. [31] mencionado en la Sección 3.1.2, tuvo como propósito la comparación de una codificación directa y generativa, encontrando que con las CPPN se podían representar fenotipos mucho más complejos, con patrones regulares, en una representación mucho más compacta que la directa.

3.1.5. Tipos de controladores

En esta sección se presenta una posible clasificación de los tipos de controladores presentes en la literatura de robótica evolutiva. Esta clasificación se centra principalmente en los controladores ya expresados (fenotipo), sin embargo, también se hace mención especial de algunas formas en las que se han representado indirectamente estos distintos tipos de controladores a través de redes neuronales.

3.1.5.1. Lazo abierto vs. Lazo cerrado

En general, todo sistema de control se puede clasificar como de lazo abierto o cerrado. En un sistema de control de lazo abierto, simplemente se incide una señal de entrada, o una perturbación, sobre un sistema dinámico de alguna índole física (mecánico, eléctrico, hidráulico, etc.), esperando cambiar al sistema de estado.

Para el caso de un sistema de lazo cerrado, se tiene una retroalimentación sobre la salida del sistema, la cual se toma en cuenta para ajustar la entrada del sistema, con base en una función de error entre una función de referencia, representando una salida deseada, y la salida real del sistema.

Para el caso de creaturas virtuales, un ejemplo de sistema de lazo cerrado es el utilizado por Sims [21] [22], el cual se caracterizaba por tener sensores a su disposición, tanto para monitorear el entorno, como el estado de los actuadores (ángulo), la señal de estos sensores provoca algún cambio de comportamiento u acción correctiva en el agente, al pasarse a las neuronas de entrada del cerebro (controlador), que a su vez posee una dinámica que incide una señal, o perturbación en el sistema dinámico de actuadores de la creatura. Mientras que un ejemplo de sistema de lazo abierto es el que se utiliza para evolucionar patrones de caminado y morfologías en robots blandos basados en voxeles [31] [4] [6], donde se incide directamente una señal basada en una función trigonométrica en los actuadores, sin tomar en cuenta ningún tipo de señal de retroalimentación que permita influir en la dinámica del controlador, logrando así emular los patrones de caminado cíclicos característicos de los seres vivos.

3.1.5.2. Paramétricos

Se trata de un esquema de control en donde los actuadores se mueven con base en funciones trigonométricas. Cada actuador suele tener asociado un conjunto de parámetros de la misma función trigonométrica. Nordmoen et al. [9] utilizaron este esquema con sus robots modulares para determinar el ángulo θ_i de una articulación i en el tiempo t a través de los parámetros de amplitud (α_i), frecuencia (ω_i), fase (ϕ_i) y compensación de amplitud (o_i), presentes en la función $\theta_i(t) = \alpha_i \sin(\omega_i t + \phi_i) + o_i$.

En el trabajo de Joachimczak et al. [7] mencionado en la Sección 3.1.3.1, utilizaron precisamente un esquema de esta naturaleza para crear patrones de caminado en sus animats y su sistema masa-resorte-amortiguador, donde los osciladores para la longitud del resorte estaban parametrizados por la frecuencia y la fase correspondientes a la salida de la red neuronal de control (GRN) de cada célula al final de la etapa de desarrollo.

En muchos de los trabajos de Cheney et al. y Kriegman et al. donde se evolucionan robots basados en voxeles se presenta algún esquema de este tipo. En [31] la frecuencia de todos los voxeles actuadores es fija, y el movimiento ondulatorio característico de los patrones de caminado, se debe a que hay dos tipos de voxel actuador, uno con una fase de π y otro con una de $-\pi$, dando pie a que el control sea completamente dependiente de la evolución de una red neuronal asociada a la morfología. En [4] se utiliza un control fuertemente inspirado en el de Joachimczak et al. [7], cada voxel tiene asociada una frecuencia y fase, obtenidas a partir de la salida de una CPPN. La principal diferencia en este caso, es que todas las salidas de frecuencia se promedian para obtener una frecuencia central; cada voxel actuador se expande y contrae a esta frecuencia, pero a una fase distinta.

3.1.5.3. Redes neuronales

El uso de redes neuronales como sistemas de control ha sido prevalente en la literatura de robótica evolutiva por su capacidad para el modelado de sistemas dinámicos al ser aproximadores universales de funciones [39].

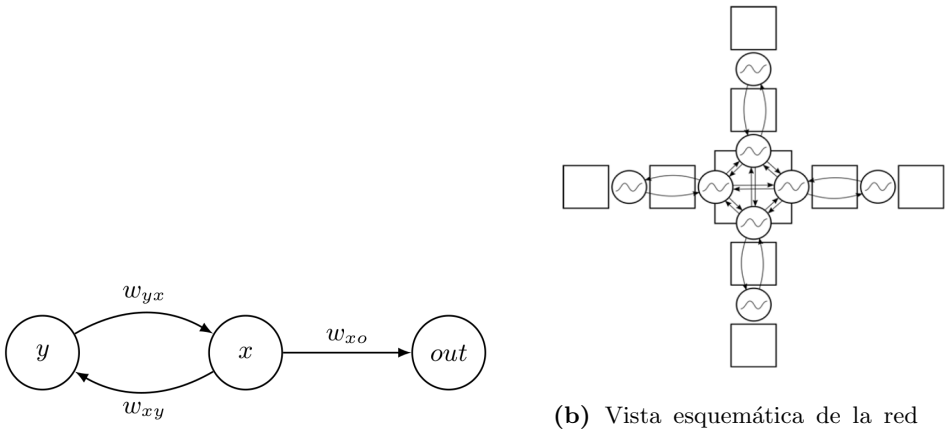
- Redes neuronales feedforward: El enfoque inicial del área fue tan solo en optimizar el control de robots con morfologías fijas. Dentro de estos trabajos se pueden distinguir aquellos que se han enfocado en la evolución de controladores para robots móviles con morfologías simples basadas en movimiento con ruedas. Se destaca el trabajo de Lund y Miglino [43], donde se evolucionaron los pesos de un controlador simple basado en una red neuronal de una sola capa para las tareas de locomoción y evasión de obstáculos a través de un algoritmo genético. También se distinguen aquellos que se han enfocado en la evolución de controladores para la tarea de caminar, en robots articulados, donde los controladores también suelen ser redes neuronales. En [41] el controlador utilizado es la red neuronal que se representa en el espacio sustrato.

- Redes neuronales recurrentes: También se ha dado el uso de redes neuronales recurrentes para tener cierta noción de mantenimiento de estados y memoria. Algunos trabajos representativos de este esfuerzo han evolucionado las topologías y pesos de este tipo de redes [44] (previo a NEAT), o tan solo los pesos [45] para robots articulados, o bien pesos y topología, con CPPN-NEAT, utilizando una variante recurrente de CPPNs bautizada CTRNN [46].
- Redes de perceptrones: Este tipo de sistema de control se encuentra ejemplificado por las creaturas de Sims, y fue explicado con anterioridad en la Sección 3.1.1.
- Central Pattern Generators (CPGs): Se trata de redes neuronales capaces de producir salidas con patrones oscilatorios o rítmicos sin necesidad de tener una entrada o un feedback de esta naturaleza [34]. Los módulos principales de estas redes pueden ser osciladores diferenciales [34] [33] [47], o de fase [42]. Para el primero, cada oscilador está compuesto de dos neuronas conectadas recursivamente, tal y como se muestra en la Figura 3.17a. Con base en dicho diagrama, el comportamiento de estos osciladores se encuentra caracterizado a partir de la siguiente ecuación diferencial:

$$\begin{aligned} \dot{x} &= w_{yx}y + b_x. \\ \dot{y} &= w_{xy}x + b_y. \end{aligned} \tag{3.4}$$

  Donde la activación de ambas neuronas x e y será periódica si w_{yx} y w_{xy} tienen signos distintos. Por su parte, la función de activación de la neurona de salida *out*, se encuentra dada por una función de activación lineal:

$$f(out) = (w_{xo} \cdot x + b) \cdot g. \tag{3.5}$$



(a) Un oscilador diferencial cuya salida es utilizada en el controlador CPG.

(b) Vista esquemática de la red CPG generada para una morfología específica.

Figura 3.17: Representación esquemática del sistema de control utilizado en [34]. En 3.17a se muestra el oscilador diferencial, en 3.17b se muestra una morfología, con el control embebido en ella. Los bloques rectangulares representan módulos pasivos, mientras que los bloques circulares representan articulaciones activas, cada una con un oscilador diferencial asociado.

3.1.5.4. Redes neuronales como representación de sistemas de control

Como ya se mencionó en la Sección 3.1.3 las redes neuronales se pueden utilizar como una forma de codificar al fenotipo, y se han utilizado como genotipo para cada una de las formas de control mencionadas anteriormente.

- Redes neuronales como representaciones de parámetros de redes neuronales (HyperNEAT): Un ejemplo es el trabajo que se mencionó anteriormente de Clune et al. [41], para evolucionar patrones de caminado tanto con FT-NEAT y HyperNEAT. Otro ejemplo más reciente es el trabajo de Jelisavcic et al. [33], donde HyperNEAT se utiliza como un mecanismo de aprendizaje en el tiempo de vida de los robots, al ajustar los pesos de las CPG, mientras que CPPN-NEAT se utiliza para evolucionar las morfologías y los controladores embebidos en estas.
- Redes neuronales representando parámetros de controladores: Se han utilizado redes neuronales CPPN para representar directamente los parámetros de los controladores de robots suaves basados en voxeles. En estos casos, los controladores están embebidos en la morfologías. El sistema dinámico a controlar, es una látice que forma un sistema masa-resorte, donde los vértices son masas y las aristas son resortes actuados por los patrones oscilatorios, de manera similar al trabajo de [7] solo que al utilizar redes CPPN, no se tiene un periodo de desarrollo. Algunos

ejemplos son: [4] [6] [8].

3.1.6. Tipos de morfologías

En esta sección se presentan los tipos de morfologías más prevalentes en la literatura de coevolución de morfología y control. Se pudieron identificar dos tipos principales en la literatura, con base en la rigidez del material y los principios mecánicos que rigen su movimiento.

3.1.6.1. Cuerpos Rígidos

Primero se tienen los tipos de morfología basadas en componentes con un rigidez alta, como algunas aleaciones de metal, fibras de carbono, etc, enlazados a través de articulaciones, formando así, un sistema mecánico de cuerpos rígidos, que se modela matemáticamente a través de cadenas cinemáticas. El movimiento de cada cuerpo rígido está restringido por sus conexiones a otros cuerpos rígidos. Es el tipo de morfología que se suele pensar cuando se piensa en robótica, desde perros robot, hasta brazos industriales ensambladores de autos, pasando por obras de ciencia ficción. Dentro de esta clasificación de morfología, nos encontramos los siguientes tipos:

- Clásicas. Se trata de morfologías compuestas de cuerpos rígidos, de dimensiones variables, y con extremidades articuladas equipadas con actuadores que les permiten moverse. Pueden permitir muchas configuraciones, como en el caso de Sims, o pueden ser morfologías mucho más restringidas, tal vez con un número ya predeterminado de extremidades y articulaciones, donde lo único que se varía son algunos parámetros dimensionales, como el que se mencionó de Nygaard et al. [10] o el de Howard et al. [48].
- Esferas. Auberbach y Bongard utilizaron morfologías de este tipo en sus trabajos del 2010 [5] y 2011 [46], haciéndolas "crecer" con un algoritmo un tanto reminiscente a búsqueda en grafos. En el trabajo del 2010, se encargaron de explorar únicamente las propiedades dinámicas de las estructuras evolucionadas, sin ningún tipo de control, haciéndolas caer desde un plano inclinado, mientras que en el del 2011, introdujeron la coevolución de controladores basados en redes recurrentes junto con estas morfologías.
- Modular. En la robótica modular se tiene un número determinado de tipos de módulo, ya sean pasivos, móviles, con sensores, de control, etc. y restricciones para conectarlos, la evolución de las morfologías en estos robots consiste en encontrar esquemas de ensamblado que permitan al agente realizar una tarea. Se puede ver como una especie de ensamblado automatizado de modelos de Lego, con la idea también de que sean resistentes a daños y reconfigurables según el ambiente en el que se desempeñen o la tarea a realizar. Esto último los hace los candidatos más atractivos para pasar de ambientes simulados a ser fabricados físicamente, o que

inclusive puedan ser evolucionados directamente en el mundo físico. Los trabajos presentados en [34] [33] [47] y [9] presentan este tipo de morfologías.

3.1.6.2. Blandas

En contraste con las morfologías basadas en cuerpos rígidos, este tipo de morfología se encuentra compuesta de materiales con baja rigidez, y se suelen modelar como sistemas mecánicos masa-resorte-amortiguador. El tipo principal de morfología en esta categoría es:

- Voxeles. Un voxel es el análogo tridimensional de un píxel, y así como una imagen se encuentra delimitada por una resolución en términos de largo y ancho en número de píxeles, una creatura basada en voxeles, se encuentra delimitada por un volumen de cierto número de voxeles, llamado espacio de diseño.

  Los robots basados en voxeles, no son más que una combinación de voxeles ya sea presentes o ausentes en este espacio de diseño, donde cada uno de los voxeles presentes, puede tener algún material de un conjunto de materiales, principalmente divididos en activos (actuadores o músculos) y pasivos (tejido blando o hueso).

  Muchos trabajos han utilizado este tipo de robots [31] [49] [4] [6] [8], porque la misma maleabilidad que tienen al ser blandos, los hace más versátiles y más fáciles de evolucionar. Además, al no estar restringidos por articulaciones y rangos de movimiento, tienen muchos más grados de libertad que sus contrapartes articulados. También parece que hay una mayor cantidad de configuraciones válidas en relación otros tipos, como rígidos o modulares.

  Otras características prometedoras, como adaptabilidad a diversos ambientes, capacidad de desempeñarse en ambientes reducidos [50] y que sus implementaciones en el mundo real representan un menor riesgo para los seres humanos que sus contrapartes sólidos [51], los hace también candidatos atractivos para pasar de únicamente simulación a fabricación.

3.1.7. Tareas

Sin duda, la tarea más común en la literatura de coevolución de morfología y control es la de aprendizaje de patrones de caminado en línea recta, además de ser comparativamente la más simple y reducida. Otras tareas que se pueden encontrar en la literatura son: aprendizaje de patrones de caminado para poder seguir objetivos [34] [22] [52], caminar con obstáculos [53], desempeñarse en ambientes reducidos [50] o competir por apoderarse de un objeto [22] [26].

Además, se han explorado escenarios donde se evolucionan robots en varias tareas, como:

- En [54] y [55], se plantea la tarea de patrones de caminado para poder seguir objetivos como dos tareas separadas: caminar en línea recta y rotar.

  En [54], se buscó maximizar la velocidad lineal y angular bajo un planteamiento multiobjetivo con NSGA-II. Se identificó a los generalistas como aquellos robots que se acercaban a los máximos alcanzados en ambos objetivos (punto de Nadir) y especialistas aquellos con valores máximos en alguno de los dos objetivos. Además, se exploraron las propiedades de las morfologías respecto a especialistas y generalistas.

  En [55], se plantea una aproximación en dos fases evolutivas. En la primera fase, se evoluciona morfología y control teniendo en mente un planteamiento multiobjetivo con NSGA-II. De dicha forma, se obtienen morfologías, junto con sus CPG embebidos como controladores, que se desempeñan adecuadamente en ambas tareas. En la segunda fase se evolucionan únicamente los pesos de los controladores, de manera independiente para cada tarea. Una vez que se tienen los controladores especializados en cada tarea y adaptados a la morfología, se utiliza una condición simple, dependiente del ángulo al que se encuentre el objetivo, para elegir entre uno o el otro para rotar o caminar en linea recta.

- En [56], se propone evolucionar poblaciones de robots por separado en dos tareas distintas (saltar escalones y caminar cubierto por una carga), para posteriormente combinarlas y lograr obtener robots que se puedan desempeñar en ambas tareas. En este trabajo, los robots se encuentran codificados como redes neuronales. Al finalizar las ejecuciones independientes dedicadas a optimizar a los robots en ambas tareas, se elige a los mejores individuos en cada tarea, de tal forma que para cada par de individuos se combinan sus redes neuronales en una sola, la cual contiene todos los nodos y conexiones de ambas. Posteriormente, se utiliza una versión multiobjetivo de NEAT para evolucionar la población resultante.

En [57], se crea un framework general para efectuar experimentos de optimización de morfologías y controladores de robots suaves en 2D. En dicho framework, se pueden incorporar distintos algoritmos de optimización (no solo evolutivos, sino también aprendizaje por refuerzo), distintas tareas, y distintos ambientes. Las tareas las dividen en dos grandes categorías: tareas de caminado y manipulación de objetos. Las tareas de caminado, tienen que ver con desplazarse de un punto a otro, y varían por el ambiente en el que se desempeñan y grado de dificultad, ya sea caminar en un piso plano, subir escaleras, saltar plataformas, o inclusive escalar muros, etc. La manipulación de objetos a veces puede tratarse de recoger un objeto y llevarlo de un punto a otro, lanzarlo, o atraparlo en el aire, de nuevo, la dificultad es variable entre cada tarea.

3.1.8. Ambientes

Otro aspecto sumamente importante en la coevolución de morfología y control, que además muchas veces está íntimamente relacionado con la tarea a realizar, es el del

ambiente donde se va a desempeñar el robot. La tarea a realizar muchas veces está condicionada al ambiente en el que se encuentra el robot, y la tarea a su vez condiciona a la morfología y control resultantes.

- En [21], el cambio de ambiente de terrestre a acuático provoca que emerjan creaturas con adaptaciones morfológicas y controles que les permiten nadar, pese a tratarse de la misma función de aptitud basada en maximizar la distancia recorrida, por lo cual se podría decir que el cambio en el ambiente implica un cambio en la tarea a realizar, específicamente de caminar a nadar.
- En [53], se explora cuantitativamente el impacto que tiene someter a los robots a distintas pistas de obstáculos de diversos grados de dificultad, en las morfologías y en la convergencia prematura en términos de la aptitud y diversidad. De nuevo, se tiene un impacto directo en la tarea, debido exclusivamente al ambiente, en este caso la tarea es la de caminar y evadir obstáculos de manera efectiva.
- En [50], de nuevo se evalúa a los robots por la distancia que logran recorrer, encontrándose inmersos en una estructura con espacio extremadamente reducido, provocando que la tarea en cuestión sea la de moverse en un espacio reducido. Los robots evolucionados en este trabajo aprendieron a arrastrarse para poder escapar.

Es importante tener en cuenta que el ambiente en donde se puedan evaluar los robots, las tareas que puedan desempeñar y las morfologías y controladores que puedan presentar, dependen completamente del simulador y motor de física que se esté utilizando. En la literatura, hay desde aquellos trabajos que implementan desde cero su propio simulador [7], hasta aquellos que utilizan simuladores conocidos, como OpenAI Gym [53], pasando por aquellos desarrollados por grupos de investigación, como Voxelyze [58], VoxCraft [59] o Evolution Gym [57], cuyo propósito es habilitar a la comunidad para que pueda efectuar experimentos de robótica evolutiva.

3.2. El problema de la convergencia prematura

La convergencia de los algoritmos evolutivos suele darse cuando la población llega a una sola solución o a un conjunto de soluciones muy similares. Lo anterior se ve reflejado como una pérdida de diversidad de las soluciones y provoca que la población deje de progresar. La pérdida de diversidad suele darse cuando los individuos de la población han convergido alrededor de óptimos locales o globales. Cuando la corrida evolutiva ha convergido a óptimos locales se dice que se ha quedado atrapada en óptimos locales, o bien, que se ha tenido una convergencia prematura. Este problema se puede deber a que en el compromiso entre exploración y explotación haya un sesgo hacía explotación. Tal sesgo adquiere mayor o menor relevancia dependiendo del problema, de manera tal que en problemas con múltiples óptimos locales, o con funciones de fitness muy accidentadas, tales como los que se presentan en la coevolución de morfología y control,

es de vital importancia tener estrategias que permitan mantener buena exploración a lo largo de la corrida evolutiva.

En particular, cuando se intenta coevolucionar la morfología y control en creaturas y robots virtuales el problema prevalente que se ha observado es que la evolución de la morfología se estanca muy pronto en un subconjunto muy similar. Lo anterior quiere decir que se pierde el potencial beneficio de coevolucionar ambos aspectos y la optimización sucede únicamente en los controladores. Se ha encontrado que esta pérdida de diversidad en la morfología es causada por el hecho de que los cambios morfológicos de una generación a otra causan grandes disrupciones en los controladores optimizados y esto a su vez causa que los individuos con cambios morfológicos recientes sean descartados por la presión de selección orientada a maximizar la aptitud en la tarea. La teoría de la cognición corporizada que viene de la psicología provee una posible explicación para este fenómeno: La inteligencia no puede puede ser separada del cuerpo, sino que resulta de las interacciones complejas entre el cuerpo y la mente [4].

3.3. ¿Cómo se ha intentado resolver este problema?

Algunas direcciones promisorias que han sido exploradas previamente han sido la imposición de mecanismos de protección de la diversidad morfológica, por ejemplo: reducir temporalmente la presión de selección con recientes cambios morfológicos, utilizar algoritmos de QD (Quality-Diversity) tales como MAP-Elites y definir un conjunto de características morfológicas deseadas en las cuales se desee descubrir calidad dentro de la diversidad misma, o evolucionar control y morfología en conjunto en una primera fase y en una segunda únicamente el controlador para dejar que se adapte a los cambios recientes en la morfología.

En este contexto, vale la pena mantener en mente el llamado teorema del "No free lunch" aplicado a problemas de optimización y búsqueda [60]. Dicho teorema, nos dice que para ciertos tipos de problemas, no existen algoritmos absolutamente superiores a otros en todos ellos (el desempeño promedio de cada algoritmo, en todos los problemas es igual), únicamente hay algoritmos mejores y peores si se considera un problema o un conjunto de problemas con ciertas propiedades específicas. Es decir, si un algoritmo A presenta mejores resultados que otro B en un problema C, se debe a propiedades del problema C y de su espacio de búsqueda, las cuales favorecen a A y su funcionamiento. Por otro lado, existirá otro problema D (no necesariamente "realista" o "práctico"), donde B supere a A, únicamente debido a las propiedades del problema D.

Así mismo, se ha encontrado que el teorema del "No free lunch" no aplica en coevolución de especies [61], donde no existe realmente una función de aptitud que sea evaluada, sino que más bien solo se tiene la noción de ganadores de encuentros y campeones de torneos. Algunos ejemplos de coevolución de morfología y control que utilizan algoritmos coevolutivos a nivel especie son [22] y [26].

La sospecha es que el mismo teorema del "No free lunch" para problemas de optimización y búsqueda, siga aplicando para coevolución de morfología y control, al menos conforme al tratamiento que se le está dando en esta tesis, dado que mantiene la noción de función de aptitud.

3.3.1. Técnicas de preservación de la diversidad

Tal y como se mencionó en la Sección 3.2, es de vital importancia tener estrategias que permitan mantener buena exploración a lo largo de la corrida evolutiva, lo cual se traduce en mantener diversidad durante la búsqueda. A continuación se presentan algunas técnicas de preservación de diversidad representativas del estado del arte en cómputo evolutivo.

3.3.2. Especiación

La población se subdivide en especies con base en la similitud entre individuos, de tal forma que:

- Cada especie tiene un genoma representativo que viene de la generación previa.
- Se tiene una métrica de distancia entre soluciones $\delta(i, j)$.
- Se decide si un individuo pertenece a cierta especie con base en un umbral o radio de nicho σ_{sh}. El individuo i pertenece al nicho o especie representada por el genoma del individuo j si $\delta(i, j) < \sigma_{sh}$. Donde σ_{sh} es un parámetro definido por el usuario.
- El genoma de un individuo se coloca en el primer nicho donde se cumpla el punto anterior.
- Si un genoma no pertenece a ningún nicho se crea una nueva especie representada por dicho genoma.

La especiación se puede llevar a cabo en combinación con fitness sharing y restricciones a la cruza, tal y como sucede en el algoritmo NEAT explicado anteriormente, sin embargo cabe recalcar que no es necesario hacerlo y cada una se puede utilizar como una técnica aparte.

3.3.3. Fitness Sharing

La aptitud de un individuo se escala de manera inversamente proporcional a la similitud que éste tenga con respecto a los otros individuos de la población, de manera que la función de aptitud se modifica perjudicando a aquellos individuos que sean similares

[62]. Tomando en cuenta lo presentado en [62], la similitud está dada por:

$$\phi(\delta(i,j)) = \begin{cases} 1 - \frac{\delta(i,j)}{\sigma_{sh}} & \text{si } \delta(i,j) < \sigma_{sh}. \\ 0 & \text{de lo contrario.} \end{cases} \tag{3.6}$$

Con lo cual, la aptitud de un individuo i estará dada por:

$$f_i' = \frac{f_i}{c_i}. \tag{3.7}$$

$$c_i = \sum_{j=1}^{n} \phi(\delta(i,j)), \tag{3.8}$$

donde c_i se conoce como conteo de nichos.

Integrando especiación, se puede ver la función de similitud ϕ como diferente de cero, si el individuo i pertenece al mismo nicho o especie al que pertenece j, e i a su vez puede pertenecer a varios nichos (aunque en especiación se coloca al individuo en el primer nicho en el que se cumpla que $\delta(i,j) < \sigma_{sh}$).

Tal y como se menciona en [62], dividir entre c_i, se puede ver como una agregación de dos objetivos: fitness y diversidad dada por el conteo de nichos.

3.3.4. Restricción a la cruza

Se puede restringir el operador de cruza para que solo individuos dentro de una misma especie se puedan cruzar, o bien, para poder cruzar únicamente individuos de distintas especies.

3.3.5. Diversidad como un objetivo

Tomando en cuenta el marco de trabajo multiobjetivo, es posible agregar alguna métrica que mida diversidad, como un objetivo extra a ser maximizado. Dicha diversidad se puede aplicar tanto en el espacio de las variables como en el de los objetivos, y suele depender del problema.

En la literatura se ha encontrado que la distancia media entre el individuo y el resto de la población lleva a los mejores resultados, por lo tanto para efectos de esta tesis la diversidad de un individuo $x \in \Omega$ respecto a la población en la generación n-ésima $P_n \subset \Omega$ se define como:

$$\begin{gathered} D(x) = \frac{1}{|P_n|} \sum_{y \in P_n} d(x,y), \\ \text{donde } D : \Omega \to \mathbb{R}^m \\ \text{y } d : \Omega \to \mathbb{R}^m \end{gathered} \tag{3.9}$$

Ahora bien, en el contexto de coevolución de morfología y control de creaturas virtuales la distancia entre individuos se puede definir de manera muy variada y depende de la representación y/o subsecuente expresión fenotípica que estos tengan. Como bien se menciona en [62], si se elige representar el genotipo como una red neuronal, calcular la distancia entre dos genotipos es un problema sumamente complejo, dado que el cálculo de la distancia entre grafos es un problema NP-Completo por si solo. Sin embargo, se puede reducir el problema al elegir un vector $\mathbf{v}^{(i)}$ que pueda expresar características deseables de la red, teniendo en cuenta una noción de distancia conveniente para la resolución del problema.

En [63], se comparan varios algoritmos, entre ellos la integración de diversidad y fitness bajo un paradigma multiobjetivo, en la tarea de evolucionar controladores (redes neuronales) de robots móviles de dos ruedas para llegar a un objetivo en un ambiente con obstáculos. Dicha integración se formula como un problema de optimización multiobjetivo:

$$\begin{aligned} &\max_{x\in\mathcal{Q}}(\mathbf{F}(x)), \\ &\text{donde } \mathbf{F}(x) = \begin{bmatrix} f(x) \\ D(x) = \frac{1}{|P_n|}\sum_{y\in P_n} d_b(x,y) \end{bmatrix} \\ &\text{y } \mathbf{F} : \mathcal{Q} \to \mathbb{R}^2. \end{aligned} \tag{3.10}$$

definiendo la distancia entre dos individuos en términos de lo que llaman el vector de comportamiento (d_b).

El vector de comportamiento $\mathbf{b}$ de un individuo x se define como:

$$\begin{aligned} \mathbf{b}(x) &= \mathbf{p}_r(T), \\ \mathbf{b} &: \mathcal{Q} \to \mathbb{R}^2. \end{aligned} \tag{3.11}$$

donde $\mathbf{p}_r(T)$ es la posición final del robot en una simulación de duración T.

La distancia entre el individuo $x \in \mathcal{Q}$ y algún otro $y \in \mathcal{Q}$ se define como la distancia euclidiana entre sus vectores de comportamiento, i.e.

$$\begin{aligned} d_b(x,y) &= ||\mathbf{b}(x) - \mathbf{b}(y)||, \\ &\text{donde } d : \mathcal{Q} \to \mathbb{R}^{(+)}. \end{aligned} \tag{3.12}$$

En [9], se evoluciona tanto morfología como control de robots modulares para la tarea de caminar la mayor distancia posible.

La morfología está dada por una codificación directa basada en árboles, donde cada nodo en el árbol es un módulo y cada arista es una conexión entre dos módulos. Hay dos tipos de módulos, no-móviles y articulaciones, cada uno soporta 5 y 3 nodos hijos.

El sistema de control se conforma de todos los controladores individuales de cada articulación, y a su vez cada controlador está dado por 5 parámetros.

