UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Aumentación de conjuntos de datos utilizando redes neuronales generativas profundas distribuidas

Exploración del uso de algoritmos coevolutivos multiobjetivo en busca de mejoras en la diversidad de las muestras generadas

Agustín Felipe Mautone Estapé

Guillermo Ripa Budelli

Andres Vidal Berriel

Tutor: Sergio Nesmachnow

Ingeniería en Computación
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Setiembre de 2022

RESUMEN

El objetivo general de este proyecto es comprender el problema de la aumentación de conjuntos de datos y las debilidades presentadas en el entrenamiento de modelos de Redes Neuronales Generativas Antagónicas (GANs). El producto de software desarrollado es una nueva versión de Lipizzaner, un framework co-evolutivo para entrenamiento distribuidos de GANs, que soporta optimización multiobjetivo. Esto implicó la implementación de funciones de aptitud multidimensionales y la incorporación de Algoritmos Evolutivos Multiobjetivo (MOEAs). Los MOEAs implementados fueron NSGA–II, FV–MOEA y MOEA/D, representantes de las tres categorías principales de MOEA: basados en Pareto, basados en indicadores y basados en decomposición. Además, se adaptó el esquema de distribución de Lipizzaner al paradigma de pasaje de mensajes con MPI. El proyecto se enfocó en el problema de la generación de conjuntos de imágenes diversos, por lo que se agregaron funciones de costo en diversdad (E–GAN y GDPP) y métricas de evaluación para estudiar la fidelidad (densidad y FID) y la diversidad (cubrimiento) de los datos generados. Los MOEAs desarrollados se evaluaron sobre el conjunto de datos CelebA y se compararon entre sí y con la versión original de Lipizzaner. Se obtuvieron resultados favorables a la superioridad de MOEA/D en cubrimiento y FID. Sin embargo, la versión original de Lipizzaner obtuvo mejores resultados en densidad que los MOEAs propuestos. En general, se concluyó que es posible utilizar MOEAs para mejorar la diversidad de los datos generados por modelos entrenados en Lipizzaner, a costa de una reducción razonable en fidelidad. Esto motiva a la continuidad de esta investigación y a la mejora de la solución diseñada, para lo que se proponen posibles líneas de trabajo futuro.

Palabras claves:
Algoritmos Evolutivos, Optimización Multiobjetivo, Redes Generativas Antagónicas, Aumentación de conjuntos, Computación de alto desempeño.

Lista de figuras

Lista de tablas

Lista de siglas

ACGAN Auxiliary Classifier Generative Adversarial Network
ADASYN Adaptive Synthetic
AE Algoritmos Evolutivos
API Application Programming Interface
BAGAN Balancing Generative Adversarial Network
CGAN Conditional Generative Adversarial Network
CIFAR Canadian Institute for Advanced Research
CNN Convolutional Neural Network
CelebA CelebFaces Attributes
DCGAN Deep Convolutional Generative Adversarial Network
DPP Determinantal Point Process
E-GAN Evolutionary Generative Adversarial Network
FID Fréchet Inception Distance
FV–MOEA Fast Hypervolume Indicator-based MOEA
GAN Generative Adversarial Network
GDPP Generative Determinantal Point Process
GTSRB German Traffic Sign Recognition Dataset
HTTP Hypertext Transfer Protocol
HV Hipervolumen
IP Identity Protocol
KS Kolmogorov-Smirnov
LAPGAN Laplacian Generative Adversarial Network
MLP Multi-Layer Perceptron
MNIST Modified National Institute of Standards and Technology
MOEA Multi Objective Evolutive Algorithm
MOEA/D MOEA Basado en Decomposición

MPI Message Passing Interface

NSGA Nondominated Sorting Genetic Algorithm

NSGA–II Nondominated Sorting Genetic Algorithm 2

PAES Pareto Archived Evolution Strategy

SLURM Simple Linux Utility for Resource Management

SMOTE Synthetic Minority Over-sampling Technique

SMS-EMOA S-Metric Selection Evolutionary Multiobjective Optimization Algorithm

SPEA Strength Pareto Evolutionary Algorithm

TVD Total Variation Distance

VEGA Vector Evaluated Genetic Algorithms

Tabla de contenidos

Capítulo 1

Introducción

En este proyecto de grado se plantea el estudio de técnicas de aumentación de datos como apoyo al entrenamiento de modelos de aprendizaje profundo. En este contexto, se abordan técnicas basadas en modelos generativos, específicamente el modelo de Redes Generativas Antagónicas (GANs, por su sigla en inglés Generative Adversarial Network), orientados a la aumentación de conjuntos de imágenes. El marco de trabajo de la investigación es el algoritmo de entrenamiento de GANs propuesto por el framework Lipizzaner [67]. Se desarrolla una extensión del framework Lipizzaner para habilitar la construcción de mejores modelos para la aumentación de datos.

El aprendizaje profundo es un conjunto de técnicas de aprendizaje automático cuyas aplicaciones incluyen la predicción de conceptos y el reconocimiento de patrones a partir de grandes volúmenes de datos. La principal característica de estas técnicas es la incorporación de múltiples capas de procesamiento no lineal, comúnmente dispuestas en modelos de redes neuronales. La adopción de técnicas de aprendizaje profundo se vio impulsada en los últimos años por el aumento exponencial del poder de cómputo y disponibilidad de datos. Debido a la alta cantidad de parámetros que deben ajustar durante su entrenamiento, los modelos de redes neuronales son altamente sensibles a la cantidad de datos disponibles, los cuales pueden ser escasos en varias áreas de aplicación, como la medicina y la astronomía.

Una de las patologías que padecen los algoritmos de aprendizaje profundo al ser entrenados con una cantidad insuficiente de datos es el sobreajuste. Los modelos sobreajustados presentan poca capacidad de extrapolación, por lo que muestran buen desempeño en datos conocidos pero obtienen inferiores

resultados al abordar nuevos escenarios. La aumentación de datos comprende una serie de técnicas para incrementar la cantidad de observaciones de un conjunto de datos, con el fin de reducir el sobreajuste en los modelos que son entrenados sobre éste.

Existen diversos métodos para aumentar conjuntos de imágenes. El uso de modelos generativos, particularmente de GANs, se ha visto impulsado recientemente como método de aumentación [74]. Los modelos generativos tienen como objetivo aprender las características de la distribución subyacente a los datos con los cuales son entrenados. Por lo tanto, si los datos de entrenamiento fueran representativos de la distribución real de la que provienen, el modelo generativo (o generador) sería capaz de crear observaciones artificiales, pero realistas. El generador entrenado permitiría, entonces, componer un nuevo conjunto de datos con un mayor número de ejemplares, lo que facilitaría el posterior entrenamiento de otro modelo de aprendizaje profundo con menor sobreajuste.

Si bien el entrenamiento de modelos generativos es una tarea inherentemente no supervisada, las GANs resuelven este problema introduciendo un segundo modelo auxiliar, denominado discriminador, que retroalimenta al modelo generador con información sobre la calidad de las imágenes generadas. Mientras el generador produce datos sintéticos a partir de ruido, el discriminador conoce los datos reales y aprende a diferenciarlos de los artificiales. Siguiendo un proceso de aprendizaje antagónico, el discriminador es entrenado de forma supervisada aprendiendo a distinguir entre imágenes reales e imágenes sintéticas (producidas por el generador) y el error en el discriminador se propaga hacia el generador, permitiéndole aprender a generar imágenes que el discriminador no sepa distinguir de las reales. El entrenamiento converge cuando el resultado del discriminador se ve degenerado al lanzamiento de una moneda, esto es, a decidir aleatoriamente si una imagen es real o no.

Las GANs han sido ampliamente utilizadas para la aumentación de conjuntos de imágenes. La bibliografía es extensa en aplicaciones de estos modelos en la medicina, como clasificación de radiografías [7, 48, 76, 78] e imágenes de microscopios [18] y análisis de señales biológicas [32, 44, 59]. También existen aplicaciones sobre reconocimiento de voz [38, 72, 73, 83, 85]; sistemas de transporte [10, 93]; detección de anomalías y fraude [47, 69]; biometría [81]; meteorología [84]; inspección de equipajes en sistemas de seguridad [88] y detección de fallas en maquinaria [91]. La cantidad de recientes aplicaciones, que

además resultan promisorias para mitigar los problemas de disponibilidad de datos en múltiples áreas de trabajo, evidencia la relevancia del desarrollo en GANs y la importancia de obtener buenos resultados en su entrenamiento.

En la literatura se propone la utilización de GANs para obtener conjuntos de datos aumentados y así mitigar problemas conocidos en el entrenamiento de modelos de aprendizaje profundo, pero el entrenamiento de GANs también padece de sus propias patologías. Dos ejemplos típicos de patologías son el desvanecimiento de gradientes, cuando el desempeño del discriminador impide el aprendizaje del generador, o el colapso modal, cuando el generador aprende apenas, o colapsa sobre, un subespacio de la distribución subyacente a los datos. Lipizzaner es un framework disponibilizado como una biblioteca de código abierto que implementa un algoritmo para entrenar GANs evitando las patologías recién mencionadas.

Lipizzaner maximiza la fidelidad de los datos generados mediante el entrenamiento de una población de GANs en un marco coevolutivo y distribuido, con el cual se incentiva implícitamente el aspecto de diversidad que evita el colapso modal. No obstante, es de interés incorporar explícitamente a la diversidad durante el entrenamiento para incentivar un total cubrimiento de la distribución de los datos de entrenamiento. Entonces, los objetivos concretos planteados en este proyecto de grado son: extender el algoritmo de optimización de Lipizzaner para dar soporte a un paradigma de optimización multiobjetivo y aplicar esta extensión a la optimización conjunta de fidelidad y diversidad de las imágenes generadas por los modelos entrenados bajo el nuevo paradigma.

Como resultado de este proyecto de grado se desarrolla una extensión del framework Lipizzaner con: 1) tres algoritmos de optimización multiobjetivo aplicados sobre las GANs, 2) dos funciones de costo de redes diseñadas específicamente para la medición de diversidad, 3) dos métricas de evaluación capaces de evaluar conjuntos sintéticos de imágenes por su relación de fidelidad y diversidad a un conjunto de imágenes base y 4) un cambio de paradigma de computación paralela hacia una solución ampliamente adoptada en centros de cómputo de supercomputación. Como principal resultado del proyecto, se destaca que uno de los algoritmos de optimización multiobjetivo desarrollados alcanza resultados comparables en fidelidad a la línea base, la implementación original de Lipizzaner, con mejores resultados de diversidad y mayor consistencia entre experimentos.

La estructura del documento es la siguiente. En el Capítulo 2 se introducen y detallan los objetivos del proyecto de grado. Luego, el Capítulo 3 presenta los fundamentos teóricos requeridos para la realización del proyecto. Se abordan los temas de Algoritmos Evolutivos Multiobjetivo (MOEAs), GANs, Lipizzaner, técnicas para aumentación de datos y las métricas y contrastes utilizados para la evaluación de los experimentos a realizar. En el Capítulo 4 se describe la metodología aplicada durante la experimentación, la extensión del framework y el relevamiento del estado del arte. Adicionalmente, se introducen los repositorios de datos a utilizar durante la experimentación. El Capítulo 5 aborda la implementación de la extensión del framework detallando la adición del soporte para MOEA y la migración a el estándar Message Passing Interface (MPI) [29]. El Capítulo 6 reporta y analiza los resultados de los experimentos realizados para evaluar el desempeño del sistema desarrollado. El capítulo se divide en cuatro secciones: un análisis de las métricas de evaluación, la configuración paramétrica, la evaluación de desempeño y la evaluación de escalabilidad. Finalmente, el Capítulo 7 formula las conclusiones del proyecto de grado y las principales líneas de trabajo futuro.

Capítulo 2

Objetivos

El objetivo general de este proyecto de grado es el estudio del problema de aumentación de datos con redes neuronales generativas antagónicas. En este contexto, se busca comprender el problema de la aumentación de conjuntos de datos de imágenes y las debilidades presentadas en el entrenamiento de GANs, para formular posibles mejoras a su entrenamiento. Específicamente, este proyecto de grado se enfoca en los problemas asociados a la diversidad de los datos sintéticos generados por GANs y a la extensión del algoritmo coevolutivo de Lipizzaner para la inclusión explícita de la diversidad en su proceso de entrenamiento. La inclusión de la diversidad en el entrenamiento implica la adaptación de Lipizzaner para resolver problemas de optimización multiobjetivo.

Los objetivos específicos del proyecto de grado, que están vinculados a la formulación del problema, a la propuesta de soluciones, y a la implementación y evaluación de las soluciones son:

1. estudiar el problema de optimización multiobjetivo y su abordaje aplicando Algoritmos Evolutivos (AE);
2. estudiar el modelo de redes neuronales generativas antagónicas, profundizando en su entrenamiento y en sus patologías más comunes, prestando especial énfasis a soluciones que busquen aumentar la diversidad de los datos generados;
3. estudiar el problema de la aumentación de datos, de modo de caracterizar la aplicabilidad de este proyecto y extender la compresión sobre las aplicaciones de otras soluciones propuestas;

4. estudiar funciones de costo que incluyan diversidad para guiar el entrenamiento de las GANs y métricas para la evaluación de modelos generativos que sean funcionales a la metodología propuesta;
5. adaptar el mecanismo de distribución del framework Lipizzaner para adoptar el estándar Message Passing Interface (MPI), para permitir la ejecución de Lipizzaner en plataformas de supercomputación como ClusterUY [56];
6. implementar el soporte para optimización multiobjetivo en Lipizzaner, incluyendo variantes de algoritmos que permitan cubrir el estado del arte de MOEAs;
7. estudiar la incidencia de los parámetros relevantes en los algoritmos implementados;
8. evaluar los algoritmos implementados tomando en cuenta la fidelidad y diversidad de los datos generados, comparándolos entre sí y con la versión original de Lipizzaner; y
9. evaluar la escalabilidad de las soluciones propuestas en comparación a la versión original de Lipizzaner, con el fin de verificar que los nuevos algoritmos no la afectan negativamente.

Capítulo 3

Fundamentos teóricos

En este capítulo se presentan los fundamentos teóricos del proyecto de grado, incluyendo definiciones y conceptos relevantes. Inicialmente, se enuncia el problema de la optimización multiobjetivo, discutiendo sus particularidades en comparación con la optimización de funciones unidimensionales y definiendo los conceptos de dominancia entre soluciones y del Frente de Pareto. Luego, se profundiza en los AEs y MOEAs. En el contexto de los MOEAs, se presentan diferentes clases y se detalla la algoritmia de algunos de ellos: el Nondominated Sorting Genetic Algorithm 2 (NSGA–II) [15] como representante de los MOEAs basados en Pareto, el Fast Hypervolume Indicator-based MOEA (FV–MOEA) [40] como representante de los MOEAs basados en indicadores y el MOEA Basado en Decomposición (MOEA/D) [90]. A continuación, se profundiza sobre modelos generativos, se presenta el framework Lipizzaner para entrenamiento distribuido de GANs y se define el problema de la aumentación de datos. Finalmente, se presentan funciones de costo para la optimización de este tipo de modelos y métricas de evaluación para cuantificar su desempeño.

3.1. Optimización multiobjetivo

En esta sección se presentan los fundamentos teóricos sobre la optimización mulitobjetivo. Primero se define el problema de la optimización multiobjetivo y luego se profundiza en la rama de algoritmos evolutivos multiobjetivo. Se presentan las variantes NSGA–II, FV–MOEA y MOEA/D que representan a las clases de MOEA basados en Pareto, en indicadores y en descomposición respectivamente.

3.1.1. Definición del problema

Un problema de optimización general consiste en encontrar soluciones óptimas en términos de la minimización de una función objetivo, dado un conjunto de restricciones que éstas deben cumplir. Formalmente, se buscan soluciones al problema planteado en la Ecuación 3.1 siendo f la función objetivo y h y g las restricciones que determinan la región de factibilidad. Una solución x se considera factible cuando pertenece a esta región. Cuando el conjunto de llegada de f es unidimensional, el problema se considera de optimización monobjetivo y puede presentar diversos óptimos locales o globales. Sin embargo, cuando f toma valores en un espacio multidimensional, el problema de optimización es multiobjetivo y a sus dimensiones se les denomina objetivos del problema. En el caso multiobjetivo no suele ser posible encontrar soluciones que minimicen simultáneamente todos los objetivos [75], por lo que la noción de optimalidad de una solución se redefine de acuerdo a la noción de dominancia.

$$\text{mín}\, f(x) \quad \text{sujeto a} \quad h(x) = 0 \quad \wedge \quad g(x) \leq 0 \tag{3.1}$$

Sea $f(x) = (f_1(x), \ldots, f_n(x))$ el objetivo de un problema multiobjetivo n-dimensional, se define la relación de dominancia [87] entre dos soluciones x e y como presenta la Ecuación 3.2.

$$\begin{aligned} x \text{ domina a } y &\iff (\forall i)\ f_i(x) \leq f_i(y) \ \wedge\ (\exists j) : f_j(x) < f_j(y) \\ x \text{ no domina a } y &\iff (\exists i) : f_i(x) > f_i(y) \ \vee\ (\forall j)\ f_j(x) \geq f_j(y) \end{aligned} \tag{3.2}$$

La segunda condición de la Ecuación 3.2 caracteriza a esta relación como un orden parcial, lo que implica que pueden haber dos soluciones que no se dominan una a otra . La no dominancia ocurre o bien cuando ambas soluciones son iguales según los objetivos, o bien cuando ambas soluciones superan a la otra en alguno de ellos. De la relación de dominancia surge la noción de solución óptima como aquella que no es dominada por ninguna otra. La noción de solución óptima, en conjunto con el orden parcial determinando por la relación de dominancia, permite definir el conjunto de soluciones óptimas, también llamado conjunto de soluciones no dominadas, conjunto de soluciones Pareto-óptimas [75, 87] o frente de Pareto del problema en cuestión (Ecuación 3.3):

$$FP = \{\, x \ :\ \ (\forall y \text{ factible})\ y \text{ no domina a } x \,\} \tag{3.3}$$

Para resolver un problema de optimización multiobjetivo se busca calcular su frente de Pareto asociado. En general, se requiere encontrar soluciones para un problema de optimización con el objetivo de apoyar a la toma de decisiones que se reducen a la elección de una o un conjunto reducido de soluciones (sub)óptimas. En un problema multiobjetivo, frente a la presencia de un conjunto potencialmente infinito de soluciones óptimas, es necesario establecer un criterio de selección. El proceso de selección de soluciones conlleva un compromiso entre los objetivos del problema y puede ser conducido a *a priori*, como parte del algoritmo de optimización, o *a posteriori*, por el tomador de decisiones.