En este caso se desea proteger la diversidad morfológica, por lo cual el vector de "comportamiento" de un individuo $x \in \mathcal{Q}$, se encuentra dado por:

$$\begin{aligned} \mathbf{b}(x) &= (m_x, j_x), \\ \mathbf{b} &: \mathcal{Q} \to \mathbb{N}^2. \end{aligned} \tag{3.13}$$

Donde m_x es el número de módulos no móviles y j_x es el número de módulos móviles.

La distancia entre dos individuos $x \in \mathcal{Q}$ e $y \in \mathcal{Q}$ está dada por:

$$d_b(x, y) = 1 - e^{-|\mathbf{b}(x) - \mathbf{b}(y)|}. \tag{3.14}$$

Mientras que la diversidad se define de la misma forma que en la Ecuación 3.9.

Cabe aclarar que la distancia que se define en este caso, es una función $d_b : \mathbb{R}^2 \to [0,1] \times [0,1] \subset \mathbb{R}^2$, por lo cual la formulación multiobjetivo queda como:

$$\begin{gathered} \max_{x \in \mathcal{Q}} (\mathbf{F}(x)), \\ \text{donde } \mathbf{F}(x) = \begin{bmatrix} f(x) \\ D_m(x) \\ D_j(x) \end{bmatrix} \\ \text{y } \mathbf{F} : \mathcal{Q} \to \mathbb{R}^3. \end{gathered}$$

3.3.6. Búsqueda divergente

El ideal del cómputo evolutivo de replicar la complejidad y sofisticación alcanzada en la naturaleza siendo utilizado como optimizador es sin duda alguna, una ambición extremadamente grande. Sin embargo, dicha ambición ha llevado a buscar codificaciones y algoritmos más complejos [64]. Según dicho ideal, únicamente debería bastar utilizar la presión de selección correctamente hacia un objetivo arbitrariamente ambicioso, pero los resultados, sobre todo en robótica evolutiva, nos dicen lo contrario [64]. Pese a que los puristas de los campos de aprendizaje de máquina y optimización han dado fuertes críticas hacia el cómputo evolutivo, denominándolo como un método ad hoc [64], una pregunta resalta como motivación para seguir explorando la alternativa de la evolución artificial: ¿cómo es que la naturaleza ha sido capaz de generar inteligencia, y si miramos hacia nuestros mejores esfuerzos para crear sistemas inteligentes artificiales con optimización tradicional, pese a ser mejores que los humanos en algunas tareas, estos palidecen en comparación, en varias áreas de la cognición humana? [64]

La búsqueda divergente propone que tal vez la idea de la evolución se ha estado aplicando de una manera un tanto incorrecta. La evolución natural es divergente, en vez de convergente. De esta idea es que surge una nueva visión: la evolución como una máquina de diversificación en lugar de optimización [64]. Esta nueva visión plantea que el problema está en que la abstracción de la evolución que se ha estado utilizando ha sido la equivocada todo este tiempo. Si volteamos a ver a la evolución natural es claro que

en ella no se optimiza nada, no se tiene planteado un objetivo claro; a excepción de una noción de recompensa por individuos no solo mejores, sino distintos (nichos evolutivos: algunos organismos que sean suficientemente distintos a sus predecesores pueden gozar de competencia reducida que los hagan más aptos para sobrevivir).

3.3.7. Novelty search

Es una radicalización de la idea anterior, dejando de lado por completo la optimización al perseguir algún objetivo concreto (por ejemplo: el robot más rápido, el más eficiente, el mejor resolviendo laberintos) y en su lugar dirigir la búsqueda únicamente por medio de una noción de novedad como consecuencia de la diversidad [64]. En Novelty Search no se optimiza ninguna función de aptitud basada en un objetivo particular para explorar el espacio de búsqueda, más bien se recompensa una métrica de novedad del individuo que nos permita saber qué tan lejos se encuentra del resto de la población (ya sea en el espacio de búsqueda o de soluciones) [65].

Una métrica de novedad que nos permite medir la dispersión en cualquier punto del espacio en cuestión es:

$$\begin{aligned} N(x) &= \frac{1}{k}\sum_{i=0}^{k} d(x,\mu_i) \\ \text{donde } \mathfrak{U} &= \{x=\mu_0,\mu_1,\cdots,\mu_i,\cdots,\mu_k\}, \\ &= KNN(x, P_j \cup \Lambda, d, k), \\ \mu_i &\in \mathfrak{U}, \\ \mathfrak{U} &\subseteq P_j \cup \Lambda, \\ P_j &\subset \mathfrak{Q}, \\ \Lambda &\subset \mathfrak{Q}, \\ x &\in P_j, \\ k &\in \mathbb{N}, \\ d &: \mathfrak{Q}^2 \to \mathbb{R}, \\ N &: \mathfrak{Q} \to \mathbb{R}. \end{aligned} \tag{3.15}$$

La cual se refiere a la distancia promedio del individuo $x \in P_j$ a sus k-vecinos más cercanos. Los k-vecinos más cercanos de x son elegidos de entre las soluciones en la población en la generación j-ésima P_j y el archivo de novedad Λ. $d(x,\mu_i)$ representa una cierta métrica de distancia dependiente del dominio. A mayor distancia promedio a sus k vecinos más cercanos, el punto x se encontrará en una región más dispersa [65]. Para dirigir el proceso de búsqueda hacia regiones más dispersas, se suele incorporar un archivo de soluciones visitadas en generaciones anteriores (archivo de novedad) Λ, las cuales para poder ser agregadas, deben cumplir con un umbral de novedad mínimo [66].

La subrutina asociada al cálculo de la novedad para un conjunto de individuos en una generación se muestra en el Algoritmo 3. Si integramos el Algoritmo 3 como la función

de evaluación en el algoritmo evolutivo genérico, es decir, como una implementación de la línea 9 en el Algoritmo 1, tendremos como resultado el algoritmo Novelty Search.

Algorithm 3: Evaluación de Novelty Search (Adaptado de [66])

Parametros: novelty_threshold, novelty_floor, min_archive_size, max_archive_size, k
Entrada : artifacts, novelty_archive
Salida : artifacts
Resultados : A cada individuo de la población se le asigna una novedad calculada como se describió anteriormente y se actualiza el archivo de novedad y sus parámetros.

```
1 for individual ∈ artifacts do
2 |   individual.novelty = average_knn_distance(individual,
  |    artifacts∪novelty_archive, k)
3 |   if (individual.novelty > novelty_threshold or novelty_archive.length <
  |    min_novelty_archive_size) then
4 |   |   novelty_archive = novelty_archive ∪ {individual}
5 |   end
6 end
7 purge_novelty_archive(novelty_archive, max_archive_size)
8 novelty_threshold = máx(adjust_archive_settings(novelty_archive,
   novelty_threshold), novelty_floor)
```

Novelty search ha tenido éxito en algunos dominios, sobre todo en aquellos cuyo espacio de evaluación de fitness se considera como "engañoso" (problemas en los que las evaluaciones basadas en objetivos fallan), dado que la evaluación basada en objetivos es susceptible a quedarse atrapada en mínimos locales en los cuales el proceso evolutivo converge a una sola solución, perdiéndose así la diversidad.

Un ejemplo de un problema en el que NS se ha aplicado exitosamente ha sido en la producción de agentes capaces de navegar en laberintos. En particular, se puede pensar en la navegación en una ciudad vieja cuyo mapa es desconocido como un problema engañoso, dado que suelen tener una estructura de caminos irregular [66].

Novelty Search ha demostrado que la habilidad de la evolución para la diversificación puede ser aprovechada para encontrar soluciones cercanas a los óptimos globales. En contraste con la noción tradicional en cómputo evolutivo de la supervivencia del más apto, se da la supervivencia del más novedoso [65].

De manera análoga a agregar la diversidad como un objetivo extra, también se puede agregar la novedad como un objetivo extra. Tal y como se llevó a cabo en el trabajo anteriormente mencionado de Mouret [63]. En dicho trabajo se compararon los regímenes multiobjetivo de diversidad+fitness y novedad+fitness, contra novedad y fitness como objetivos separados, en la tarea de evolucionar controladores de robots móviles en forma de redes neuronales para la tarea de la resolución de un laberinto. Lo me-

jores resultados se obtuvieron utilizando novedad+fitness, aunque únicamente fueron marginalmente mejores que diversidad+fitness.

3.3.8. Calidad-Diversidad

El problema con la visión de Novelty Search es que utiliza la diversidad como un medio para lograr el fin usual de encontrar un mínimo global [64], es decir, como un optimizador. Además la idea de eliminar por completo los objetivos quiere decir que se pierde por completo la noción de aquellas soluciones que son buenas y malas, algo que pareciera ser crucial en la mayoría de las aplicaciones.

En contraste con la búsqueda de óptimos globales, los métodos de calidad-diversidad (Quality Diversity/QD), buscan emular la manera en que la evolución funciona en la naturaleza, de tal manera que en una sola corrida se encuentran diversas formas de resolver un mismo problema, todas igualmente viables.

El objetivo de los algoritmos de QD es encontrar una colección de individuos máximamente diversa con respecto al espacio de soluciones posibles, en el que cada miembro es máximamente apto en su propio nicho evolutivo [64]. Los algoritmos QD balancean la búsqueda de diversidad que encuentra muchos nichos evolutivos, con búsquedas locales de calidad dentro de cada nicho. En un mismo nicho, se focaliza la competencia entre los individuos, produciendo de dicha forma, los mejores individuos posibles de manera local al nicho. Este tipo de búsqueda es capaz de encontrar soluciones diversas mientras mejora las ya existentes [64].

En QD, la medición de diversidad de un individuo x, y su posición en el espacio de características o comportamientos $\mathfrak{B}$, se da a través de su caracterización en dicho espacio con un vector de características $\mathbf{b}(x)$, el cuál representa una abstracción de las acciones y/o características del fenotipo de x en el contexto del problema que se intente resolver. Cabe hacer mención que dicha abstracción es equivalente a la del vector de comportamiento utilizada en [62], y que fue mencionada en la Sección 3.3.5. Otros nombres con los que se le conoce en la literatura de QD son: Behavior Space and Behavior Characterization (BC) [64] y Descriptor Space and Behavioral Descriptor (BD) [67] [68]. La elección de las características de interés es decisión del usuario, pero se debe buscar que dicha elección sea representativa del problema y del como es que las posibles soluciones lo resuelven [67]. La idea de poder representar a las soluciones a través de un vector de características es similar a una reducción de dimensionalidad en aprendizaje de máquinas, en el sentido de la re-expresión de los datos de un espacio de alta dimensionalidad, y por lo tanto, difícil de tratar, a un subespacio de más baja dimensionalidad que sea más fácil de tratar. La gran diferencia es que en este caso se trata de una proyección libre de modelo donde las dimensiones, y características representadas en estas son elegidas manualmente, mientras que en aprendizaje de máquinas y aprendizaje profundo se utilizan algoritmos tales como el bien conocido análisis de componentes principales, el cual realiza una proyección lineal de un espacio de alta dimensionalidad a otro de baja dimensionalidad, o las redes neuronales profundas, las cuales pueden actuar como

selectores de características además de modelos de aprendizaje. En ciertos modelos de aprendizaje profundo, como los auto-codificadores, se utilizan y entrenan un par de redes neuronales en la tarea de compresión de datos desde un espacio de características de alta dimensionalidad a un espacio de baja dimensionalidad denominado espacio latente (red codificadora), y en la descompresión desde el espacio latente al espacio de características original (red decodificadora). El componente codificador puede ser visto como una generalización no lineal de PCA, con el espacio latente como una aproximación de los componentes principales [69].

El problema general a optimizar planteado en los algoritmos de QD es encontrar, para cada punto $\mathbf{b} \in \mathfrak{B}$, los parámetros x, con el mayor valor de fitness [67], i.e.:

$$\begin{aligned} & x^* = \arg\max_x F(x). \\ & \text{s.t. } \mathbf{b} = \mathbf{b}(x). \\ & \forall \mathbf{b} \in \mathfrak{B}. \end{aligned} \tag{3.16}$$

En general los algoritmos QD funcionan al muestrear con cierta granularidad $\mathfrak{B}$, y para cada región muestreada, obtener las mejores soluciones. En estos algoritmos la diversidad tiene una mayor prioridad que la calidad, y por lo tanto se visitan la mayor cantidad de regiones del espacio, pese a que estas puedan tener menor potencial respecto a calidad.

En QD, el espacio de características $\mathfrak{B}$ se divide en t nichos, $\mathbf{N} = \{N_1, N_2, ..., N_t\}$. Donde $\mathbf{N}$ debe cubrir a $\mathfrak{B}$. Cada punto $\mathbf{b}_j$ pertenece a un nicho N_i, de acuerdo a una regla de proximidad. El objetivo de los algoritmos QD es maximizar una métrica de calidad Q con respecto a cada nicho [64].

La promesa de los algoritmos de QD es que además de ser competentes en la realización de tareas orientadas tradicionalmente a optimización, puedan superar lo que es posible hacer con algoritmos tradicionales de optimización [64]. Este tipo de algoritmos también han sido llamados "algoritmos de iluminación", dado que revelan (o "iluminan") el mejor rendimiento alcanzable en cada región del espacio de fenotipos.

Cabe recalcar que este tipo de algoritmos son distintos a simplemente utilizar un marco de trabajo multiobjetivo, agregando la novedad o la diversidad como un objetivo más, tal y como se mencionó en las secciones 3.3.5 y 3.3.7. La distinción se da en la localidad de la búsqueda de calidad que se da en QD. A diferencia del caso puramente multiobjetivo, donde la noción de calidad de un individuo x es global; en QD, la noción de calidad es local al nicho N_i, donde el fenotipo de x se encuentre de acuerdo a $\mathbf{b}(x)$, y por lo tanto relativa a la calidad de los fenotipos encontrados hasta ese momento en N_i.

3.3.9. Novelty Search with Local Competition (NSLC)

A diferencia de lo que sucede en la naturaleza, la evolución de creaturas virtuales en mundos virtuales tiende a converger prematuramente a una sola morfología porque la selección recompensa de manera greedy el tipo de morfología que sea más fácil de explotar [70]. Se propone utilizar NS para ayudar a mitigar la convergencia prematura al recompensar morfologías novedosas [70]. En el algoritmo de NSLC lo que se busca es utilizar un marco de trabajo multiobjetivo, específicamente de dos objetivos: la calidad y la novedad. Sin embargo la aproximación a optimización multiobjetivo es un tanto especial en este caso. En los algoritmos QD, no se busca una optimización global, como sucedería en una aproximación multiobjetivo tradicional, sino que se busca encontrar nichos lo más diversos posibles mediante el objetivo de novedad expuesto por Novelty Search, y dentro de cada nicho obtener soluciones de calidad a través del concepto de "competencia local", el cual consiste en evaluar al individuo de acuerdo al porcentaje de individuos que supera en términos de aptitud dentro de su propio nicho, es decir, busca localmente a los mejores dentro de un mismo nicho. No se promueve un concepto de "mejor" en el sentido global, este algoritmo no se fija en los nichos que sean mejores en sí.

En el trabajo de Lehman y Stanley [70], se propone utilizar el ordenamiento de no dominancia, en el sentido de Pareto, con dos objetivos: calidad de movimiento evaluada sobre una cierta métrica de distancia recorrida (en este caso la euclídea) en un cierto tiempo de simulación por un individuo, y la novedad evaluada como una subrutina a través de Novelty Search en un espacio morfológico tridimensional (En este caso estas características morfológicas conforman un BC) compuesto por altura, masa y número de articulaciones activas (actuadores). La métrica de novedad estará dada por la distancia euclidiana promedio a los k vecinos más cercanos (de la misma manera que en NS puro) y el objetivo de calidad estará dado por la proporción de esos mismos k vecinos más cercanos que se superan con respecto a la métrica de calidad. En NSLC, los nichos emergen al calcular los k vecinos más cercanos al individuo y la calidad de un cierto individuo se calcula de manera local al nicho como la proporción de individuos en el nicho que son superados por éste.

El nicho N de un individuo x se define como los k vecinos más cercanos al individuo en el espacio $\mathfrak{B}$ de acuerdo a una distancia medida respecto a $\mathbf{b}$: $d_{\mathbf{b}}(.,.)$, i.e.:

$$\begin{aligned} N_x &= \{x = \mu_0, \mu_1, \cdots, \mu_i, \cdots, \mu_k\}, \\ &= KNN(x, P_j \cup \Lambda, d_{\mathbf{b}}, k). \end{aligned} \tag{3.17}$$

Es así como se puede plantear NSLC como un problema multi-objetivo de dos objetivos, de la siguiente forma:

$$
\begin{aligned}
\max_{x \in \mathcal{Q}} \left\{ \mathbf{F}(\mathbf{x}) = \begin{bmatrix} Q(x) \\ N(x) \end{bmatrix} \right\}, \\
Q(x) = k - ||\{\mu_i \ \forall i \in \{0, \cdots, k\} \mid \mu_i \in N_x, x \in \mathcal{Q} \text{ y } F(x) < F(\mu_i)\}||, \\
N(x) = \frac{1}{k} \sum_{i=0}^{k} d_{\mathbf{b}}(x, \mu_i), \\
\mathbf{F} : \mathcal{Q} \to \mathbb{N} \times \mathbb{R} \\
d_{\mathbf{b}} : \mathcal{Q}^2 \to \mathbb{R} \\
Q : \mathcal{Q} \to \mathbb{N}, \\
N : \mathcal{Q} \to \mathbb{R}.
\end{aligned}
\tag{3.18}
$$

En Lehman y Stanley [70], donde se propuso por primera vez este algoritmo y fue aplicado a coevolución de morfología y control, se encontró un trade-off muy sustancial: pese a obtener resultados más diversos en términos de morfología, los individuos más aptos obtenidos por NSLC fueron ligeramente peores que los individuos más aptos de un algoritmo evolutivo tradicional. En [70] se propone como posible hipótesis que a diferencia de lo que sucede en el mundo natural, en la evolución artificial, se utilizan poblaciones con tamaño fijo, y que esto limita la capacidad de exploración de cada nicho en busca de mejores individuos dado que el explorar nuevos nichos, implica necesariamente redirigir recursos fuera de los nichos que ya se han descubierto. Este es otro ejemplo del techo de complejidad con el que se ha encontrado el área.

Nota: Cabe aclarar que hay muchos casos en el mundo natural donde las poblaciones se pueden modelar como aproximadamente constantes, tal y como sucede en los sistemas depredador-presa, sin embargo, en esta tesis se presume que probablemente a lo que se referían los autores de [70], es a aquellos sistemas que no se pueden modelar como poblaciones constantes.

3.3.10. MAP-Elites

A diferencia de NSLC y NS, donde los nichos emergen como parte del proceso de búsqueda de NS y el archivo de novedad que éste mantiene, en el algoritmo de MAP-Elites (Multidimensional Archive of Phenotypic Elites) el espacio de comportamiento es explícitamente subdividido en nichos ("bins"/"recipientes") creados al discretizar $\mathfrak{B}$ [64]. En este caso, cada bin guarda al individuo más apto (élite) que se ha producido hasta el momento. A diferencia de NSLC, se deshecha el concepto de población en cada generación en favor del conjunto de élites descubiertas en cada bin [49]. En este sentido, no se está evolucionando una población, sino el archivo de élites [68]. El proceso de búsqueda en MAP-Elites consiste en que en cada generación se elige alguna élite de algún nicho con probabilidad uniforme y se le aplica algún operador de variación (mutación o cruza) al genotipo, posteriormente se obtiene el fenotipo junto con su vector de características correspondiente y se evalúa, después con base en cada uno de los

valores del vector de BC del fenotipo se ve a que nicho pertenece. Finalmente se compara la aptitud de la élite del nicho al que el nuevo individuo pertenece, si la aptitud del nuevo individuo es mayor, éste se mapea al nicho como el nuevo individuo élite [49]. Como suele ser común, este proceso se repite hasta que algún criterio de paro dependiente del problema se cumpla. Dicho proceso puede ser masivamente paralelizable [49] si se considera un esquema de bloqueo pesimista, donde cada proceso bloquee la elección del bin, la mutación y la potencial sustitución en el nuevo bin. También se puede considerar un esquema de batches, donde el paralelismo es implícito, de manera similar al que se encuentra en algoritmos evolutivos poblacionales, dicho esquema generaliza el ciclo anteriormente descrito, para la elección de un solo bin en una iteración, a la elección sin reemplazo de varios bins, en cada iteración. El proceso descrito, se puede expresar en términos del Algoritmo 1, al considerar las siguientes subrutinas:

1. Inicialización del algoritmo: La inicialización de MAP-Elites depende de la discretización del espacio de características en bins o nichos y la inicialización de un número de individuos P, que se puede considerar como la población inicial, situando cada individuo en su bin correspondiente, dado $\mathbf{b}(x)$.

   - Discretización de $\mathfrak{B}$: Dado un espacio de características $\mathfrak{B}$ de dimensión n, definido como:

     $$\mathfrak{B} = \{\mathbf{b} \in \mathbb{R}^n | l_i \leq \mathbf{b}_i \leq u_i \ \forall i \in \{1, \cdots, n\}\}. \tag{3.19}$$

     Donde l_i y u_i, son los límites inferior y superior de la característica i-ésima, delimitando los valores posibles de cada característica. La discretización b_i de una cierta característica $\mathbf{b}_i$, con cierta granularidad o número de puntos q_i, estará dada por:

     $$b_i = \{l_i + k_i \Delta \mathbf{b}_i | k_i \in \{0, \cdots, q_i - 1\}\}, \tag{3.20}$$

     $$\text{donde: } \Delta \mathbf{b}_i = \frac{u_i - l_i}{q_i - 1}.$$

     La discretización B del espacio $\mathfrak{B}$ se define como:

     $$B = b_1 \times b_2 \cdots \times b_i \cdots \times b_n.$$

   - Inicialización de la población y ubicación de los individuos en el espacio discretizado B: Se toma una muestra aleatoria de individuos de tamaño P; se inicializan P genotipos aleatoriamente, se expresan como fenotipos, y se evalúan.

     Para poder situar a un cierto individuo x en la discretización B, se tiene que tomar el vector de características $\mathbf{b}(x)$, y ubicar el bin en el que éste se puede colocar, dado el rango de valores a los que pertenece en cada característica.

El número de esténciles, o subdivisiones por característica está dado por: $s_i = q_i - 1$. Por lo que tenemos:

$$\Delta \mathbf{b}_i = \frac{u_i - l_i}{s_i}. \tag{3.21}$$

Si queremos el índice del esténcil en el cuál cae un individuo x en la dimensión i-ésima, dado el valor de $\mathbf{b}(x)_i$, basta con hacer un simple despeje de k_i en la expresión resultante de sustituir 3.21 en 3.20. Considerando que los índices de los esténciles comienzan en cero, la expresión que determina el índice del esténcil de la dimensión i-ésima en el cuál se sitúa el individuo x, dado su vector de características $\mathbf{b}(x)$:

$$J_i = \text{mín}\left(\left\lfloor \frac{(\mathbf{b}(x)_i - l_i)s_i}{u_i - l_i} \right\rfloor, s_i - 1\right). \tag{3.22}$$

Con esto, se puede llevar un vector de características en los números reales, a un vector de características entero, donde cada entrada determina la subdivisión o esténcil al que pertenece, dado el valor de la característica. Es decir:

$$(b_1, b_2, \cdots, b_i, \cdots, b_n) \rightarrow (J_0, J_1, \cdots, J_i, \cdots, J_n).$$

Dado este mapeo, se puede obtener el índice del bin al cual pertenece el individuo x como:

$$I_{\mathbf{b}(x)} = J_0 + \sum_{j=1}^{n}\left(J_j \prod_{i=0}^{j-1} s_i\right). \tag{3.23}$$

2. Supervivencia: Cada individuo de una población de hijos de tamaño P es evaluado y situado en algún bin con base en su vector de características $\mathbf{b}$, si el hijo es más apto que la élite del bin, o el bin está vacío, el hijo es la nueva élite.
3. Selección: Cada individuo de una población de padres de tamaño P es elegido con probabilidad uniforme de alguna de las élites presentes en el grid.

Vale la pena hacer mención que en [68] y [67], los autores presentan un marco de trabajo para la generalización de algoritmos QD. La propuesta de dicho marco de trabajo se efectúa a través de la abstracción de las partes principales que componen a todos, y se da un ejemplo de como se puede utilizar para generar nuevos algoritmos QD. En dichos trabajos se hace la distinción entre los dos tipos de archivos principales que se pueden tener en un algoritmo QD, el de NSLC lo bautizan como un archivo no estructurado (o simplemente archivo) y al de MAP-Elites como "grid".

3.3.11. Multi-BC QD

Una característica importante de un espacio de características adecuado es que se alinee con la noción de calidad. La alineación en el contexto de un BC se define como el grado

en el que encontrar diversidad, lleva también a encontrar mayor aptitud [64]. Se ha encontrado que buscar diversidad utilizando un BC con mayor grado de alineación da mejores resultados, simplemente porque la búsqueda de soluciones diversas también lleva a mejores soluciones en el sentido de calidad, y por lo tanto, se mitiga el problema de convergencia prematura. Algunos ejemplos donde se pueden distinguir los efectos de tener BC alineados y desalineados son:

- En el dominio de resolución de laberintos por agentes, donde se busca encontrar la topología y pesos de una red neuronal que los controle a través de entradas dadas por sensores de proximidad (para detección de obstáculos), y por sensores de navegación (que le digan al robot la dirección del objetivo) y salidas que controlan la aceleración lineal y angular. Un BC asociado simplemente a la posición final del agente en el laberinto estará altamente alineado, dado que un agente que encuentre continuamente nuevas posiciones eventualmente llegará a la posición final en el laberinto [64].
- En coevolución de morfología y control de creaturas virtuales, la búsqueda de morfologías novedosas en muchos casos no lleva a mejores ciclos de caminado [70] (al menos en el corto plazo, sin protección de diversidad [8] y sin introducir desarrollo [6]), sin embargo es algo que ha sido muy buscado para atender al problema de la convergencia prematura de la morfología respecto al control, dado que la convergencia prematura suele ser una consecuencia de la pérdida de diversidad.

El grado de alineación es extremadamente difícil de medir a priori, sin embargo se puede anticipar considerando las siguientes dos propiedades:

1. Cada comportamiento está asociado con un rango reducido de valores de aptitud.
2. El valor máximo de fitness en regiones adyacentes del BC está correlacionado.

Soros et.al. [64] mencionan que cuando se busca diversidad utilizando BCs desalineados, los algoritmos QD fallan en obtener las soluciones óptimas y en los problemas más complejos, pueden fallar completamente en hacer algún progreso hacia soluciones siquiera viables. Es por ello que proponen el uso de nuevas variantes de los algoritmos MAP-Elites y NSLC capaces de manejar múltiples BCs. A continuación se presentan las variantes más representativas

3.3.11.1. Multi-BC Novelty Search with Local Competition (MNSLC)

Esta extensión al algoritmo NSLC incorpora como un tercer objetivo, una métrica de novedad alineada a la calidad, y puede ser especialmente útil en dominios donde la novedad utilizada en NSLC esté desalineada con la calidad. Este segundo BC incorporaría características deseables que según el juicio del tomador de decisiones pudiesen representar una potencial alineación a la calidad. Lo anterior da pie a imaginar potenciales variaciones con más de dos BC, con varias nociones de competencia local, etc., utilizando un algoritmo evolutivo multiobjetivo capaz de lidiar con la maldición

de la dimensionalidad. Cabe aclarar que con cada BC que se introduzca, se tendrá que introducir un nuevo archivo de novedad, junto con la distancia definida para ese BC para poder calcular la novedad.

3.3.11.2. Multi-BC MAP-Elites (ME-ME)

De manera similar a MNSLC, este algoritmo introduce un nuevo BC alineado a la calidad. De manera que en cada generación, se selecciona la misma cantidad de padres de cada grid, y sus hijos se mapean a ambos archivos independientemente (un hijo puede guardarse, en ambos, uno o ningún archivo).

3.3.12. Optimización multimodal

Este tipo de algoritmos surgieron como precursores a los QD. Como su nombre lo indica, operan en espacios de búsqueda multimodales, es decir, con múltiples óptimos locales y su objetivo no es el descubrimiento del óptimo global, sino de múltiples óptimos locales, de manera similar a los algoritmos QD. La principal diferencia reside en que se enfocan en diversidad genética (en contraste con la diversidad de comportamiento en QD), y aplican únicamente para problemas cuya representación en el espacio de los genotipos y los fenotipos es la misma, por ejemplo: funciones matemáticas. Una limitante asociada a este enfoque, es el aliasing genético: la diversidad del genotipo puede no estar necesariamente asociada a diversidad del fenotipo o del comportamiento, esto puede suceder en mapeos complejos del genotipo a fenotipo, tal y como sucede en robótica evolutiva [23].

Capítulo 4

Experimentos

En este capítulo se presenta la configuración experimental de esta tesis. Se siguió un marco de trabajo lo más general posible para la realización de experimentos de creaturas virtuales y robótica evolutiva, trabajando sobre los componentes principales identificados en los capítulos 2 y 3.