3.1.2. Algoritmos Evolutivos Multiobjetivo

Los algoritmos evolutivos son métodos de optimización basados en heurísticas inspiradas por el concepto de evolución en la biología. La idea básica de modelización de un algoritmo evolutivo, representada en el esquema de la Figura 3.1, consiste en mantener una población de individuos y simular su evolución en la naturaleza: los individuos pueden mutar, generar descendencia, interactuar entre sí y sobrevivir según su aptitud relativa. El proceso evolutivo se repite hasta alcanzar algún criterio de parada y a cada iteración se le llama generación.

Las componentes principales de un algoritmo evolutivo se definen como:

1. Población de individuos: esta es una representación, directa o indirecta, de la solución al problema.
2. Proceso de selección, basado en una métrica de aptitud sobre las soluciones candidatas.
3. Proceso de transformación, mediante el cual se construyen nuevos individuos a partir de la "composición genética" de las soluciones existentes.

La forma en que los individuos interactúan dentro de la población se define por artefactos del algoritmo, denominados operadores evolutivos. En general, se incorporan operadores de inicialización, que determinan la población inicial; de mutación, que permiten generar nuevas soluciones a partir de una existente; de recombinación o cruzamiento, que permiten fusionar soluciones para generar nuevas que posean características de todas ellas y de selección, que permiten seleccionar las soluciones que trascenderán entre generaciones.

Figura 3.1: Ciclo de un algoritmo evolutivo.

Los operadores evolutivos se diseñan a modo de mantener poblaciones diversas para evitar el estancamiento en óptimos locales. Las poblaciones paulatinamente convergen a soluciones de alto desempeño o, equivalentemente, con buena aptitud. Las características de los AEs, especialmente el mantenimiento de una población diversa como aspecto central, inspiraron su utilización para la resolución de problemas de optimización multiobjetivo. Un problema de optimización multiobjetivo puede traducirse a un modelo evolutivo codificando las soluciones como individuos y definiendo una función de aptitud inversamente proporcional al objetivo multidimensional.

Diversos MOEAs se han propuesto para estimar el frente de Pareto de una función multiobjetivo. Esencialmente, es posible clasificar los MOEA según la incidencia de la noción de optimalidad de Pareto en su algoritmia. Por un lado, los MOEAs basados en Pareto utilizan explícitamente la dominancia entre soluciones para converger al frente de Pareto. Ejemplos de algoritmos basados en Pareto son NSGA y NSGA–II. Por otro lado, los algoritmos no basados en pareto utilizan otras heurísticas para explorar de manera inteligente el espacio de soluciones factibles. Los algoritmos que no consideran explícitamente la noción de dominancia de Pareto se subclasifican en dos categorías: 1) algoritmos basados en indicadores, como el hipervolúmen y 2) algoritmos basados en descomposición (MOEA/D).

A seguir, se presentan las tres clases de algoritmos evolutivos, enfatizando en un algoritmo representante de cada una de ellas: NSGA–II como algoritmo basado en Pareto; FV–MOEA como algoritmo basado en indicadores y MOEA/D como algoritmo basado en decomposición.

3.1.3. MOEAs basados en Pareto: NSGA–II

Los algoritmos evolutivos basados en Pareto se caracterizan por utilizar explícitmente los conceptos de dominancia entre soluciones y de Frente de Pareto en su proceso de optimización. Esta clase de algoritmo constituye a los primeros intentos de incorporar funciones de aptitud efectivamente multidimensionales.

En contraposición a los métodos clásicos de resolución de problemas de optimización multiobjetivo, Schaeffer propuso en 1984 el Vector Evaluated Genetic Algorithms (VEGA) [66] que, si bien fue el primer MOEA en considerar explícitamente la función objetivo como un vector, presentaba fuertes sesgos a ciertas regiones del frente de Pareto. Como posible solución para mi-

tigar este fenómeno, Goldberg [26] propuso en 1989 la incorporación, en el ciclo evolutivo, de procedimientos de ordenación según el orden de dominancia y de penalización de soluciones en regiones de alta densidad de la población (niching). El objetivo del primer procedimiento sería el de explicitar la idea de optimalidad de Pareto de las soluciones en el algoritmo; mientras que el niching se aplicaría para estimular el cubrimiento del frente.

El Nondominated Sorting Genetic Algorithm (NSGA) es un algoritmo evolutivo propuesto por Srinivas y Deb en el 1994 e inspirado por la propuesta de Goldberg para resolver los sesgos en la exploración del espacio de soluciones por el algoritmo VEGA. La principal característica de NSGA es la introducción de una función de aptitud que asigna aptitud a las soluciones según la distancia máxima, en órdenes de dominancia, entre la solución en cuestión y alguna de las soluciones no dominadas. La función de aptitud se realiza computando el conjunto de soluciones no dominadas para la población, asignándole el mayor valor posible de aptitud y removiéndolo temporalmente para repetir el proceso en el restante de la población. De esta forma, en la i-ésima iteración se consiguen todas las soluciones que en la $(i-1)$-ésima iteración solamente eran dominadas por las soluciones no dominadas. En cada paso, se le asigna a las soluciones no dominadas una aptitud menor al asignado a las extraídas en el paso anterior, estableciendo un ranking según la relación de orden parcial determinada por la dominancia.

Además de la explicitud del orden de dominancia entre las soluciones al asignar la aptitud, NSGA incorpora el niching de soluciones mediante fitness sharing. La técnica de niching consiste en penalizar la aptitud de las soluciones de forma directamente proporcional a la cantidad de soluciones vecinas en un entorno de radio $\sigma_{sharing}$. De esta forma, se favorecen aquellas soluciones en regiones menos densas de la población. La distancia en consideración es la distancia fenotípica, esto es, calculada sobre las representaciones (genes) de las soluciones.

Si bien esta definición de aptitud direcciona el proceso evolutivo uniformemente hacia las áreas no dominadas de la región factible, NSGA es objetivo de crítica debido a la falta de un elitismo más acentuado que acelere la convergencia; a la necesidad de especificar el parámetro de sharing y al alto costo computacional del algoritmo de ordenación. Con el fin de solucionar los problemas de NSGA, Deb, Pratab y Agarwal propusieron en el 2002 una nueva versión, NSGA–II.

En NSGA–II, el proceso de ordenación por orden de dominancia se redefine a costa de un compromiso entre complejidad computacional y almacenamiento en memoria. Se reduce un orden de complejidad computacional, a la vez que se aumenta un orden en el requisito de memoria: sea N el tamaño de la población y M la cantidad de objetivos, mientras NSGA presentaba complejidad computacional de orden $O(MN^3)$ y costo de almacenamiento de orden $O(N)$, NSGA–II realiza un procedimiento cuyo resultado es el mismo con costo computacional de orden $O(MN^2)$ y requisito de memoria de orden $O(N^2)$.

Para resolver la sensibilidad de NSGA al parámetro de sharing y a la distancia entre individuos, en NSGA–II se introduce una nueva técnica de niching, denominada crowding. Esta técnica se diferencia del fitness sharing en dos aspectos principales: no requiere configuración paramétrica y puede utilizar únicamente los valores de los objetivos, por lo que permite desacoplar el algoritmo de la distancia fenotípica. El procedimiento inicia ordenando las soluciones según su desempeño en cada objetivo. El índice de crowding, como estimador de la densidad de la población local a una solución, se define como infinito para los extremos, mientras que para el resto se promedian las diferencias entre los objetivos de la solución anterior y la siguiente.

El índice de crowding i_c es utilizado en conjunto al ranking de dominancia i_r para guiar el proceso de selección de manera elitista. Ambos valores se componen en un operador de comparación $\succ$ entre dos soluciones x e y definido en la ecuación Ecuación 3.4

$$x \succ y \iff i_r(x) < i_r(y) \ \vee \ (i_r(x) = i_r(y) \ \wedge \ i_c(x) > i_c(d)) \tag{3.4}$$

Esto es, $x \succ y$ cuando y domina a x o cuando pertenecen al mismo frente de dominancia e y está en una región menos densa que x. Una vez generada la descendencia de la población actual, se selecciona según este operador a los N individuos. El procedimiento se realiza de forma eficiente manteniendo el tamaño de cada frente. Se agregan tantos frentes cuantos sea posible a la siguiente población. En el caso de que sea necesario partir un frente para completar los N individuos, se ordena a las soluciones de ese frente según i_c y se seleccionan cuantos sean necesarios.

En general, NSGA–II presenta resultados prometedores en comparación con otros MOEAs elitistas como el Pareto Archived Evolution Strategy (PAES) [41] y el Strength Pareto Evolutionary Algorithm (SPEA) [94]. Además, NSGA–II presenta desempeño equiparable al del sucesor del último, SPEA-2, especialmente en espacios objetivos de baja dimensionalidad [95].

3.1.4. MOEAs basados en indicadores: FV–MOEA

La ventaja de los algoritmos basados en indicadores proviene de usar las medidas de desempeño como Hipervolumen (HV) y la métrica epsilon directamente en la selección de descendencia, maximizando directamente las funciones objetivo.

S-Metric Selection Evolutionary Multiobjective Optimization Algorithm (SMS-EMOA) [6] es un algoritmo basado en indicadores que propone maximizar directamente el hipervolumen dominado. El proceso se desarrolla en un formato de estado estable combinando un operador de selección basado en la medida de hipervolumen y el concepto de clasificación no-dominada. El formato de estado estable define que se incluya (y retire) un único nuevo individuo a la población en cada iteración, por lo que SMS-EMOA tiene la restricción de depender fuertemente de operaciones de mutación que generen gran diversidad en la población. Además, al requerir calcular el hipervolumen en cada iteración para todos los frentes en la clasificación no-dominada incurre en un alto costo computacional.

La dependencia de SMS-EMOA de operadores agresivos de mutación y su alto costo computacional motivo a que Jiang y col. [40] presentaran el algoritmo Fast Hypervolume Indicator-based MOEA (FV–MOEA). FV–MOEA busca actualizar de forma eficiente las contribuciones que las diferentes soluciones realizan al hipervolumen. El método utilizado para medir las contribuciones de las soluciones al hipervolumen reduce el tiempo de cómputo del cálculo exacto de las contribuciones mediante la eliminación de soluciones irrelevantes y el traspaso de información de contribuciones entre iteraciones. Por último, reduce los tiempos de cómputo al momento de generar cambios en la población utilizando un modelo de lote similar al de NSGA–II.

FV–MOEA no requiere de ningún tipo de inicialización o método de reproducción especial de la población, permitiendo una fácil adaptación dentro del algoritmo co-evolutivo de Lipizzaner.

En específico, FV–MOEA se concentra en la selección de soluciones. Utiliza el método de clasificación no-dominada de NSGA–II para ordenar las soluciones en sus diferentes rangos definidos por los frentes de Pareto consecutivos a los que pertenecen, seleccionando para la siguiente población a los individuos de los primeros k+1 frentes $F_1, F_2, ..., F_k, ...F_n$ tal que $|\bigcup_{i=1}^{k} F_i| < NP$ y $|\bigcup_{i=1}^{k+1} F_i| \geq NP$ siendo NP tamaño de población. Es recién en el frente k+1 donde se aplica el cálculo de contribuciones al hipervolumen que cada solución del frente aporta.

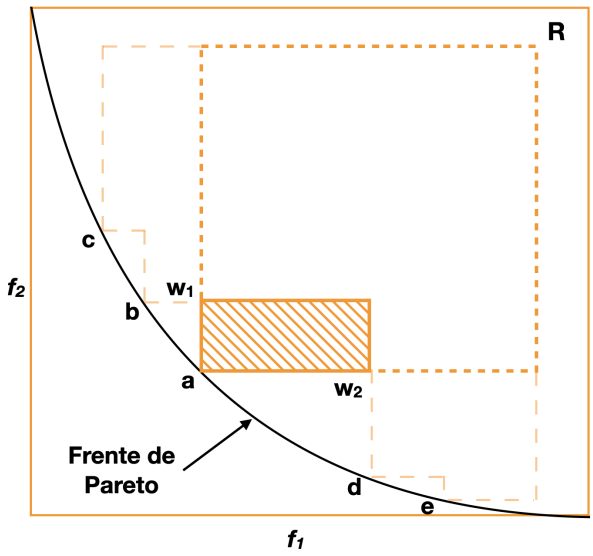


Figura 3.2: Contribución al hipervolumen para una solución a.

Tomando la Figura 3.2 como ejemplo ilustrativo, siendo las soluciones no dominadas $S = a, b, c, d, e$ pertenecientes al frente de Pareto óptimo, $R = r$ el punto de referencia único tomando en base a los valores máximos (o mínimos) de los objetivos de todas las soluciones del frente y el hipervolumen alcanzado por las población y el punto de referencia como el área de la figura definida por el polígono *cbadeRc*. Una forma directa para calcular la contribución de una solución a toma la forma de la Ecuación 3.5 planteada por Beume, Naujoks y Emmerich en el algoritmo SMS-EMOA donde HV corresponde al hipervolumen.

$$v(a, S, R) = HV(S, R) - HV(S - a, R) \tag{3.5}$$

El algoritmo de FV–MOEA presenta una alternativa donde se consiguen los valores *peores-no-dominados* de la solución a, es decir las máximos valores de sus vecinos, no dominados por la solución. En la Figura 3.2 representados por w_1 y w_2. Desde este punto la contribución, en vez de requerir el cálculo de un nuevo hipervolumen entero se define simplemente por el hipervolumen de un único punto, logrando la Ecuación 3.6. De esta forma se eliminan iterativamente las soluciones de menor contribución hasta alcanzar $|\bigcup_{i=1}^{k+1} F_i| = NP$.

$$v(a, S, R) = HV(a, R) - HV(\{w_1, w_2\}, R) \tag{3.6}$$

3.1.5. MOEAs basados en descomposición: MOEA/D

Los algoritmos basados en descomposición buscan reducir la complejidad computacional de la optimización multiobjetivo mediante la descomposición en subproblemas de optimización unidimensional. Se basan en el principio de que una solución pareto óptima, dadas las condiciones apropiadas, también será solución de un problema cuyo objetivo es la agregación de las funciones a optimizar. Además de la ventaja en complejidad computacional, el cambio a problemas escalares permite la aplicación de operadores de algoritmos evolutivos tradicionales y de técnicas de optimización unidimensional.

El algoritmo original propuesto por Zhang y Li propone asignar un vecindario a cada subproblema, lo que restringe el intercambio de información entre subproblemas. La propuesta consiguió resultados mejores o similares a NSGA–II en varios problemas con un costo computacional altamente inferior, además de conseguir una distribución bastante uniforme de soluciones. Para determinar los vecindarios se propone utilizar la distancia entre los vectores de agregación asignados a cada subproblema, ya que se supone que las soluciones óptimas a dos subproblemas con vectores cercanos serán similares.

Para la descomposición del problema de optimización multiobjetivo, el algoritmo original propone tres variantes: suma ponderada, Tchebycheff e intersección de borde. En Lipizzaner se cuenta con una implementación de MOEA/D de suma ponderada con muestreo uniforme de los pesos, que asigna el vector de pesos dependiendo de la posición en la grilla. Por ejemplo, para un problema de dos objetivos y una grilla de $n \times n$ el valor de la agregación en la celda (i, j) será: $F(x) = \frac{i}{n} f_1(x) + \frac{j}{n} f_2(x)$.

La selección de los pesos en MOEA/D es muy importante para la obtención de soluciones distribuidas uniformemente en el frente. Utilizar pesos tomados de forma uniforme consigue buenos resultados si la geometría del frente es sencilla, sin embargo, ante problemas con frentes complejos (no continuos, con escalas muy distintas y más) esta forma de seleccionar pesos puede llevar a no encontrar las soluciones de mayor calidad. Para encontrar siempre las soluciones de mayor calidad se han publicado diversos artículos proponiendo formas innovadoras de asignar los pesos [39]. Una de las lineas de investigación busca aprovechar las soluciones encontradas para detectar que tan dispersa esta cada solución llevando un historial de las soluciones encontradas. Con este historial ajusta los pesos de los subproblemas para enfocar la búsqueda en las zonas en las que menos soluciones fueron encontradas.

Una extensión de MOEA/D fue propuesta por Farias [23], en la cual mantiene una población externa de soluciones no dominadas, la cual utiliza para generar nuevos vectores de pesos. Este propone mejoras al algoritmo propuesto por Qi [60] y utiliza el enfoque de descomposición de Tchebycheff. La descomposición del problema multiobjetivo se plantea según la Ecuación 3.7 donde λ es el vector de pesos y z^* un vector objetivo utópico, por ejemplo $z_j^* = \text{mín}\,\{f_j(\boldsymbol{x}) \mid \boldsymbol{x} \in \Omega\}$ en caso de minimización.

$$\begin{aligned} &\text{minimizar } g^{TCH}\left(x \mid \lambda, z^*\right) = \text{máx}_{1\leq j\leq m}\left(\lambda_j \left|f_j(\boldsymbol{x}) - z_j^*\right|\right), \\ &\text{sujeto a } \boldsymbol{x} \in \Omega \end{aligned} \tag{3.7}$$

El proceso de ajuste de pesos ocurre en dos pasos. Primero se eliminan soluciones de la población que se esta evolucionando ordenándolos según su nivel de dispersión (SL, por su sigla en inglés *Sparsity Level*) usando la Ecuación 3.8 y se eliminan el top 5 % de las soluciones con menor nivel de dispersión.

$$SL\left(ind^j, pop\right) = \prod_{i=1}^{m} L_2^{NN_i^j} \tag{3.8}$$

El segundo paso consiste en agregar nuevos vectores de pesos. Se consiguen las soluciones más dispersas de la población externa (usando la misma medida) y actualizándolos según 3.9, siendo z^* el vector objetivo utópico mencionando anteriormente y f_i^{sp} la evaluación de la función f_i en el individuo elegido para generar el nuevo vector de pesos. Este proceso no se realiza para el último 10 % de las generaciones.

$$\lambda^{sp} = \left(\frac{\frac{1}{f_1^{sp} - z_1^*}}{\sum_{k=1}^{m} \frac{1}{f_k^{sp} - z_k^*}}, \cdots, \frac{\frac{1}{f_m^{sp} - z_m^*}}{\sum_{k=1}^{m} \frac{1}{f_k^{sp} - z_k^*}} \right), \prod_{j=1}^{m} \left(f_j^{sp} - z_j^* \right) \neq 0 \tag{3.9}$$

El algoritmo de Farias presentó resultados superiores a otras metodologías en más del 70 % de los casos y se aclara que no presento mejoras para frentes de pareto convexos y disconexos. Además de que todavía queda espacio para investigar cual es la mejor configuración para la frecuencia de adaptación de los pesos.

Otra de las lineas de extensión de MOEA/D atacan el problema de utilización de recursos, ya que se ha notado que MOEA/D puede desperdiciar recursos evolucionando individuos que no son prometedores. La técnica predominante restringe la cantidad de subproblemas que se actualizan en cada iteración.

El artículo de Lavinas [45] mostró excelentes resultados comparado al estado del arte aplicando un algoritmo muy sencillo de asignación de recursos. El algoritmo consiste en introducir un parámetro *ps* para cada subproblema. Este parámetro determina la probabilidad de que un subproblema sea actualizado en la siguiente iteración. Los resultados del artículo mostraron que $ps = 0.1$ mejoraba los resultados respecto al algoritmo de MOEA/D original y venciendo a varios algoritmos en el estado del arte. Los investigadores dejaron además la puerta abierta a la investigación de utilizar un valor de *ps* variable a lo largo del entrenamiento.

3.2. Aprendizaje Automático

Esta sección introduce los conceptos generales sobre aprendizaje automático y aprendizaje profundo. Luego, se presentan las familias de modelos de aprendizaje profundo utilizadas en este proyecto de grado.

3.2.1. Conceptos generales sobre aprendizaje automático

El aprendizaje automático es una rama de la inteligencia artificial y la computación, esta rama incluye a los algoritmos capaces de aprender median-

te la experiencia y el uso de datos. La definición brindada por Goodfellow, Bengio y Courville es [27] *"Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas T y la medida de rendimiento P, si su desempeño en tareas en T, medido por P, mejora con la experiencia E"* .

Este proyecto de grado trabaja sobre la tarea (T) de síntesis de nuevos ejemplares. En este tipo de tareas el algoritmo es entrenado para generar ejemplares similares a los vistos en el conjunto de entrenamiento. La aplicación de aprendizaje automático a esta tarea es especialmente útil cuando se necesita generar una cantidad grande de nuevos ejemplares, donde hacerlo manualmente sería una tarea lenta y tediosa.

Para medir el rendimiento de los modelos entrenados (P) se utilizan medidas de similitud entre la distribución de los datos generados y la distribución de entrenamiento, como se profundiza en la sección 3.7.1.

Los conjuntos de datos (E) que se utilizan en este proyecto de grado no están etiquetados y se busca aprender sus propiedades estructurales, por lo que se puede categorizar como un algoritmo no supervisado [27].

Para resolver la tarea de síntesis de nuevos ejemplares, este proyecto de grado utiliza técnicas de aprendizaje profundo. El aprendizaje profundo es una rama del aprendizaje automático que se basa en la aplicación de redes neuronales profundas para intentar capturar la distribución subyacente de datos no estructurados. Las siguientes secciones explican los conceptos relacionados a la aplicación de aprendizaje profundo para el modelado de la tarea de síntesis.

3.2.2. Redes neuronales artificiales

Las redes neuronales artificiales (ANN, por su sigla en inglés Artificial Neural Networks) son modelos de aprendizaje automático capaces de procesar información en un proceso inspirado en el funcionamiento de las redes neuronales biológicas presentes en los cerebros animales. Una ANN esta compuesta por capas usualmente dispuestas en una capa de entrada, una capa de salida y múltiples capas intermedias. Cada capa esta compuesta por un conjunto de nodos llamados neuronas. Cada neurona esta conectada a las neuronas de la capa siguiente (similar a la sinápsis en el cerebro) y pueden tener un peso asociado a cada conexión entre neuronas basado en varios criterios.

Cuando una neurona recibe un valores de entrada y los procesa (proceso conocido como activación) el valor de salida se calcula como la suma de todas las activaciones que llegan a la neurona. En la formulación más básica de una ANN, cada activación es ponderada por el peso asociado a cada conexión y luego se aplica una función no lineal a la suma de las activaciones (por ejemplo la unidad rectificada lineal [1]), como se muestra en el diagrama de la Figura 3.3. El diagrama también presenta valores b_i, llamados sesgos de la red.

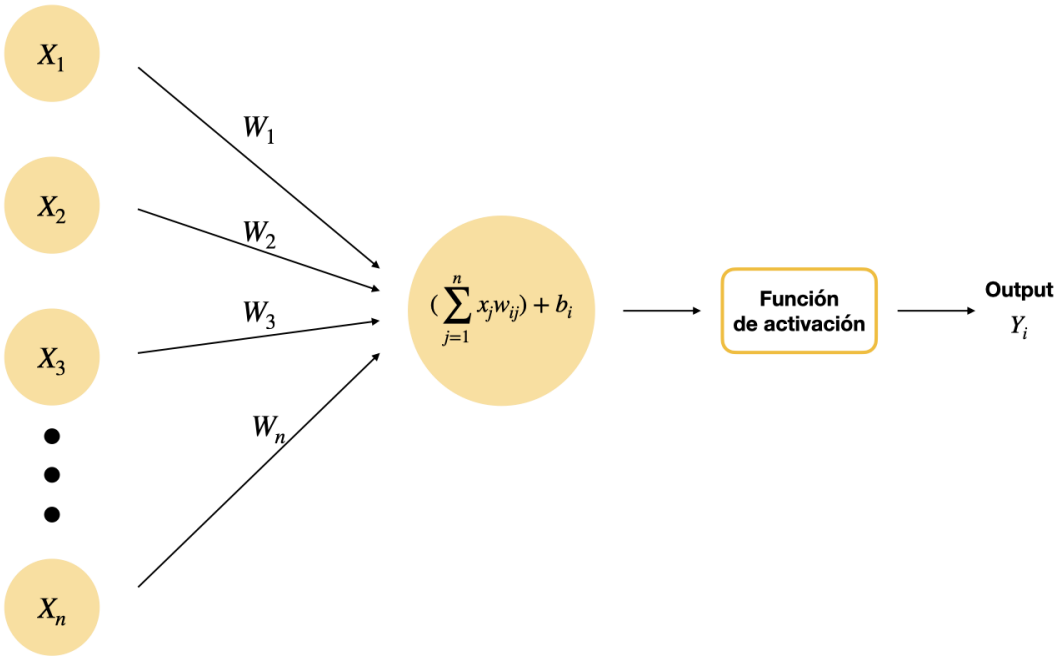


Figura 3.3: Activación de una neurona

Durante el entrenamiento se ajustan los parámetros de la red (el conjunto de pesos y sesgos asociados a las conexiones entre neuronas) para lograr reducir una función de costo. Los ajustes son realizados mediante el uso de descenso por gradiente y propagación hacia atrás [46].

Los perceptrones multi-capa (MLP) con más de una capa intermedia son el modelo básico del aprendizaje profundo. En los MLP la información fluye desde la entrada hasta la salida en un solo sentido, sin ningún tipo de ciclo. La primera capa de un perceptrón es denominada la capa de entrada y la última es la capa de salida. Todas las capas intermedias se denominan capas ocultas. La cantidad de capas ocultas se denomina profundidad y de aquí surge el término aprendizaje profundo.

3.2.3. Redes neuronales convolucionales

Las capas convolucionales son lo que distingue a las redes neuronales convolucionales (CNN) del resto de las redes neuronales artificiales. Estas redes tienen la capacidad de analizar datos correlacionados espacialmente y han demostrado tener la capacidad de capturar contenido semántico de imágenes y otros tipos de datos de alta dimensionalidad [35].

Las capas convolucionales consisten de un filtro que se aprende durante el entrenamiento. El filtro es convolucionado con los datos de entrada durante la inferencia de la red. Un filtro es un conjunto de pesos dispuesto en forma de matriz. La operación de convolución entre este filtro y el vector de entrada (una imagen puede ser un vector de entrada) es equivalente a deslizar el filtro a través del vector de entrada, calculando el producto escalar entre ambos en cada posición. La figura 3.4 muestra una aplicación del filtro.

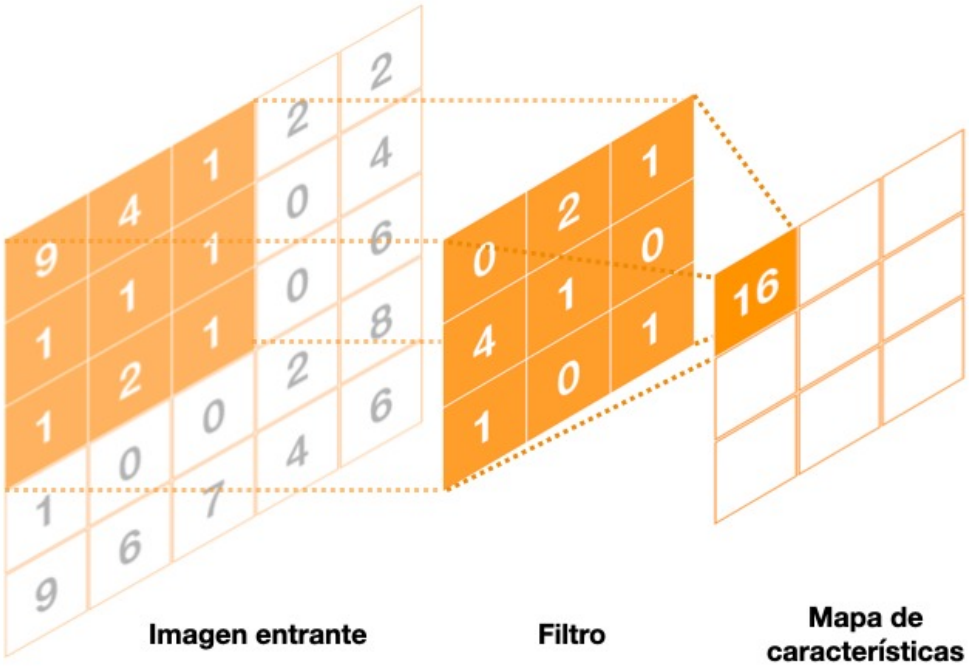


Figura 3.4: Capa de convolución

Luego de la convolución de un filtro con la entrada se obtiene un *mapa de características*. Al concatenar varias capas convolucionales se logra que los mapas de características condensen información semántica sobre la imagen, por ejemplo: bordes, esquinas, colores y otros [89].

Las redes convolucionales reducen la cantidad de parámetros requeridos para procesar entradas de alta dimensionalidad gracias a que los pesos del filtro son compartidos. Estas redes no requieren una capa de entrada con el mismo tamaño que los datos de entrada, como sí sería necesario en un MLP. Por otra parte, un MLP no aprovecha la estructura espacial de los datos, mientras que el filtro deslizante permite mantener la localidad entre capas.

Otra variación de las capas convoluciones son las capas convoluciones-transpuestas, las cuales aumentan la dimensión de los mapas de características usando parámetros que se aprenden durante el entrenamiento. Las capas con convoluciones transpuestas son comunes en redes de segmentación, generación o encoder-decoder [21, 61]. La Figura 3.5 presenta como una convolución con un kernel de 2×2 se aplica a un vector de entrada de 2×2.

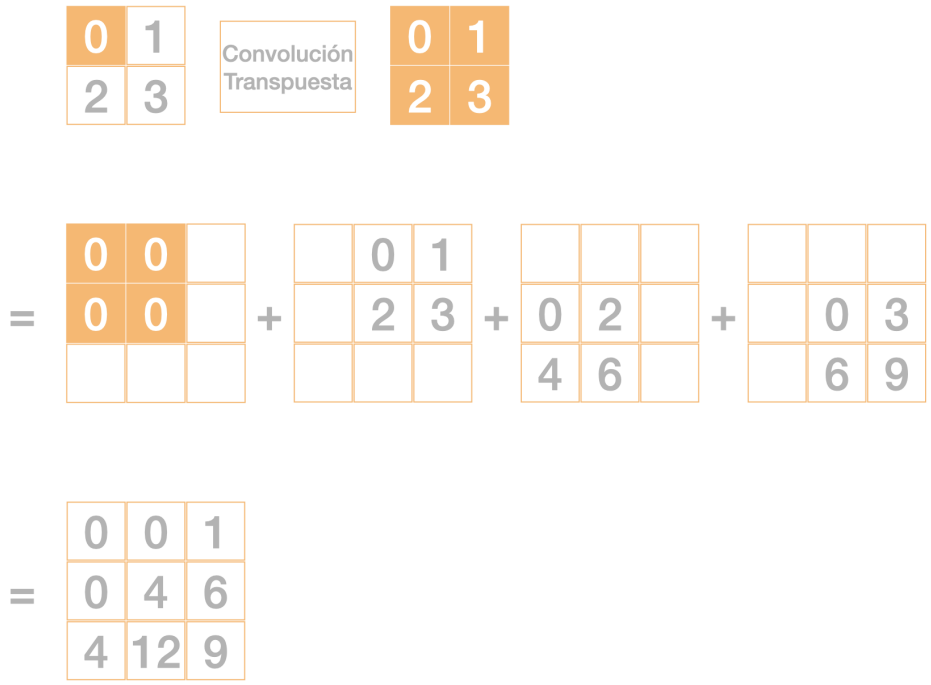


Figura 3.5: Convolución transpuesta con un kernel de 2×2

3.3. Redes neuronales generativas antagónicas

En esta sección se profundiza sobre los aspectos teóricos de las GANs, propuesta de entrenamiento de modelos generativos presentada por Goodfellow

y col. [28] que se utiliza en el desarrollo de este proyecto de grado. Las GANs forman parte de una familia de redes neuronales llamadas redes generativas profundas [63] las cuáles a su vez son parte de la familia de modelos generativos. Los modelos generativos se definen en términos de modelos probabilísticos como un descriptor de cómo un conjunto de datos es generado. Al tomar muestras de un modelo generativo, es posible generar nuevos datos [24].

3.3.1. Arquitectura y entrenamiento de las GAN

Las GANs fueron descriptas por primera vez en 2014 por Goodfellow y alcanzaron notoriedad gracias a su habilidad de generar datos sintéticos realistas aprendiendo a partir de distribuciones complejas. A continuación se presentan la arquitectura de las GAN y su proceso de entrenamiento.

Las GANs están compuestas por dos redes distintas, el generador y el discriminador, y su entrenamiento se basa en alternar la actualización del generador y del discriminador usando back-propagation siguiendo un paradigma de juego minimax. La alternancia en la actualización del generador y del discriminador es para realizar un entrenamiento antagónico, donde el generador intenta generar un dato sintético tan realista que al ser evaluado por el discriminador este no sea capaz de diferenciar si es un dato real o un dato sintético. Tomando una muestra de ruido $z \sim p(z)$ como entrada, el generador G devuelve un nuevo dato sintético $G(z)$ que pertenece a una distribución p_g, al principio del entrenamiento p_g no es similar a la distribución p_{data} de los datos originales. Por otra parte, el discriminador D se encarga de distinguir los datos reales $x \sim p_{data}(x)$ de los datos generados $G(z) \sim p_g(G(z))$. Siguiendo la Ecuación 3.10, durante el proceso de entrenamiento el generador actualiza sus pesos utilizando como función objetivo la clasificación del discriminador intentando (buscando aproximar p_g a p_{data}) mientras el discriminador es entrenado de forma supervisada para clasificar entre datos reales y datos sintéticos.

$$\min_G \max_D E_{x\sim p_{\text{data}}} [\log D(x)] + E_{z\sim p_z}[\log 1 - (D(G(z)))] \tag{3.10}$$

En teoría, el juego de minimax termina cuando $p_g = p_{data}$, por lo que el discriminador responde aleatoriamente si la imagen es real o falsa. De todas formas, como se presenta en la sección 3.3.2, la convergencia de las GANs es un área de investigación aún activa ya que no siempre se consigue que p_g sea similar a p_{data}.

Las redes DCGAN son una extensión de las GAN que utilizan capas convolucionales y convolucionales-transpuestas en el discriminador y el generador respectivamente. El generador recibe un vector de ruido (comúnmente muestreado de una distribución uniforme), el cual expande mediante la utilización de convoluciones transpuestas, produciendo como salida una imagen. La Figura 3.6 muestra la arquitectura del generador. DCGAN demostró generar resultados más realistas que la arquitectura de GAN original, especialmente en el caso de imágenes [61].

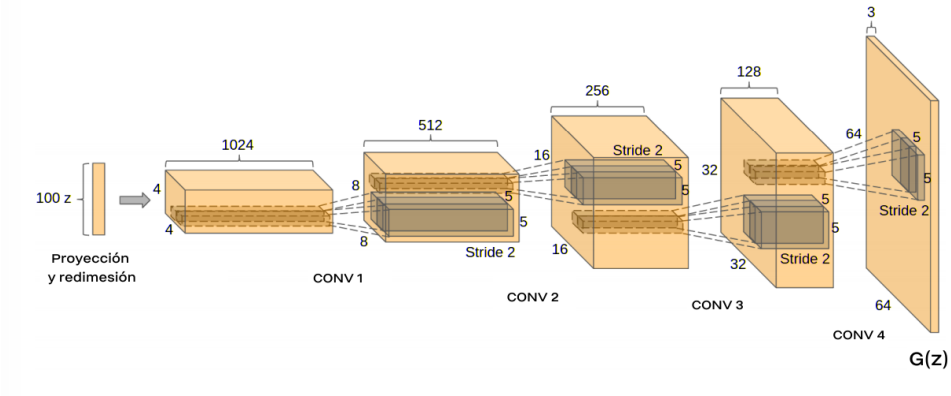


Figura 3.6: Arquitectura del generador de la red DCGAN. Diagrama basado en el presentado por Radford, Metz y Chintala [61].

3.3.2. Patologías de entrenamiento de las GAN

En la práctica, el entrenamiento de GANs sufre de patologías que causan problemas de convergencia debido a que la función de costo no es convexa para redes neuronales. Goodfellow y col. [28] detectaron la no convergencia como una causa del mal desempeño de las GAN. La no convergencia se debe a que las actualizaciones basadas en gradiente buscan un punto estacionario con gradiente cero (el discriminador no distingue verdadero de falso), causando una variedad de degeneraciones del entrenamiento, por ejemplo: el colapso modal, el colapso del discriminador, el estancamiento en oscilación de las funciones de costo y el desvanecimiento de gradientes.

Estudios recientes han demostrado que la función objetivo del entrenamiento puede llegar a un valor óptimo aún cuando el generador está lejos de haber aprendido la distribución [5]. Un generador siempre está buscando producir un dato sintético que confunda al discriminador, pero si este encuentra un dato

sintético que sea especialmente bueno en confundir al discriminador, el generador podría producir únicamente ese dato y así conseguir siempre el valor mínimo en la Ecuación 3.10. Si el generador genera siempre el mismo dato, el discriminador puede aprender a siempre predecir falso para ese dato, pero si el generador cae en un mínimo local, el resultado será un generador capaz de reproducir una variedad muy baja de datos sintéticos. Esta patología es la que se conoce como colapso modal.

Otro patología común es cuando el discriminador se vuelve muy bueno distinguiendo entre los datos generados y los reales. Se probó que, con esta patología, a medida que el discriminador mejora los gradientes del generador se desvanecen o presentan un comportamiento masivamente inestable [4].