4.1. Representación

Se eligió tomar como base la representación de morfología y control de Cheney et al. [4] porque utiliza un genotipo diploide, que a través de dos redes neuronales CPPN codifica ambos aspectos de las creaturas. Esto se traduce en poder hacer variaciones en alguno de los dos aspectos sin afectar el otro. Además, es posible medir la diversidad genotípica y complejidad de las redes neuronales de uno u otro aspecto y cómo se relaciona con la diversidad fenotípica, así como proponer algunos indicadores relevantes para detectar y medir el fenómeno de la convergencia prematura de la morfología respecto al control.

Así mismo, como ya se ha observado en la literatura [31], la utilización de CPPNs como codificación indirecta suele llevar a mejores resultados en cuanto a facilidad de evolución, al ser capaces de producir morfologías y controladores con patrones regulares.

Para las CPPN utilizadas, en general se tiene una región $\Omega \subset \mathbb{R}^3$ que se refiere un espacio de diseño tridimensional acotado arbitrariamente:

$$\Omega = \begin{cases} x & \in [a_1, b_1]. \\ y & \in [a_2, b_2]. \\ z & \in [a_3, b_3]. \end{cases} \tag{4.1}$$

En el caso particular de los experimentos, se considera un espacio de diseño normalizado:

$$\Omega = \begin{cases} x & \in [0,1]. \\ y & \in [0,1]. \\ z & \in [0,1]. \end{cases} \tag{4.2}$$

4.2. Morfología

Para los experimentos desarrollados en esta tesis se eligió evolucionar robots con morfologías blandas basadas en voxeles (softbots), al igual que en el trabajo de Cheney et al. [4]. Se eligieron por su promesa en cuanto a versatilidad, mayor facilidad para su evolución, y el potencial para ser llevados del mundo virtual al físico, tal y como se mencionó en la Sección 3.1.6.2. El funcionamiento de este tipo de morfología se mencionó brevemente en el capítulo anterior (3.1.2), sin embargo difiere en cuanto a [31], donde el enfoque es el uso de distintos materiales, y el control está embebido en los materiales para voxeles activos, con desfases fijos de 0 y $\pi/2$, respecto a una frecuencia fija.

Para los experimentos efectuados se tienen morfologías con únicamente dos tipos de voxeles, activos y pasivos. Los voxeles activos toman el papel de músculos o actuadores, cada uno efectúa expansiones y contracciones a una misma frecuencia, pero con compensaciones de fase distintas, permitiendo así obtener movimientos similares a la propagación de una onda. Una morfología válida se crea en un espacio de diseño discreto, con una resolución fija de 5x5x5, y se compone de una combinación de cualquier cantidad de voxeles tales que sean contiguos.

Se utilizó una red neuronal CPPN, tal y como se explica en la Sección 3.1.3.2 y se define en la Ecuación 3.2, para codificar la morfología. La red neuronal $CPPN_m$ tiene un número fijo de entradas y de salidas. Se tienen cuatro entradas (x, y, z, r), que corresponden a las tres coordenadas euclídeas del espacio de diseño tridimensional, más una radial (con la esperanza de incentivar la aparición de patrones radiales). Análogamente se tienen dos salidas, $(p, m) \in \mathbb{R}^2$, donde p indica la presencia o ausencia del voxel en la morfología y m indica el material (activo o pasivo).

La manera en la cual se hace "crecer" la morfología de un softbot s a partir de su $CPPN_m$, es haciendo consultas individuales a la red neuronal, respecto a cada uno de los elementos del espacio de diseño, si $p \in (-\infty, 0)$ el voxel se queda vacío, y si $p \in [0, \infty)$, el voxel pertenece a la morfología, y será ya sea pasivo, si $m \in (-\infty, 0)$, y activo, si $m \in [0, \infty)$. En la Figura 4.1 se ejemplifica lo anterior.

Nota: Se considera que las redes neuronales CPPN de estos experimentos son funciones únicamente de (x, y, z), y no también de r, porque se puede decir que están conectadas, a la entrada, a una neurona con función de activación $r = \sqrt{x^2 + y^2 + z^2}$.

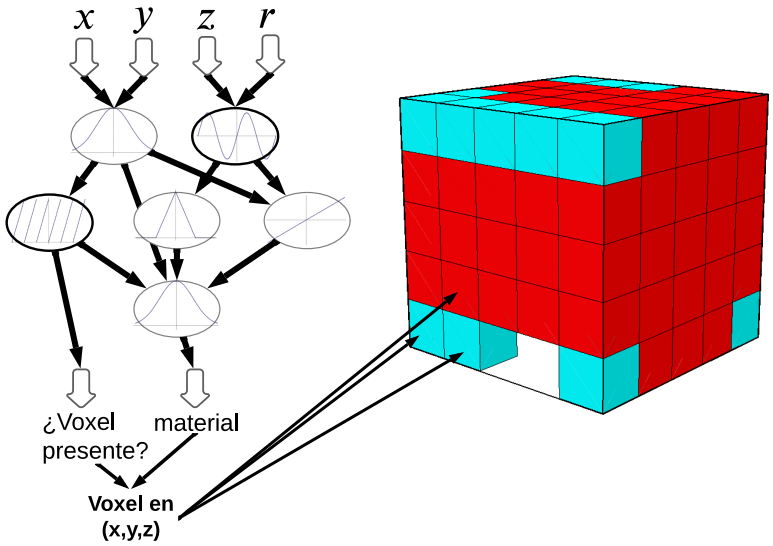


Figura 4.1: Ejemplo de la CPPN de morfología y su expresión en el fenotipo [31].

Con todo ello, se puede expresar la evaluación de $CPPN_m$ como una función $\vec{m}_s : \Omega \rightarrow \mathbb{R}^2$.

$$\vec{m}_s(x,y,z) = CPPN_m(x,y,z) = \begin{bmatrix} p(x,y,z) \\ m(x,y,z) \end{bmatrix} \tag{4.3}$$

Y a su vez, la evaluación de $CPPN_m$, da lugar a la expresión de la morfología como:

$$M_s(x,y,z) = \begin{cases} Voxel_{nil} & \text{si } p(x,y,z) \in (-\infty, 0). \\ Voxel_p & \text{si } p(x,y,z) \in [0,\infty) \text{ y } m(x,y,z) \in (-\infty, 0) \text{ , } (x,y,z) \in \Omega. \\ Voxel_a & \text{si } p(x,y,z) \in [0,\infty) \text{ y } m(x,y,z) \in [0,\infty). \end{cases} \tag{4.4}$$

4.3. Control

Para el caso del sistema de control, éste se conforma por el conjunto de todos aquellos voxeles activos dentro de la morfología, definido como:

$$\mathcal{A}(s) = \{M_s(x,y,z) | (x,y,z) \in \Omega \text{ y } M_s(x,y,z) = Voxel_a\} \tag{4.5}$$

Una vez que se tiene la morfología, se toman todos los voxeles del sistema de control, y dadas las coordenadas de cada uno $\Omega_{\mathcal{A}} = \{(x,y,z) | (x,y,z) \in \Omega \text{ y } M_s(x,y,z) = Voxel_a\}$, se consulta una segunda red $CPPN_c$ correspondiente al control. Dicha red evaluada $CPPN_c(x,y,z)$ nos va a dar los parámetros de control de cada voxel activo.

En $CPPN_c$, las entradas son las mismas que en la red de morfología, y las salidas también son dos, $(\phi, \tau) \in \{[0, 2\pi] \times [-1, 1]\}$, donde $\phi(x,y,z)$ corresponde al desfase que

tendrá el voxel actuador respecto a una frecuencia global f_g y $\tau(x, y, z)$ corresponde al periodo.

Respecto a $\phi(x, y, z)$, puede tomar valores escalados de $[0, 2\pi]$ radianes, lo cual se logra conectando siempre la salida u a una neurona con función de activación logística escalada por 2π, donde $u(x, y, z) : \Omega_{\mathcal{A}} \to \mathbb{R}$. La evaluación de ϕ se define de la siguiente forma:

$$\phi(u(x, y, z)) = \frac{2\pi}{1 + e^{-u(x,y,z)}}. \tag{4.6}$$

Para el caso del periodo $\tau(x, y, z)$, éste se utiliza para obtener la frecuencia global f_g, promediando la frecuencia obtenida para cada voxel activo $f(x, y, z) = \frac{1}{\tau(x,y,z)}$, $(x, y, z) \in \Omega_{\mathcal{A}}$:

$$f_g = \frac{10}{\frac{1}{||\Omega_{\mathcal{A}}||} \sum\limits_{(x,y,z)\in\Omega_{\mathcal{A}}} f(x, y, z) + 1.5}. \tag{4.7}$$

El valor de τ se obtiene conectando la salida v de la red neuronal a una función de activación logística sesgada en el intervalo $[-1, 1]$, donde $v(x, y, z) : \Omega_{\mathcal{A}} \to \mathbb{R}$. La evaluación de τ se define de la siguiente forma:

$$\tau(v(x, y, z)) = \frac{2}{1 + e^{-v(x,y,z)}} - 1. \tag{4.8}$$

Además, el cálculo de f_g tiene discontinuidades en $\frac{1}{||\Omega_{\mathcal{A}}||} \sum\limits_{(x,y,z)\in\Omega_{\mathcal{A}}} f(x, y, z) = -\frac{2}{3}$ y $\tau(v(x, y, z)) = 0$, por lo tanto, todos aquellos softbots cuyo cálculo de f_g lleve a errores numéricos de divisiones entre cero, serán evaluados con una aptitud de cero.

Con base en lo anterior, se puede expresar la evaluación de $CPPN_c$ como una función $\vec{c}_s : \Omega_{\mathcal{A}} \to \{[0, 2\pi] \times [-1, 1]\}$.

$$\vec{c}_s(x, y, z) = CPPN_c(x, y, z) = \begin{bmatrix} \phi(x, y, z) \\ \tau(x, y, z) \end{bmatrix} \tag{4.9}$$

Que a su vez, da lugar a la expresión de los parámetros del control como:

$$C_s(x, y, z) = \begin{bmatrix} f = f_g \\ \phi = \phi(x, y, z) \end{bmatrix}, (x, y, z) \in \Omega_{\mathcal{A}}. \tag{4.10}$$

Cheney et al. [4] mencionan que la inspiración para este tipo de control se encuentra en el trabajo de Joachimczak et al. [7](explicado brevemente en el Capítulo 3), solo que en su caso las morfologías son tridimensionales, y se utiliza un motor físico dedicado a simulaciones basadas en voxeles llamado Voxelyze [58]. El funcionamiento del motor

de física utilizado para efectuar las simulaciones se sale del alcance de esta tesis, sin embargo, Cheney et al. mencionan que a grandes rasgos es similar al presentado en [7]. En dicho trabajo se describe el sistema de control de los Animats como un sistema masa-resorte-amortiguador, donde las masas corresponden a los centroides de las células (nodos) al final del periodo de desarrollo, y los resortes a las aristas formadas a partir de la triangulación de Delaunay de los nodos. La longitud en reposo L_0 de cada resorte es determinada al efectuar la triangulación de Delaunay.

La actuación se logra al alterar la longitud de reposo de acuerdo a la siguiente señal:

$$L(t) = [1 + A\sin(\frac{2\pi t}{\tau_1} + \phi_1) + A\sin(\frac{2\pi t}{\tau_2} + \phi_2)] \cdot L_0.$$

donde τ_1, τ_2 y ϕ_1, ϕ_2 corresponden a los periodos y fases de cada célula al extremo del resorte.

La modificación de la longitud de reposo sirve como señal de entrada el sistema, el cual se rige por un sistema de dos ecuaciones diferenciales.

Primero se tiene la ley de Hooke amortiguada:

$$F_s = -kx - c\frac{dx}{dt} \ ,$$

donde todos los resortes tienen la misma constante k, c es el coeficiente de amortiguamiento, y $x = L(t) - L_0$.

Adicionalmente, cada resorte tiene una fuerza de presión F_p, actuando de manera perpendicular al resorte y hacia el exterior de la región triangular a la que pertenece, representando que cada región triangular tiene una presión de equilibrio proporcional al área inicial de la región S_0, e inversamente proporcional al área en el tiempo t, $S(t)$, que limita la expansión/compresión excesiva.

Dicha fuerza se encuentra dada por:

$$F_p = c_p L(t) \left(\frac{S_0}{S} - 1\right),$$

donde c_p se refiere al coeficiente de presión global.

Al igual que los Animats, los softbots también se controlan a través de un sistema masa resorte amortiguador, para el caso de los softbots, los nodos se encuentran al centro de cada voxel y debido a que las morfologías se obtienen a partir de voxeles y no triangulaciones, forman una látice, tal y como la que se muestra en la Figura 4.2.

Este sistema es de lazo abierto, dado que no se toma en cuenta ningún tipo de señal de error o de sensores como realimentación para el sistema de control, únicamente se incide una señal de entrada al sistema.

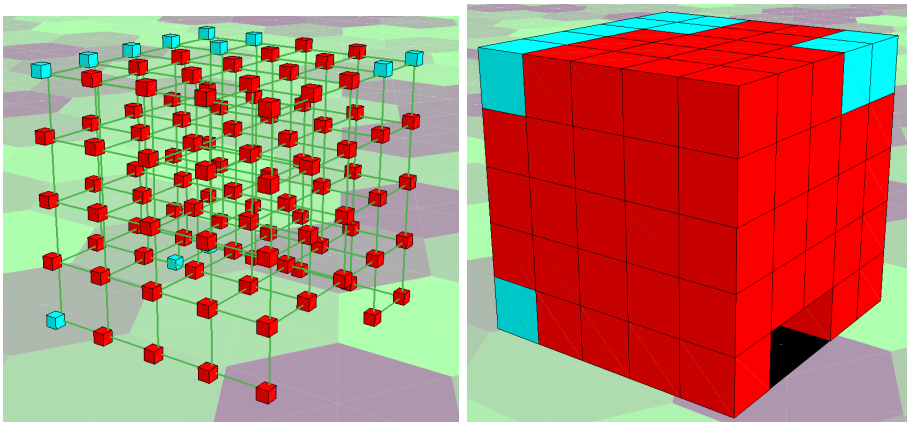


Figura 4.2: Ejemplo de la látice de masa-resorte-amortiguador de un softbot evolucionado en uno de los experimentos, renderizado en VoxCAD [58].

4.4. Definición de Softbot

De acuerdo a las definiciones para morfología y control expuestas en las Secciones 4.2 y 4.3, es posible definir a un softbot s, del tipo que se evoluciona en los experimentos, con base en su genotipo para la morfología (Ecuación 4.3) y control (Ecuación 4.9), así como su fenotipo para la morfología (Ecuación 4.4) y control (Ecuación 4.10).

El genotipo G_s de un softbot s se define como:

$$G_s = (CPPN_m, CPPN_c, \vec{m}_s(x,y,z), \vec{c}_s(x,y,z)) \tag{4.11}$$

Y el fenotipo P_s de s, se define como:

$$P_s = (M_s(x,y,z), C_s(x,y,z)) \tag{4.12}$$

Con lo cual, un softbot s se define como:

$$s = (G_s, P_s) \tag{4.13}$$

4.5. Tarea

La tarea que se eligió para todos los experimentos de este trabajo, fue la de aprendizaje de patrones de caminado, sin duda, la tarea más común en la literatura de coevolución de morfología y control. Sin embargo no es la única, y probablemente en la practica, podría interesar más el aprendizaje de patrones de caminado para poder seguir objetivos [34] [22] [52], caminar con obstáculos y en distintos ambientes [53], desempeñarse

en ambientes reducidos [50] o competir por apoderarse de un objeto [22] [26]. A pesar de ello, es comparativamente la más simple y reducida, y el interés de esta investigación no se encuentra en explorar distintos ambientes o tareas (pese a ser rutas de investigación igualmente válidas). El interés se encuentra principalmente en la incidencia de la convergencia prematura de morfología respecto a control, y la comparación o benchmarking de distintos algoritmos con mecanismos de protección a la diversidad morfológica en cuanto a distintos indicadores pertinentes.

4.6. Ambiente simulado

El ambiente simulado en donde se desempeñan los softbots es un plano en el motor de física Voxelyze [58], en su versión acelerada por GPU utilizando CUDA, bautizada como Voxcraft [59].

Originalmente los experimentos fueron implementados en el motor Voxelyze y se tenía conciencia de los altos requerimientos computacionales de lograr ejecutarlos, por lo que se decidió, con base en los parámetros y resultados de los experimentos de Cheney et al. [4], que bastaría efectuar 10 ejecuciones, cada una con 3000 generaciones y poblaciones de 30. Pero al hacer la prueba, el tiempo de los experimentos era muy alto, y además, al examinar más cuidadosamente la literatura, se vio que 10 corridas probablemente no bastarían para obtener resultados estadísticamente significativos.

Se volvió aparente que la causa de este cuello de botella era el simulador físico Voxelyze, el cual se ejecuta en el procesador directamente. Por ello, se tomó el código existente de la representación y expresión del fenotipo y se adaptó para utilizar en su lugar Voxcraft, el cual fue desarrollado para sustituir a Voxelyze y paralelizar masivamente la ejecución de múltiples experimentos, así como escalar a experimentos con múltiples agentes simulados interactuando en el ambiente [71]. En pruebas preliminares, se observó una mejora promedio del 300 %, en cuanto a tiempo de ejecución, para poblaciones de 30 individuos, y aún mayor en poblaciones más grandes (por el aprovechamiento del paralelismo masivo de la GPU), por lo que se decidió que hacer el esfuerzo de migrar a Voxcraft valía la pena.

4.7. Algoritmos y parámetros

Primeramente definimos el espacio de búsqueda de softbots sin restricciones $\mathbb{S}'$. El espacio de diseño, de acuerdo a su resolución es de $5 \times 5 \times 5$ voxeles, tiene un volumen de $5^3 = 125$ voxeles. El espacio $\mathbb{S}'$ se compone de todos los softbots que puedan ser expresados en ese espacio de diseño. Para poder definir el espacio de búsqueda de los experimentos $\mathbb{S}$, se restringió $\mathbb{S}'$ para prevenir la aparición de morfologías triviales, como un solo voxel activo, o que todos los voxeles posibles fuesen activos. Para ello, se

agregaron una serie de restricciones al porcentaje mínimo y máximo de voxeles ocupados y al porcentaje mínimo y máximo de voxeles activos.

Dichas restricciones provocan que $\mathbb{S}$ se defina como:

$$\begin{aligned}\mathbb{S} = \{s | s \in \mathbb{S}' \text{ y} \\ 7 \leq M_a(s) + M_p(s) \leq 112 \\ \lceil 0.1 \cdot (M_a(s) + M_p(s)) \rceil \leq M_a(s) \leq \lfloor 0.9 \cdot (M_a(s) + M_p(s)) \rfloor\},\end{aligned} \tag{4.14}$$

donde:

$$\begin{aligned} M_a(s) &= ||\mathcal{A}(s)||, \\ M_p(s) &= ||\mathcal{P}(s)||, \\ M_p(s), M_a(s) &: \mathbb{S}' \to [0, 125] \subset \mathbb{N}, \\ \mathbb{S} &\subset \mathbb{S}'. \end{aligned} \tag{4.15}$$

Las funciones $\mathcal{P}(s)$ y $\mathcal{A}(s)$ se definen como:

$$\begin{aligned}\mathcal{A}(s) = \{s.P_s.M_s(x,y,z) | (x,y,z) \in \Omega \text{ y } s.P_s.M_s(x,y,z) = Voxel_a\} \\ \mathcal{P}(s) = \{s.P_s.M_s(x,y,z) | (x,y,z) \in \Omega \text{ y } s.P_s.M_s(x,y,z) = Voxel_p\}\end{aligned}, \ s \in \mathbb{S}'. \tag{4.16}$$

Los experimentos se realizaron con cinco algoritmos distintos (Algoritmo evolutivo Mono-objetivo (SO), MOEA: Aptitud-Novedad (QN), NSLC, MAP-Elites, MNSLC), con 20 ejecuciones de cada uno para obtener resultados estadísticamente significativos.

Para identificar los parámetros con los que los algoritmos se tenían que correr, especialmente con los que presentan mecanismos de protección a la diversidad, se efectuaron ejecuciones en problemas de prueba con funciones multimodales, en lugar hacerlo directamente en el problema de coevolución de morfología y control, dado el alto coste computacional de éste último. Fue de particular interés, encontrar un conjunto de parámetros que conciliara el hecho de que a mayor número de individuos por generación, se sacaría más provecho de las simulaciones en GPU.

Se corrieron los algoritmos QN, NSLC y MAP-Elites en los problemas de prueba Rastrigin, Ackley, Eggholder, y Schaffer4, con especial atención a los parámetros utilizados para éste último, dado que el dominio para el cual está definido ($x_1, x_2 \in [-100, 100]$) es similar en tamaño al BC desalineado definido respecto al número de voxeles pasivos y activos $M_p(s), M_a(s)$:

Una vez que se identificaron los parámetros con los problemas de prueba, se hicieron algunas corridas de preliminares con menos generaciones (200), con el propósito de refinar los parámetros obtenidos. Con ello, se llegó a los siguientes parámetros para todos los experimentos:

- Mutación ($p_m = \frac{1}{2}$) - A cada red neuronal $CPPN_m$ y $CPPN_c$ se le aplicará una mutación con probabilidad p_m. La mutación a aplicar a la red neuronal elegida $CPPN_i, i \in m, c$ será elegida aleatoriamente y puede ser alguna de:

- Agregar nodo - Se elige aleatoriamente una conexión $(u, v) \in CPPN_i.E$, con un peso $w_o = w(u, v)$. Tal y como sucede en NEAT, se agrega un nodo nuevo q entre u y v, de tal forma que se tenga al camino $u \mapsto q \mapsto v$, donde las conexiones (u, q) y (q, v) tendrán pesos de $w(u, q) = 1$ y $w(q, v) = w_o$. Se elige aleatoriamente una función de activación de entre A.
- Quitar nodo - Se elige aleatoriamente un nodo oculto conectado u y se quita, cuidando mantener la conectividad de cualquier camino que pase por u.
- Agregar conexión - Se elige aleatoriamente un par de nodos u, v de la red $CPPN_i$, tales que $(u, v) \notin CPPN_i.E$, $u, v \in CPPN_i.V_{ocultos}$ y $u \neq v$. Una vez elegido el par de nodos, se crea la conexión (u, v) y se elige un peso aleatorio ya sea de $w(u, v) = -0.1$ o $w(u, v) = 0.1$.
- Quitar conexión - Se elige aleatoriamente una conexión y se quita.
- Cambiar función de activación - Se elige algún nodo oculto o de salida y se cambia de manera aleatoria la función de activación por una distinta a la que tenía originalmente.
- Mutar peso - Se elige aleatoriamente una conexión (u, v) y se modifica su peso original $w_o = w(u, v)$ con base en una distribución normal, de tal forma que el peso nuevo es: $w(u, v) = \text{máx}(-1, \text{mín}(1, \mathcal{N}(w_o, 0.25)))$.

- $population\_size = 100$.
- $generations = 1000$.
- Resolución del espacio de diseño: $[5, 5, 5]$.
- $k = 200$: Número de vecinos más cercanos considerados para calcular la novedad de entre los individuos del archivo, y de la población actual.
- $novelty\_threshold = 12$: Valor que toma el umbral de novedad inicialmente.
- $novelty\_floor = 1$: El valor mínimo que puede tomar el umbral de novedad.
- $max\_archive\_size = 1500$: Tamaño máximo de archivo.
- $min\_archive\_size = 1$: Tamaño mínimo del archivo.
- Dimensiones de los grids : $[25, 25]$. Se trata de cuantos bins considera la discretización de los grids, por cada dimensión del BC desalineado (M_p, M_a).

Se tienen en total dos archivos y dos grids. Los dos archivos se utilizan para medir novedad alineada y desalineada. Los dos grids de fitness y novedad alineada, para medir cobertura y QD-scores. En el caso del grid de fitness, MAP-Elites también lo utiliza para su proceso de búsqueda en lugar de una población.

4.8. Implementación de los algoritmos

Se decidió no implementar los algoritmos evolutivos desde cero, pese a la ventaja que pudiera representar (mayor personalización, aprendizaje, etc.), con el propósito de reducir lo más posible la afectación de los resultados por culpa de errores humanos. Se utilizó el framework de algoritmos evolutivos multiobjetivo implementado en python PyMOO [13], por su fuerte enfoque orientado a objetos, modularidad y extensibilidad. Permitiendo integrar, de manera limpia, cohesiva y simple, las implementaciones propias de Novelty Search, Novelty Search with Local Competition y MAP-Elites, al algoritmo evolutivo genérico (Algoritmo 1), tal y como se describió en el capítulo 3, además de las variaciones de dichos algoritmos tratadas más adelante, y la función de evaluación basada en simulaciones de Voxcraft.

4.8.1. Algoritmo evolutivo Mono-objetivo (SO)

Se tomó como base el algoritmo evolutivo simple de un sólo objetivo $(\mu+\lambda)$, tal y como se describe en el Algoritmo 1. El problema a resolver fue:

$$\begin{aligned} &\max_{s\in\mathbb{S}} \left(F(s) = ||\vec{r}_s(t=T)||\right), \\ &\text{donde } F : \mathbb{S} \to \mathbb{R}. \end{aligned} \tag{4.17}$$

La función de aptitud $F(s)$ se describe a detalle más adelante, en la Sección 5.1.1.1.

Este algoritmo no presenta ningún tipo de protección a la diversidad morfológica, y sirve como punto de partida para identificar el fenómeno de la convergencia prematura de morfología respecto a control, y comparar con otros algoritmos que si presentan protección a la diversidad morfológica.

4.8.2. MOEA con dos objetivos: Aptitud-Novedad (QN)

Tal y como se menciona al final de la Sección 3.3.7, se puede integrar la novedad como un segundo objetivo además de la aptitud. Luego entonces, se tiene que en este experimento el problema a resolver fue:

$$\begin{aligned} &\max_{s\in\mathbb{S}}(\mathbf{F}(s)), \\ &\text{donde } \mathbf{F}(s) = \begin{bmatrix} F(s) \\ N_u(s) \end{bmatrix} \\ &\text{y } \mathbf{F} : \mathbb{S} \to \mathbb{R}^2. \end{aligned} \tag{4.18}$$

La función de novedad morfológica o desalineada $N_u(s)$ se describe a detalle más adelante, en la Sección 5.1.2.2.

Para este algoritmo se utilizó la implementación de NSGA-II del framework PyMOO.

Con este algoritmo, se intenta responder a las preguntas:

- ¿Cómo afecta la adición de la novedad morfológica como segundo objetivo al fenómeno de la convergencia prematura de morfología respecto a control?
- ¿Cómo afecta la adición de la novedad morfológica como segundo objetivo a la aptitud de las soluciones?
- ¿Cómo se compara esta aproximación multiobjetivo que utiliza aptitud global, a sus contrapartes QD que consideran aptitud local (NSLC y MNSLC), en cuanto al fenómeno de convergencia prematura y a la aptitud de las soluciones?

4.8.3. NSLC

Se trata del mismo algoritmo que se describe en la Sección 3.3.9. El problema que se trata de resolver es:

$$\begin{aligned} \max_{s\in\mathbb{S}} \left\{ \mathbf{F}(s) = \begin{bmatrix} Q(s) \\ N_u(s) \end{bmatrix} \right\}, \\ \text{donde } \mathbf{F} : \mathbb{S} \to \mathbb{N} \times \mathbb{R}. \end{aligned} \tag{4.19}$$

Recordemos, con base en lo descrito en la Sección 3.3.9, que $Q \in \mathbb{N}$ se refiere a la calidad local o score de competencia local al nicho del individuo $s \in \mathbb{S}$ en la población de la generación j-ésima $P_j \subset \mathbb{S}$, con respecto a dicha población, y el archivo de novedad desalineada Λ_u:

$$\begin{aligned} Q(s) = k - ||\{\mu_i \ \forall i \in \{0, \cdots, k\} | \mu_i \in N_s, s \in \mathbb{S} \text{ y } F(s) < F(\mu_i)\}||, \\ N_s = \{s = \mu_0, \mu_1, \cdots, \mu_i, \cdots, \mu_k\}, \\ = KNN(s, P_j \cup \Lambda_u, d_m, k). \end{aligned} \tag{4.20}$$

Se utilizó la implementación de NSGA-II del framework PyMOO, sustituyendo el crowding distance por el cálculo de la diversidad genética que se describe en la Ecuación 5.21.

La inclusión de este algoritmo en los experimentos intenta responder a:

- ¿Cómo afecta el uso de la competencia local en lugar de aptitud global al fenómeno de la convergencia prematura de morfología respecto a control?
- ¿Cómo afecta el uso de la competencia local en lugar de aptitud global a la aptitud de las soluciones?
- ¿Cómo afecta el uso de la diversidad genética en lugar del crowding distance?

- ¿Cómo se compara con un algoritmo QD con presión de selección basada en la aptitud global (ME) y con otro Multi-BC, en cuanto al fenómeno de convergencia prematura y a la aptitud de las soluciones?

4.8.4. MAP-Elites (ME)

Se trata del mismo algoritmo descrito en 3.3.10, sólo que se hace la modificación descrita en [68] de hacer la selección de los padres proporcional a la aptitud. Este algoritmo se incluye para intentar responder a:

- ¿Cómo afecta a un algoritmo QD como MAP-Elites la introducción de una presión de selección global?
- ¿Sigue manteniendo las características deseables de los algoritmos QD?
- ¿Cómo se compara con un algoritmo Multi-BC QD?