Goodfellow y col. [28] propusieron una reformulación de la función de costo de GAN en la cual el generador busca incrementar la probabilidad de que el discriminador cometa un error, en lugar de minimizar la probabilidad de que el discriminador no se equivoque. Esta reformulación provoca que los gradientes de la función de costo del generador se mantengan altos a lo largo del entrenamiento, contribuyendo positivamente a la convergencia de la red.

Existen otras propuestas que buscan mejorar la estabilidad del entrenamiento. Tres de estas propuestas son:

- La red generativa antagónica condicional (CGAN). En esta propuesta la red recibe también cuál clase debe generar, por lo que tiene como objetivo aprender $p(x|y)$ en vez de $p(x)$ [53].
- La red generativa antagónica laplaciana (LAPGAN). En esta propuesta en vez de entrenar con las imágenes en toda su resolución, el entrenamiento comienza con imágenes en menor resolución y aumenta la resolución a medida que avanza el entrenamiento [17]. El nombre surge de la utilización de pirámides laplacianas para generar imágenes con diferentes niveles de detalle.
- La red generativa antagónica de Wasserstein (WGAN). En esta propuesta se utiliza la función de costo de Wasserstein, que está diseñada para mejorar los problemas de desvanecimiento de gradientes y colapso modal. Esta función de costo busca la minimización entre la distancia de dos distribuciones de probabilidad, la de los datos reales y los generados. En esta propuesta el discriminador pasa a llamarse critico, ya que este no aprende a diferenciar si un dato es real o no, en cambio aprende a

asignar valores más grandes a los datos reales que a los sintéticos (sin restringirse al intervalo [0, 1]). El colapso modal aumenta la distancia entre las distibuciones de datos reales y sintéticos. Esta función de costo penaliza la distancia entre ambas distribuciones, evitando así el colapso modal.

3.4. Lipizzaner

En esta sección se presenta el framework Lipizzaner, un sistema escalable y distribuido para el entrenamiento co-evolutivo de GANs. Lipizzaner aplica algoritmos evolutivos para intentar resolver las patologías de entrenamiento de colapso modal y desvanecimiento de gradientes. Las siguientes secciones presentan el framework completo de Lipizzaner, especialmente las etapas del entrenamiento y la implementación de distribución de redes para su entrenamiento distribuido.

3.4.1. Algoritmo de entrenamiento

El algoritmo de entrenamiento de Lipizzaner es relevante para lograr comprender tanto los requerimientos de comunicación entre nodos como los cambios presentados en este proyecto de grado. A continuación se detalla el funcionamiento de los algoritmos 1 y 2. En la Figura 3.7 se presenta un diagrama del algoritmo de entrenamiento.

La topología utilizada en el algoritmo de Lipizzaner es la de una grilla circular. Por lo que cada celda tiene cuatro vecinos (norte, sur, este y oeste). Cada celda comenzará con un discriminador y un generador locales (inicializado aleatoriamente) y además recibirá los generadores y discriminadores de sus vecinos.

Una vez las cinco GANs están en la celda, se realizará el proceso de selección, este busca elegir el mejor par de individuos (generador y discriminador). Para hacerlo se puede usar cualquier operador de selección. Un ejemplo de operador de selección consiste en evaluar todos los discriminadores contra todos los generadores y realizar una selección aleatoria ponderada por el desempeño obtenido en la evaluación.

Algoritmo 1 `BasicCoevGANs`$(P_u, P_v, \mathcal{L}, \{\alpha_i\}, \{\beta_i\}, I)$ [68, Algoritmo 1].
Entrada:
P_u : población de Generadores
P_v : población de Discriminadores
$\{\alpha_i\}$: probabilidad de selección
$\{\beta_i\}$: probabilidad de mutación
I : número de iteraciones por paso de entrenamiento por población
$\mathcal{L}$: función objetivo de GAN
Salida:
P_u : Población de Generadores evolucionados
P_v : Población de Discriminadores evolucionados

---

for i **in** $range(I)$ **do**
    // **Evaluate** P_u **and** P_v
    $f_{u_1 \ldots u_T} \leftarrow 0$
    $f_{v_1 \ldots v_T} \leftarrow 0$
    for each u_i **in** P_u**, each** v_j **in** P_v **do**
        f_{u_i} $-= \mathcal{L}(u_i, v_j)$
        f_{v_j} $+= \mathcal{L}(u_i, v_j)$
    end for
    // **Ordenar** P_u **y** P_v
    $u_{1\ldots T} \leftarrow u_{s(1)\ldots s(T)}$ `with` $s(i) = \texttt{argsort}(f_{u_1 \ldots u_T}, i)$
    $v_{1\ldots T} \leftarrow v_{s(1)\ldots s(T)}$ `with` $s(j) = \texttt{argsort}(f_{v_1 \ldots v_T}, j)$
    // **Selección**
    $u_{1\ldots T} \leftarrow u_{s(1)\ldots s(T)}$ `with` $s(i) = \texttt{argselect}(u_{1\ldots T}, i, \{\alpha_i\})$
    $v_{1\ldots T} \leftarrow v_{s(1)\ldots s(T)}$ `with` $s(j) = \texttt{argselect}(v_{1\ldots T}, j, \{\alpha_j\})$
    // **Mutación & Reemplazo**
    $u_{1\ldots T} \leftarrow \texttt{replace}(\{u_i\}, \{u'_i\})$ `with` $u'_i = \texttt{mutate}(u_i, \beta_i)$
    $v_{1\ldots T} \leftarrow \texttt{replace}(\{v_j\}, \{v'_j\})$ `with` $v'_j = \texttt{mutate}(v_j, \beta_j)$
end for
return P_u, P_v

Algoritmo 2 `CoevGANs`$(P_u, P_v, \mathcal{L}, \{\alpha_i\}, \{\beta_i\})$ [68, Algoritmo 2].
Entrada:
P_u : población de Generadores
P_v : población de Discriminadores
$\{\alpha_i\}$: probabilidad de selección
$\{\beta_i\}$: probabilidad de mutación
I : número de iteraciones por paso de entrenamiento por población
m : largo del lado de grilla cuadrada
$\mathcal{L}$: función objetivo de GAN
Salida:
P_u^* : ensamble evolucionado de generadores
w^* : vector de pesos de ensamble evolucionado

---

repeat
    // Coevolución Espacial de poblaciones de Generadores y Discriminadores
    parfor k in $range(m^2)$ **do**
        $\hat{P}_u^k, \hat{P}_v^k \leftarrow$ `BasicCoevGANs`$(P_u^k, P_v^k, \mathcal{L}, \{\alpha_i\}, \{\beta_i\}, I)$
        $P_u^{k,1} \leftarrow$ `TopN`$(\hat{P}_u^k, n = |P_u^{k,1}|)$
        $P_v^{k,1} \leftarrow$ `TopN`$(\hat{P}_v^k, n = |P_v^{k,1}|)$
    end parfor
    // Evolución de pesos de ensamble de generadores
    $w^1, \ldots, w^{m^2} \leftarrow$`(1+1)-ES`$(w^1, \ldots, w^{m^2}, g, \{P_u^k\})$
until training converged
$P_u^*, w^* \leftarrow \arg\max_{P_u^k, w^k : 1 \leq k \leq m^2} g\Big(\sum_{\substack{u_i \in P_u^k \\ w_i \in w^k}} w_i G_{u_i}\Big)$
return P_u^*, w^*

Luego se aplica el operador de mutación elegido en Lipizzaner que consiste en entrenar un *epoch* utilizando descenso por gradiente al discriminador y generador seleccionados.

Finalmente se reemplaza el generador y discriminador locales por los nuevos individuos y se comienza una nueva iteración.

Una vez cumplida la cantidad de iteraciones deseada se genera un ensamble con los modelos producidos por las celdas. Primero, todas las celdas envían su generador a un proceso de control, que se encarga de generar el ensamble. Luego, se le asigna un peso w_i a cada generador que determina la probabilidad de ser utilizado en la predicción. Finalmente, para conseguir el vector de pesos se utiliza nuevamente un algoritmo evolutivo que optimiza este vector basado en una métrica de desempeño.

A diferencia de los esquemas evolutivos clásicos, el resultado de la optimización es una solución única formulada a partir de todo el conjunto de soluciones. El ensamble da una solución que aproxima al conjunto de datos de entrenamiento mejor que cada modelo de manera independiente.

3.4.2. Sistema de distribución

El sistema de intercambio de mensajes en Lipizzaner esta implementado sobre le protocolo HTTP. La implementación utiliza una API escrita con la biblioteca Flask de Python. A continuación se detallan los mensajes y módulos que están involucrados en el proceso de comunicación entre procesos.

El algoritmo comienza con la creación de los procesos clientes y maestro. Se debe elegir en que nodo y puerto iniciar la ejecución de cada proceso. Los procesos abren una API en el puerto asignado y se quedan esperando el mensaje de inicio de ejecución.

El maestro reconoce los puertos de red donde se han expuesto a los clientes, los puertos pueden ser provistos en la configuración o pueden descubrirse automáticamente haciendo una búsqueda en toda la red. Una vez identificados todos los clientes, se crea un hilo para cada uno que se encarga de chequear que se mantienen funcionando, lo que se conoce como *heartbeat*. Los hilos envían un mensaje a cada cliente en un intervalo fijo de tiempo y espera una respuesta, de no recibirla, se reinicia el cliente o se finaliza todo el algoritmo.

Luego, el maestro envía a cada cliente el mensaje de inicio de ejecución y finalmente se quedará bloqueado hasta que los clientes terminen de ejecutar el algoritmo.

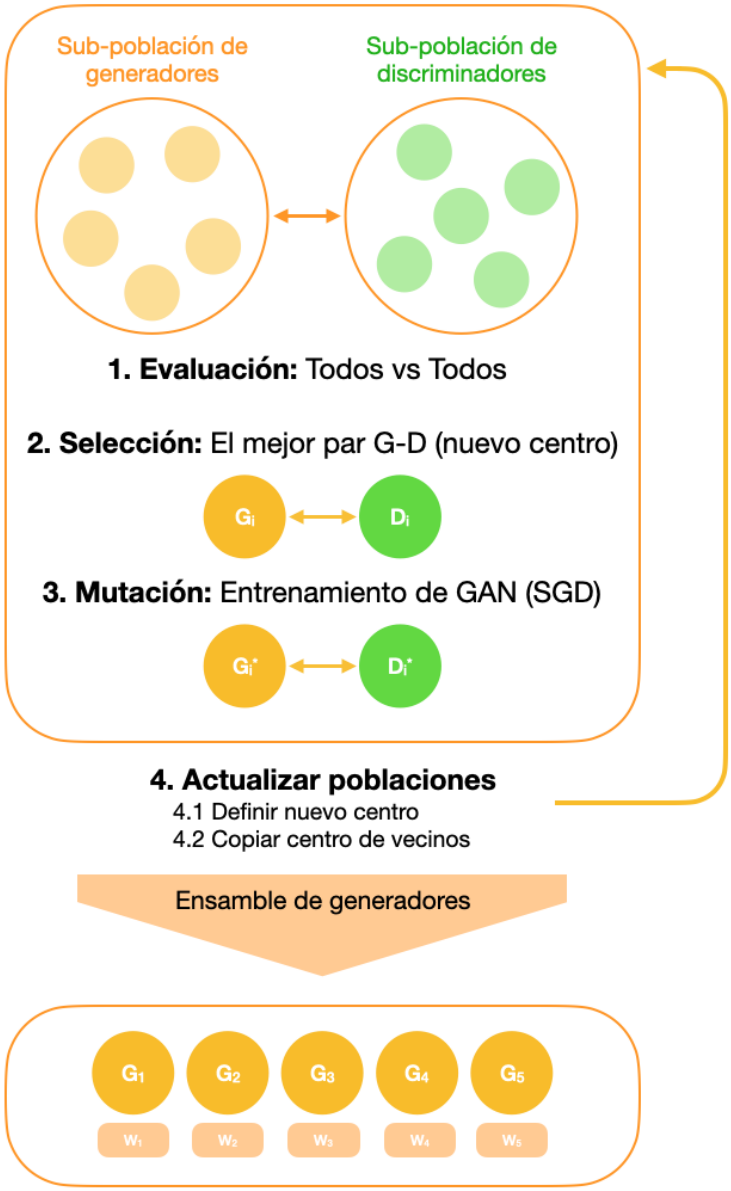


Figura 3.7: Algoritmo de entrenamiento de una celda y optimización de ensamble.

Una vez los clientes reciben el mensaje para comenzar a ejecutar crean un nuevo hilo en el proceso que se encarga de llevar la ejecución del algoritmo mientras que el proceso principal se encarga de atender los pedidos que lleguen al cliente.

Al inicio de cada iteración (nueva generación) cada cliente envía un pedido a cada uno de sus vecinos para que respondan con la red que fue seleccionada más recientemente en sus celdas. De esta forma se evita la sincronización de las iteraciones, ya que siempre se responde con la última selección, sin importar si los nodos se encuentran en diferentes iteraciones.

Se utiliza una clase que se encarga de controlar el acceso de lectura/escritura a las redes locales, evitando condiciones de carrera entre el thread que corre el algoritmo y el de control.

Cuando todos los clientes finalizaron sus iteraciones envían un mensaje de finalizado al maestro como respuesta al *heartbeat*. Una vez todos notificaron su finalización, el maestro realiza un pedido HTTP a todos los clientes para que envíen los generadores y discriminadores. Con las redes obtenidas finalmente se genera el ensamble mencionado en la sección anterior.

Otro aspecto interesante, es la forma en que cada nodo determina sus vecinos. Todo el vecindario es controlado por una clase que implementa las operaciones de comunicación con otros vecinos y almacena la información de la topología. Las direcciones IP de los nodos son compartidas por el maestro al inicio de la ejecución. Se encuentran ordenadas en una lista y se asume que cada índice corresponde a una celda comenzando de la esquina superior izquierda, continuando hacia la derecha y luego hacia abajo. Ver Tabla 3.1 para un ejemplo del mapeo de índices a celdas. Luego para la celda (i,j) su vecino derecho será el que se encuentra en el índice $(i, j+1) = (i * \sqrt{\#nodos} + j + 1)\ mod\ \#nodos$ siendo $\#nodos$ la cantidad de clientes y asumiendo que $\sqrt{\#nodos}$ es un numero natural (es necesario para que la grilla sea cuadrada).

(i,j)	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8

Tabla 3.1: Mapeo de índices de la lista a una grilla de 3×3

Algunos de los problemas con esta forma de manejar la ejecución distribuida son:

- Requiere que los nodos tengan visibilidad de las IP y puerto de los otros nodos. Esta funcionalidad no esta permitida en muchos clusters ya que podría ser una vulnerabilidad de seguridad. Esto restringe los entornos en los que se puede ejecutar.
- Es necesario el balance de carga manual de los procesos. No se cuenta con ningún sistema para determinar una distribución óptima de los procesos en los nodos y debe ser realizado manualmente, lo que agrega trabajo a la hora de ejecutar el algoritmo.
- Introduce un alto nivel de complejidad ya que requiere la implementación de hilos de control para poder atender la API mientras se ejecuta el experimento. Debido al uso de hilos surge la necesidad de control de acceso a los recursos para evitar condiciones de carrera.
- Para conseguir las redes de los vecinos es necesario hacer un pedido de envío, incrementando la cantidad de mensajes necesarios y la demora en poder continuar el entrenamiento.

3.5. Aumentación de datos

La aumentación de datos encompasa un conjunto de técnicas diseñadas para incrementar el tamaño de conjuntos de datos a partir de la transformación o sobre-muestreo de sus ejemplares [74]. Las técnicas de aumentación de datos son aplicadas bajo la suposición de que más información puede ser extraída de un conjunto de datos para mejorar el desempeño de modelos de aprendizaje automático. Esta sección introduce los conceptos generales de la aumentación de datos y sus utilidades. Luego, se presentan técnicas clásicas de aumentación de datos. Por último, se presentan técnicas de aumentación de datos utilizando redes neuronales artificiales.

3.5.1. Aumentación de datos para resolver el desbalance de clases

Las técnicas de aumentación de datos son aplicadas así con el fin de contar con un conjunto de datos extendido y así mejorar el entrenamiento de modelos

de aprendizaje automático. Uno de los problemas que resuelven las técnicas de aumentación de datos es el problema de desbalance de clases en tareas de clasificación. A continuación se presenta el problema del desbalance de clases, los tipos de desbalance y las métricas utilizadas para evaluar a los modelos de aprendizaje automático entrenados en conjuntos desbalanceados.

Los conjuntos de datos utilizados en problemas de aprendizaje automático cuentan con una o más etiquetas para cada uno de los ejemplares del conjunto. Las etiquetas de un conjunto de datos se repiten dentro del conjunto y a los grupos de etiquetas repetidas se les llama clases. Un conjunto de datos es desbalanceado si sus clases no están representadas de manera equitativa [9]. Los conjuntos de datos desbalanceados son muy comunes hoy en día, por ejemplo en aplicaciones médicas, transacciones fraudulentas o detección de artículos falsos. En el entrenamiento de modelos de clasificación se busca entrenar un modelo de aprendizaje automático capaz de clasificar las clases de un conjunto de datos y además, la clase con menos ejemplares (la clase minoritaria) suele ser la clase de mayor interés y la clase para la cual es necesario tener un buen desempeño en la predicción. Sin embargo, el desempeño de los modelos de aprendizaje automático se ve afectado si el conjunto de entrenamiento es reducido alcanzando un bajo desempeño especialmente en las clases poco representadas. Las técnicas de aumentación de datos surgen de la necesidad de balancear la representación de las clases para alcanzar un mejor rendimiento en los modelos de aprendizaje automático.

El desbalance de clases ocurre tanto entre las clases como intra-clase Sampath y col. El desbalance intra-clase ocurre cuando existe algún sesgo indeseado dentro de la clase. Por ejemplo, en un conjunto de datos con una clase de perros, se puede separar esta clase por el color, variación en la pose o raza de los animales. Cualquier tipo de sesgo en estas subcategorías podría causar que el modelo aprenda correlaciones no deseadas, por ejemplo si se cuenta con un conjunto de perros etiquetados en base a su color y todos los perros etiquetados marrones son de la misma raza el modelo puede aprender a fijarse la raza del perro en vez de su color para clasificarlo marrón.

La evaluación de modelos predictivos en conjuntos desbalanceados debe hacerse cuidadosamente. En un conjunto balanceado, es útil medir los ejemplos clasificados correctamente sobre ejemplos totales pero no es útil en un conjunto desbalanceado. Si un modelo ignora completamente la clase minoritaria obtendrá un buen resultado. Para mitigar evitar problemas de evaluación en

conjuntos desbalanceados se utilizan métricas como precisión y exhaustividad 3.11 que consideran la capacidad predicativa en cada una de las clases.

$$\begin{aligned} \text{Precisión} &= \frac{|\{ \text{ documentos relevantes } \} \cap \{ \text{ documentos recuperados } \}|}{|\{ \text{ documentos recuperados } \}|} \\ \text{Exhaustividad} &= \frac{|\{ \text{ documentos relevantes } \} \cap \{ \text{ documentos recuperados } \}|}{|\{ \text{ documentos relevantes } \}|} \end{aligned} \tag{3.11}$$

En una encuesta realizada en 2021 [65] se propuso una taxonomía que divide en tres categorías a los problemas de desbalance de conjuntos de datos: A nivel de imágenes en clasificación, a nivel de objetos en detección de objetos y a nivel de píxeles en segmentación. Lo mencionado anteriormente y en lo que se basa este proyecto de grado es en mitigar los problemas de la primera categoría mencionada aunque algunas de partes del trabajo desarrollado podrían también ser aplicadas para atacar las otras dos categorías de desbalance.

3.5.2. Métodos tradicionales de aumentación de datos

Existen tres métodos tradicionales para intentar mejorar el desempeño de los modelos de aprendizaje automático en las clases minoritarias:

- Sobre-muestreo. Es el proceso de muestrear una clase con mayor frecuencia de la que ésta representa en el conjunto de datos. Se distinguen dos grandes categorías. Muestreo con reemplazo, donde cada ejemplar puede ser muestreado repetidas veces y muestreo sintético, donde al muestrear una clase se aplica un algoritmo que permite la generación de nuevos ejemplares de dicha clase.
- Descarte de ejemplares. La técnica consiste en descartar ejemplares de las clases más respresentadas hasta que la cantidad de ejemplares sea similar a la cantidad de ejemplares de las clases minoritarias. Algunas técnicas más avanzadas buscan seleccionar de forma inteligente cuales ejemplares eliminar y de esta forma preservar la mayor diversidad intra-clase posible.
- Pesos en el objetivo. Asignar pesos en la función objetivo de un modelo de aprendizaje automático, que de mayor importancia a los errores en las clases minoritarias permite mejorar el aprendizaje sin realizar cambios en el conjunto de datos. Este método no siempre es útil ya que ante la presencia de pocos ejemplares puede resultar fácilmente en sobre-ajuste en las clases minoritarias.

Debido a que la utilización de sobre-muestreo con remplazamiento suele causar el sobre-ajuste del modelo en la clase minoritaria [9], es común utilizar técnicas que generen muestras sintéticas. Las técnicas de generación de muestras sintéticas buscan crear nuevos ejemplares de la clase minoritaria de forma que no sean exactamente iguales a los del conjunto de datos, esto permite al modelo ampliar los bordes de decisión y así conseguir un modelo más robusto. El desarrollo de técnicas para la generación de nuevos ejemplos es un área activa de investigación y es una de las temáticas que se abordarán durante esta sección. A continuación se presentan técnicas de generación sintética de datos.

3.5.2.1. SMOTE: Sobre-muestreo sintético de minorías

El sobre-muestreo sintético de minorías (SMOTE) es un método clásico para la aumentación de conjuntos de datos tabulares. Es utilizado como línea base en diversos artículos del estado del arte para comparar los resultados contra nuevas técnicas debido a su fácil aplicación y buenos resultados.

La forma en que SMOTE genera nuevos ejemplos es mediante la interpolación de muestras de la clase minoritaria. Se seleccionan dos ejemplares cercanos de la misma clase, se haya su diferencia y se la multiplica por un número aleatorio para luego sumar la diferencia multiplicada a uno de los ejemplares. Para elegir los pares de ejemplares itera sobre los ejemplares de la clase a sobre-muestrear y se utilizan los k vecinos más cercanos, siendo k la cantidad de veces que se desea multiplicar la clase (por ejemplo para duplicar la clase usar $k = 2$). SMOTE funciona bajo la suposición de que la interpolación entre ejemplares generará un nuevo ejemplar válido, forzando una ampliación en la zona de decisión de la clase minoritaria [9].

SMOTE ha demostrado que los árboles de decisión entrenados con SMOTE son de mucho menor profundidad que los árboles obtenidos mediante sobre-muestreo con remplazamiento. Los árboles de decisión de baja profundidad son usualmente modelos que presentan menor sobre-ajuste a los datos de entrenamiento. También se probo que si además de utilizar sobre-muestreo con SMOTE se agrega descarte de ejemplares en las clases con mayor cantidad de ejemplos, los resultados son mejores que los obtenidos realizando solo el descarte de ejemplares [9].

SMOTE posee muchas variantes que fueron desarrolladas en los últimos años, como SMOTE-NC que trata con el problema de variables no continuas

[9] o ADASYN el cual propone agregar ruido a la interpolación y adaptar la cantidad de ejemplos generados a la dificultad que tenga el modelo para aprender a distinguir a cada clase [31].

3.5.2.2. Manipulación básica de imágenes

Para el problema de clasificación de imágenes, las primeras técnicas efectivas de aumentación de datos fueron las transformaciones simples sobre los conjuntos de datos. El volteo vertical de la imagen, cambios en el espacio de colores y recortes aleatorios son ejemplos de transformaciones simples de imágenes [74]. Los modelos de clasificación de imágenes deben superar problemas de puntos de vista, oclusión, iluminación, escala y fondos adversos, entre otros. El objetivo de la aumentación de datos es introducir invarianzas (imágenes similar con la misma etiqueta) dentro de los mismos conjuntos de datos para que los modelos logren generalizar mejor el conjunto de datos con el que fueron entrenados.

A continuación se describen algunas de las técnicas de manipulación (o transformación) de imágenes y las consideraciones necesarias para la apropiada aplicación de cada una de las técnicas. Las técnicas que se presentan a continuación tienen diferentes niveles de preservación de la clase post-transformación, es decir que dependiendo del dominio donde se aplique la transformación algunos datos pueden perder la información válida para su clase. En el dominio de conjuntos de imágenes de objetos por ejemplo, un recorte aleatorio sobre una imagen puede remover por completo al objeto descripto en su etiqueta.

- Volteo. El volteo de imágenes es una de las transformaciones más simples y su variante de volteo en el eje horizontal resulta más común que el volteo vertical. La transformación ha probado ser útil en conjuntos de datos como CIFAR-10 e ImageNet pero la transformación no es segura en conjuntos de datos de dígitos, como MNIST.
- Espacio de colores. Comúnmente las imágenes digitales se codifican utilizando tres canales de color R, G y B para codificar la cantidad de color rojo, verde y azul de cada píxel, respectivamente. Manipulaciones sobre los canales de color resultan útiles y son simples en su implementación. Las opciones más simples van desde utilizar solo uno de los canales y suben en complejidad hasta manipular los canales para cambiar su brillo, saturación, etc.

- Recortes aleatorios. Los recortes de imágenes se utilizan para cambiar las dimensiones esperadas por el modelo de altura y ancho. La manipulación también permite presentar al modelo recortes parciales de las clases a generalizar, por ejemplo una parte del cuerpo de un animal y no al animal entero.
- Rotaciones. Las redes neuronales convolucionales no pueden procesar la rotación de una imagen, es decir que no son capaces de diferenciar que un gato rotado 90 grados sigue siendo un gato, por lo que se ven beneficiadas por la inclusión de rotaciones al conjunto de datos para generar robustez en el modelo. El grado de rotación es definitorio para la seguridad de preservación de etiquetas.
- Traslación. El movimiento de una imagen en una dirección aleatoria puede ayudar a reducir el sesgo posicional del conjunto de datos. Si todas las imágenes están perfectamente centradas, como es el caso en muchos conjuntos de rostros, sería necesario evaluar al modelo únicamente en rostros perfectamente centrados resultando en un modelo poco robusto.
- Inyección de ruido. Más allá de las transformaciones geométricas, la inserción de ruido consiste en sumar una matriz de valores aleatorios a la matriz de la imagen.
- Filtros de kernel. Los filtros de kernel son populares en el área de procesamiento de imágenes para agudizar o desenfocar imágenes. Los filtros funcionan aplicando la multiplicación de una matriz sobre la imagen, dependiente de la transformación deseada (filtro Gaussiano, filtro de ejes). Una desventaja de este tipo de aumentaciones es su similaridad con la mecánica interna de las CNNs por lo que los filtros podrían ser implementados directamente como parte de la red sin necesidad de aplicarlos al conjunto de datos de antemano.

La mayoría de las transformaciones mencionadas previamente son combinables para lograr una distorsión mayor de la imagen. La combinación de transformaciones no siempre es beneficioso. En casos donde el conjunto de datos es reducido, expandir el conjunto de datos mediante transformaciones simples puede resultar contra-producente y tender al sobre-ajuste del modelo. El diseño del conjunto de transformaciones a realizar sobre las imágenes debe realizarse con cuidado.

3.5.3. Redes neuronales para aumentación de datos

Esta subsección presenta como se pueden utilizar redes neuronales para aumentar conjuntos de datos y resolver los problemas de desbalance inter- e intra- clases.

3.5.3.1. Desbalance inter-clases

Un enfoque para solucionar el problema de desbalance inter-clases utilizando redes neuronales es utilizar GANs para realizar sobremuestreo sintético. Similar a la técnica SMOTE pero utilizando una GAN para generar las muestras sintéticas como presenta la Figura 3.8.

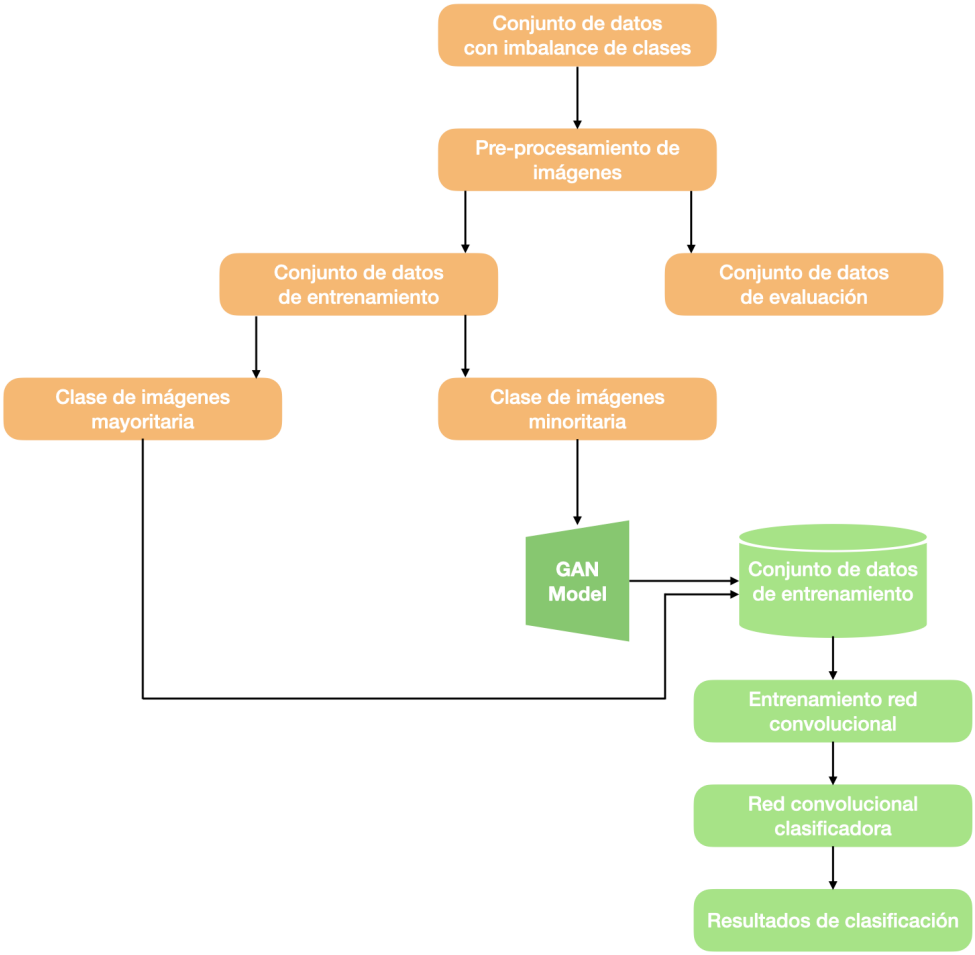


Figura 3.8: Diagrama de flujo de técnica de sobre muestreo con GANs

Liqaa y Saud [49] aplicaron existosamente la técnica de sobre-muestreo utilizando una GAN. Entrenaron una DCGAN sobre la clase minoritaria de su conjunto de datos de entrenamiento. Luego la utilizaron para generar muestras sintéticas y así igualar la cantidad de ejemplares de la clase minoritaria. El clasificador entrenado en el conjunto desbalanceado de Malaria obtuvo una precisión de 50 %, mientras que cuando se agregaron los ejemplares sintéticos la precisión subió a un 94,5 %.

El artículo de BAGAN [52] es otro caso de éxito. El artículo de la BAGAN propuso que la clase minoritaria de un conjunto de datos puede no tener suficientes datos para entrenar una GAN, por lo que propusieron incluir todas las clases durante el entrenamiento adversarial de la GAN. Utilizar todo el conjunto de datos en el entrenamiento permitió al generador aprender características útiles de la clase mayoritaria y usarlas en la clase minoritaria. Luego, se utilizó un codificador pre-inicializado como generador que permite condicionar la generación a una clase objetivo. Esta técnica fue evaluada en cuatro conjuntos (MNIST, CIFAR-10, Flowers y GTSRB) y comparada contra otras técnicas de aumentación de datos con GANs y contra técnicas tradicionales. Se demostró que BAGAN era mejor que las alternativas que utilizaban GANs en todos los casos pero solo alcanzaba mejores resultados que las técnicas de aumentación tradicionales cuando las clases tenían características sensibles a la orientación (por ejemplo los números en MNIST).

Los conjuntos de datos sensibles a la rotación son los conjuntos donde más se aplica aumentación de datos ya que la mayor parte de los estudios están basados en conjuntos de datos médicos. Muramatsu y col. [54] investigaron utilizar una Cycle-GAN [92] entrenada en imágenes de lesiones de diversos órganos para generar nuevos ejemplos de mamografías benignas y malignas. El artículo resultó en pequeñas mejoras en la precisión del modelo de clasificación. Un conjunto de datos reciente es el COVID-19 Chest X-ray [11]. Sobre este conjunto los autores de CovidGAN [79] entrenaron una variante de AC-GAN logrando una mejora del 10 % en la precisión del modelo de clasificación. Guan and Loew [30] entrenaron una GAN utilizando recortes de imágenes de mamografías. Los recortes eran de zonas normales y zonas afectadas (cáncer o tumor). El estudio demostró que mezclar los datos reales con los generados mejoraban los resultados, sin embargo, las técnicas de aumentación de datos tradicionales obtuvieron mejores resultados.

Resumiendo, el área de aumentación de datos con GANs esta en pleno desarrollo y aún es necesaria más investigación para probar su efectividad. Bajo ciertas condiciones las GANs pueden ser útiles para el balanceo de conjuntos, resultando en mejoras significativas en la precisión de los modelos finales y reduciendo el sobreajuste. Cuando las imágenes tienen características sensibles a la orientación las GANs demostraron ser mejores que los métodos tradicionales, ya que los métodos tradicionales habitualmente realizan trasformaciones afines sobre las imágenes provocando alteraciones no deseadas [52]. Además, se demostró buena efectividad cuando las imágenes son de baja resolución, ya que la GAN es capaz de generar imágenes indistinguibles con las reales, en cambio si la resolución es alta las texturas y colores presentan un mayor desafío para la generación de datos sintéticos. Finalmente, se vio que utilizando GANs y aumentaciones tradicionales para la aumentación de conjuntos de entrenamiento se logra consistentemente una reducción del sobreajuste en el entrenamiento de modelos de clasificación [30, 65].

3.5.3.2. Desbalance intra-clase

Para atacar el problema de desbalance intra-clase se necesitan un conjunto de técnicas que sean independientes de las etiquetas, ya que en la mayoría de conjuntos de datos las características que diferencian a ejemplares dentro de una misma clase no están etiquetadas. En conjuntos de datos de personas, se pueden tener imágenes de una misma persona con poca variedad entre imágenes (imágenes tomadas en secuencia por ejemplo) mientras que otras personas pueden contar con imágenes tomadas en diferentes días, horarios o ángulos. La falta de etiquetas genera la necesidad de introducir técnicas que permitan detectar características no etiquetadas y así lograr generar ejemplos variados para cada clase.

Hase y col. [34] propusieron una técnica para detectar las diferencias intra-clases utilizando un codificador que permite la transformación de las imágenes en vectores para luego agruparlos y detectar distintos subconjuntos de imágenes con características comunes dentro de las clases. Una vez detectadas las diferentes sub-clases el generador se entrenó para generar imágenes que logren balancear el conjunto de datos a nivel inter- e intra-clase. Se demostró que el método propuesto genera imágenes diversas y mejoró la precisión del modelo de clasificación entrenado en CIFAR-10.

Donahue y col. [20] utilizaron redes siamesas en combinación con GANs para generar imágenes de una misma persona bajo distintas circunstancias (puntos de vista, luz y escala son algunos de ellos). Buscaron separar los espacios latentes de cada identidad y de esta forma lograron, dentro de una misma identidad (clase) generar ejemplares de diferentes partes de su distribución. Para lograr la separación utilizaron dos generadores y a la red siamesa como discriminador. Partieron el espacio de entrada en identidad y observación (las observaciones de una identidad son distintas imágenes de una misma persona). Luego, pasaron por cada generador un vector de la misma identidad pero distintas observaciones y la red siamesa se encargaba de decidir si las dos imágenes representaban a la misma persona o no. Los experimentos con conjuntos de datos de humanos y sistemas de verificación de identidad demostraron que el algoritmo era capaz de generar imágenes bajo diferentes circunstancias sin alterar la identidad.

Todavía se requiere más investigación para lograr entrenar modelos de generación sintética que mitiguen el problema de desbalance intra-clase. La generación de atributos faciales, la reidentificación de personas y reidentificación de vehículos son algunos ejemplos de áreas donde son escazos los artículos intentando resolver el problema de desbalance intra-clase.

3.6. Funciones de costo de diversidad

En esta sección se presentan funciones de costo de diversidad utilizadas en el desarrollo de GANs: E–GAN y GDPP. La investigación de estas métricas es relevante para definir funciones de aptitud multidimensionales que permitan optimizar conjuntamente a la fidelidad y a la diversidad de los modelos generados por Lipizzaner.