4.8.5. Multi-BC NSLC (MNSLC)

Se trata del mismo algoritmo que se describe en la Sección 3.3.11. El problema que se trata de resolver es:

$$\max_{s \in \mathbb{S}} \left\{ \mathbf{F}(s) = \begin{bmatrix} Q(s) \\ N_u(s) \\ N_a(s) \end{bmatrix} \right\}, \tag{4.21}$$

donde $\mathbf{F} : \mathbb{S} \to \mathbb{N} \times \mathbb{R}^2$.

La función de novedad de comportamiento o alineada $N_a(s)$ se describe a detalle más adelante, en la Sección 5.1.1.2.

Con la introducción de este algoritmo a los experimentos, hacemos las siguientes dos preguntas:

- ¿Puede agregar a NSLC un tercer objetivo de novedad en un BC que se supone como alineado a la calidad, mitigar el problema del progreso lento de los algoritmos QD, cuando la diversidad de interés (morfológica) se encuentra desalineada a la calidad?
- ¿Se pierde alguna característica deseable de los algoritmos QD? En especial deseable para la convergencia prematura de morfología respecto a control.

Capítulo 5

Análisis comparativo de la convergencia prematura

En este capítulo se presentan a detalle los indicadores considerados para realizar la comparación de los experimentos de coevolución de morfología y control de creaturas virtuales realizados en esta tesis, teniendo en mente el problema de la convergencia prematura de morfología respecto a control. Posteriormente, se presentan los resultados conforme a cada indicador, en forma de gráficos de estadística descriptiva, para comparar cuantitativamente los algoritmos elegidos en los experimentos, establecer un ordenamiento respecto los indicadores y finalmente discutir a detalle los resultados.

5.1. Principales indicadores

Con el propósito de comparar los distintos algoritmos respecto a 3 aspectos pertinentes (diversidad fenotípica, genotípica, y en el desempeño de la tarea), se introdujeron 31 indicadores originalmente, agrupados en esos aspectos mencionados. Sin embargo, al realizar regresiones lineales en toda la distribución de datos experimentales, para cada par de indicadores, y obtener coeficientes de correlación, se determinó que se iba a desechar algún indicador de cualquier par que tuviese un coeficiente de correlación $r \geq .85$, con el fin de reducir los indicadores únicamente a aquellos que proveyeran la mayor cantidad de información. Fue así que se hizo la reducción a 11 indicadores, los cuales se presentan a continuación.

5.1.1. Indicadores de desempeño de la tarea

El propósito de estos indicadores es el de medir el desempeño de los softbots en la tarea de caminar.

5.1.1.1. Aptitud

La función de aptitud respecto a la tarea de aprendizaje de patrones de caminado, fue la distancia recorrida por el softbot en una simulación de $T = 5[s]$, $||\vec{r}_s(t = T)||$, donde $||.||$ se refiere a la norma L_2 o distancia euclidiana:

$$\begin{aligned} F(s) &= ||\vec{r}_s(t = T)||, \\ \text{donde } F(s) &: \mathbb{S} \to \mathbb{R}. \end{aligned} \tag{5.1}$$

5.1.1.2. Novedad del comportamiento o alineada

Una noción de diversidad que pareciera estar bastante alineada con la aptitud o calidad, en el contexto de los experimentos a realizar, es definirla en el espacio tridimensional de simulación en el que se desempeñan los softbots, dado que sería medida en el espacio de características (BC) donde justamente se mide la aptitud de los robots.

Con inspiración directa en [63], definimos la distancia de comportamiento entre dos softbots s_1 y s_2 como la norma L_2 del vector formado por las posiciones de ambos al final de una simulación con duración T:

$$\begin{aligned} d_{\vec{r}}(s_1, s_2) &= ||\vec{r}_{s_1}(T) - \vec{r}_{s_2}(T)||, \\ d_{\vec{r}} &: \mathbb{S}^2 \to \mathbb{R}^{(+)}. \end{aligned} \tag{5.2}$$

Con dicha distancia, se puede definir la novedad del comportamiento, o bien, novedad alineada (dado que estaríamos definiéndola con base en la distancia definida sobre un BC alineado).

El cálculo de la novedad alineada N_a de un softbot s, con base en la Ecuación 3.15 y el Algoritmo 3, necesita un archivo de novedad alineada Λ_a.

Haciendo uso de la Ecuación 5.2 y sustituyendo en 3.15, se obtiene:

$$\begin{aligned} N_a(s) &= \frac{1}{k} \sum_{i=0}^{k} d_{\vec{r}}(s, \mu_i), \\ N_a &: \mathbb{S} \to \mathbb{R}^{(+)}. \end{aligned} \tag{5.3}$$

5.1.1.3. Indicadores QD

En general, la métrica principal de los algoritmos QD es el indicador QD. Para éste, se toma un grid, y sea cual sea la métrica de calidad (Q) utilizada en el criterio de selección de élites en cada bin, se suma para todos los bins, dando como resultado un indicador capaz de informar la calidad local total o acumulada de las soluciones encontradas hasta el momento, o la calidad en la diversidad encontrada hasta el momento.

Sea $Q(x_i)$ la calidad del individuo x (o para efectos prácticos, cualquier atributo de x) con mayor calidad encontrada hasta el momento en el bin con índice i del grid G, donde $G(i) = x_i$. El indicador QD se define como:

$$QD - score = \sum_{i=0}^{|G|-1} Q(x_i = G(i)) \quad x_i \in \mathbb{Q}. \tag{5.4}$$

Cabe mencionar que este tipo de indicador, junto con la cobertura (mencionada más adelante), al derivarse del algoritmo MAP-Elites, se calculan siempre con referencia a un archivo (grid), en contraste con todos los indicadores anteriormente mencionados, que se encuentran referidos a la población, recordando que en MAP-Elites la estructura principal que se está evolucionando es el grid y no la población, como en los demás algoritmos.

También vale la pena hacer la aclaración de que los dos grids utilizados se basan en la discretización del BC definido más adelante, en la Sección 5.1.2.

5.1.1.4. Indicador QD_{ff} (Grid de Fitness / Fitness)

Se introdujo un grid G_f donde el criterio principal para conservar la élite en cada bin es el fitness (Ecuación 5.1). En cada generación evaluada, se efectúa un ciclo de selección de MAP-Elites y se obtiene la calidad sustituyendo $Q(x_i) = F(s_i)$ en 5.4.

$$QD_{ff} = \sum_{i=0}^{|G_f|-1} F(s_i = G_f(i)) \quad s_i \in \mathbb{S}. \tag{5.5}$$

5.1.1.5. Indicador QD_{anan} (Grid de Novedad alineada / Novedad alineada)

Se introdujo un grid G_{an} con base en el BC alineado propuesto, donde el criterio de selección para cada ciclo de MAP-Elites fue la novedad alineada N_a, en lugar del fitness F. De igual forma que en el indicador anterior, se suma la noción de calidad que se tiene para cada bin, en este caso la novedad alineada, dando como resultado una métrica que nos dice la calidad encontrada respecto a la diversidad morfológica en las soluciones de un algoritmo, entendiéndose en este caso, la calidad como una noción de diversidad supuestamente alineada a la calidad.

$$QD_{an} = \sum_{i=0}^{|G_{an}|-1} N_a(s_i = G_{an}(i)) \quad s_i \in \mathbb{S}. \tag{5.6}$$

Cabe aclarar que la novedad alineada en este grid se calcula no solo con respecto a los miembros del grid y la población actual, sino también con respecto a los miembros del archivo de novedad alineada, esto porque dicho archivo mantiene un histórico de las soluciones que se han generado a lo largo de la corrida evolutiva, y no solo a las élites. Si se considerara solo a las élites contra la población actual, se podría considerar de manera errónea a miembros de la generación actual como más novedosos de lo que realmente son y a miembros del grid como menos novedosos, o viceversa, llevando a posibles errores en la selección de élites.

5.1.2. Indicadores de diversidad fenotípica

Estos indicadores tienen como propósito medir la diversidad de los softbots en términos del genotipo expresado en la morfología.

5.1.2.1. Diversidad de morfología

Como bien se mencionó en la Sección 3.3.5, si se desea introducir diversidad en coevolución de morfología y control de creaturas virtuales, se necesita una noción de distancia, la cual, puede ser muy variada. Más aún, si nos decidiéramos por medir distancias entre las redes neuronales que codifican a la morfología (grafos), y utilizáramos una distancia definida para grafos, como por ejemplo una distancia de edición, nos encontraríamos que dicha distancia para grafos dirigidos y cíclicos es un problema NP-Completo por sí solo. Sin embargo, se puede introducir la noción de vector y espacio de características (BC), haciendo una reducción de dimensionalidad libre de modelo.

Se tomó inspiración de [9] para definir el vector de características morfológicas de un softbot s como:

$$\begin{aligned} \mathbf{b}_m(s) &= (M_p(s), M_a(s)), \\ \mathbf{b}_m &: \mathbb{S} \to \mathbb{N}^2. \end{aligned} \tag{5.7}$$

donde M_p y M_a, son el número de voxeles pasivos y activos respectivamente, de la morfología del softbot s, tal y como se definen en la Ecuación 4.15.

Con ello, se define la distancia morfológica entre dos softbots s_1 y s_2 como:

$$\begin{aligned} d_m(s_1, s_2) &= ||\mathbf{b}_m(s_1) - \mathbf{b}_m(s_2)||, \\ \mathbf{b}_m &: \mathbb{S}^2 \to \mathbb{R}^{(+)}. \end{aligned} \tag{5.8}$$

Y a su vez, se puede definir la diversidad morfológica de un softbot s respecto a la población en la generación n (P_n), sustituyendo 5.8 en 3.9:

$$\begin{aligned} D_m(s) &= \frac{1}{|P_n|} \sum_{s' \in P_n} d_m(s, s'), \\ D_m &: \mathbb{S}^2 \to \mathbb{R}^{(+)}. \end{aligned} \tag{5.9}$$

5.1.2.2. Novedad de morfología o desalineada

Tal y como se mencionó en la Sección 3.3.11, en coevolución de morfología y control, la búsqueda de morfologías novedosas, suele no llevar a mejores patrones de caminado, por lo tanto el BC definido anteriormente, sobre el cual a su vez, se definió la distancia morfológica, es un BC desalineado. De igual forma si definimos la novedad utilizando dicha distancia, tendremos una novedad morfológica, o desalineada.

Para el cálculo de dicha novedad desalineada N_u de un softbot s, se hace uso de la Ecuación 5.8 y se sustituye en 3.15, para obtener:

$$\begin{aligned} N_u(s) &= \frac{1}{k}\sum_{i=0}^{k} d_m(s, \mu_i), \\ N_u &: \mathbb{S} \to \mathbb{R}^{(+)}. \end{aligned} \tag{5.10}$$

5.1.2.3. Indicador QD_{fun} (Grid de Fitness / Novedad desalineada)

Se entiende como una métrica sobre la novedad desalineada acumulada encontrada en el grid de fitness. Para su cálculo, tomamos $Q(x_i) = N_u(s_i)$, en la Ecuación 5.4 y al igual que para QD_{anan}, se toma en cuenta el grid, la población y el archivo de novedad desalineada.

$$\begin{aligned} QD_{fun} &= \sum_{i=0}^{|G_f|-1} N_u(s_i = G_f(i)), \\ s_i &\in \mathbb{S}. \end{aligned} \tag{5.11}$$

Para este caso, tal vez el nombre de métrica QD pueda quedar un poco en duda, dado que la calidad utilizada para el grid es el fitness y no la novedad desalineada, sin embargo, vale la pena recordar que la definición que se está tomando para la métrica QD en esta tesis, conforme a la Sección 5.1.1.3, permite utilizar cualquier métrica como base para Q.

5.1.2.4. Cobertura

Dado un grid, se trata simplemente de la proporción entre el número de bins descubiertos y el número de bins totales.

Se puede expresar como:

$$\mathcal{C} = \frac{|\{G(i) \forall i \in \{0, ..., |G| - 1\} | G(i) \neq nil\}|}{|G|}. \tag{5.12}$$

Cabe recalcar que para un mismo algoritmo, cualquier cantidad de grids definidos en un mismo BC siempre tendrán la misma cobertura, dado que se está cuantificando la proporción del espacio de un BC que se ha visitado al menos una vez, independientemente de la métrica de calidad que se esté utilizando para actualizar el grid.

5.1.3. Indicadores de diversidad genética

Estos indicadores surgen en la implementación de NSLC, por la necesidad de definir una diversidad entre genotipos un poco más rigurosa que la que se define en el paper original de dicho algoritmo [70]. En ese trabajo, se expresa la necesidad de sustituir al crowding distance de NSGA-II como mecanismo promotor de la diversidad a lo largo de los frentes de Pareto, por un indicador de diversidad genotípica. Se alega que una diversidad definida en el espacio de los objetivos de competencia local y novedad no necesariamente va a reflejar la diversidad de interés (en nuestro caso, la diversidad de morfologías). Por ejemplo, se podría tener dos individuos con exactamente los mismos scores de competencia local y novedad, sin embargo, al ser ambos scores relativos a los k individuos más cercanos, estos dos individuos podrían encontrarse en regiones del espacio de morfología completamente distintas (es decir, ser individuos morfológicamente distintos). Sin embargo, el crowding distance penalizaría fuertemente a estos dos individuos, potencialmente descartándolos. Lehman y Stanley [70] definen la diversidad genética con base en una distancia basada en el conteo de nodos y aristas de los grafos que definen a la morfología y control de los robots que están evolucionando.

Para la distancia genotípica que se define en esta tesis, se aprovechó el hecho de que se está trabajando con genotipos basados en redes neuronales, para definir una distancia genotípica basada en las salidas de las CPPN que codifican a la morfología y control.

Partiendo de que una red neuronal es un aproximador universal de funciones [39], se puede ver a una red neuronal CPPN con m neuronas de entrada y n neuronas de salida, como una función:

$$\mathbf{F} : \mathbb{R}^m \to \mathbb{R}^n. \tag{5.13}$$

Entonces, es posible definir una métrica o pseudo-métrica que capture la noción de distancia entre funciones con dicha forma.

Recordando el producto interior entre dos funciones escalares de variable escalar $f, g : \mathbb{R} \to \mathbb{R}$, en el intervalo $[a, b]$, definido como:

$$\langle f|g\rangle|_a^b = \int_a^b f(x)g(x)dx. \tag{5.14}$$

la distancia entre f y g se puede ver como $d(f, g) = \|f - g\|$.

Si la norma de la función f está relacionada con el producto interior, por medio de:

$$\|f\| = \langle f|f\rangle^{\frac{1}{2}}. \tag{5.15}$$

entonces:

$$\|f - g\| = \langle f - g|f - g\rangle^{\frac{1}{2}} = \sqrt{\int_a^b (f - g)(f - g)dx} = \sqrt{\int_a^b (f(x) - g(x))^2 dx}. \tag{5.16}$$

Extendiendo a campos vectoriales $\mathbf{F}, \mathbf{G} : \mathbb{R}^m \to \mathbb{R}^n$, podemos definir una distancia entre campos vectoriales sobre una región del espacio $\Omega \in \mathbb{R}^m$ como:

$$\|\mathbf{F} - \mathbf{G}\| = \langle \mathbf{F} - \mathbf{G} | \mathbf{F} - \mathbf{G} \rangle^{\frac{1}{2}} = \sqrt{\int_{\Omega} (\mathbf{F} - \mathbf{G})^T (\mathbf{F} - \mathbf{G}) d\Omega}. \tag{5.17}$$

Para las CPPN utilizadas en estos experimentos, se tiene que la región Ω se trata precisamente del espacio de diseño:

$$\Omega = \begin{cases} x & \in [a_1, b_1]. \\ y & \in [a_2, b_2]. \\ z & \in [a_3, b_3]. \end{cases} \tag{5.18}$$

Para un espacio de diseño con una resolución de $R = X \times Y \times Z$, se tiene que:

$$\Delta\Omega_i = \frac{b_1 - a_1}{X} \frac{b_2 - a_2}{Y} \frac{b_3 - a_3}{Z}. \tag{5.19}$$

Con lo cual, la distancia definida anteriormente, en su versión discreta queda:

$$d(\mathbf{F}, \mathbf{G}) \approx \sqrt{\sum_{i=0}^{R-1} (\mathbf{F}(x_i, y_i, z_i) - \mathbf{G}(x_i, y_i, z_i))^T (\mathbf{F}(x_i, y_i, z_i) - \mathbf{G}(x_i, y_i, z_i)) \Delta\Omega_i},$$
$$(x_i, y_i, z_i) \in \Omega. \tag{5.20}$$

5.1.3.1. Diversidad genética

Dada la Ecuación 5.20, tomamos las redes neuronales de morfología y control como la función de la expresión de genes $\vec{g}_s$ de un softbot s, tal que $\vec{g}_s : \mathbb{R}^3 \to \mathbb{R}^4$. Podemos decir que la diversidad genética de s, con respecto a la población en la generación n ($P_n \subset \mathbb{S}$), sustituyendo 5.20 en 3.9, es:

$$D_g(s) = \frac{1}{|P_n|} \sum_{s' \in P_n} d(\vec{g}_s, \vec{g}_{s'}),$$
$$D_g : \mathbb{S}^2 \to \mathbb{R}^{(+)}. \tag{5.21}$$

5.1.3.2. Diversidad genética de control

Si ahora solo tomamos la red neuronal de control como la función de expresión de genes para el controlador de un softbot s, $\vec{c}_s : \mathbb{R}^3 \to \mathbb{R}^2$, podemos hacer lo mismo que para la Ecuación 5.21:

$$D_{gc}(s) = \frac{1}{|P_n|} \sum_{s' \in P_n} d(\vec{c}_s, \vec{c}_{s'}),$$
$$D_{gc} : \mathbb{S}^2 \to \mathbb{R}^{(+)}. \tag{5.22}$$

5.1.3.3. Diversidad genética de morfología

Si esta vez hacemos lo mismo pero ahora tomamos la red neuronal de morfología de un softbot s, tendremos que:

$$\begin{aligned} D_{gm}(s) &= \frac{1}{|P_n|} \sum_{s' \in P_n} d(\vec{m}_s, \vec{m}_{s'}), \\ D_{gm} &: \mathbb{S}^2 \to \mathbb{R}^{(+)}. \end{aligned} \tag{5.23}$$

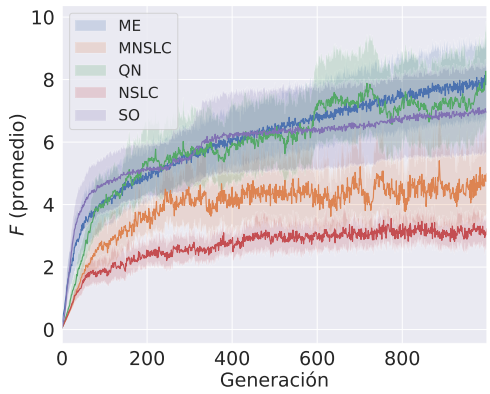
5.2. Resultados

En esta sección se presentan los resultados de los experimentos en forma de gráficas de estadística descriptiva de los distintos indicadores descritos en la Sección 5.1. Adicionalmente, se presentan otros resultados interesantes, ya sea de otros indicadores y/o de otras visualizaciones.

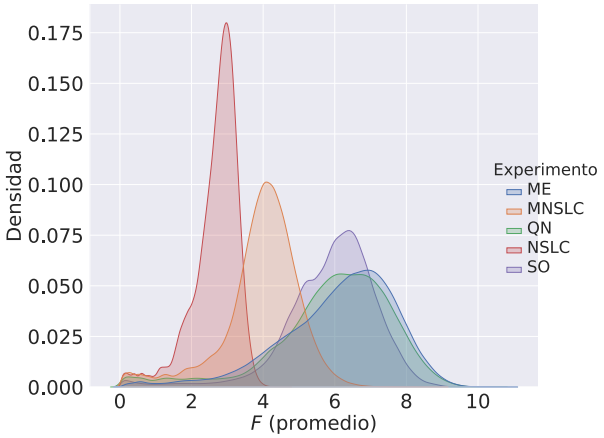
5.2.1. Resultados de indicadores de desempeño de la tarea

En esta sección se presentan gráficas de convergencia, box-plots y densidades aproximadas a través KDE, para cada uno de los indicadores mencionados en la Sección 5.1. Para las gráficas de convergencia y KDE se utilizó el método de remuestreo conocido como bootstrapping, el cual consiste en tomar muestras aleatoriamente con reemplazo. En este caso, se tomaron 1000 muestras por generación de entre las 20 ejecuciones. En las gráficas de convergencia se muestra el promedio de las ejecuciones en colores opacos, y el intervalo de confianza $ci = \pm 95\,\%$, como franjas transparentes.

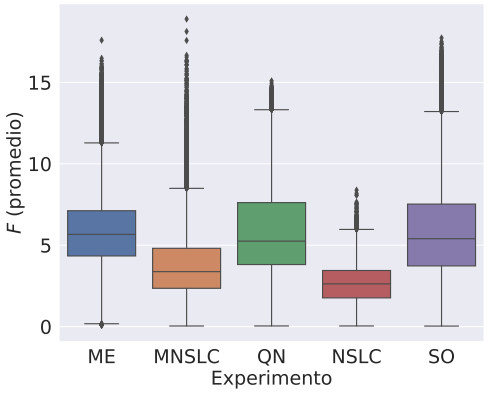
5.2.1.1. Fitness F



(a) Convergencia de F.



(b) KDE de F.



(c) BoxPlots de F.

Figura 5.1: Fitness F.

En la Figura 5.1a se puede observar que NSLC posee un crecimiento bastante lento en cuanto a fitness, algo que suele ser característico de los algoritmos QD con novedad definida en un BC desalineado, tal y como se menciona en [64].

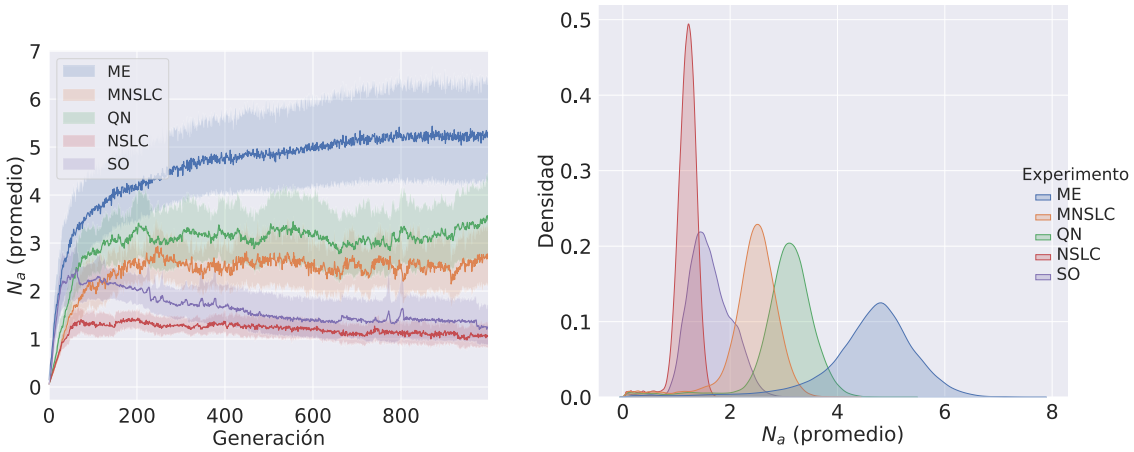
MNSLC claramente presenta un crecimiento mayor que NSLC, y también un tanto más ruidoso, algo que se evidencia tanto al examinar visualmente la Figura 5.1a, como al examinar la gran cantidad de outliers presentes en la Figura 5.1c. Más aún, parece ser que el BC presuntamente alineado, lo es en efecto, dado este crecimiento mayor al comparar ambos algoritmos. Tanto ME, QN y SO, tienen un patrón de convergencia similar en este indicador, con ME presentando un rendimiento un poco superior (en

vista de la Figura 5.1a, la mediana en la Figura 5.1c, y la densidad estimada en la Figura 5.1b, sesgada un poco más a la derecha), seguido de QN y SO.

SO parece tener un comportamiento un poco más estable que QN, mientras que QN parece tener un crecimiento marginalmente más rápido que SO (con base en la Figura 5.1a y la Figura 5.1b, la Figura 5.1c muestra medianas casi iguales para ambos).

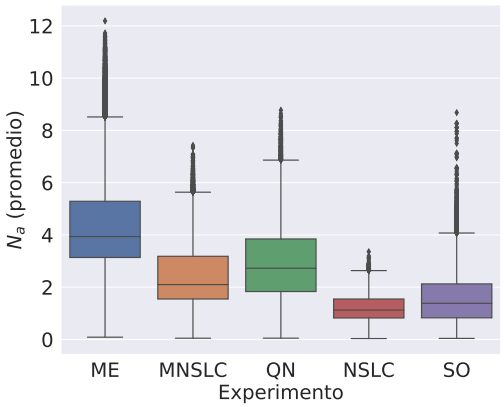
Ahora bien, más adelante, se espera que MNSLC se desempeñe al menos tan bien como NSLC en los indicadores donde se espera que un algoritmo QD presente mejores resultados.

5.2.1.2. Novedad alineada N_a



(a) Convergencia de N_a.

(b) KDE de N_a.



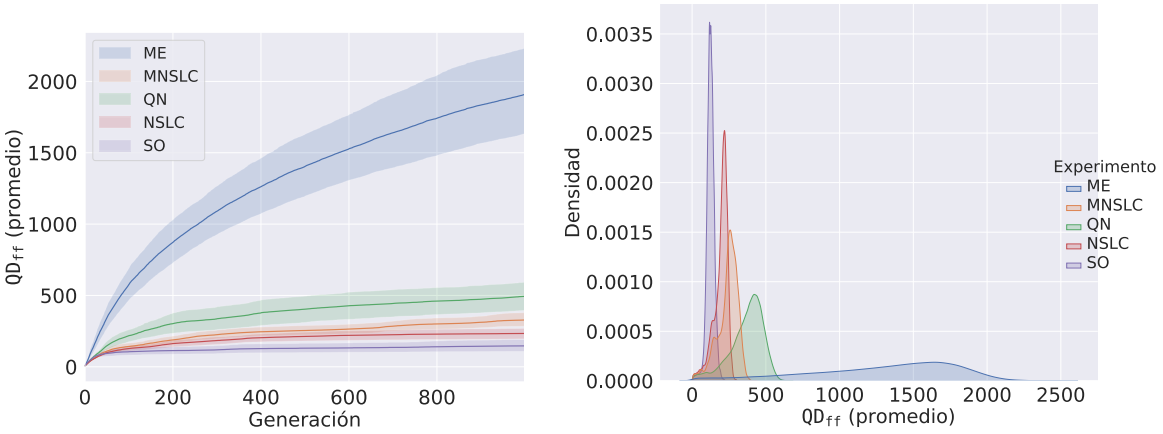
(c) BoxPlots de N_a.

Figura 5.2: Novedad alineada N_a.

En las Figuras 5.2a, 5.2b y 5.2c es interesante ver que ME y QN, dos experimentos que utilizan algoritmos con una presión de selección orientada a la aptitud global, superen a otro, que explícitamente maximiza la novedad alineada, como uno de sus objetivos (MNSLC). Este hecho confirma en parte que el BC propuesto como alineado, efectivamente lo está, pero también se podría tener una relación causal parcial con la diversidad morfológica, puesto que si sólo estuviese relacionado con la aptitud, al examinar las Figuras 5.2a y 5.2c, SO tendría un comportamiento similar a ME y QN, cosa que no sucede, tiene más cercanía con NSLC.

Pareciera que hay una parte de la novedad alineada que se debe al fitness: si una solución logra recorrer una distancia mucho mayor que cualquiera de las anteriores, aunque sea en la misma trayectoria que todas las demás, tendrá una alta novedad alineada. Mientras que hay otra parte de la novedad alineada que se podría deber a la novedad morfológica, si tomamos en cuenta que cuando se busca novedad morfológica, sucede lo descrito en [4], es decir, que cambios en la morfología sobre controladores optimizados, equivalen a cambios en el ambiente, y que estos cambios llevan a comportamientos erráticos, entonces se tendrán cambios en las trayectorias de los softbots, que podrían llevar a comportamientos con una novedad alineada más alta, pero fitness bajo.

5.2.1.3. Indicador $\texttt{QD}_{\texttt{ff}}$



(a) Convergencia de $\texttt{QD}_{\texttt{ff}}$.

(b) KDE de $\texttt{QD}_{\texttt{ff}}$.

(c) BoxPlots de $\texttt{QD}_{\texttt{ff}}$.

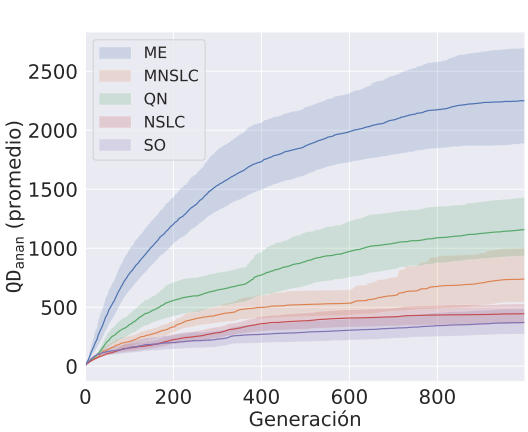
Figura 5.3: Indicador $\texttt{QD}_{\texttt{ff}}$.