3.6.1. Fitness de diversidad de E-GAN

Wang y col. [80] presentaron un framework para el entrenamiento de GANs que utiliza algoritmos evolutivos. El framework de entrenamiento cuenta con una población de generadores sobre los que aplica un proceso evolutivo y un único discriminador. A continuación se presenta el proceso de entrenamiento evolutivo y en específico la función de diversidad E-GAN.

El discriminador se actualiza junto con el generador ganador de cada generación. En cada generación se réplica a un generador en varios generadores y a la nueva población se le aplican mutaciones basadas en actualizaciones mediante descenso por gradientes con diferentes funciones de costo.

Para seleccionar el mejor generador de entre los generadores mutados, se evalúa a cada uno mediante una función de fitness. La función de fitness está compuesta por la suma de dos funciones de fitness, una función para evaluar la fidelidad y otra para evaluar la diversidad de las muestras generadas, la primera corresponde a la función de costo tradicional de las GANs y la segunda se presenta a continuación [80].

La función de fitness de diversidad de Evolutionary Generative Adversarial Network (E-GAN) se basa en que si el generador colapsa a una pequeña región de la distribución de salida (únicamente genera datos sintéticos similares entre sí, poco diversos), el discriminador clasifica a los datos sintéticos colapsados como falsos con alta probabilidad. Cuando el discriminador clasifica con alta probabilidad de ser falso a un dato sintético, provoca un gradiente de gran magnitud. El artículo propone usar el logaritmo del gradiente del discriminador para evaluar si un generador está colapsado o es capaz de generar datos sintéticos diversos. La función de fitness para diversidad de E-GAN queda definida por la Ecuación 3.12.

$$- log \, \| \nabla D - E_x[log \, D(x)] + E_z[log \, 1 - (D(G(z)))] \| \tag{3.12}$$

En el artículo se utiliza la función de diversidad exclusivamente para la selección de generadores. Si el generador obtiene un puntaje de diversidad alto, quiere decir que los gradientes del discriminador fueron bajos. Indicando que los ejemplares generados no fueron detectados con alta confianza. De esta forma incentivando a que los generadores que lograron confundir al discriminador permanezcan en la población.

3.6.2. Función de costo GDPP

Elfeki y col. [22] presentaron una función de costo no supervisada inspirada en Procesos Puntuales Determinantes (DPP, por su sigla en inglés *Determinantal Point Processes*) para aliviar la patología de colapso modal en el

entrenamiento de modelos generativos como la GAN. Esta sección presenta el proceso mediante el cuál se compuso la función de costo GDPP y cómo se aplica en su artículo original.

DPP es un modelo probabilístico utilizado para resolver problemas de selección con restricciones de diversidad. Por ejemplo, la tarea de resumir documentos o videos. Resolver un problema de selección de este tipo requiere cuantificar la diversidad de 2^N subconjuntos, siendo N el tamaño del conjunto base (En el caso de un documento N puede modelarse como la cantidad de palabras del documento).

Un proceso puntual $\mathcal{P}$ sobre un conjunto $\mathcal{V}$ es una medida de probabilidad en el conjunto potencia 2^N, donde $N = |\mathcal{V}|$ es el tamaño del conjunto base. Un proceso puntual $\mathcal{P}$ es llamado *determinante (en ingles "determinantal")* o DPP si, dado un subconjunto aleatorio Y muestreado según P, se cumple la Ecuación 3.13 para algún kernel de similaridad simétrico $L \in \mathbb{R}^{NxN}$, donde L_S es el kernel de similaridad del subconjunto S. L debe ser una matriz real semi-definida positiva $L \succeq I$; ya que representa una medida probabilística.

$$\forall S \subseteq Y, \mathcal{P}(S \subseteq Y) \propto \det(L_S) \tag{3.13}$$

Un DPP es una distribución sobre los subconjuntos de un conjunto fijo. Por ejemplo, los resultados seleccionados dentro de una base de datos. Un DPP sobre un conjunto de tamaño N puede ser modelado como un vector binario de largo N donde sus variables están correlacionadas negativamente entre sí. La correlación se deriva de una matriz que define una medida global de similaridad entre pares (el kernel L), de esta forma los elementos más parecidos tienen menor probabilidad de ocurrir juntos. DPP asigna mayor probabilidad a los subconjuntos más diversos [43].

Un problema de DPP es que se vuelve ineficiente computacionalmente en el dominio de generación de datos, donde se debería cuantificar la diversidad de los 2^N subconjuntos de datos, N siendo el tamaño del conjunto de cada lote de datos generados. Otro problema de DPP para la generación de datos es que además de querer incentivar la diversidad de los datos generados, es de interés generar datos diversos que se asemejen a la distribución real. Para resolver ambos problemas Elfeki y col. optaron por modelar la diversidad de los datos reales y los generados utilizando sus kernels DPP y buscar durante el entrenamiento minimizar la distancia entre los kernel DPP, llamando GDPP

a la función que se busca minimizar. Para construir los kernels se utiliza una descomposición como matriz de Gram (Ecuación 3.14) donde $q(e_i) \geq 0$ es una medida de calidad para los elementos del conjunto, pero como los ejemplos son seleccionados aleatoriamente sin criterio de calidad se puede asumir que $q\left(e_i\right) = 1; \forall i \in 1, 2, \ldots, B$. Y $\phi_i \in \mathbb{R}^D; D \leq N$ y $\left\|\phi_i\right\|_2 = 1$ es usado como representación normalizada de un elemento.

$$\mathcal{P}(S \subseteq Y) \propto \det\left(\phi(S)^\top \phi(S)\right) \prod_{e_i \in S} q^2\left(e_i\right) \tag{3.14}$$

Sin embargo, lograr acercar dos kernels es una optimización sin restricciones por lo que, para simplificar la optimización, en el artículo se pasa a trabajar con los valores y vectores propios de ambos kernels. Trabajar con los valores y vectores propios de los kernels permite reducir el problema de optimización de acercar ambos kernels a la regresión de las magnitudes de los valores propios y la orientación de los vectores propios.

Siguiendo la Ecuación 3.14, para GDPP se elige $\phi(\cdot)$ como la activación de la ultima capa oculta del discriminador, la cual es normalizada según la norma euclidiana. Luego, la función de costo GDPP $\mathcal{L}^{DPP}$ se puede descomponer en dos factores, magnitud de diversidad $\mathcal{L}_m$ y estructura de diversidad $\mathcal{L}_s$ (Ecuación 3.15), donde λ^i_{real} y λ^i_{fake} son los i^{esimo} valores propios de los kernels y $\hat{\lambda}^i_{\text{real}}$ la versión normalizada usando min-max para escalar la similaridad coseno entre los valores propios v^i_{fake} y v^i_{real}.

$$\begin{aligned} &\mathcal{L}^{DPP} = \mathcal{L}_m + \mathcal{L}_s = \\ &\sum_i \left\|\lambda^i_{\text{real}} - \lambda^i_{\text{fake}}\right\|_2 - \sum_i \hat{\lambda}^i_{\text{real}} \cos\left(v^i_{\text{real}}, v^i_{\text{fake}}\right) \end{aligned} \tag{3.15}$$

En el artículo de GDPP se modifica únicamente la función de costo del generador como se presenta en la ecuación Ecuación 3.16.

$$\mathcal{L}_g = \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] + \mathcal{L}^{DPP} \tag{3.16}$$

3.7. Herramientas de evaluación

En esta sección se presentan las herramientas utilizadas para la evaluación de los algoritmos propuestos. Primeramente, se discute sobre métricas de evaluación de modelos generativos: la Frèchet Inception Distance (FID) [37]; la

Total Variation Distance (TVD) y las adaptaciones de las métricas clásicas precisión y exhaustividad [64] y sus versiones mejoradas, densidad y cubrimiento. Luego, se presentan los contrastes de hipótesis sobre los que se apoya el análisis estadístico de los resultados: Shapiro–Wilk [71], Anderson–Darling [2] y D'Agostino-Pearson [13] para estudiar la normalidad de las muestras y Kruskall-Wallis [42] y Wilcoxon [82] para comparar los resultados de los algoritmos.

3.7.1. Métricas para evaluación de generadores

La evaluación de modelos generativos es un problema difícil de definir y resolver, especialmente cuando se trabaja con datos de alta dimensión como las imágenes [37]. Frecuentemente, se debe recurrir al juicio humano para determinar la calidad de los ejemplares generados [8, 64]. En este proyecto de grado se introducen en Lipizzaner cuatro nuevas métricas (precisión, exhaustividad, densidad y cubrimiento). Las nuevas métricas se agregan al reporte de los resultados obtenidos junto a las métricas de evaluación FID y TVD extendiendo la metodología de Schmiedlechner y col. A Continuación se presentan la 6 métricas mencionadas.

3.7.1.1. Fréchet Inception Distance

En los últimos años se ha optado por medir la calidad de los modelos generativos para generación de imágenes mediante la métrica FID. La métrica FID utiliza los valores de activación del modelo Inception v3 cómo representación de menor dimensionalidad de las imágenes. Luego utiliza las representaciones para obtener la media y la co-varianza de los conjuntos de datos reales y generados. Finalmente, la métrica es calculada como la distancia entre ambas distribuciones medida según la distancia de Fréchet [37].

FID es utilizada por su capacidad de diferenciar entre resultados con diferentes niveles de realismo y también por su capacidad de diferenciar entre resultados con diferentes niveles de diversidad. La fidelidad y la diversidad de las imágenes son dos dimensiones distintas de un conjunto de imágenes por lo que se vuelve difícil diferenciar entre un conjunto muy fiel a la distribución

original y uno que comparte su diversidad. Lipizzaner cuenta con una implementación de FID para evaluar los resultados del entrenamiento. Para paliar el problema de contar con una única métrica evaluando dos dimensiones se introducen las métricas de evaluación precisión y exhaustividad para resultados de redes generativas.

3.7.1.2. Total Variation Distance

TVD es una métrica diseñada para medir la diferencia entre dos conjuntos de probabilidad, siendo 3.17 la ecuación que la expresa en dominios finitos y discretos, siendo μ y ν conjuntos en E. Esta métrica está implementada en Lipizzaner y sigue la formulación para espacios finitos y discretos, por lo que solo es posible calcularla para conjuntos de datos etiquetados y con un clasificador pre-entrenado. Si se cuenta con etiquetas y con un clasificador, como es el caso del conjunto de datos MNIST por ejemplo, primero se obtiene la distribución de clases para el conjunto de entrenamiento, luego se clasifican las imágenes generadas y se mide la diferencia entre las clases clasificadas para el conjunto general y el generado.

$$\|\mu - \nu\|_{TVD} = \frac{1}{2} \sum_{x \in E} |\mu(x) - \nu(x)| \tag{3.17}$$

3.7.1.3. Precisión y densidad

La precisión [64] es una métrica de fidelidad de los datos sintéticos en relación a los datos reales. Se analizan los datos del conjunto conformado por la unión de los radios-k de los datos reales, al cual se denomina variedad real. Se define el radio-k de un punto como el espacio de los puntos que se encuentran a una distancia menor que el punto en cuestión y el más lejano de sus k vecinos más cercanos. Los radios-k se calculan considerando únicamente los datos reales, y la precisión se calcula como la como la proporción de datos sintéticos pertenecientes a la variedad real. La Figura 3.9 ilustra el cálculo de la precisión sobre una muestra de ejemplo.

La densidad [55] es una métrica de fidelidad que se plantea para mejorar a la precisión. Se calcula la variedad real de la misma forma que para la precisión, uniendo los radios-k de los datos reales considerando únicamente el conjunto de datos reales. Sin embargo, a cada dato falso se le asigna un punto por cada radio-k al que pertenezca dentro del colector de datos reales. Luego se suman

estos puntos y se dividen por el producto de la cantidad de datos falsos y el k elegido. La Figura 3.9 ilustra el cálculo de la precisión sobre una muestra de ejemplo.

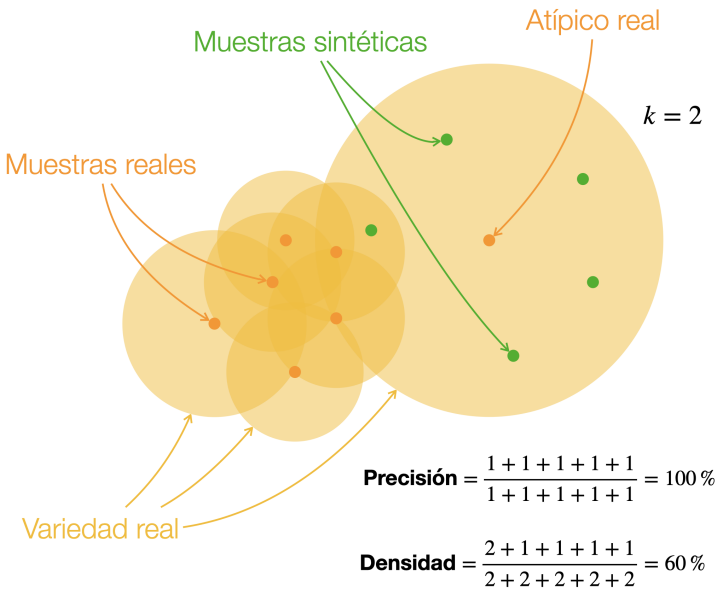


Figura 3.9: Ilustración del cálculo de precisión y densidad. Se identifica en naranja la muestras reales y en verde las muestras sintéticas. El radio-k utilizado para el cálculo es igual a 2.

3.7.1.4. Exhaustividad y cubrimiento

La exhaustividad [64], es una métrica de diversidad de los datos sintéticos en relación a los reales. Se analiza la variedad sintética, conformada por la unión de los radios-k de los datos sintéticos considerando únicamente el conjunto de datos sintéticos. Entonces, la exhaustividad se calcula como la proporción de datos reales en la variedad sintética. La Figura 3.10 ilustra el cálculo de la precisión sobre una muestra de ejemplo.

El cubrimiento es una métrica de fidelidad que se plantea para mejorar a la exhaustividad. El cubrimiento utiliza los radios-k de los datos reales considerando únicamente el conjunto de datos reales. Se calcula el cubrimiento como el cociente entre la cantidad de radios-k que contienen alguna muestra sintética y la cantidad de datos reales.

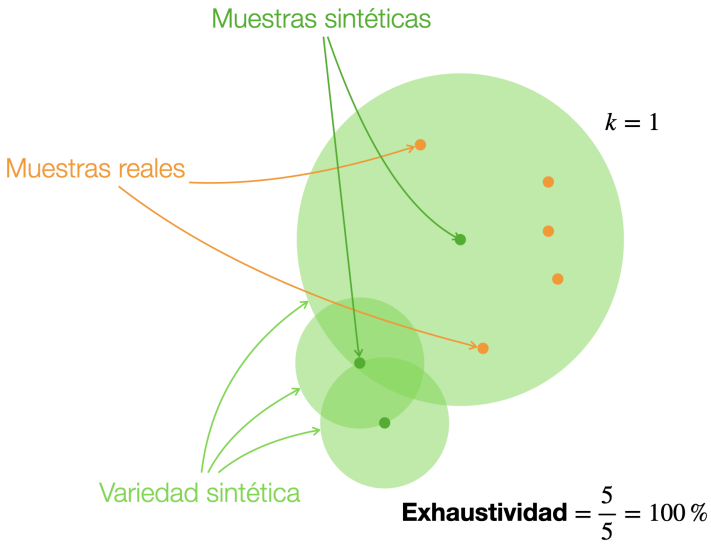


Figura 3.10: Ilustración del cálculo de exhaustividad. Se identifica en naranja la muestras reales y en verde las muestras sintéticas. El radio-k utilizado para el cálculo es igual a 1.

3.7.1.5. Comentarios generales

La precisión representa la cercanía de los resultados falsos respecto a los reales mientras que la exhaustividad muestra si los resultados falsos cubren la distribución de los datos reales. Estas métricas son una forma de diferenciar las dimensiones de fidelidad y diversidad y lograr medirlas independientemente, aunque presentan diversos problemas relacionados a la elección de hiperparámetros.

La sensibilidad a puntos aislados es uno de los problemas al elegir hiperpárametros. Con las definiciones de precisión y exhaustividad, en el caso de contener puntos aislados en cualquiera de los conjuntos - reales o falsos - rápidamente ambas métricas se disparan a 100 %. Otros problemas de precisión y exhaustividad incluyen la sensibilidad a la forma de las distribuciones, a la cantidad de vecinos elegidos para los radio-k y la poca robustez a distintas representaciones de espacio latente para un mismo conjunto de imágenes. Se entiende por sensibilidad a cuando se observa una gran desviación en el valor de la métrica al utilizar distintos hiperparámetros. En busca de reducir los problemas mencionados de precisión y exhaustividad se introducen dos nuevas métricas llamadas densidad y cubrimiento.

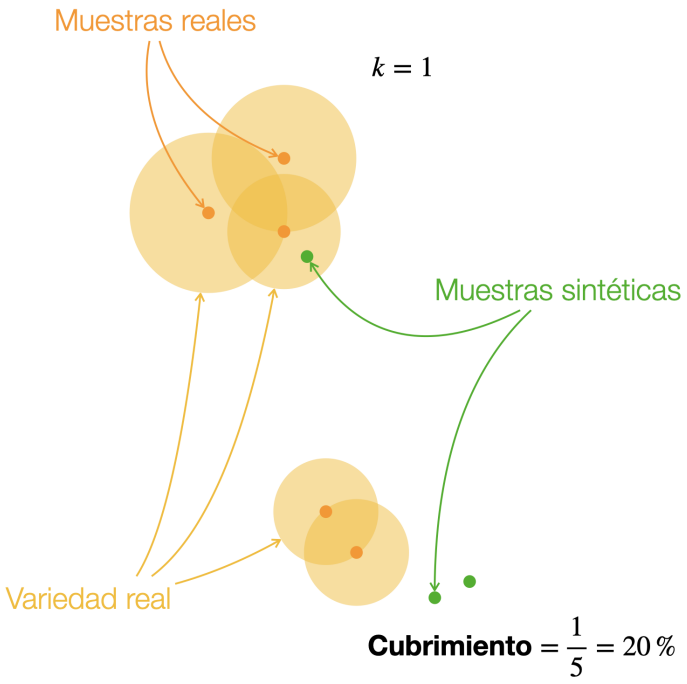


Figura 3.11: Ilustración del cálculo de cubrimiento. Se identifica en naranja la muestras reales y en verde las muestras sintéticas. El radio-k utilizado para el cálculo es igual a 1.