Es de esperarse que ME sea el experimento con mejores resultados (Figuras 5.3a, 5.3b y 5.3c) en un indicador que mide la calidad total alcanzada en un grid sobre el BC desalineado y considerando que el algoritmo utilizado es MAP-Elites sobre ese mismo grid, con una presión de selección orientada a la calidad, es decir, buscando maximizar precisamente este $\texttt{QD}_{\texttt{ff}}$.

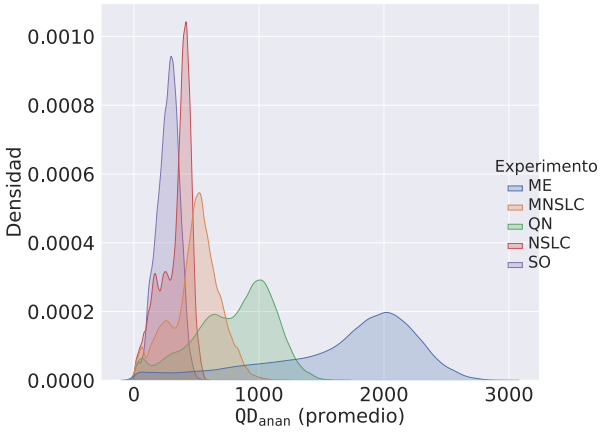
Ahora bien, cabe señalar que en los otros cuatro experimentos se tiene un ordenamiento similar al observado en las Figuras 5.2, solo que en este caso, SO obtuvo los peores resultados, probablemente no por falta de calidad en las soluciones, sino por falta de

exploración del BC desalineado, mientras que NSLC fue el segundo peor, pero más bien por falta de calidad de las soluciones, no por falta de exploración. MNSLC pareciera mitigar esa falta de calidad de las soluciones de NSLC, pero sin lograr encontrar una calidad tan alta como QN, que presenta mejor capacidad explotativa.

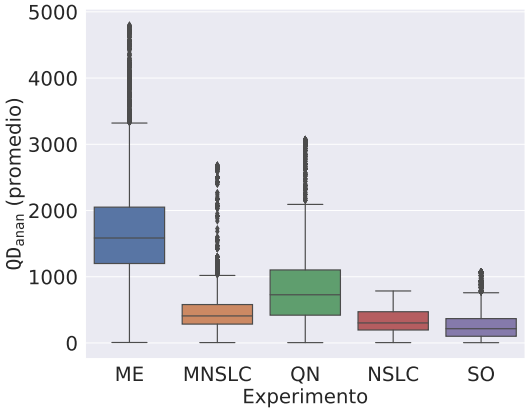
5.2.1.4. Indicador $\texttt{QD}_{\texttt{anan}}$



(a) Convergencia de $\texttt{QD}_{\texttt{anan}}$.



(b) KDE de $\texttt{QD}_{\texttt{anan}}$.



(c) BoxPlots de $\texttt{QD}_{\texttt{anan}}$.

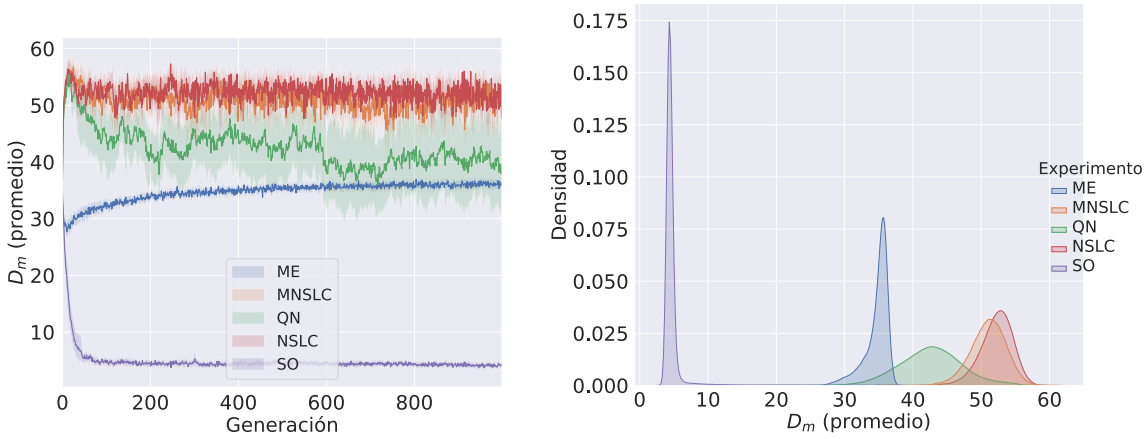
Figura 5.4: Indicador $\texttt{QD}_{\texttt{anan}}$.

Para este indicador, se pueden señalar algunos puntos similares a los observados en N_a y $\texttt{QD}_{\texttt{ff}}$. Se vuelve a evidenciar el alto grado de alineación de N_a, primero porque ME es el experimento con mejores resultados (Figuras 5.4a, 5.4b y 5.4c), maximizando $\texttt{QD}_{\texttt{ff}}$ y no $\texttt{QD}_{\texttt{anan}}$, y segundo porque los cuatro experimentos restantes tienen el mismo

ordenamiento que para el indicador anterior. Lo anterior también señala al sesgo hacia la explotación de SO, el sesgo a la exploración de NSLC la mitigación de MNSLC sobre el sesgo de NSLC, y tanto en QN como ME, los efectos de una explotación a nivel global sobre la diversidad descubierta. Además, tal y como se muestra en las Figuras 5.4, la diferencia en escalas (Figura 5.4b y 5.4c)) y el crecimiento mayor (Figura 5.4a) que presentan especialmente: QN y MNSLC, y en menor medida: NSLC y SO, respecto a lo mostrado en las Figuras 5.3, hace evidente, al menos hasta cierto punto, que la novedad alineada considera, no solo la distancia recorrida, sino la trayectoria, y por lo tanto, indirectamente la diversidad morfológica.

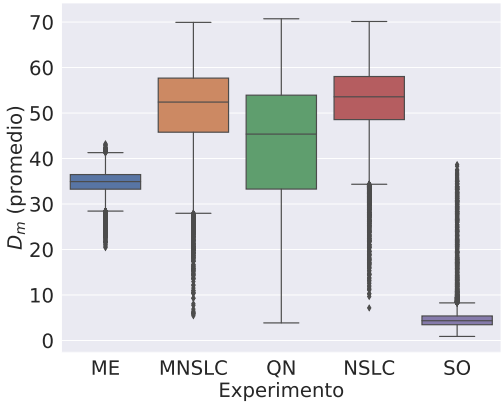
5.2.2. Resultados de indicadores de diversidad fenotípica

5.2.2.1. Diversidad de morfología D_m



(a) Convergencia de D_m.

(b) KDE de D_m.



(c) BoxPlots de D_m.

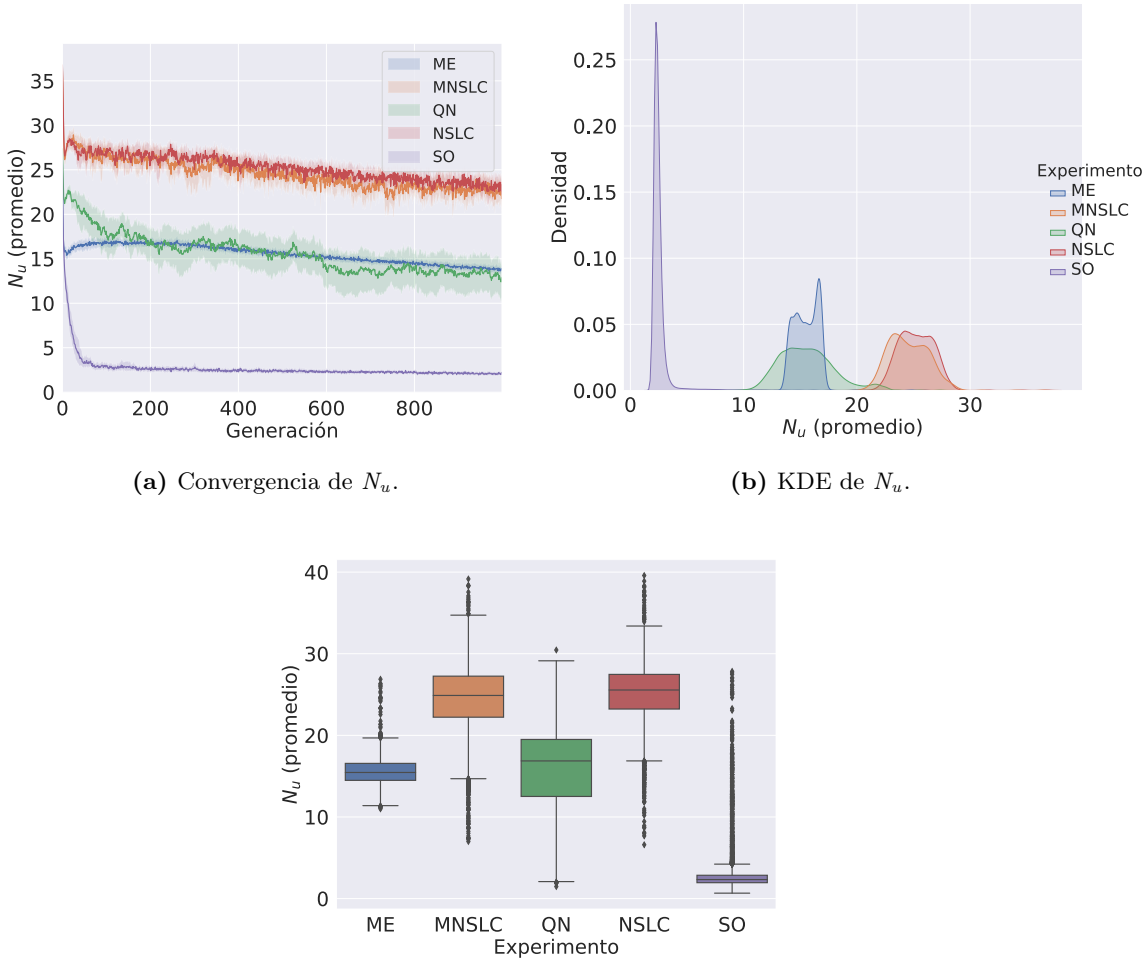
Figura 5.5: Diversidad de morfología D_m.

Con este indicador son bastante claros y contrastantes los efectos de la pérdida de la diversidad morfológica cuando no se tiene ningún tipo de protección a la diversidad explícito (SO), respecto a cuando si se tiene (QN, ME, MNSLC, NSLC). En la Figura 5.5c se puede ver claramente como SO presenta la menor diversidad morfológica y se separa dramáticamente de los demás experimentos. A grandes rasgos, los algoritmos que presentan menor pérdida de diversidad morfológica, son los puramente QD (NSLC y MNSLC), seguidos de QN, el cual integra a la calidad global en lugar de la local y a

la novedad desalineada (o morfológica) en un marco de trabajo multiobjetivo. Después se tiene a ME, que no presenta ningún objetivo de novedad desalineada, ni presión de selección explícita hacia ésta, y más bien encuentra diversidad morfológica por medio de un grid definido en el BC desalineado y de maximizar $\texttt{QD}_{\texttt{ff}}$. En ME, se hace evidente la presión de selección de padres hacia la aptitud global, con la pérdida de diversidad a nivel población, que ésta representa respecto a sus contrapartes QD "puros" (NSLC y MNSLC). En la Figura 5.5a se puede observar lo siguiente:

- En SO, se observa un comportamiento decreciente que se estabiliza en valores bajos.
- En ME, se observa que en las primeras generaciones se da un comportamiento decreciente, posiblemente porque en esas etapas tempranas de la optimización, aún no se ha explorado gran parte del BC de morfologías, sin embargo, no pasan muchas generaciones antes de que se detenga el comportamiento decreciente y se pase a un comportamiento donde se incrementa lentamente la diversidad de la morfología, hasta estabilizarse en un valor de 35.
- En todos los algoritmos que utilizan NSGA-II, y explícitamente un objetivo de N_u se puede observar un comportamiento de crecimiento muy rápido.
- En QN, se puede distinguir mayor varianza, y un comportamiento decreciente justo después de ese periodo inicial de rápido crecimiento. Este comportamiento es más acentuado que en NSLC y MNSLC, al punto de casi llegar al mismo valor en el que se estabiliza ME.
- Para el caso de los algoritmos QD en este rubro (NSLC y MNSLC), se observa un comportamiento ligeramente decreciente, que se detiene casi inmediatamente, estabilizándose rápidamente en un rango de valores.

5.2.2.2. Novedad desalineada N_u



(a) Convergencia de N_u.

(b) KDE de N_u.

(c) BoxPlots de N_u.

Figura 5.6: Novedad desalineada N_u.

Con N_u se tiene una historia similar a la de D_m. Tal y como se muestra en la Figura 5.6a, MNSLC y NSLC tienen la mayor novedad desalineada, con un decremento acentuado en las primeras generaciones, seguido de una estabilización en un decremento bastante lento, de carácter mas o menos lineal. Es interesante observar que ME y QN convergen a valores similares, con ME quedándose con valores ligeramente mayores al final de las corridas evolutivas, especialmente si se considera que QN presenta explícitamente el objetivo N_u, evidenciando el efecto de utilizar una calidad global en contraste con su versión local.

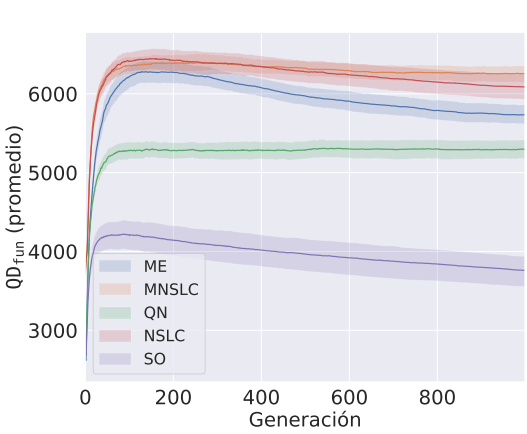
Se puede observar en las Figuras 5.6b y 5.6c que QN también tiene mayor varianza, evidenciando los objetivos en conflicto de calidad global y novedad desalineada.

En la Figura 5.6a ME presenta una pérdida de novedad más acentuada que los tres experimentos anteriormente mencionados, evidenciando de nuevo esa presión de selección orientada a la calidad global, en contraste con sus contrapartes QD "puros", sin embargo vuelve a presentar aquel crecimiento lento inmediatamente después, que se asemeja a una gráfica logarítmica, para posteriormente presentar un decremento más o menos lineal.

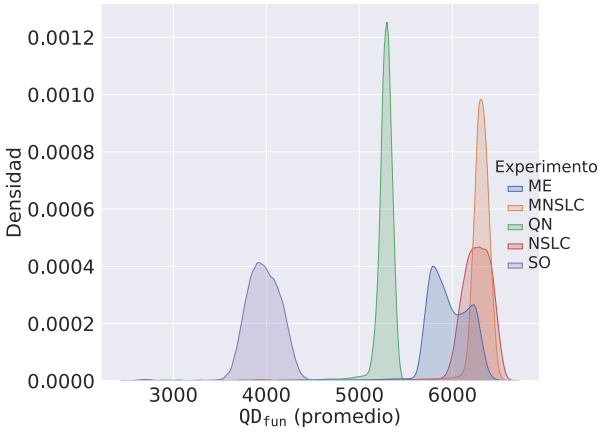
En la Figura 5.6c se puede ver que si bien ME presenta una mediana de novedad un poco menor que QN, también presenta un comportamiento menos ruidoso, y por lo tanto una varianza mucho menor.

Finalmente, SO presenta un comportamiento similar al que tuvo en D_m (Figura 5.6c), de nuevo se tienen outliers con valores bastante altos, seguramente debidos a mutaciones que derivaron en individuos eliminados por el operador de selección puramente explotativo, o bien porque en las primeras generaciones se tienen individuos más novedosos.

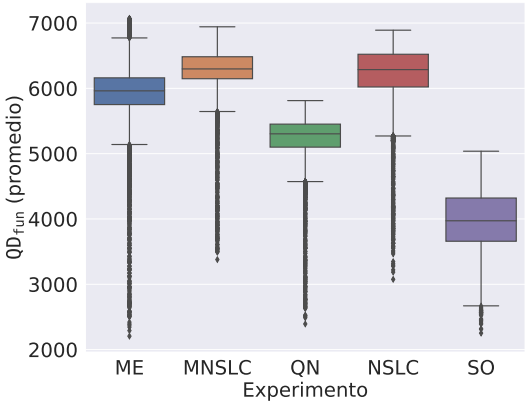
5.2.2.3. Indicador $\texttt{QD}_{\texttt{fun}}$



(a) Convergencia de $\texttt{QD}_{\texttt{fun}}$.



(b) KDE de $\texttt{QD}_{\texttt{fun}}$.



(c) BoxPlots de $\texttt{QD}_{\texttt{fun}}$.

Figura 5.7: Indicador $\texttt{QD}_{\texttt{fun}}$.

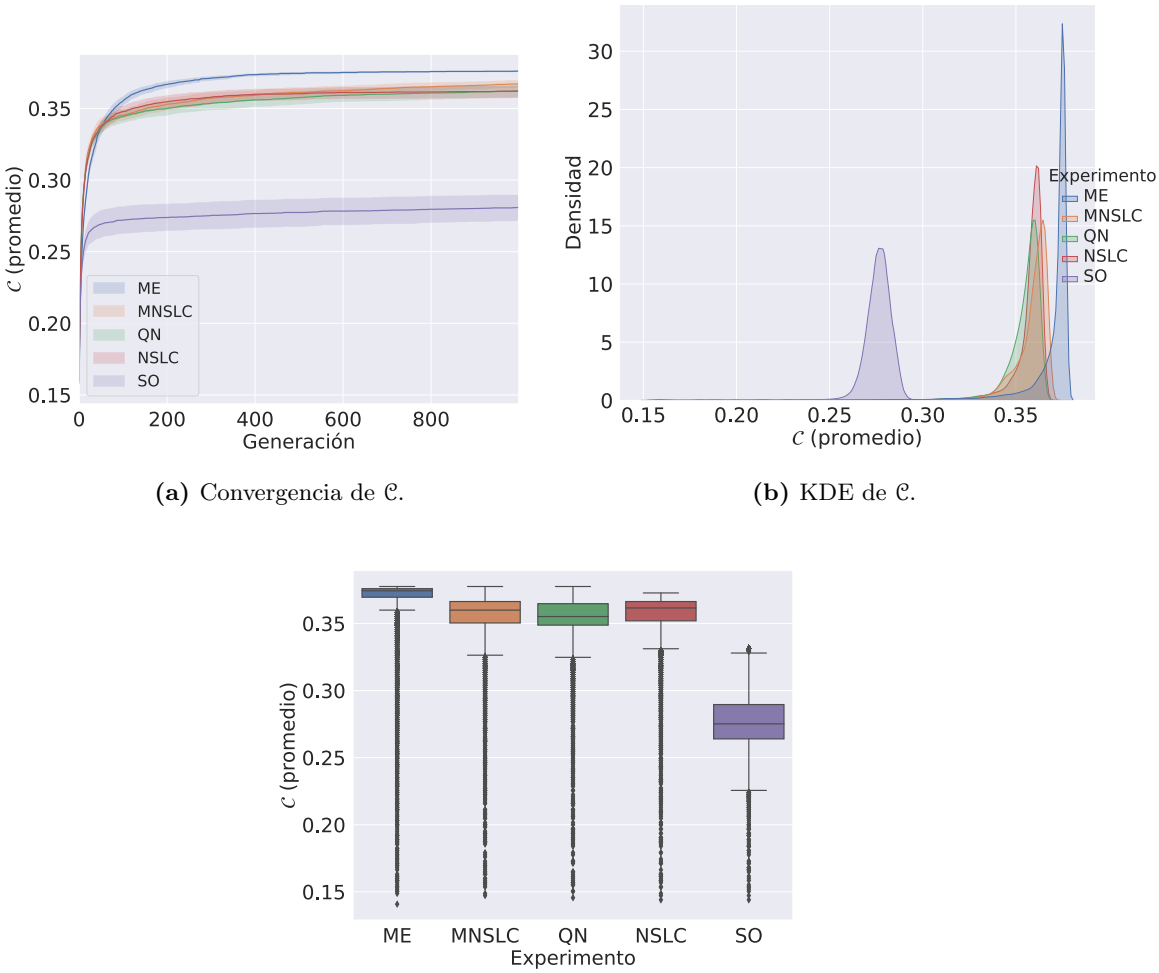
Al igual que en los otros indicadores QD que se examinaron anteriormente y como se puede observar en la Figura 5.7a, se tiene un crecimiento similar a una función logarítmica, sin embargo parece ser que cuando se comienza a reducir la rapidez con la que sucede el descubrimiento de nuevas áreas del espacio BC desalineado, se comienza perder la novedad morfológica de las élites en el grid.

Por su parte, la cercanía del comportamiento de ME con NSLC y MNSLC (Figuras 5.7a, 5.7b y 5.7c) se puede explicar con que el los indicadores QD son relativos al grid, y ME evoluciona un grid y no una población, por lo que al mantener soluciones

distribuidas en el BC de morfología, está también buscando implícitamente diversidad y por lo tanto novedad, finalmente situándolo en un comportamiento más cercano a los algoritmos QD "puros" que a QN.

También examinando el comportamiento de SO en este indicador (Figuras 5.7a, 5.7b y 5.7c), se puede observar que inclusive tratándose de un experimento sin ningún tipo de protección a la diversidad, al utilizar un grid para guardar las soluciones encontradas, se encuentra cierta novedad, derivando en un comportamiento relativo a los otros experimentos, que es visiblemente mejor para SO, que el que se observó en la novedad desalineada poblacional N_u.

5.2.2.4. Cobertura $\mathcal{C}$



(a) Convergencia de $\mathcal{C}$.

(b) KDE de $\mathcal{C}$.

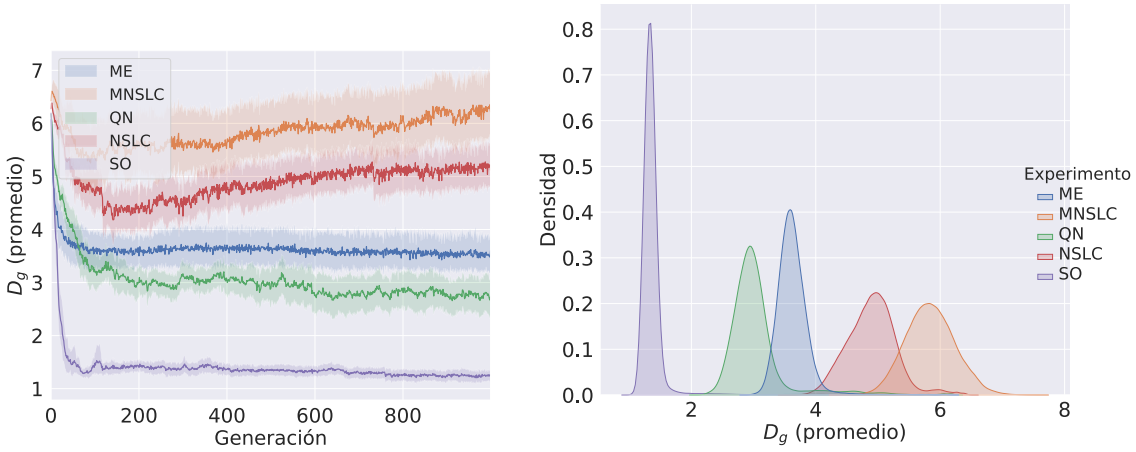
(c) BoxPlots de $\mathcal{C}$.

Figura 5.8: Cobertura $\mathcal{C}$.

Tal y como se puede observar en las Figuras 5.8a, 5.8b y 5.8c, inclusive empleando una probabilidad de selección proporcional a la calidad global, ME presenta una cobertura superior a MNSLC, NSLC y QN. Una selección estocástica de bins en un grid, aún sesgada de manera explotativa, encuentra más regiones del espacio de características que la búsqueda de novedad. Lo anterior seguramente sería más notorio si la selección fuera proporcional, es decir si se hubiese utilizado MAP-Elites puro.

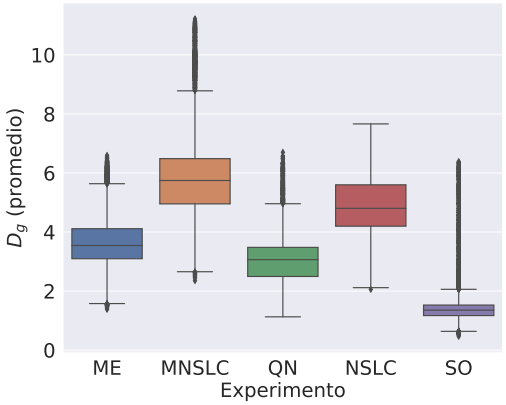
5.2.3. Resultados de indicadores de diversidad genética

5.2.3.1. Diversidad genética D_g



(a) Convergencia de D_g.

(b) KDE de D_g.



(c) BoxPlots de D_g.

Figura 5.9: Diversidad genética D_g.

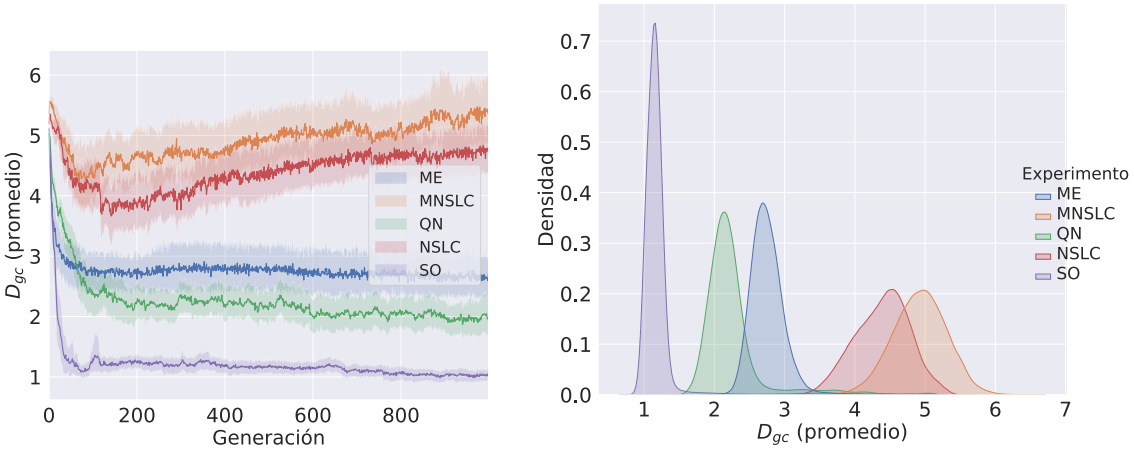
Al observar el comportamiento de este indicador en las Figuras 5.9a, 5.9b, 5.9c y el de los demás indicadores de diversidad, parece haber cierta relación entre la diversidad genética como se define en esta tesis, y la diversidad del fenotipo. En cuanto a este indicador, cada uno de los experimentos presenta un comportamiento característico y fácilmente distinguible a simple vista. Como se puede observar en la Figura 5.9a, ningún experimento se encuentra algún tipo de crecimiento repentino al principio de las corridas evolutivas, es decir, en todos los experimentos hay pérdida de diversidad

genética en mayor o menor medida, un fenómeno que es prevalente en la literatura. Respecto a la Figura 5.9a, también se puede observar que:

- Comenzando con SO, se encuentra un comportamiento de rápida pérdida de diversidad genética, muy alineado a todos los indicadores de diversidad fenotípica poblacional anteriormente mencionados (N_u y D_m).
- QN muestra la segunda mayor pérdida de diversidad genética, mientras que ME muestra la tercera, probablemente por el uso del grid en el espacio de características morfológicas, y de manera bastante alineada a los comportamientos encontrados en los indicadores de diversidad fenotípica poblacional. De nuevo, parece ser que la introducción de la presión de selección proporcional a la calidad global, separa a ME de los demás algoritmos QD.
- Para el caso de NSLC y MNSLC, se tiene un comportamiento ligeramente distinto, donde uno supera al otro por un margen bastante más amplio. En este caso, tenemos a MNSLC superando a NSLC, que quizás se pueda explicar porque la búsqueda de novedad alineada lleva a mayor diversidad en los controladores. Además, se puede observar un comportamiento ligeramente creciente, posterior a la pérdida de diversidad inicial.

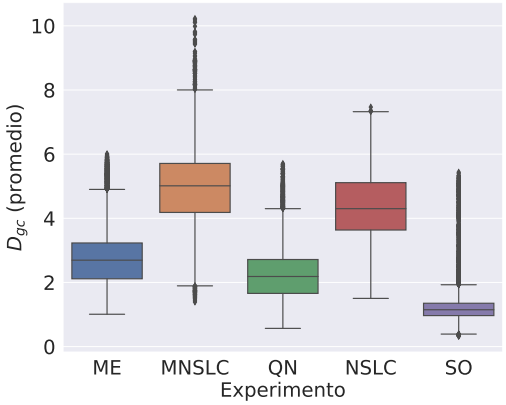
Es interesante que los dos experimentos que presentan competencia local (o bien calidad local), y más aún, que los algoritmos QD (ME, NSLC y MNSLC), sean los que presentan la mayor diversidad genética. Pareciera que NSLC y MNSLC, los cuales representan más puramente las características de los algoritmos QD, que son la división en nichos (no estructurados en este caso) del espacio de características y la noción de calidad local, son los que presentan mayor diversidad genética, mientras que ME, al introducir una presión de selección proporcional a la calidad global, y no dar una probabilidad uniforme a todos los nichos, lo hace perder diversidad genética, tal y como sucedió con N_u y D_m.

5.2.3.2. Diversidad genética de control D_{gc}



(a) Convergencia de D_{gc}.

(b) KDE de D_{gc}.



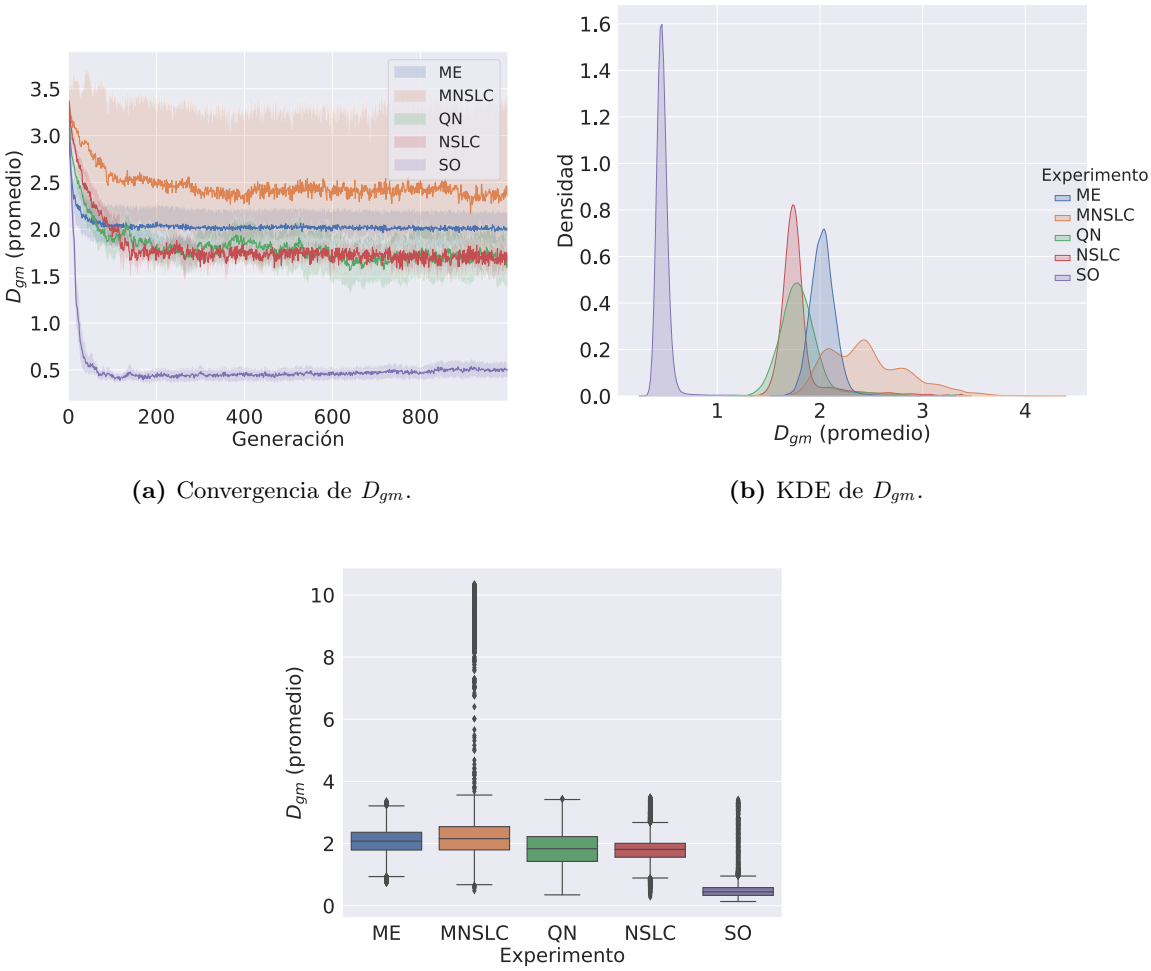
(c) BoxPlots de D_{gc}.

Figura 5.10: Diversidad genética de control D_{gc}.

Tal y como se evidencia en las Figuras 5.10a, 5.10b y 5.10c, este indicador presenta el mismo ordenamiento que D_g. En este caso, se puede observar que la novedad alineada podría ser provocada (al menos parcialmente) por una mayor diversidad en el comportamiento, y a su vez, por una mayor diversidad en los controladores. Más aún, los dos experimentos que presentan competencia local, son los que presentan mayor diversidad en los controladores, si observamos el comportamiento de QN en la Figura 5.10a, el cual también presenta novedad de morfología como segundo objetivo, se podría argumentar que buscar maximizar el score de competencia local, incrementa la diversidad genética de los controladores. Podríamos ir un paso más allá, y al observar ME, argumentar

que los algoritmos QD presentan mayor diversidad genética de control, pero de nuevo, al introducir la presión de selección proporcional a la calidad global, dicha diversidad genética del control se erosiona.

5.2.3.3. Diversidad genética de morfología D_{gm}



(a) Convergencia de D_{gm}.

(b) KDE de D_{gm}.

(c) BoxPlots de D_{gm}.

Figura 5.11: Diversidad genética de morfología D_{gm}.

En la Figura 5.11a, SO muestra un comportamiento decreciente mucho más rápido que sus contrapartes con protección a la diversidad. En esa misma Figura, es interesante ver que los comportamientos de NSLC y QN son tan similares. Pareciera ser, que la diversidad genética morfológica ignora completamente la competencia local, a diferencia de D_{gc}. Por su parte, MNSLC, cuya única diferencia con NSLC es la introducción

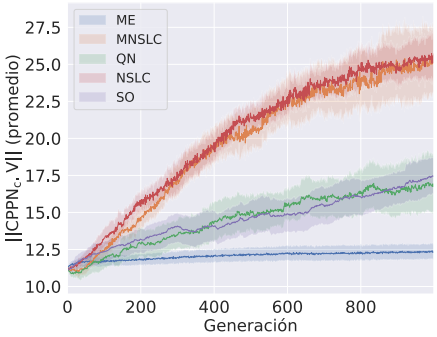
de un tercer objetivo correspondiente a la novedad alineada, posee la mayor D_{gm}, que pudiese deberse a que la novedad alineada en parte es provocada por cambios en trayectorias asociados a cambios en morfologías. Las Figuras 5.11b y 5.11c reflejan el mismo ordenamiento discutido para 5.11a. MNSLC presenta una gran cantidad de outliers visibles en la Figura 5.11c, que a su vez se ven reflejados en la forma de la KDE con bootstrapping (Figura 5.11b), con una forma alargada y sin llegar a formar la característica distribución normal que se obtiene al aplicar el muestreo en los demás experimentos.

5.2.4. Resultados de otros indicadores y otras visualizaciones

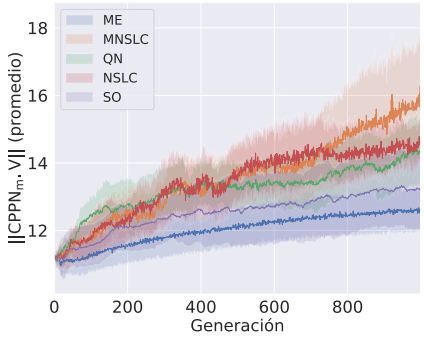
En esta sección se presentan otros indicadores y visualizaciones, además de los presentados anteriormente. Para el caso de los indicadores extra, pese a no ser relevantes para la elección del algoritmo ganador, si representan resultados interesantes. Además se muestran visualizaciones auxiliares para algunos de los indicadores principales.

5.2.4.1. Complejidad de las redes neuronales

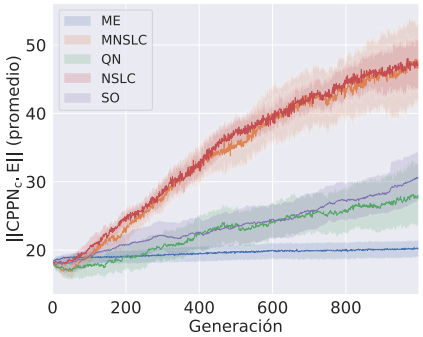
A continuación se muestran tres indicadores muy simples, pero representativos, de la complejidad de las redes neuronales de control y morfología, para cada uno de los experimentos. Estos tres indicadores son el número de nodos (Figuras 5.12a y 5.12b), aristas (Figuras 5.12c y 5.12d) y la suma de pesos total (Figuras 5.12e y 5.12f).



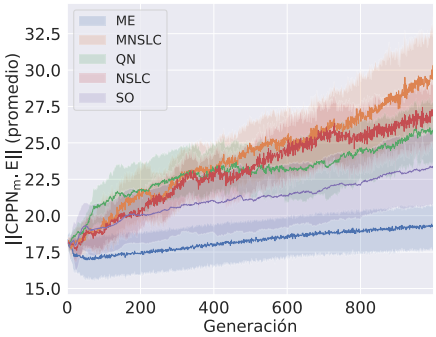
(a) Convergencia de nodos, CPPN de control.



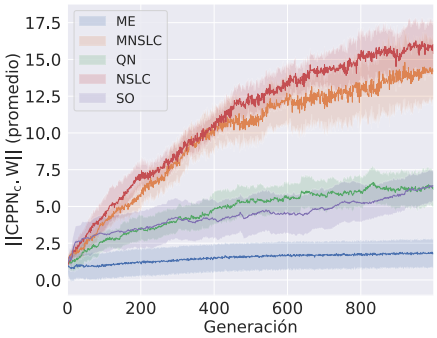
(b) Convergencia de nodos, CPPN de morfología.



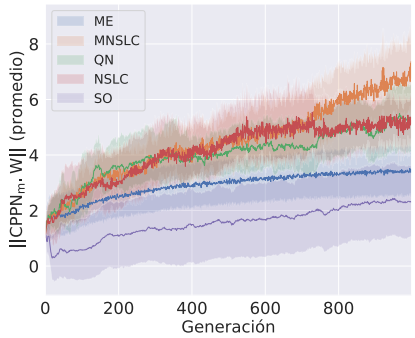
(c) Convergencia de aristas, CPPN de control.



(d) Convergencia de aristas, CPPN de morfología.



(e) Convergencia de suma de pesos, CPPN de control.



(f) Convergencia de suma de pesos, CPPN de morfología.

Figura 5.12: Complejidad de las CPPN.

Es muy interesante que el experimento que mantuvo las redes neuronales tanto de morfología, como de control, con menor complejidad, fue ME, mientras que MNSLC

y NSLC fueron consistentemente los que obtuvieron las redes neuronales de mayor tamaño y suma de pesos. Algo que podría ser deseable, es encontrar soluciones diversas y de alta calidad, con los genotipos lo menos complejos posibles, lo cual parece ser que ME logra.

5.2.4.2. El espacio de comportamiento

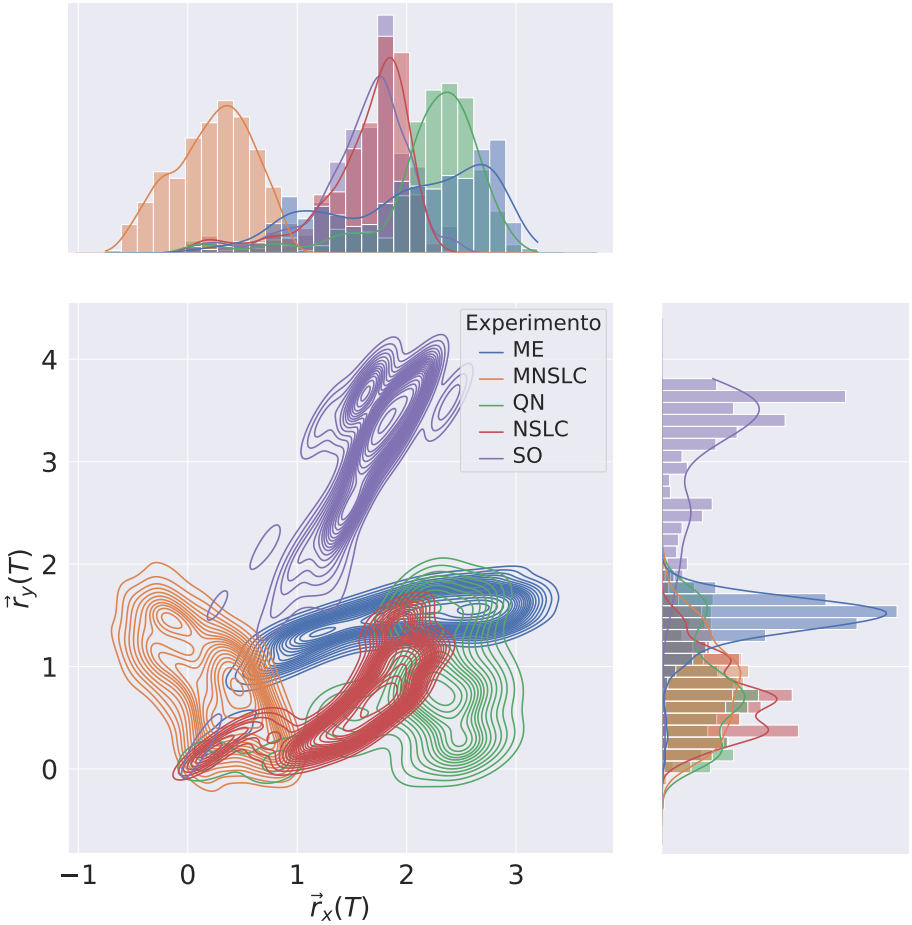


Figura 5.13: KDE conjunta de espacio de comportamiento.

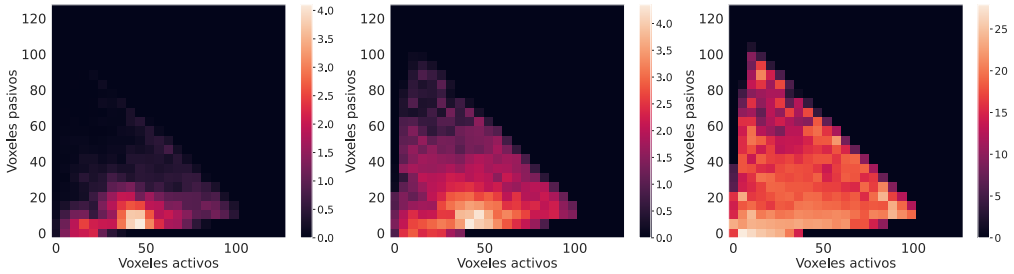
La Figura 5.13 muestra las curvas de contorno de la distribución conjunta de las posiciones de los softbots al final de las simulaciones, en el plano XY. Es una forma de examinar visualmente los efectos de la novedad alineada y el fitness.

- Por ejemplo, para SO, se hace evidente por qué la novedad alineada es tan baja, pero el fitness es alto, y es por la naturaleza explotativa del experimento, que lo lleva únicamente a encontrar una familia muy reducida de trayectorias.

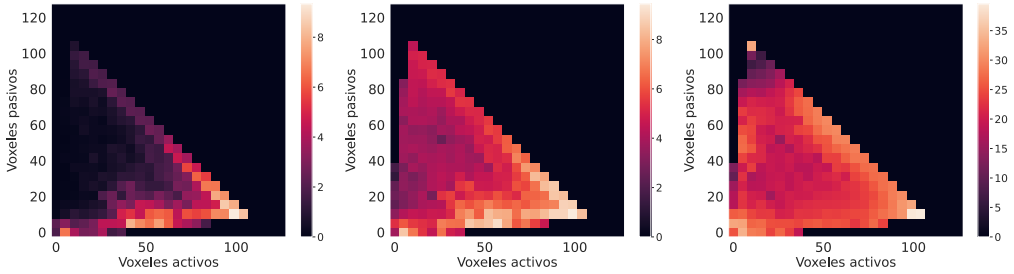
- El caso de ME, hace evidente el resultado alto en fitness y alto en novedad alineada, dado que encontró soluciones de calidad a lo largo de distintas trayectorias.
- QN presenta una historia similar a ME.
- MNSLC presenta una gama de trayectorias, pero cerca del origen, lo cual pone en evidencia un fitness más bajo que SO, ME y QN, y novedad alineada menor que ME y QN.
- NSLC presenta una historia similar a MNSLC, en cuanto a soluciones más cercanas al origen, y también hace patente una menor novedad alineada que MNSLC, al tener una menor gama de trayectorias.

5.2.4.3. El grid de fitness (G_f)

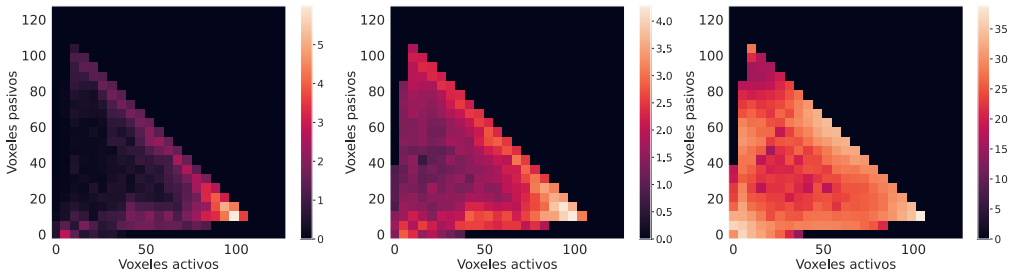
En las Figuras 5.14 y 5.15 se muestran las visualizaciones correspondientes al grid, o mapa de características del fitness resultante de cada experimento (renglones). El eje horizontal corresponde a los voxeles activos, mientras el vertical a los pasivos. La primera columna muestra el promedio de todas las corridas del fitness máximo encontrado para cada bin, la segunda columna muestra la novedad alineada promedio de las soluciones guardadas en cada bin, y finalmente la tercera columna muestra la novedad desalineada promedio de las soluciones guardadas en cada bin. Con este grid, en cada generación de cada algoritmo se calculó $\texttt{QD}_{\texttt{ff}}$ y $\texttt{QD}_{\texttt{fun}}$.



(a) Promedio de F (SO). **(b)** Promedio de N_a (SO). **(c)** Promedio de N_u (SO).

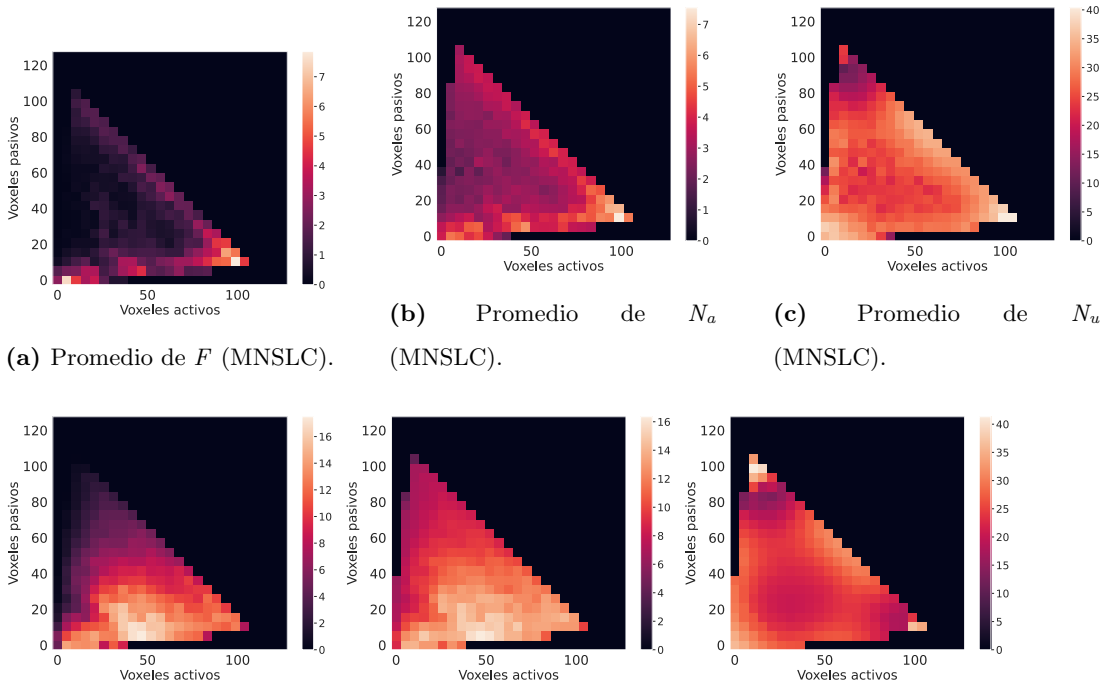


(d) Promedio de F (QN). **(e)** Promedio de N_a (QN). **(f)** Promedio de N_u (QN).



(g) Promedio de F (NSLC). **(h)** Promedio de N_a (NSLC). **(i)** Promedio de N_u (NSLC).

Figura 5.14: Grid de Fitness (G_f).



(a) Promedio de F (MNSLC). **(b)** Promedio de N_a (MNSLC). **(c)** Promedio de N_u (MNSLC).

(d) Promedio de F (ME). **(e)** Promedio de N_a (ME). **(f)** Promedio de N_u (ME).

Figura 5.15: Grid de Fitness (G_f).

Es notorio el comportamiento puramente explotativo de SO al examinar la Figura 5.14a, donde se puede observar cómo la concentración de los valores más altos de fitness se encuentra reducida a un área muy pequeña. Si observamos la Figura 5.3, el comportamiento de SO en $\texttt{QD}_{\texttt{ff}}$ encaja con lo observado. Cuando observamos la Figura 5.14c, nos fijamos en la escala y la comparamos con la escala en los demás experimentos, podremos observar que a grandes rasgos, se alinea con lo observado en el indicador $\texttt{QD}_{\texttt{fun}}$, SO tiene los valores más bajos de novedad desalineada en el grid de fitness, sin embargo, no son tan bajos como los que presenta a nivel poblacional, evidenciando la capacidad de un grid para encontrar novedad en el BC en el que está definido. Observando la Figura 5.14b podemos encontrar en cierto sentido una mezcla entre las Figuras 5.14a y 5.14c, por un lado, observamos de nuevo la alineación al fitness al darnos cuenta que la región con mayor novedad alineada, coincide a grandes rasgos con la región de mayor fitness en la Figura 5.14a, y por el otro lado, se observa una iluminación similar hasta cierto punto a la presentada en la Figura 5.14c, reflejando quizás la relación que se tiene entre la novedad alineada y la morfológica.

Examinando el grid de QN en las Figuras 5.14d, 5.14e y 5.14f, se puede observar en primer lugar que los valores de fitness se dispersan en una región más amplia y en segundo lugar, que esta región se encuentra mayormente en los bordes, o bien limitando con

las restricciones en cuanto al espacio de diseño, respecto a la novedad desalineada, se pueden notar áreas iluminadas, que asemejan ligeramente clusters, seguramente consecuencia de ese objetivo de novedad desalineada que se tiene. Finalmente en la novedad alineada se tiene un algo similar a lo que se observó en SO, donde las regiones de mayor novedad alineada coinciden con las regiones de mayor fitness, a la vez que se tiene una mayor porción del espacio iluminado, reflejando la parte de la novedad alineada que se puede deber a la novedad desalineada.

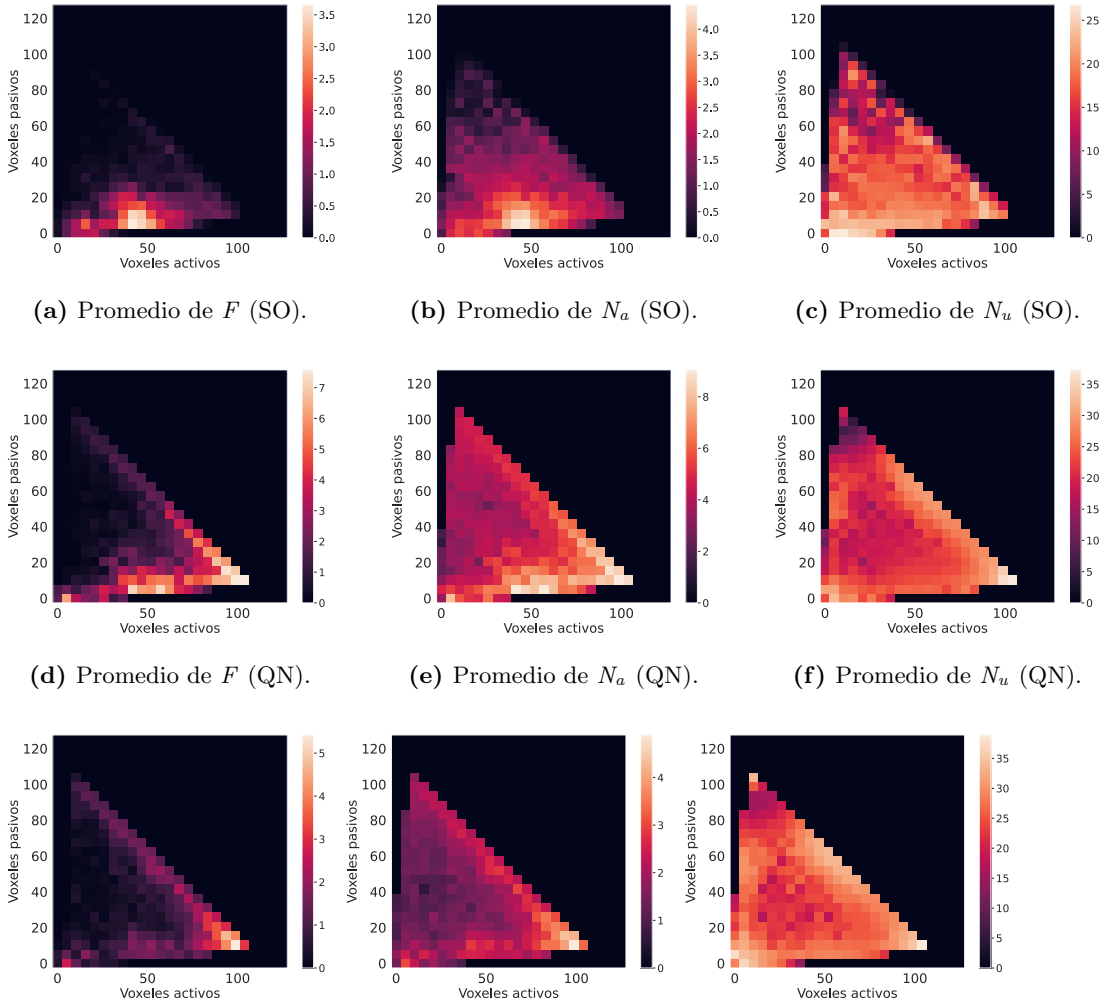
NSLC presenta una historia similar a QN, solo que en las Figuras 5.14g y 5.14h se pueden ver valores de fitness y novedad alineada más bajos con base en la escala. Así mismo, en la Figura 5.14i se puede observar más claramente la formación de los clusters debidos la búsqueda de novedad desalineada.

MNSLC por su parte presenta un comportamiento similar a NSLC, sólo que con valores de fitness y novedad alineada más altos con base en la escala, tal y como se puede ver en las Figuras 5.15a, 5.15b y 5.15c.

Mientras que para el caso de ME es evidente la capacidad de exploración y el alto nivel de calidad encontrado en cada bin, tal y como se muestra en las Figuras 5.15d, y 5.15e, ambos aspectos logrados gracias a la maximización de $\texttt{QD}_{\texttt{ff}}$. Finalmente, en la Figura 5.15f se puede notar una iluminación un tanto más "difuminada" en comparación con NSLC y MNSLC, con valores altos de novedad desalineada, pero un tanto menos frecuentes, que a grandes rasgos podría explicar lo observado en el indicador $\texttt{QD}_{\texttt{fun}}$ cuando comparamos ME con NSLC y MNSLC.

5.2.4.4. El grid de novedad alineada (G_{an})

En esta sección, se muestran en las Figuras 5.16 y 5.17, los grids, o mapas de características correspondientes a la novedad alineada, la primera columna corresponde al fitness promedio de las soluciones guardadas en cada bin, la segunda muestra el promedio de todas las corridas de la novedad alineada máxima encontrada para cada bin, y la tercera a la novedad desalineada promedio de las soluciones guardadas en cada bin. Con este grid, en cada generación de cada algoritmo se calculó $\texttt{QD}_{\texttt{anan}}$.