3.7.2. Contrastes de hipótesis de normalidad

En esta sección se presentan los contrastes de hipótesis utilizados para evaluar la normalidad de los resultados, los cuales son importantes para determinar las condiciones bajo las que se comparan los resultados obtenidos por los algoritmos. Se presentan los contrastes de D'Agostino–Pearson; de Anderson–Darling y de Shapiro–Wilk. Los coeficientes de asimetría g_1 o $\sqrt{b_1}$ y de curtosis b_2 son conceptos centrales en la noción de normalidad y están definidos en la Ecuación 3.18, dónde $\mu_k = \mathbb{E}\left[(X - \mathbb{E}X)^k\right]$ es el k-ésimo momento centrado en la media y $\sigma = \sqrt{\mu_2}$ es la desviación estándar de la variable aleatoria X en consideración.

$$g_1 = \sqrt{b_1} = \frac{\mu_3}{\sigma^3} \qquad b_2 = \frac{\mu_4}{\sigma^4} \tag{3.18}$$

Cuando una variable aleatoria Z es normal, su cuarto momento centrado es $\mu_4 = 3\sigma^4$, por lo que su curtosis es $b_2 = 3$, permitiendo definir el estadístico de exceso de curtosis g_2 respecto a una distribución normal en la Ecuación 3.19.

$$g_2 = b_2 - 3 = \frac{\mu_4}{\sigma^4} - 3 \tag{3.19}$$

Tanto la asimetría como la curtosis son características de la forma de una distribución de probabilidad. El coeficiente de asimetría mide el grado de asimetría de la distribución, tomando como eje de simetría a la línea vertical definida por su media. Por otro lado, a mayores valores de curtosis se observa mayor concentración de los valores de la distribución en torno a su media o sobre sus colas y menor concentración en las regiones intermedias.

En el análisis estadístico realizado en este proyecto de grado, X es la variable aleatoria que representa el valor de una métrica de evaluación específica (FID, densidad o cubrimiento) para la salida de una ejecución de un algoritmo específico (NSGA–II, FV–MOEA, o MOEA/D). Por lo tanto, para N ejecuciones de un algoritmo se cuenta con una muestra de $x_1, \ldots, x_N$ compuesta por N realizaciones de la variable X. En este contexto, los momentos μ_k de X se pueden estimar mediante los momentos muestrales $\hat{\mu}_k = \frac{1}{n}\sum_{i=1}^{N}\left[(x_i - \hat{\mu}_1)^k\right]$ y el desvío estándar se puede estimar insesgadamente utilizando $\hat{\sigma} = S = \sqrt{\frac{1}{n}\hat{\mu}_2}$, observando que $\hat{\mu}_1$ equivale a la media muestral $\overline{x} = \frac{1}{n}\sum_{i=1}^{N} x_i$ y es un estimador insesgado de la media μ. Los estadísticos que estiman asimetría, curtosis y exceso de curtosis: $\hat{g}_1$, $\hat{b}_2$ y $\hat{g}_2$, respectivamente; se derivan de las definiciones de sus parámetros análogos utilizando los estimadores $\hat{\mu}_k$ y S mencionados anteriormente.

3.7.2.1. Test de D'Agostino-Pearson

El test de D'agostino–Pearson plantea la hipótesis nula de que la muestra $X_1, \ldots, X_N$ en consideración proviene de una variable aleatoria X que se distribuye con normalidad. Se plantean los estadísticos Z_1 [13] y Z_2 [3] como transformaciones de la asimetría y la curtosis. Z_1 y Z_2 están definidos en las ecuaciones 3.20 y 3.21, respectivamente, dónde δ, α, son constantes derivadas por D'Agostino y Pearson; A es una constate derivada por Anscombe y Glynn [3]; $\mu_2(g_1)$ y $\mu_2(g_2)$ son las varianzas de g_1 y g_2 derivadas por Pearson [57] y $\mu_1(g_2)$ es la media de g_2, también derivada por Pearson [57]

$$Z_1 = \delta \operatorname{arcsinh}\left(\frac{g_1}{\alpha\sqrt{\mu_2(g_1)}}\right) \tag{3.20}$$

$$Z_2 = \sqrt{\frac{9A}{2}} \left[1 - \frac{2}{9A} - \left(\frac{1 - 2/A}{1 + \frac{g_2 - \mu_1(g_2)}{\sqrt{\mu_2(g_2)}}} \sqrt{2/(A-4)} \right) \right] \tag{3.21}$$

Los estadísticos Z_1 y Z_2 se componen en el estadístico $K^2 = (Z_1)^2 + (Z_2)^2$, que se distribuye χ^2 con dos grados de libertad bajo la hipótesis nula [12]. Esta información puede utilizarse para calcular el p-valor del test.

3.7.2.2. Test de Anderson–Darling

El test de Anderson–Darling establece la hipótesis nula de que la muestra $X_1, \ldots, X_N$, proviene de una variable aleatoria X que se distribuye con normalidad. Se plantea el estadístico A^2 definido en la Ecuación 3.22, dónde Φ es la distribución acumulada de la normal estándar, $Y_1, \ldots, Y_N$ es la muestra normalizada y ordenada crecientemente. La normalización de la muestra implica la estimación de sus primeros dos momentos (la media y la varianza) según mencionado anteriormente.

$$A^2 = -n - \sum_{i=1}^{n} \frac{2i-1}{n} \left[\ln(\Phi(Y_i)) + \ln\left(1 - \Phi(Y_{n+1-i})\right) \right] \tag{3.22}$$

A^2 se deriva de un criterio de distancia entre la función de distribución acumulada empírica, obtenida de la muestra normalizada, y la función de distribución acumulada de la Normal estándar. Los p-valores están precalculados para los valores críticos más usuales en la biblioteca de estadística Scipy [77].

3.7.2.3. Test de Shapiro–Wilk

El test de Shapiro–Wilk tiene como hipótesis nula que la muestra $X_1, \ldots, X_N$ se distribuye con normalidad. Se utilizan los estadísticos de orden $X_{(i)}$, que representan al i-ésimo elemento de la muestra en orden creciente. Sea una muestra aleatoria $Z_1, \ldots, Z_N$ independiente e idénticamente distribuida proveniente de una población normal, se definen sus estadísticos de orden $Z_{(1)}, \ldots, Z_{(N)}$, el vector de medias $\mathbf{m} = (m_1, \ldots, m_N)$ dónde $m_i = \mathbb{E}Z_{(i)}$ y $\mathbf{V} \in M_{N \times N}$ la matriz de covarianzas dónde $\mathbf{V}_{ij} = \mathrm{COV}(Z_{(i)}, Z_{(j)})$. Se plantea el estadístico W definido en la Ecuación 3.23, dónde $x_{(i)}$ es la realización de $X_{(i)}$ y s^2 es la varianza muestral observada.

$$W = \frac{\left(\sum_{i=1}^{N} a_i x_{(i)}\right)^2}{ns^2} \quad : \quad (a_1, \ldots, a_N) = \frac{\mathbf{m}^T \mathbf{V}^{-1}}{\|\mathbf{m}^T \mathbf{V}^{-1}\|} \tag{3.23}$$

La distribución de W es desconocida, por lo que los valores críticos y los p-valores se calculan mediante estimaciones de Monte Carlo.

3.7.3. Comparación de muestras

En esta sección se introducen los contrastes de hipótesis que son utilizados para comparar las muestras, obtenidas al ejecutar los distintos algoritmos. En este proceso se consideran dos muestras aleatorias $\vec{X} = X_1 \ldots X_n$ e $\vec{Y} = Y_1, \ldots Y_n$ provenientes de dos poblaciones X e Y independientes, definidas sobre el mismo soporte, y cuyas distribuciones están determinadas por funciones de distribución acumulada F_x y F_y. A partir de las muestras $\vec{X}$ e $\vec{Y}$, se desea establecer una relación de orden entre X e Y que modele la noción de las realizaciones de una variable suelen tomar mayores valores que las de la otra.

El orden estocástico [70] se define según la Ecuación 3.24 para X e Y definidas en un soporte S: X es estocásticamente mayor o igual a Y cuando la probabilidad de que X sea mayor que cualquier valor del soporte es mayor que la probabilidad de que Y sea mayor que ese mismo valor. Alternativamente, esta idea puede expresarse según las funciones de distribución acumulada F_x y F_y (Ecuación 3.25). Finalmente, en la Ecuación 3.26 se define la versión estricta de este orden, en la cual se exige que F_x sea estrictamente mayor que F_y en algún punto.

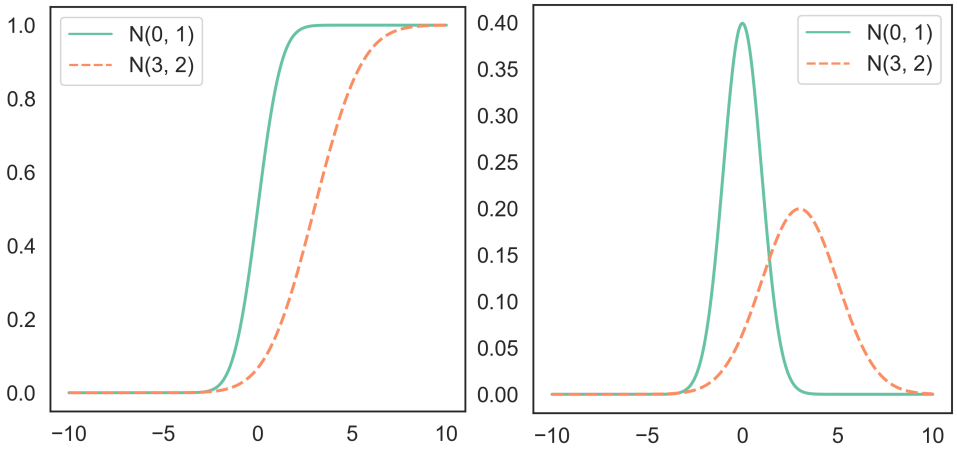
$$X \geq_{st} Y \iff (\forall x \in S)\, \mathbb{P}(X \geq x) \geq \mathbb{P}(Y \geq x) \tag{3.24}$$

$$X \geq_{st} Y \iff (\forall x \in S)\, F_x(x) \geq F_y(x) \tag{3.25}$$

$$X >_{st} Y \iff (\forall x \in S)\, F_x(x) \geq F_y(x) \quad \wedge \quad (\exists x \in S)\, F_x(x) > F_y(x) \tag{3.26}$$

En la Figura 3.12a se muestran las funciones de distribución acumulada F_x y F_y de una variable normal estándar $X \sim N(0, 1)$ y de una normal con media igual a 3 y desvío estándar igual a 2, $Y \sim N(3, 2)$. En este caso F_x acota superiormente a F_y en toda la recta real (el soporte de X e Y) por lo

que X es estocásticamente mayor que Y. Por otro lado, en la Figura 3.12b se presentan las densidades de X e Y y se observa que, a pesar de que $X >_{st} Y$, la última es la distribución que acumula más probabilidad en valores altos del soporte. Se destaca, entonces, que las mayores variables según el criterio de orden estocástico son aquellas que acumulan probabilidad más rápido y no aquellas que suelen realizarse en mayores valores.



(a) F_x **(b)** f_x

Figura 3.12: Funciones de distribución acumulada F_x y de densidad f_x de variables aleatorias normales. En verde, la normal estándar, en naranja, una normal con media 3 y desvío 2

En virtud de la definición de orden estocástico y su interpretación, se define el concepto de dominancia estocástica [58] entre variables aleatorias. Esta relación establece un orden inverso al determinado por el orden estocástico, lo cual facilita su interpretación. Para el contexto de este proyecto de grado, es de interés la dominancia estocástica de primer órden, definida según la Ecuación 3.27 para dos variables aleatorias X e Y con soporte S.

$$X \succ_{(1)} Y \iff Y >_{st} X \tag{3.27}$$

Se dice que X domina estocásticamente a Y cuando X es estrictamente menor que Y según el criterio de orden estocástico. La interpretación de esta relación es directa y coincide con la noción de que X suele tomar mayores valores que Y, por lo que es ampliamente utilizada en el campo de la teoría de la decisión.

En este proyecto de grado, dados una métrica de evaluación ε, un algoritmo α y un número de ejecuciones n, se considera una muestra $\vec{X}(\varepsilon, \alpha, n) =$

$X_1 \dots X_n$ dónde cada X_i corresponde al valor observado de ε sobre los resultados del algoritmo α en la i-ésima ejecución. Sean dos algoritmos α_1 y α_2, se obtienen dos muestras $\vec{X}_1 = \vec{X}(\varepsilon, \alpha_1, n) \sim A_1$ y $\vec{X}_2 = \vec{X}(\varepsilon, \alpha_2, n) \sim A_2$ y se desea estudiar la relación de dominancia estocástica entre A_1 y A_2. Si se cumpliera $A_1 \succ_1 A_2$, podría concluirse que α_1 tiene mejor desempeño en ε que α_2.

Se utilizan contrastes de hipótesis para estudiar la relación de dominancia estocástica entre A_1 y A_2 a partir de las realizaciones de las muestras X_1 y X_2. Se destacan las pruebas no paramétricas de Wilcoxon; de Mann-Whitney [51]; de Kruskall-Wallis y de Kolmogorov-Smirnov [36]. Se entiende que las muestras X_1 y X_2 son independientes, por lo que se descarta la prueba de Wilcoxon – recomendada para muestras emparejadas. La hipótesis nula de la prueba de Mann-Whitney se interpreta en su forma más general como la igualdad entre las muestras estudiadas y solamente admite el estudio de orden mediante dominancia estocástica o comparación de medianas cuando las muestras provienen de la misma distribución, a menos de una traslación [19, 33]. La prueba de Kruskall-Wallis es una extensión de la de Mann-Whitney, permitiendo su aplicación a más de dos muestras en simultáneo, y requiere las mismas hipótesis que ésta para el estudio del orden entre muestras.

La prueba de Kolmogorov-Smirnov (KS), estudia la bondad de ajuste entre una muestra y una distribución teórica (KS de una muestra) o entre dos muestras (KS de dos muestras). En el último caso, admite variantes para estudiar la dominancia estocástica de las muestras en ambas direcciones. Por este motivo, se enfoca en la prueba de Kruskall-Wallis para analizar la (des)igualdad de las muestras obtenida y en la prueba de Kolmogorov-Smirnov para analizar la dominancia estocástica sin imponer hipótesis fuertes.

3.7.3.1. Prueba de Kruskall-Wallis

La prueba de Kruskall-Wallis es una extensión del contraste de Mann-Whitney tiene como hipótesis nula que, dadas k muestras aleatorias $\vec{X}_1 \dots \vec{X}_k$, éstas provienen de la misma distribución. Si todas las muestras tuvieran semejante escala y forma, se podría interpretar la hipótesis alternativa como que al menos una de las muestras proviene de una distribución estocásticamente dominante [19, 33]. En otro caso, se interpreta simplemente que alguna de las muestras proviene de una distribución distinta.

Sean k muestras, n_i el número de observaciones en la i-ésima muestra y $N = \sum_{i=1}^{k} n_i$ el número total de observaciones; el cómputo de esta prueba requiere ordenar los datos de las k en una única lista, atribuyéndoles un rango $r_{ij} \in 1 \dots N$ a cada observación i de la muestra j. Sean $\overline{r}_j$ el promedio de los rangos de la muestra j y $\overline{r}$ el promedio de todos los rangos, el estadístico H de Kruskall-Wallis está definido en la Ecuación 3.28.

$$H = (N-1)\frac{\sum_{i=1}^{k} n_i(\overline{r}_i - \overline{r})^2}{\sum_{i=1}^{k}\sum_{i=1}^{n_i}(\overline{r}_{ij} - \overline{r})^2} \tag{3.28}$$

Existen tablas de probabilidad para calcular el p-valor del test a partir de la realización del estadístico H, aunque su distribución también se puede aproximar como una χ^2_{k-1}.

3.7.3.2. Prueba de Kolmogorov-Smirnov

La prueba de Kolmogorov-Smirnov es una prueba de bondad de ajuste de una muestra a una distribución teórica, o entre dos muestras. En el caso de dos muestras, puede utilizarse no solamente para analizar si estas provienen de la misma población (KS de dos colas), sino también para estudiar la dominancia estocástica entre ellas (KS de una cola). En el último caso, dadas dos muestras $\vec{X} \sim F_x$ e $\vec{Y} \sim F_y$, la hipótesis nula de la prueba es que $(\forall x)\, F_x(x) \geq F_y(x)$. O bien, que X es estocásticamente mayor o igual que Y ($X \geq_{st} Y$), o incluso que (con cierto nivel de aproximación) Y domina estocásticamente a X ($Y \succ_{(1)} X$).

El estadístico de Kolmogorov-Smirnov D para dos muestras está basado en sus funciones de distribución acumulada empírica como se define en la Ecuación 3.29, dónde $\hat{F}_x(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{x_i \leq x\}$, n es la cantidad de observaciones en la muestra y $\mathbb{1}$ es la función indicatriz.

$$D = \sup_x |\hat{F}_x(x) - \hat{F}_y(x)| \tag{3.29}$$

En este caso, también existen tablas de distribución precalculadas que permiten calcular el p-valor o los valores críticos del estadístico para rechazar, o no, la hipótesis nula.

3.8. Message Passing Interface

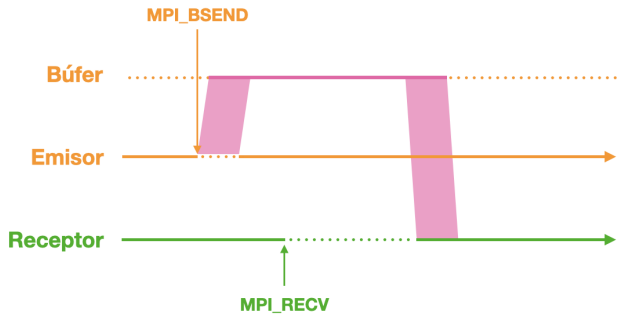
MPI es un estándar de pasaje de mensajes entre procesos que incluye definiciones relacionadas a las comunicaciones punto a punto, a las comunicaciones colectivas, a las topología y al manejo de ambientes de ejecución. Existen diversas bibliotecas, como MPICH, que implementan el estándar y facilitan su adopción por usuarios que requieran utilizar el paradigma de computación distribuida.

Las comunicaciones punto a punto refieren al pasaje de mensajes entre un par de procesos realizado en un canal unidireccional. Uno de los procesos (emisor) es responsable por enviar el mensaje y el otro (receptor) por recibirlo. Los datos del mensaje deben guardarse en un búfer provisto por el usuario y el envío puede realizarse de forma bloqueante o no bloqueante y de forma síncrona o asíncrona. En la Figura 3.13 se representan los modos de comunicación.