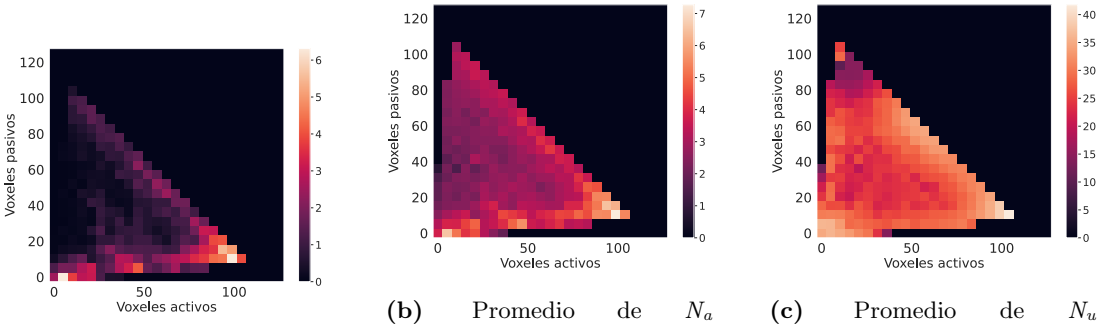


(a) Promedio de F (SO). **(b)** Promedio de N_a (SO). **(c)** Promedio de N_u (SO).

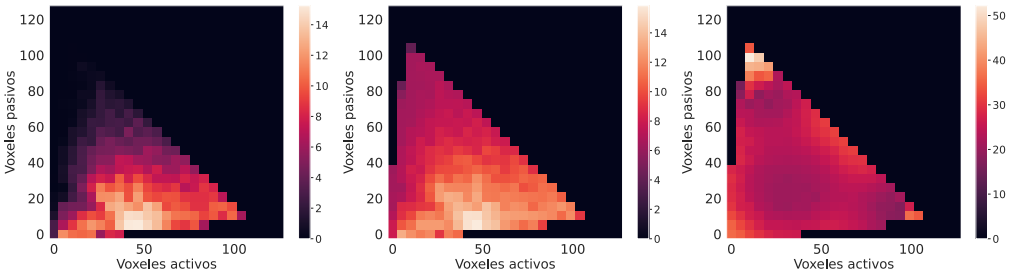
(d) Promedio de F (QN). **(e)** Promedio de N_a (QN). **(f)** Promedio de N_u (QN).

(g) Promedio de F (NSLC). **(h)** Promedio de N_a (NSLC). **(i)** Promedio de N_u (NSLC).

Figura 5.16: Grid de novedad alineada (G_{an}).



(a) Promedio de F (MNSLC). **(b)** Promedio de N_a (MNSLC). **(c)** Promedio de N_u (MNSLC).



(d) Promedio de F (ME). **(e)** Promedio de N_a (ME). **(f)** Promedio de N_u (ME).

Figura 5.17: Grid de novedad alineada (G_{an}).

Tabla 5.1: Comparativa de valores de F en ambos grids

	ME	MNSLC	QN	NSLC	SO
Valor de p	1.8145E-40	2.6456E-40	1.8145E-40	1.8145E-40	3.5638E-38
Máximo en G_f	1.7489E+01	7.8451E+00	9.3372E+00	5.9968E+00	4.0998E+00
Máximo en G_{an}	1.5243E+01	6.3108E+00	7.5813E+00	5.4226E+00	3.6584E+00
Promedio en G_f	3.0541E+00	5.2439E-01	7.8994E-01	3.7359E-01	2.3427E-01
Promedio en G_{an}	2.1768E+00	4.3722E-01	6.4724E-01	3.3545E-01	2.0067E-01

Tabla 5.2: Comparativa de valores de N_u en ambos grids

	ME	MNSLC	QN	NSLC	SO
Valor de p	8.1401E-01	6.1950E-10	3.2329E-15	2.7449E-05	1.7267E-04
Máximo en G_f	4.1325E+01	4.0365E+01	3.9545E+01	3.8818E+01	2.7901E+01
Máximo en G_{an}	5.2193E+01	4.1764E+01	3.7304E+01	3.8912E+01	2.6793E+01
Promedio en G_f	9.1732E+00	1.0007E+01	8.4679E+00	9.7398E+00	6.0168E+00
Promedio en G_{an}	9.4751E+00	9.9145E+00	8.3163E+00	9.7000E+00	6.0649E+00

Tabla 5.3: Comparativa de valores de N_a en ambos grids

	ME	MNSLC	QN	NSLC	SO
Valor de p	2.0877E-40	1.1234E-34	4.4950E-29	1.7652E-24	5.3136E-15
Máximo en G_f	1.6353E+01	7.5394E+00	9.4449E+00	4.2785E+00	4.3498E+00
Máximo en G_{an}	1.5844E+01	7.2729E+00	9.0443E+00	4.8916E+00	4.4680E+00
Promedio en G_f	4.0596E+00	1.2292E+00	1.9203E+00	7.2673E-01	6.0451E-01
Promedio en G_{an}	3.6001E+00	1.1831E+00	1.8548E+00	7.1174E-01	5.9325E-01

En general, se puede observar una gran similitud en cuanto a las regiones iluminadas en cada indicador entre el grid de novedad alineada G_{an} y el grid de fitness G_f. Dicha similitud se debe, en primer lugar, a que la cobertura en ambos grids es la misma. En segundo lugar, podemos observar que la similitud no sólo se limita a iluminar las mismas regiones, sino que además cada región se ilumina con una intensidad muy similar.

Quizás las mayores diferencias se encuentran en las escalas. Para el caso del fitness (F) en G_{an} para todos los algoritmos, se puede observar que se tiende a alcanzar valores ligeramente menores a los alcanzados en G_{an}. Si observamos ambos grids, para todos los experimentos (Figuras 5.14a, 5.16a, 5.14d, 5.16d, 5.14g, 5.16g, 5.15a, 5.17a, 5.15d y 5.17d), se puede ver la diferencia entre los valores máximos de las escalas. Cotejando con los valores en la Tabla 5.1, en primer lugar tenemos valores de $p \leq 0.05$ que indican significancia estadística, por lo que las distribuciones de fitness (F) en ambos grids (G_f y G_{an}) son distintas. En segundo lugar, tenemos diferencias pequeñas en cuanto a los máximos alcanzados y el valor promedio. Dichas diferencias siempre favorecieron a G_f. Ello es de esperarse considerando que G_f mantiene a las élites de F, mientras que G_{an} mantiene a las élites de N_a. Otra observación interesante es que los dos grids con menor diferencia entre valores máximos de F (SO y NSLC), coinciden con los experimentos

con menor valor de $\texttt{QD}_{\texttt{ff}}$, es decir, la diferencia pequeña de F es simplemente por la escala.

Es interesante que en el caso de la novedad alineada (N_a), no sucede algo análogo a lo que sucedió con F, es decir, que el valor máximo de N_a fuese mayor en G_{an} que en G_f, en todos los experimentos. Si observamos los grids de SO (Figuras 5.14b y 5.16b) y NSLC (Figuras 5.14h y 5.16h), y cotejamos con lo observado en la Tabla 5.3, son los únicos experimentos en los que se cumple lo anterior, con diferencias bastante pequeñas, pero estadísticamente significativas ($p \leq 0.05$). Además, SO y NSLC son los mismos experimentos que presentaron las menores diferencias de valores máximos de F de acuerdo a lo observado en la Tabla 5.1. En el resto de los experimentos: QN (Figuras 5.14e y 5.16e), MNSLC (Figuras 5.15b y 5.17b) y ME (Figuras 5.15e y 5.17e), los valores máximos de N_a en G_f son mayores a los de G_{an}, aunque con diferencias aproximadas bastante reducidas ($p \leq 0.05$), tal y como se muestra en la Tabla 5.3.

En cuanto a la novedad desalineada (N_u), es interesante observar que pese a las diferencias visibles de escala en G_{an} y G_f para ME (Figuras 5.15f y 5.17f), no existe suficiente evidencia para poder concluir que dichos valores vengan de distribuciones distintas, al no poderse rechazar la hipótesis nula ($p > 0.05$). Otro resultado bastante interesante es el obtenido en MNSLC (Figuras 5.15c y 5.17c) donde la diferencia es mínima en cuanto al valor máximo, pero bastante grande en cuanto al valor promedio. El grid con mayor valor promedio es G_f, mientras que el que tiene mayor valor máximo es G_f. Con ello se hace evidente que en G_f se logró mantener soluciones con menor variabilidad en cuanto a N_u y más novedosas en el sentido morfológico. Ello puede sugerir que cuando se agrega la novedad alineada como tercer objetivo a NSLC, las soluciones con más fitness, tienden a poseer una mayor novedad morfológica, mientras que las soluciones más novedosas en comportamiento, tienen menor fitness. En el resto de los experimentos: SO (Figuras 5.14c y 5.16c), QN (Figuras 5.14f y 5.16f) y NSLC (Figuras 5.14i y 5.16i), los valores máximos y promedio de N_u en G_f son mayores a los de G_{an}, y al igual que en N_a, con diferencias bastante reducidas. Lo anterior se puede cotejar en la Tabla 5.2, donde además se observa que las diferencias entre ambos grids son estadísticamente significativas ($p \leq 0.05$).

En resumen, todo lo anterior sugiere nuevamente, un alto nivel de alineación entre N_a y F. Específicamente: La similitud entre los patrones de iluminación de ambos grids, y las bajas diferencias entre valores máximos y valores promedio de F, N_u y N_a, pese a que cada uno mantenía las élites de F y N_a, respectivamente.

5.3. Discusión

En esta sección se expone la metodología basada en teoría de votaciones para elegir el mejor algoritmo de la manera más justa posible, tomando todas las consideraciones que sean necesarias. Posteriormente, se discuten los hallazgos relacionados a la aplicación de dicha metodología y a los resultados expuestos anteriormente, para finalmente rela-

cionar ambos. Esta metodología será planteada en su forma más general, tomando en cuenta varios indicadores y problemas de prueba [72], sin embargo, será igualmente útil para nuestro caso, con un solo problema de prueba y varios indicadores. Posteriormente se aplica esta metodología a los experimentos e indicadores principales, obteniendo al final, de la manera más exhaustiva y multifactorial, qué algoritmo presenta el comportamiento más deseable para enfrentarse al problema de la convergencia prematura de morfología respecto al control, comprometiendo lo menos posible otros aspectos deseables, especialmente de carácter explotativo, para encontrar soluciones no solo diversas morfológicamente, sino también funcionales.

Por completitud, vale la pena explicar, que con base en la metodología para el análisis de resultados en benchmarking de algoritmos de optimización, expuesta en [72], lo realizado en la Sección 5.2 sería lo que corresponde al primer paso, es decir, análisis exploratorio de datos, en esta sección se realizará el análisis confirmatorio, que consiste en comparar cada par de algoritmos, respecto a cada par de problemas de prueba e indicadores $(P_i, I_j) \in \mathbb{P} \times \mathbb{I}$, a través de pruebas de hipótesis.

5.3.1. ¿Qué prueba de hipótesis utilizar?

Como ya se dijo, se harán comparaciones de pares, y además no se puede asumir normalidad en los datos (no se cumplen las condiciones para poder utilizar estadística paramétrica), por lo tanto utilizaremos una prueba de estadística no paramétrica, específicamente Wilcoxon Rank-Sum. Además, dado el alto costo computacional de los experimentos, evitaremos hacer el total de experimentos por hipótesis ($5 \cdot 4 \cdot 11 = 220$), y nos limitaremos a hacer veinte corridas por experimento, por lo cual se hace Wilcoxon Rank-Sum con corrección de Bonferroni. Se elige $\alpha = 0.05/220 = 0.000227$ como valor mínimo de p para poder rechazar la hipótesis nula.

5.3.2. Expresando las preferencias de (P_i, I_j)

Teniendo una forma de comparar a los algoritmos, podemos ordenarlos respecto a tal comparación. En el caso más simple se tienen 2 algoritmos, 1 problema y 1 indicador.

- Hacemos múltiples ejecuciones independientes
- Aplicamos la prueba estadística elegida, con 3 posibles resultados
  - $A_1 \prec A_2$. Si $\mu_1 > \mu_2$; $p < \alpha$
  - $A_2 \prec A_1$. Si $\mu_2 > \mu_1$; $p < \alpha$
  - $A_1 \sim A_2$. Si $p \geq \alpha$.

Para n candidatos (algoritmos), $\mathbb{A} = A_1, \cdots, A_n$, un problema P_1 y un indicador I_1, tendríamos una matriz que expresa las preferencias de (P_1, I_1):

	A_1	$\cdots$	A_n
A_1	–	$\uparrow$	$\downarrow$
$\vdots$	$\downarrow$	–	$\downarrow$
A_n	$\uparrow$	$\leftrightarrow$	–

Se puede leer por renglón, de izquierda a derecha como

- $A_i \prec A_j = A_i \uparrow A_j$.
- $A_j \prec A_i = A_i \downarrow A_j$.
- $A_i \sim A_j = A_i \leftrightarrow A_j$

A su vez, dicha tabla puede ser resumida en un conteo:

- +1 al algoritmo preferido.
- -1 al algoritmo que pierde.
- 0 indiferencia.

Con dicho conteo, podemos ordenar de mayor a menor, y dicho ordenamiento expresaría la preferencia de (P_1, I_1). Si deseáramos generalizar para:

$$\mathbb{P} = \{P_1, \cdots, P_m\}$$

$$\mathbb{I} = \{I_1, \cdots, I_l\}$$

Entonces necesitaríamos obtener las $\frac{||\mathbb{P}||\cdot(||\mathbb{I}||-1)}{2}$ preferencias de $\mathbb{P} \times \mathbb{I}$. Para nuestro caso, únicamente tenemos un problema, por lo que tenemos una preferencia por cada uno de los once indicadores. Obtenemos la media y desviación estándar de todos los indicadores, para cada algoritmo, tal y como se muestra en la Tabla 5.4:

Tabla 5.4: Indicadores por algoritmo.

Indicador	ME	MNSLC	QN	NSLC	SO
N_a	**4.61**(**2.21**)	2.41(1.28)	3.00(1.55)	1.19(0.52)	1.62(1.04)
D_{gc}	2.76(0.86)	**4.92**(**1.19**)	2.25(0.80)	4.39(1.05)	1.20(0.49)
$\mathcal{C}$	**0.37**(**0.02**)	0.36(0.02)	0.35(0.02)	0.36(0.02)	0.28(0.02)
F	**6.10**(**2.81**)	3.94(2.53)	5.96(3.20)	2.68(1.21)	5.91(3.07)
D_g	3.63(0.83)	**5.84**(**1.36**)	3.04(0.86)	4.90(0.99)	1.41(0.56)
D_m	34.72(2.71)	51.01(8.68)	42.56(15.20)	**52.24**(**7.70**)	4.86(3.20)
D_{gm}	2.03(0.47)	**2.47**(**1.61**)	1.81(0.56)	1.80(0.41)	0.50(0.33)
$\mathtt{QD}_{\mathtt{anan}}$	**1678.72**(**881.86**)	493.25(359.01)	803.88(547.56)	335.00(182.46)	269.88(203.65)
$\mathtt{QD}_{\mathtt{ff}}$	**1295.69**(**705.72**)	240.92(101.44)	371.71(217.51)	193.60(82.90)	125.29(76.88)
$\mathtt{QD}_{\mathtt{fun}}$	5929.06(453.52)	**6278.15**(**333.42**)	5254.00(334.64)	6233.63(418.72)	3967.39(475.09)
N_u	15.54(1.48)	24.58(3.68)	15.65(5.25)	**25.25**(**3.06**)	2.68(1.86)

Con dicha tabla, junto con las pruebas de hipótesis, podemos obtener los resultados de los encuentros para cada indicador (las cifras junto a las flechas son los valores de p), tal y como se muestra en las Tablas 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14 y 5.15.

Tabla 5.5: Resultados de encuentros para N_a.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑
MNSLC	0.0000E+00 ↓	–	0.0000E+00 ↓	0.0000E+00 ↑	0.0000E+00 ↑
QN	0.0000E+00 ↓	0.0000E+00 ↑	–	0.0000E+00 ↑	0.0000E+00 ↑
NSLC	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–	0.0000E+00 ↓
SO	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↑	–

Tabla 5.6: Resultados de encuentros para D_{gc}.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	0.0000E+00 ↓	0.0000E+00 ↑	0.0000E+00 ↓	0.0000E+00 ↑
MNSLC	0.0000E+00 ↑	–	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑
QN	0.0000E+00 ↓	0.0000E+00 ↓	–	0.0000E+00 ↓	0.0000E+00 ↑
NSLC	0.0000E+00 ↑	0.0000E+00 ↓	0.0000E+00 ↑	–	0.0000E+00 ↑
SO	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–

Tabla 5.7: Resultados de encuentros para $\mathcal{C}$.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑
MNSLC	0.0000E+00 ↓	–	1.0515E-204 ↑	1.8353E-05 ↑	0.0000E+00 ↑
QN	0.0000E+00 ↓	1.0515E-204 ↓	–	1.6669E-142 ↓	0.0000E+00 ↑
NSLC	0.0000E+00 ↓	1.8353E-05 ↓	1.6669E-142 ↑	–	0.0000E+00 ↑
SO	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–

Tabla 5.8: Resultados de encuentros para F.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	0.0000E+00 ↑	4.4480E-15 ↑	0.0000E+00 ↑	4.6503E-16 ↑
MNSLC	0.0000E+00 ↓	–	0.0000E+00 ↓	0.0000E+00 ↑	0.0000E+00 ↓
QN	4.4480E-15 ↓	0.0000E+00 ↑	–	0.0000E+00 ↑	3.9406E-01 ↔
NSLC	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–	0.0000E+00 ↓
SO	4.6503E-16 ↓	0.0000E+00 ↑	3.9406E-01 ↔	0.0000E+00 ↑	–

Tabla 5.9: Resultados de encuentros para D_g.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	0.0000E+00 ↓	0.0000E+00 ↑	0.0000E+00 ↓	0.0000E+00 ↑
MNSLC	0.0000E+00 ↑	–	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑
QN	0.0000E+00 ↓	0.0000E+00 ↓	–	0.0000E+00 ↓	0.0000E+00 ↑
NSLC	0.0000E+00 ↑	0.0000E+00 ↓	0.0000E+00 ↑	–	0.0000E+00 ↑
SO	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–

Tabla 5.10: Resultados de encuentros para D_m.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↑
MNSLC	0.0000E+00 ↑	–	0.0000E+00 ↑	4.5780E-52 ↓	0.0000E+00 ↑
QN	0.0000E+00 ↑	0.0000E+00 ↓	–	0.0000E+00 ↓	0.0000E+00 ↑
NSLC	0.0000E+00 ↑	4.5780E-52 ↑	0.0000E+00 ↑	–	0.0000E+00 ↑
SO	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–

Tabla 5.11: Resultados de encuentros para D_{gm}.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	3.3759E-143 ↓	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑
MNSLC	3.3759E-143 ↑	–	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑
QN	0.0000E+00 ↓	0.0000E+00 ↓	–	3.4341E-02 ↔	0.0000E+00 ↑
NSLC	0.0000E+00 ↓	0.0000E+00 ↓	3.4341E-02 ↔	–	0.0000E+00 ↑
SO	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–

Tabla 5.12: Resultados de encuentros para $\texttt{QD}_{\texttt{anan}}$.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑
MNSLC	0.0000E+00 ↓	–	0.0000E+00 ↓	0.0000E+00 ↑	0.0000E+00 ↑
QN	0.0000E+00 ↓	0.0000E+00 ↑	–	0.0000E+00 ↑	0.0000E+00 ↑
NSLC	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–	0.0000E+00 ↑
SO	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–

Tabla 5.13: Resultados de encuentros para $\texttt{QD}_{\texttt{ff}}$.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑	0.0000E+00 ↑
MNSLC	0.0000E+00 ↓	–	0.0000E+00 ↓	0.0000E+00 ↑	0.0000E+00 ↑
QN	0.0000E+00 ↓	0.0000E+00 ↑	–	0.0000E+00 ↑	0.0000E+00 ↑
NSLC	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–	0.0000E+00 ↑
SO	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–

Tabla 5.14: Resultados de encuentros para $\texttt{QD}_{\texttt{fun}}$.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	0.0000E+00 ↓	0.0000E+00 ↑	0.0000E+00 ↓	0.0000E+00 ↑
MNSLC	0.0000E+00 ↑	–	0.0000E+00 ↑	1.0960E-30 ↑	0.0000E+00 ↑
QN	0.0000E+00 ↓	0.0000E+00 ↓	–	0.0000E+00 ↓	0.0000E+00 ↑
NSLC	0.0000E+00 ↑	1.0960E-30 ↓	0.0000E+00 ↑	–	0.0000E+00 ↑
SO	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–

Tabla 5.15: Resultados de encuentros para N_u.

Algoritmo	ME	MNSLC	QN	NSLC	SO
ME	–	0.0000E+00 ↓	3.1311E-39 ↓	0.0000E+00 ↓	0.0000E+00 ↑
MNSLC	0.0000E+00 ↑	–	0.0000E+00 ↑	4.4832E-97 ↓	0.0000E+00 ↑
QN	3.1311E-39 ↑	0.0000E+00 ↓	–	0.0000E+00 ↓	0.0000E+00 ↑
NSLC	0.0000E+00 ↑	4.4832E-97 ↑	0.0000E+00 ↑	–	0.0000E+00 ↑
SO	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	0.0000E+00 ↓	–

Con los resultados de todos los encuentros, podemos obtener las puntuaciones para cada algoritmo, en cada indicador (Tabla 5.16).

Tabla 5.16: Puntuaciones por algoritmo en cada indicador.

Indicador	ME	MNSLC	NSLC	QN	SO
N_a	4	0	-4	2	-2
D_{gc}	0	4	2	-2	-4
$\mathcal{C}$	4	2	0	-2	-4
F	4	-2	-4	1	1
D_g	0	4	2	-2	-4
D_m	-2	2	4	0	-4
D_{gm}	2	4	-1	-1	-4
$\mathtt{QD}_{\mathtt{anan}}$	4	0	-2	2	-4
$\mathtt{QD}_{\mathtt{ff}}$	4	0	-2	2	-4
$\mathtt{QD}_{\mathtt{fun}}$	0	4	2	-2	-4
N_u	-2	2	4	0	-4

Para finalmente obtener las preferencias de cada indicador, e indicar el número de incidencias que tuvo cada permutación (Tabla 5.17).

Tabla 5.17: Incidencia de permutaciones.

1	2	3	4	5	Incidencias
ME	MNSLC	NSLC	QN	SO	1
ME	QN	MNSLC	NSLC	SO	2
ME	QN	MNSLC	SO	NSLC	1
ME	QN	SO	MNSLC	NSLC	1
MNSLC	ME	QN	NSLC	SO	1
MNSLC	NSLC	ME	QN	SO	3
NSLC	MNSLC	QN	ME	SO	2

5.3.3. Eligiendo al ganador

Finalmente, podemos resumir la Tabla 5.17 en la Tabla 5.18.

Tabla 5.18: Conteo de Borda.

Posición	ME	MNSLC	NSLC	QN	SO
1	5	4	2	0	0
2	1	3	3	4	0
3	3	3	1	3	1
4	2	1	3	4	1
5	0	0	2	0	9
Conteo de Borda	42	43	33	33	14

Donde el conteo de Borda, se obtiene de la siguiente manera, para cada algoritmo:

$$CB(ME) = 5(5) + 4(1) + 3(3) + 2(2) + 1(0) = 42$$

$$CB(MNSLC) = 5(4) + 4(3) + 3(3) + 2(1) + 1(0) = 43$$

$$CB(NSLC) = 5(2) + 4(3) + 3(1) + 2(3) + 1(2) = 33$$

$$CB(QN) = 5(0) + 4(4) + 3(3) + 2(4) + 1(0) = 33$$

$$CB(SO) = 5(0) + 4(0) + 3(1) + 2(1) + 1(9) = 14$$

A través de conteo de Borda, el algoritmo ganador es $MNSLC$, seguido de cerca por ME.

Con base en el conteo de Borda, el ordenamiento de los experimentos queda: MNSLC, ME, NSLC, QN, SO.

Otra forma de elegir al ganador, es a través de el método de Condorcet. En dicho método:

- Se hace una matriz de $n \times n$.
- Se rellena con las veces que un candidato A_i sea preferido a A_j.
- Un ganador de Condorcet, es aquel que gana todos sus "encuentros" uno a uno. Es decir, aquel A_i que haya sido preferido sobre A_j, más veces que las que A_j fue preferido sobre A_i $\forall j; i \neq j$.

Obteniendo dicha matriz para nuestros experimentos y conjunto de indicadores (Tabla 5.19):

Tabla 5.19: Método de Condorcet.

	ME	MNSLC	NSLC	QN	SO
ME	0	5	6	9	11
MNSLC	6	0	9	7	10
NSLC	5	2	0	6	9
QN	2	4	4	0	10
SO	0	1	2	0	0

Con el método de Condorcet, se tiene de nuevo que el ganador es $MNSLC$. Ahora bien, como es notorio, la victoria de $MNSLC$ fue marginal contra ME, bajo ambos criterios.

En cuanto a conteo de Borda tenemos que MNSLC tuvo una puntuación de 43, contra los 42 de ME, con ME siendo el primer lugar 5 veces, una vez más que MNSLC, lo cual hubiese situado a ME como ganador por criterio de mayoría, la diferencia la marcó las 2 veces más que MNSLC fue cuarto lugar. En cuanto al método de Condorcet, como ya se mencionó, la victoria de MNSLC también fue marginal, con 6 encuentros ganados, contra los 5 de ME. Si examinamos la Tabla 5.16, se hace evidente que las victorias de MNSLC se centraron casi exclusivamente en los indicadores que consideran puramente diversidad (D_m, N_u, D_g, D_{gc}, D_{gm}, $\texttt{QD}_{\texttt{fun}}$) con excepción de coverage ($\mathcal{C}$),

con lo cual es más que claro que MNSLC podrá encontrar mayor diversidad, tanto morfológica, como de control, pero ME es superior en todos los criterios que consideran aptitud global y calidad local, encontrando no solo mayor aptitud (F), sino que también calidad acumulada ($\texttt{QD}_{\texttt{ff}}$), novedad alineada acumulada ($\texttt{QD}_{\texttt{anan}}$), y novedad alineada ($N_a$).

Ahora recordemos las preguntas hechas en 4.8.4, con el análisis anterior hemos respondido cómo se compara ME, con un algoritmo Multi-BC QD, para responder cómo afecta a un algoritmo QD como MAP-Elites la introducción de una presión global y si sigue manteniendo las características deseables de los algoritmos QD, tenemos que hacer la comparación contra NSLC.

Regresando a examinar el conteo de Borda 5.18, podemos observar que ME supera ampliamente a NSLC. Ahora bien, examinando el método de Condorcet 5.19, nos damos cuenta de que ME supera marginalmente a NSLC, con 6 victorias, contra 5 de NSLC, ganando en casi los mismos indicadores que a MNSLC, con la adición de D_{gm}, por lo que nos encontramos con una historia similar a ME y MNSLC. Considerando las características deseables de un algoritmo QD, podemos encontrar que ME fue superior a MNSLC y NSLC en cuanto a calidad encontrada en la diversidad, pero en general, inferior en cuanto a las nociones de diversidad consideradas.

Cuando consideramos los resultados de MNSLC contra NSLC, parece que la victoria tanto respecto al conteo de Borda, como respecto al método de Condorcet, fue con un margen bastante considerable.

Examinando los encuentros uno a uno en la Tabla 5.19, podemos ver que MNSLC ganó 9 de los 11 encuentros contra NSLC, mientras que NSLC ganó los restantes. Más aún, si examinamos los dos indicadores en los que NSLC ganó, podemos observar que la única ventaja que tiene NSLC es que alcanza un poco más de diversidad y novedad morfológica (D_m, N_u), específicamente de 1.23 y 0.67 en promedio respectivamente, diferencias pequeñas, pero estadísticamente significativas. En ambos indicadores NSLC se encuentra en primer lugar, seguido inmediatamente por MNSLC. Por lo tanto, podemos decir que, pese a que existe una pérdida de diversidad y novedad morfológica respecto NSLC y ambos indicadores son clave para la convergencia prematura de morfología, esta pérdida es mínima, aunque estadísticamente significativa.

Análogamente, tomando en cuenta los 9 indicadores en los que MNSLC fue superior, podemos observar que, no solo MNSLC fue superior en todos aquellos relacionados a la aptitud en el desempeño de la tarea, también lo fue en dos de los cuatro pertenecientes a diversidad del fenotipo y en todos los de diversidad del genotipo. Más aún, los indicadores de diversidad fenotípica en los que se desempeño mejor fueron los referidos al grid de fitness. Tomando en cuenta las observaciones que respectan a la comparación de MNSLC y NSLC, parece ser que la adición de la novedad alineada, en primer lugar, y como ya se mencionó en los resultados, es en efecto una noción de novedad alineada adecuada, y en segundo lugar, mitiga de manera efectiva el problema del progreso lento de la aptitud en algoritmos QD, específicamente en NSLC, sin comprometer

prácticamente el desempeño en cuanto a diversidad morfológica.

Comparando el desempeño de QN respecto a sus contrapartes QD (MNSLC y NSLC), nos encontramos con que MNSLC tiene un desempeño superior a QN tanto en el conteo de Borda, como en el método de Condorcet, mientras que QN y NSLC tienen un desempeño igual respecto a Borda y NSLC supera a QN respecto a Condorcet por un encuentro. Podemos notar que las 4 victorias de QN sobre MNSLC fueron exclusivamente en los indicadores que consideran calidad global o local, y análogamente, las 7 victorias de MNSLC sobre QN fueron exclusivamente en los indicadores que consideran diversidad genotípica y fenotípica. En cuanto a las 5 victorias de QN sobre NSLC, tenemos una situación muy similar a la anterior, solo que se agrega al conteo de victorias de QN, una victoria muy marginal en la diversidad genética de la morfología (D_{gm}), por lo tanto, podemos decir que tomando como referencia QN, el uso de competencia local lleva a resultados más diversos tanto del genotipo, como del fenotipo, mientras que el uso de la aptitud global, lleva a resultados con mayor calidad global, y con mayor calidad sobre la diversidad encontrada.

Ya hablamos sobre las ventajas y desventajas relativas entre los distintos algoritmos que presentan protección a la diversidad, ahora hablaremos sobre las ventajas y desventajas absolutas que presentan los algoritmos con protección a la diversidad respecto no presentar ningún tipo de protección a la diversidad, utilizando el algoritmo genético simple (SO) como referencia.

En cuanto al desempeño de QN respecto a SO se refiere, el primero supera al segundo con más del doble de puntos en cuanto a conteo de Borda, y QN gana todos sus encuentros contra SO, a excepción de uno, donde hubo un empate (no se pudo rechazar la hipótesis nula), el cual corresponde a la aptitud, es decir, la afectación de la adición de la novedad morfológica como segundo objetivo a la aptitud de las soluciones al problema de optimización de morfología y control de creaturas virtuales en la tarea de patrones de caminado, cuando el primer objetivo es la calidad global, es nula. Y mientras que no hay afectación alguna en la aptitud, si hay mejoras en cuanto al resto de los indicadores relacionados con la aptitud respecta ($\texttt{QD}_{\texttt{ff}}$, $\texttt{QD}_{\texttt{anan}}$, N_a), en cuanto a todos los indicadores relacionados con la diversidad tanto genética como del fenotipo. Es decir, la adición de la novedad morfológica, no solo no afecta a la aptitud global, sino que mejora criterios relacionados, como la novedad alineada, la novedad alineada encontrada en la diversidad morfológica, y la calidad en la diversidad morfológica. Además de que mejora en todos los aspectos relacionados a la diversidad, incluida la morfológica, y por lo tanto, la robustez ante el fenómeno de la convergencia prematura de la morfología respecto al control.

Comparando NSLC contra SO, encontramos de nuevo una ventaja de más del doble de puntos con respecto al conteo de Borda y NSLC gana 9 de sus 11 encuentros contra SO, SO supera a NSLC en los dos indicadores poblacionales relacionados a aptitud (F y N_a), y pierde en cualquier otro indicador que incorpore alguna noción de diversidad, desde los indicadores de calidad en la diversidad ($\texttt{QD}_{\texttt{ff}}$, $\texttt{QD}_{\texttt{anan}}$), hasta los indicadores

de diversidad del genotipo y del fenotipo, por lo tanto podemos afirmar que: agregar novedad morfológica y cambiar la aptitud global por local, leva a resultados más diversos tanto del genotipo, como del fenotipo y con mayor calidad sobre la diversidad encontrada, pero con peor aptitud global.

Pasando a MNSLC contra SO, nos encontramos ahora con el mejor algoritmo contra el peor respecto al ordenamiento que nos da el conteo de Borda. Esta vez, MNSLC gana 10 encuentros de 11, mientras que SO supera a MNSLC únicamente en un indicador, la aptitud global, con lo cual, es notorio el efecto que genera agregar como un tercer objetivo la novedad alineada, puesto que MNSLC ahora supera a SO en dicho indicador. Con estas observaciones podemos afirmar: agregar como segundo y tercer objetivo a la novedad morfológica y alineada, y cambiar la aptitud global por local, lleva a resultados más diversos tanto del genotipo, como del fenotipo, con mayor calidad sobre la diversidad encontrada, y con mayor novedad alineada, pero con peor aptitud global.

Finalmente, haciendo la comparación entre ME y SO, ahora nos encontramos con el segundo lugar por muy poco, contra el último lugar. En este caso, ME es mejor que SO en todos y cada uno de los indicadores, con lo cual podemos decir que agregar una presión de selección proporcional a la aptitud global a MAP-Elites, provoca que el algoritmo resultante, no solo llegue a soluciones más diversas y con mayor calidad sobre la diversidad, sino también a soluciones más aptas globalmente.

NSLC posee el peor progreso en cuanto a aptitud se refiere, algo que es consistente con lo encontrado en la literatura [64], sin embargo, es difícil saber, solo con este indicador, si sólo se trata de un progreso lento, o de un estancamiento total, MNSLC por su parte logra superar a NSLC, pero pareciera ser, al examinar las últimas generaciones en la gráfica de convergencia, que ya no crecería mucho en generaciones posteriores a la 1000. Ahora bien, en los otros algoritmos podemos ver que en esas últimas generaciones sigue habiendo una tendencia creciente, de las cuales ME, presenta la más clara, sin embargo, incluso SO sigue mostrando indicios de que seguirá creciendo más allá de la generación 1000, por lo tanto, es posible que de haber dejado a los experimentos correr por más generaciones, se hubiese observado la convergencia de SO a un valor final de aptitud.

Como se ha mencionado varias veces, la novedad de comportamiento, es en efecto, una novedad alineada a la aptitud, sin embargo, se puede obtener una novedad alineada con fitness alto con una trayectoria no muy novedosa, en caso de que la creatura recorra mayor distancia que cualquiera de las soluciones anteriores dentro de su propio nicho, pero también puede que la novedad alineada sea baja, en caso de que no se de esa superación. Otra forma en la que se puede alcanzar la novedad alineada es mediante trayectorias novedosas, las cuales, no necesariamente se asocian a valores altos de aptitud, y pueden asociarse a cambios en la morfología que a su vez lleven a comportamientos erráticos. Puede ser, entonces que la razón por la cual ME y QN superan a MNSLC en novedad alineada, sea porque ambos algoritmos son capaces de encontrar tanto soluciones morfológicamente novedosas, como soluciones con mayor aptitud global que MNSLC, cubriendo ambos escenarios en los que se puede dar una alta novedad

alineada, es decir, por trayectorias diversas, y progreso consistente de aptitud global. Ello se podría examinar, explorando los ángulos de las trayectorias de cada individuo y la cantidad de veces por generación que se dieron superaciones del individuo con aptitud máxima. Esa última métrica, se podría también relacionar con el progreso relativo a la aptitud, y por lo tanto la convergencia al valor de aptitud final. Por lo tanto, también se podría probar si una novedad alineada más alta, se relaciona con un mayor progreso en términos de aptitud.

En cuanto a los hallazgos principales que la examinación de la diversidad morfológica nos provee, podemos mencionar que sin duda hay un claro efecto en la introducción de mecanismos de protección a la diversidad, que permite poblaciones mucho más diversas (i.e. con valores más altos de distancia morfológica promedio), y que no solo se tenga una pérdida de la diversidad morfológica más lenta, sino que la diversidad morfológica se mantenga, y por lo tanto se evite la convergencia prematura de la morfología respecto al control. Ahora bien, respecto a la novedad desalineada se puede mencionar algo muy similar, solo que en este caso, no se mantiene para ningún algoritmo, sino que decrece lentamente. Cabe recalcar que este indicador está muy relacionado con los parámetros que se hayan elegido para el archivo, y al menos en los problemas de prueba, también se podían encontrar gráficas con forma logarítmica (con valores de k pequeños), o que llegaban a un valor máximo y decrecían (señal de que el archivo había llegado a su tamaño máximo).

La cobertura nos provee información sobre la capacidad para explorar distintas morfologías que tiene cada algoritmo, y de manera consistente con todos los indicadores que incorporan algún tipo de diversidad, SO se encuentra muy por debajo respecto a los demás experimentos. ME supera a todos los demás experimentos, pese a tener una probabilidad de selección de bins proporcional a la aptitud, sin embargo, seguramente una probabilidad de selección uniforme lograría mayor cobertura. Complementando con los mapas de características podemos observar que, en cuanto a fitness, ME encuentra regiones mucho más uniformes, mientras que los algoritmos que incorporan novedad, encuentran soluciones cuyo fitness se concentra mucho más en las regiones limítrofes del espacio de búsqueda. Para el caso de SO, la concentración de fitness se encuentra sesgada hacia una región muy reducida, que a su vez se encuentra contenida en ME, donde justamente, se tiene el mayor fitness, ahora bien, en ME el valor de fitness encontrado en esta región es mucho mayor, lo cual sugiere la posible explotación de dicha región a lo largo de varias generaciones.

Así mismo, se encontró una pérdida de la diversidad genética en mayor a menor medida en todos los algoritmos, con SO presentando la mayor pérdida de diversidad genética, de manera consistente con indicadores de diversidad del fenotipo a nivel poblacional. Los experimentos que presentan menor pérdida de la diversidad genética son los algoritmos QD. Así como en novedad y diversidad morfológica, ME tuvo el peor desempeño de los tres algoritmos QD, estos indicadores son poblacionales, por lo tanto, tiene sentido argumentar que al introducir una selección de padres proporcional a la calidad global, en lugar de uniforme, ciertos individuos tengan mayor probabilidad de ser seleccionados

como padres de la siguiente generación, y que entonces se elijan individuos (inclusive, más de una vez) que compartan ciertas características que los hacen más aptos, y por lo tanto que sean más similares entre sí, seleccionando un conjunto de individuos menos diversos y novedosos. Más aún, vale la pena recordar que tanto MNSLC como NSLC, después de efectuar el ordenamiento de no dominados de NSGA-II, eligen los supervivientes del último frente con un ranking basado en la diversidad genética, en lugar del crowding distance, lo cual puede también jugar un papel no solo en la diversidad genética, sino también en novedad y diversidad morfológica. Si comparamos con QN, podemos argumentar algo similar respecto a cómo la presión de selección global afecta a la diversidad, y al posible papel que pueda jugar el uso de la diversidad genética como criterio de corte en el último frente de NSGA-II. No parece demasiado riesgoso asumir que la diversidad genética, diversidad morfológica y novedad morfológica más alta que presentan MNSLC y NSLC se deban no sólo a la competencia local, sino al uso de la diversidad genética, sin embargo, no se efectuó la recopilación de los datos necesarios para poder determinar esto de manera contundente.

La diversidad genética de control presenta el mismo ordenamiento que la diversidad genética, y parece provocar una mayor diversidad de comportamiento, que a su vez provoca una mayor novedad alineada (al tener a MNSLC, como el mejor en este indicador). Más aún los dos experimentos con competencia local, presentan mayor diversidad en los controladores y se podría argumentar que buscar maximizar el score de competencia local y la novedad alineada incrementa la diversidad genética de los controladores.

Es interesante ver que NSLC y QN son tan similares en cuanto a diversidad genética de morfología, parece que la diversidad genética morfológica, ignora la competencia local, y únicamente se ve interesada por la diversidad morfológica y el incremento a la novedad alineada que ésta también conlleva.

Capítulo 6

Conclusiones

En este capítulo se presentan las conclusiones de este trabajo de tesis con base en los objetivos, la metodología propuesta, los resultados obtenidos de los experimentos, y la subsecuente discusión de dichos resultados.

Con base en la metodología propuesta, se puede decir que se tuvo éxito en la proposición y uso del marco de trabajo general para la realización de experimentos de creaturas virtuales y robótica evolutiva, al menos desde un punto de vista práctico. Si bien hizo falta ahondar en la implementación del algoritmo general para crear experimentos de robótica evolutiva, formalizarlo, y comentar la arquitectura del sistema que se implementó. La estructura del capítulo 4 en sí presentó de manera implícita la identificación de las características en común de los experimentos de robótica evolutiva y el método general para proponer los experimentos, al detallar de manera ordenada cada uno de los componentes de los experimentos y los factores que se tomaron en cuenta para elegir un tipo específico de cada componente.

Dado que los experimentos comparativos estuvieron mayormente centrados en el estudio de la convergencia prematura de morfología respecto a control, la mayoría de los indicadores que se utilizaron para la comparación buscaban medir diversidad, ya fuera del genotipo o fenotipo y poblacional o basada en un grid, por ello es que se encontró como ganador absoluto a MNSLC, y no a ME, pese a que posee un mayor equilibrio entre explotación y exploración, logrando tener una protección a la diversidad competitiva con NSLC y MNSLC, y una capacidad explotativa que no solo no se ve afectada por la incorporación de esta protección a la diversidad, sino que supera a la capacidad explotativa de SO.

MNSLC fue superior a ME, únicamente porque habían más indicadores que favorecían a la exploración, que a la explotación, y a su vez habían más indicadores de exploración que de explotación porque se buscaba encontrar al mejor algoritmo en cuanto a convergencia prematura de morfología respecto al control, por lo tanto, ME es un peor algoritmo que MNSLC en cuanto la protección de la diversidad morfológica, pero pro-

bablemente representa una mejor opción en la práctica, para coevolución de morfología y control de creaturas virtuales.

En el caso más general, comparando ME contra MNSLC y NSLC, podemos volver a afirmar lo mismo, ME es un peor algoritmo que sus contrapartes QD "puros" (sin calidad global) en cuanto a protección de la diversidad morfológica, y por lo tanto, en cuanto a la convergencia prematura de morfología respecto a control.

En lo que respecta a NSLC y MNSLC, la novedad de comportamiento, es en efecto una novedad alineada, y la adición de esta como un tercer objetivo, mitiga el progreso lento de NSLC, sin comprometer el desempeño en cuanto a la convergencia prematura de morfología respecto a control. Tomando como referencia QN, el uso de la competencia local en lugar de la calidad global, lleva a resultados más diversos, mientras que utilizar calidad global, lleva a resultados con mayor calidad global y mayor calidad sobre la diversidad, por lo tanto, MNSLC y NSLC llevan a resultados más deseables que QN en cuanto a la convergencia prematura de morfología respecto a control.

La comparación entre los distintos algoritmos que presentan protección a la diversidad nos ayuda a identificar ventajas y desventajas relativas, y nos da una idea de los compromisos que cada uno efectúa respecto al otro. Sin embargo, una visión más objetiva respecto a qué gana y qué pierde cada algoritmo, y que nos puede ayudar a identificar ventajas y desventajas absolutas, es comparar contra un algoritmo base, en este caso, un algoritmo sin ningún tipo protección a la diversidad (SO).

En cuanto a QN, se encontró que la afectación de la adición de la novedad morfológica como segundo objetivo en la aptitud de las soluciones, cuando el primer objetivo es la calidad global, es nula. Además, QN mejora criterios relacionados, como la novedad alineada, la novedad alineada encontrada en la diversidad morfológica, la calidad en la diversidad morfológica, y en todos los aspectos relacionados a la diversidad, incluida la morfológica. Por lo tanto, la robustez de QN ante el fenómeno de la convergencia prematura de la morfología respecto al control es mayor.

En NSLC, se encontró que agregar novedad morfológica y cambiar la aptitud global por local, lleva a resultados más robustos contra la convergencia prematura de morfología respecto a control y con mayor calidad sobre la diversidad encontrada, pero con peor aptitud global.

Respecto a MNSLC, se encontró que agregar como segundo y tercer objetivo a la novedad morfológica y alineada, y cambiar la aptitud global por local, lleva a resultados más robustos contra la convergencia prematura de morfología respecto a control, con mayor calidad sobre la diversidad encontrada, y con mayor novedad alineada, pero con peor aptitud global.

Finalmente, en cuanto a ME, agregar una presión de selección proporcional a la aptitud global a MAP-Elites, provoca que el algoritmo resultante, no solo llegue a soluciones más robustas contra la convergencia prematura de morfología respecto a control y con mayor calidad sobre la diversidad, sino también a soluciones más aptas globalmente.

6.1. Trabajo futuro

El trabajo realizado en esta tesis sienta las bases para explorar nuevos caminos y plantear nuevos experimentos, desde ajustes mínimos hasta algoritmos nuevos. Pasando por distintas morfologías, controladores y ambientes, se puede seguir el mismo marco de trabajo, y utilizar el mismo código como base, por supuesto, algunas rutas supondrán más cambios y más trabajo que otras, pero entre las opciones de trabajo futuro más realistas y promisorias se encuentran:

- La utilización de modelos subrogados, como los descritos en [67], que permitan modelar la costosa evaluación de función de aptitud como un proceso estocástico, reduciendo así el número de evaluaciones necesarias y permitiendo escalar los experimentos a poblaciones más grandes, mayor número de generaciones, más algoritmos a incluir en la comparación e inclusive un barrido de parámetros ("parameter sweep") exhaustivo de los algoritmos en el problema de coevolución de morfología y control de creaturas virtuales.
- Tareas distintas, como patrones dirigidos de caminado, tales como los que se evolucionaron en [34] y [52].
- Sustituir el BC desalineado que se obtuvo a través de una reducción de dimensionalidad manual, por una a red neuronal generativa, como una VAE, entrenada en las salidas de las redes de control y morfología, y explorar ese espacio latente por medio de LVE (Latent Variable Evolution), de manera inspirada en [69].

Bibliografía

[1] J. C. Bongard, "Evolutionary robotics," *Commun. ACM*, vol. 56, p. 74–83, Aug. 2013. 13, 14

[2] S. Doncieux, N. Bredeche, J.-B. Mouret, and A. E. G. Eiben, "Evolutionary robotics: What, why, and where to," *Frontiers in Robotics and AI*, vol. 2, p. 4, 2015. 13, 14, 34

[3] S. Nolfi and D. Floreano, "Evolutionary robotics. the biology, intelligence, and technology of self-organizing machines," 2001. 2001 (2nd print), 2000 (1st print). 13

[4] H. Lipson, V. SunSpiral, J. Bongard, and N. Cheney, "On the difficulty of co-optimizing morphology and control in evolved virtual creatures," *Artificial Life*, pp. 226–233, 2016. 13, 14, 54, 55, 58, 59, 62, 77, 78, 80, 83, 99

[5] J. E. Auerbach and J. C. Bongard, "Evolving cppns to grow three-dimensional physical structures," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, (New York, NY, USA), p. 627–634, Association for Computing Machinery, 2010. 14, 34, 58

[6] S. Kriegman, N. Cheney, and J. Bongard, "How morphological development can guide evolution," *Scientific Reports*, vol. 8, Sep 2018. 14, 54, 58, 59, 75

[7] M. Joachimczak, R. Suzuki, and T. Arita, "Artificial metamorphosis: Evolutionary design of transforming, soft-bodied robots," *Artificial Life*, vol. 22, pp. 271–298, 08 2016. 14, 55, 57, 61, 80, 81

[8] N. Cheney, J. Bongard, V. SunSpiral, and H. Lipson, "Scalable co-optimization of morphology and control in embodied machines," *Journal of The Royal Society Interface*, vol. 15, 2018. 14, 58, 59, 75

[9] J. Nordmoen, F. Veenstra, K. O. Ellefsen, and K. Glette, "Quality and diversity in evolutionary modular robotics," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 2109–2116, 2020. 14, 55, 59, 65, 92

[10] T. Nygaard, E. Samuelsen, and K. Glette, "Overcoming initial convergence in multi-objective evolution of robot control and morphology using a two-phase approach," pp. 825–836, 03 2017. 14, 37, 38, 39, 58

[11] A. Eiben and J. Smith, *Introduction To Evolutionary Computing*, vol. 45. 01 2003. 19, 20, 21

[12] P. J. Bentley and D. W. Corne, "An introduction to creative evolutionary systems," in *Creative Evolutionary Systems* (P. J. Bentley and D. W. Corne, eds.), The Morgan Kaufmann Series in Artificial Intelligence, pp. 1–75, San Francisco: Morgan Kaufmann, 2002. 20, 22, 24

[13] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020. 21, 29, 86

[14] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992. 23

[15] D. Goldberg, G. David Edward, D. Goldberg, and V. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley series in artificial intelligence, Addison-Wesley Publishing Company, 1989. 23

[16] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992. 24, 25

[17] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA, USA: MIT Press, 1994. 25

[18] M. Emmerich and A. Deutz, "A tutorial on multiobjective optimization: fundamentals and evolutionary methods," *Natural Computing*, vol. 17, 09 2018. 29

[19] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014. 29

[20] J. Blank, K. Deb, and P. C. Roy, *Investigating the Normalization Procedure of NSGA-III: 10th International Conference, EMO 2019, East Lansing, MI, USA, March 10-13, 2019, Proceedings*, pp. 229–240. 01 2019. 29

[21] K. Sims, "Evolving virtual creatures," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, (New York, NY, USA), p. 15–22, Association for Computing Machinery, 1994. 33, 34, 35, 36, 37, 44, 54, 61

[22] K. Sims, "Evolving 3d morphology and behavior by competition," *Artif. Life*, vol. 1, p. 353–372, July 1994. 33, 54, 59, 62, 82, 83

[23] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, pp. 131–162, 2007. 34, 36, 43, 44, 45, 48, 49, 50, 76

[24] T. Geijtenbeek and N. Pronost, "Interactive character animation using simulated physics: A state-of-the-art review," *Comput. Graph. Forum*, vol. 31, p. 2492–2515, Dec. 2012. 34

[25] W. Mcculloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 127–147, 1943. 35, 46

[26] T. Miconi and A. Channon, "Analysing co-evolution among artificial 3d creatures," in *Artificial Evolution* (E.-G. Talbi, P. Liardet, P. Collet, E. Lutton, and M. Schoenauer, eds.), (Berlin, Heidelberg), pp. 167–178, Springer Berlin Heidelberg, 2006. 59, 62, 83

[27] T. Miconi and A. Channon, "A virtual creatures model for studies in artificial evolution," vol. 1, pp. 565 – 572 Vol.1, 10 2005. 35, 36, 44

[28] G. S. Hornby and J. B. Pollack, "Body-brain co-evolution using l-systems as a generative encoding," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, (San Francisco, CA, USA), p. 868–875, Morgan Kaufmann Publishers Inc., 2001. 36, 37, 38, 40, 44, 53

[29] G. Hornby and J. Pollack, "Creating high-level components with a generative representation for body-brain evolution," *Artificial life*, vol. 8, pp. 223–46, 02 2002. 36, 44, 53

[30] H. Lipson and J. Pollack, "Automatic design and manufacture of robotic lifeforms," *Nature*, vol. 406, pp. 974–8, 09 2000. 38, 40

[31] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding," *SIGEVOlution*, vol. 7, p. 11–23, Aug. 2014. 39, 54, 55, 59, 77, 78, 79

[32] M. A. Bedau, J. S. McCaskill, N. H. Packard, and S. Rasmussen, *From Directed to Open-Ended Evolution in a Complex Simulation Model*, pp. 293–299. 2000. 40

[33] M. Jelisavcic, K. Glette, E. Haasdijk, and A. E. Eiben, "Lamarckian evolution of simulated modular robots," *Frontiers in Robotics and AI*, vol. 6, p. 9, 2019. 40, 56, 57, 59

[34] G. Lan, M. Jelisavcic, D. M. Roijers, E. Haasdijk, and A. E. Eiben, "Directed locomotion for modular robots with evolvable morphologies," in *Parallel Problem Solving from Nature*, 2018. 40, 56, 57, 59, 82, 141

[35] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[36] P. Krcah, *Evolutionary development of robotic organisms*. PhD thesis, Universidad de Charles, Praga, 2013.

[37] H. Lipson, "Principles of modularity, regularity, and hierarchy for scalable systems," *Journal of Biological Physics and Chemistry*, vol. 7, 12 2007.

[38] P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, *A Developmental Model for the Evolution of Complete Autonomous Agents*, pp. 393–401. 1996. 45

[39] G. V. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989. 48, 55, 94

[40] K. Stanley, D. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial life*, vol. 15, pp. 185–212, 02 2009. 49, 50, 51, 52, 54

[41] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock, "Evolving coordinated quadruped gaits with the hyperneat generative encoding," in *2009 IEEE Congress on Evolutionary Computation*, pp. 2764–2771, 2009. 52, 53, 55, 57

[42] V. Matos and C. P. Santos, "Towards goal-directed biped locomotion: Combining cpgs and motion primitives," *Robotics and Autonomous Systems*, vol. 62, no. 12, pp. 1669–1690, 2014. 53, 56

[43] H. Lund and O. Miglino, "From simulated to real robots," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 362–365, 1996. 55

[44] P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, *Co-evolution of Pursuit and Evasion II: Simulation Methods and Results*, pp. 506–515. 1996. 56

[45] G. Hornby, S. Takamura, O. Hanagata, M. Fujita, and J. Pollack, "Evolution of controllers from a high-level simulator to a high dof robot," pp. 80–89, 06 2000. 56

[46] J. Auerbach and J. Bongard, "Evolving complete robots with cppn-neat: the utility of recurrent connections," in *GECCO '11*, 2011. 56, 58

[47] M. D. Carlo, D. Zeeuwe, E. Ferrante, G. Meynen, J. Ellers, and A. Eiben, "Influences of artificial speciation on morphological robot evolution," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 2272–2279, 2020. 56, 59

[48] D. Howard, T. Lowe, and W. Geles, "Diversity-based design assist for large legged robots," *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 2020. 58

[49] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *ArXiv*, vol. abs/1504.04909, 2015. 59, 72, 73

[50] N. Cheney, J. Bongard, and H. Lipson, "Evolving soft robots in tight spaces," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, (New York, NY, USA), p. 935–942, Association for Computing Machinery, 2015. 59, 61, 83

[51] F. Schmitt, O. Piccin, L. Barbé, and B. Bayle, "Soft robots manufacturing: A review," *Frontiers in Robotics and AI*, vol. 5, 2018. 59

[52] G. Lan, M. De Carlo, F. van Diggelen, J. M. Tomczak, D. M. Roijers, and A. Eiben, "Learning directed locomotion in modular robots with evolvable morphologies," *Applied Soft Computing*, vol. 111, p. 107688, 2021. 59, 82, 141

[53] E. H. Stensby, K. O. Ellefsen, and K. Glette, "Co-optimising robot morphology and controller in a simulated open-ended environment," in *Applications of Evolutionary Computation*, pp. 34–49, Springer International Publishing, 2021. 59, 61, 82

[54] M. De Carlo, E. Ferrante, J. Ellers, G. Meynen, and A. E. Eiben, "The impact of different tasks on evolved robot morphologies," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, (New York, NY, USA), p. 91–92, Association for Computing Machinery, 2021. 60

[55] E. de Bruin, J. Hatzky, B. Hosseinkhani Kargar, and A. E. Eiben, "A multi-brain approach for multiple tasks in evolvable robots," in *Applications of Evolutionary Computation* (J. Correia, S. Smith, and R. Qaddoura, eds.), (Cham), pp. 129–144, Springer Nature Switzerland, 2023. 60

[56] T. Kimura, Z. Jin, R. Niiyama, and Y. Kuniyoshi, "Evolving soft robots to execute multiple tasks with combined-cppn-neat," in *2016 IEEE/SICE International Symposium on System Integration (SII)*, pp. 409–414, 2016. 60

[57] J. S. Bhatia, H. Jackson, Y. Tian, J. Xu, and W. Matusik, "Evolution gym: A large-scale benchmark for evolving soft robots," 2022. 60, 61

[58] J. Hiller and H. Lipson, "Dynamic simulation of soft multimaterial 3d-printed objects," *Soft Robotics*, vol. 1, no. 1, pp. 88–101, 2014. 61, 80, 82, 83

[59] S. Liu, D. Matthews, S. Kriegman, and J. Bongard, "Voxcraft-sim, a gpu-accelerated voxel-based physics engine." `https://github.com/voxcraft/voxcraft-sim`, 2020. 61, 83

[60] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997. 62

[61] D. Wolpert and W. Macready, "Coevolutionary free lunches," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 721–735, 2005. 62

[62] J.-B. Mouret and S. Doncieux, "Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity," in *2009 IEEE Congress on Evolutionary Computation*, pp. 1161–1168, 2009. 64, 65, 69

[63] J.-B. Mouret, *Novelty-Based Multiobjectivization*, vol. 341, pp. 139–154. 02 2011. 65, 68, 90

[64] J. Pugh, L. Soros, and K. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, 07 2016. 66, 67, 69, 70, 72, 75, 97, 136

[65] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary Computation*, vol. 19, no. 2, pp. 189–223, 2011. 67, 68

[66] I. Omelianenko, "Hands-on neuroevolution with python," 2019. 67, 68

[67] K. Chatzilygeroudis, A. Cully, V. Vassiliades, and J.-B. Mouret, "Quality-diversity optimization: a novel branch of stochastic optimization," *ArXiv*, vol. abs/2012.04322, 2020. 69, 70, 74, 141

[68] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 245–259, 2018. 69, 72, 74, 88

[69] A. Gaier, A. Asteroth, and J.-B. Mouret, "Automating representation discovery with map-elites," *ArXiv*, vol. abs/2003.04389, 2020. 70, 141

[70] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, (New York, NY, USA), p. 211–218, Association for Computing Machinery, 2011. 71, 72, 75, 94

[71] S. Kriegman, D. Blackiston, M. Levin, and J. Bongard, "Kinematic self-replication in reconfigurable organisms," *Proceedings of the National Academy of Sciences*, vol. 118, no. 49, p. e2112672118, 2021. 83

[72] T. Bartz-Beielstein, C. Doerr, J. Bossek, S. Chandrasekaran, T. Eftimov, A. Fischbach, P. Kerschke, M. López-Ibáñez, K. M. Malan, J. H. Moore, B. Naujoks, P. Orzechowski, V. Volz, M. Wagner, and T. Weise, "Benchmarking in optimization: Best practice and open issues," *ArXiv*, vol. abs/2007.03488, 2020. 125