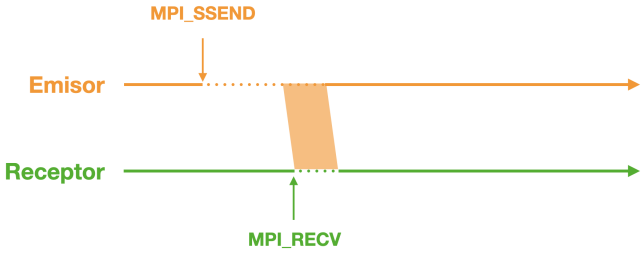
Un envío es bloqueante cuando retorna luego de la liberación del búfer, lo que puede ocurrir porque MPI copió el mensaje a un búfer interno (Figura 3.13a) o porque la comunicación se realizó mediante el rendezvous con el proceso emisor (Figura 3.13b). Por otro lado, un envío es no bloqueante cuando retorna sin garantizar la liberación del búfer, lo que permite continuar el procesamiento en el emisor a medida que ocurre la copia (Figura 3.13c). Además, un envío es síncrono cuando no retorna hasta que ocurre el rendezvous y el mensaje es recibido por el receptor (Figura 3.13b).

El envío estándar de un mensaje en MPI se realiza mediante el comando `MPI_SEND`. En el modo estándar, la implementación de MPI decide si el envío será bloqueante o no, dependiendo del estado del sistema y de la disponibilidad de recursos. Un envío en modo bloqueante puede ser invocado con los comandos `MPI_BSEND` o `MPI_SSEND`. En el primer caso, se fuerza la utilización del búfer, mientras que en el segundo se fuerza el sincronismo del mensaje y no es necesario utilizar el búfer. Por otro lado, para enviar un mensaje en modo no bloqueante (y asíncrono, consecuentemente) se debe utilizar la instrucción `MPI_ISEND`. Las Figuras 3.13a, 3.13b y 3.13c ilustran los flujos de datos y las líneas de ejecución entre los procesos emisor y receptor utilizando `MPI_BSEND`, `MPI_SSEND` y `MPI_ISEND`, respectivamente.

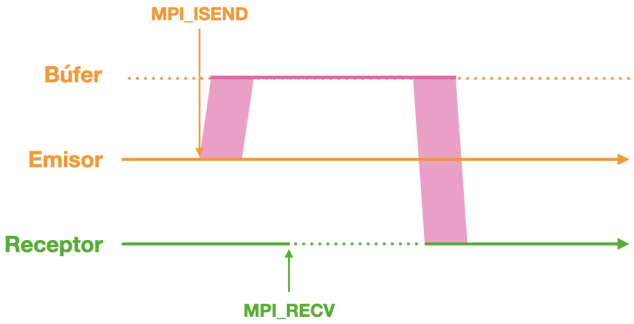
Las comunicaciones colectivas refieren al pasaje de mensajes replicados a todos los miembros de un grupo. En MPI, los procesos se agrupan en intracomunicadores que facilitan las rutinas de comunicación entre los procesos del



(a) Envío asíncrono y bloqueante con `MPI_BSEND`. Con esta instrucción se fuerza la utilización del búfer y el proceso emisor se bloquea mientras los datos se copian al proceso.



(b) Envío síncrono y bloqueante con `MPI_SSEND`. El proceso emisor se bloquea hasta que el receptor está pronto para recibir el mensaje. Los datos se transmiten directamente en el rednezvous, por lo que no es necesario utilizar el búfer.



(c) Envío asíncrono y no bloqueante con `MPI_ISEND`. El emisor instruye a MPI la copia de los datos al búfer y continúa su procesamiento inmediatamente.

Figura 3.13: Representación gráfica de los distintos modos de envío en comunicaciones punto a punto de MPI. Se representa la ejecución en el tiempo de cada proceso participante: el emisor y el receptor. La línea puntada indica que el proceso está suspendido, mientras que la línea sólida indica que se encuentra en ejecución. También se representa el estado del buffer: punteado para libre, sólido para ocupado. Entre las líneas de ejecución se representa la transmisión de datos.

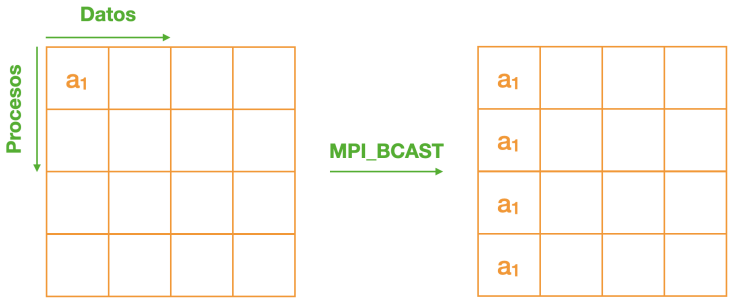
grupo. Todos los procesos en un ecosistema MPI se agrupan en un intracomunicador global (`MPI_WORLD`), en el cual se identifican por rangos entre 1 y N, siendo N la cantidad total de procesos. A partir de `MPI_WORLD` pueden instanciarse otros comunicadores que contengan subconjuntos de procesos o faciliten la implementación de topologías convenientes para la lógica de negocio. Ejemplos de intracomunicadores que implementan topologías son la grilla cartesiana (`MPI_Cart`) y la estructura de grafo (`MPI_Graph`). Este tipo de intracomunicador introduce la noción de localidad entre procesos, proveyendo utilidades que permiten diseñar en función de esta noción (por ejemplo, para obtener a los vecinos de un proceso en la grilla) y posibilitando la optimización en la asignación de recursos (por ejemplo, asignando procesos cercanos en la topología a nodos de cómputo cercanos físicamente).

Las comunicaciones colectivas se clasifican en tres categorias: uno a todos, todos a uno y todos a todos. Ejemplos de comunicación colectiva en estas categorías son las funciones `MPI_BCAST`, `MPI_GATHER` y `MPI_ALLGATHER`, respectivamente. En la Figura 3.14 se representan gráficamente estas funciones, presentando el estado inicial de los datos de cada proceso de un intracomunicador de cuatro procesos y el estado resultante al finalizar la rutina de comunicación.

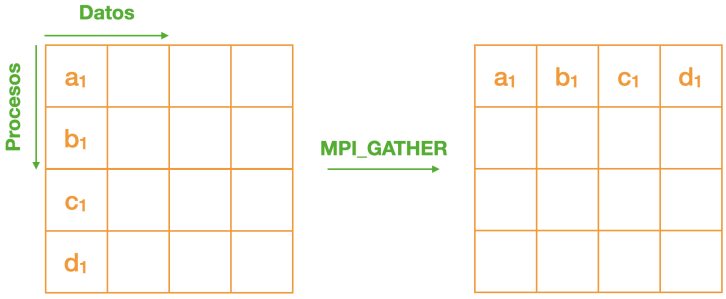
Al utilizar `MPI_BCAST` se realiza una difusión amplia del dato provisto por el proceso emisor a todos los procesos del intracomunicador. En este caso, se cuenta con un único proceso emisor y N procesos receptores, por lo que se categoriza como una comunicación de uno a todos. El proceso emisor es el único que cuenta con datos para enviar, que son obtenidos por todos los procesos restantes al finalizar la comunicación. Esto se ilustra en la Figura 3.14a.

Por otro lado, `MPI_GATHER` se clasifica dentro de las comunicaciones de todos a uno, por lo que se cuenta con N procesos emisores y un proceso receptor. En este caso, todos los procesos tienen datos para enviar, que serán obtenidos por el proceso receptor al finalizar la comunicación. El proceso receptor concluye con un arreglo de datos indexados por proceso, como se muestra en la Figura 3.14b.

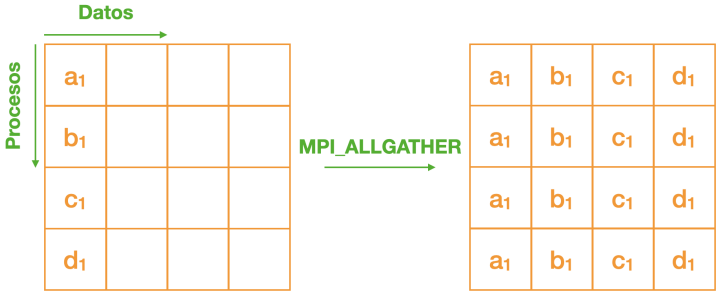
Finalmente, en la rutina de comunicación de todos a todos `MPI_ALLGATHER` se cuenta con N emisores y N receptores. De manera semejante a `MPI_GATHER`, todos los procesos tienen datos para enviar, aunque los datos deben ser propagados a todos los procesos restantes. Como se observa en la Figura 3.14c, todos los procesos obtienen el mismo arreglo de datos indexado por proceso.



(a) Envío colectivo con `MPI_BCAST`. Este es un ejemplo de envío uno a todos, en el cual el proceso emisor envía el mismo dato a todos los procesos del intracomunicador. Todos los procesos concluyen la comunicación con el mismo dato único.



(b) Envío colectivo con `MPI_GATHER`. Este es un ejemplo de envío todos a uno, en el cual cada proceso emisor envía un dato distinto al proceso receptor, que concluye la comunicación con un arreglo de datos indexado por proceso.



(c) Envío colectivo con `MPI_ALLGATHER`. Este es un ejemplo de envío todos a todos, en el cual cada proceso envía el mismo dato a todos los restantes. Es semejante a `MPI_GATHER`, pero todos los procesos concluyen la comunicación con el mismo arreglo de datos indexado por proceso.

Figura 3.14: Representación gráfica de distintas rutinas de comunicación colectiva en MPI para un intracomunicador con cuatro procesos. Se presentan matrices en el plano Procesos×Datos, que representan arreglos de datos en cada proceso del intracomunicador. A la izquierda se muestra la matriz de datos inicial y a la derecha se muestra la matriz resultante al concluir un protocolo de comunicación colectiva determinado.

Además de las rutinas de comunicación ya mencionadas, MPI cuenta con rutinas para sincronizar procesos en un intracomunicador, como `MPI_BARRIER`, y para verificar la existencia de mensajes para ser recibidos, como `MPI_PROBE` (bloqueante) y `MPI_IPROBE` (no bloqueante). `MPI_BARRIER` es una rutina de comunicación colectiva que bloquea a todos los procesos del intracomunicador hasta que todos llegan al mismo punto del programa (la barrera). Por otro lado, `MPI_PROBE` simula la recepción de un mensaje sin consumirlo, lo cual es de utilidad para monitorear la cola de mensajes previniendo la suspensión del procesamiento. Otras primitivas importantes para el funcionamiento de MPI son `MPI_INIT` y `MPI_FINALIZE`, que se encargan de inicializar y finalizar la estructura de comunicación para cada proceso.

Capítulo 4

Metodología

La metodología seguida en el desarrollo de este proyecto de grado se organizó en tres etapas. La investigación inició con la conformación de una reseña del estado del arte en las áreas de interés (aumentación de conjuntos de datos, redes neuronales generativas antagónicas y algoritmos coevolutivos multiobjetivo) y sus intersecciones. Luego de la reseña del estado del arte se realizó la extensión del framework Lipizzaner para el soporte de funciones multiobjetivo y la adaptación de su esquema de distribución. Finalmente, se realizó el análisis experimental de las soluciones implementadas sobre el framework. En este capítulo se presenta el diseño metodológico de cada una de esas etapas, resaltando las decisiones tomadas y los pasos seguidos.

4.1. Metodología de investigación sobre el estado del arte

Para delinear el estado del arte se realizó una investigación bibliográfica en publicaciones recientes según el área. Para los temas de aumentación de conjuntos de datos y redes neuronales generativas antagónicas se favorecieron publicaciones con menos de cinco años de antigüedad para caracterizar dos aspectos relevantes: el problema abordado, definiéndolo, explorando sus dificultades y conociendo dónde se presenta; y el modelo escogido, entendiendo cómo funciona, en qué casos es aplicable y cuáles son las principales dificultades en su entrenamiento. En el contexto del estudio de las dificultades del entrenamiento de GANs se estudiaron indicadores para la diversidad de los datos generados, que deben utilizarse para guiar el entrenamiento del modelo o para apoyar su evaluación.

Respecto al tema de algoritmos coevolutivos multiobjetivo se optó por estudiar la teoría sobre computación evolutiva, algoritmos coevolutivos y optimización multiobjetivo. El objetivo del estudio fue comprender el marco de trabajo de la investigación, dado que Lipizzaner es un algoritmo evolutivo, así como poder seleccionar las técnicas de optimización multiobjetivo a implementarse en el framework. El estudio incluyó el análisis de publicaciones que presentan a los algoritmos evolutivos multiobjetivo clásicos y también el análisis de propuestas más modernas.

4.2. Metodología de adaptación de Lipizzaner al estándar Message Passing Interface

La extensión del framework Lipizzaner, base del desarrollo de este proyecto de grado, se divide en dos partes. Por una parte, la reimplementación del esquema de distribución original (escrita bajo el modelo cliente-servidor, basado en *sockets* y el protocolo HTTP) según el paradigma de pasajes de mensajes. Por otra parte, la implementación de las estructuras y algoritmos necesarios para soportar el procesamiento de la función de aptitud multidimensional y el manejo de múltiples soluciones óptimas dispuestas en frentes no dominados.

Fue necesario reimplementar el esquema de distribución de Lipizzaner para la ejecución de los experimentos en la plataforma de computación de alta performance, ClusterUY, el Centro Nacional de Computación. La reimplementación se debió a que el paradigma de computación distribuida trabaja sobre middleware específico y no utilizando implementaciones de bajo nivel. Se optó, entonces, por la distribución del algoritmo con el paradigma de pasaje de mensajes, que se corresponde con el pasaje de mensajes entre las poblaciones de individuos del algoritmo teórico de Lipizzaner y su uso es ampliamente difundido en plataformas de cómputo de alto desempeño gracias a su estandarización mediante el estándar MPI.

La metodología utilizada para realizar la adaptación consistió en el análisis de las estructuras existentes en Lipizzaner, con énfasis en las estructuras relacionadas al ciclo principal de entrenamiento y a la distribución de los individuos. Aunque Lipizzaner contempla el caso especial de entrenamiento de GANs condicionales supervisadas, se acotó la implementación al entrenamiento de GANs no-supervisadas, debido a que éstas son el objeto de este proyecto

de grado. El procedimiento de adaptación fue útil para evaluar el acoplamiento de las estructuras en el protocolo de comunicación original, identificando las estructuras que debían ser descartadas y las que debían ser reutilizadas. Finalmente, se diseñó e implementó la nueva arquitectura, utilizando la biblioteca mpi4py [14].

4.3. Metodología de extensión de Lipizzaner para problemas multiobjetivo

El ciclo de entrenamiento de Lipizzaner consiste en repetir cinco pasos principales:

1. Intercambio de los mejores individuos con el vecindario.
2. Evaluación de las poblaciones de generadores y discriminadores.
3. Selección de generadores y discriminadores para la mutación.
4. Evaluación de la nueva generación.
5. Selección de generadores y discriminadores para el reemplazo.

Lipizzaner optimiza un problema mono-objetivo en su ciclo evolutivo utilizando la función de costo de los generadores y discriminadores como el objetivo a optimizar. La función de costo se evalúa en los puntos 2 y 4 del ciclo de entrenamiento.

La motivación detrás de extender Lipizzaner para que sea capaz de resolver problemas multiobjetivo es poder optimizar un objetivo que se enfoque específicamente en la diversidad de los resultados generados y mantener la optimización del objetivo ya optimizado en framework original. Se agregaron dos funciones de costo para GANs: E-GAN y GDPP. Ambas funciones de costo fueron diseñadas para evaluar la capacidad de los generadores de crear conjuntos de datos sintéticos diversos. Estas funciones de costo se agregaron a Lipizzaner como los nuevos objetivos enfocados en la diversidad de los resultados generados.

En lo que refiere a la extensión de Lipizzaner para el soporte de algoritmos de optimización multiobjetivo, basado en el estudio sobre el estado del arte de MOEAs, se optó por implementar tres algoritmos: NSGA–II, MOEA/D y FV–MOEA. Los tres algoritmos son buenos representantes de las principales clases de MOEAs. NSGA–II es un MOEA clásico basado en Pareto. MOEA/D

es un MOEA clásico que aplica la descomposición del problema de optimización multiobjetivo. FV–MOEA es un MOEA vanguardista que se basa en indicadores. La metodología utilizada para el desarrollo de los tres algoritmos consistió en implementar los tres algoritmos como operadores evolutivos dentro de la arquitectura distribuida de Lipizzaner y disponibilizar las tres configuraciones en la interfaz de usuario de la biblioteca. Los tres algoritmos se aplican específicamente en la selección de generadores en el paso 5 del ciclo de entrenamiento.

4.4. Metodología de evaluación experimental

Los algoritmos implementados en este proyecto de grado se someten a un proceso de evaluación experimental, cuyo objetivo es generar evidencia sobre el desempeño y la escalabilidad. Este proceso se divide en tres partes: 1) configuración paramétrica, 2) evaluación de desempeño y 3) evaluación de escalabilidad. En esta sección se explica la metodología seguida en cada uno de los tres componentes de la evaluación experimental y se introducen los conjuntos de datos utilizados durante la ejecución de los experimentos.

4.4.1. Repositorios de datos

Se optó por utilizar conjuntos de datos relativamente pequeños y ampliamente adoptados. Específicamente, la evaluación experimental se realizó sobre los conjuntos de datos clásicos Modified National Institute of Standards and Technology (MNIST) [16] y CelebFaces Attributes (CelebA) [50]. En esta subsección se introducen brevemente ambos conjuntos.

4.4.1.1. MNIST

El conjunto de datos MNIST esta compuesto por imágenes de dígitos escritos a mano. El conjunto de datos surge de la unión de dos conjuntos de datos de un mismo instituto, los cuales consisten de dígitos escritos por estudiantes de liceos estadounidenses y por empleados del *American Census Bureau*. Algunas de las características de este conjunto son:

- Cuenta con 70.000 imágenes de dígitos escritos a mano del 0 al 9, de las cuales 60.000 componen el conjunto de entrenamiento y 10.000 el de evaluación.
- Cada imagen es de un tamaño de 28×28 y un solo canal de color (imágenes en blanco y negro).

Desde su publicación en 1999, MNIST se ha convertido en uno de los conjuntos más populares para comparar algoritmos de visión por computadora. En la Figura 4.1 se pueden ver algunos ejemplares del conjunto.

Figura 4.1: Ejemplos del conjunto de evaluación de MNIST.

4.4.1.2. CelebA

El conjunto de datos CelebA esta compuesto por más de doscientas mil imágenes de caras de celebridades, cada una con cuarenta atributos anotados. Las imágenes contienen gran variedad de poses y de fondos. Algunas de las características de este conjunto son:

- Cuenta con 10.117 identidades distintas.
- Cuenta con 202.599 imágenes de caras.
- Incluye coordenadas de 5 puntos por imagen indicando ambos ojos, extremo derecho e izquierdo de la boca, y la nariz.
- 40 atributos incluyendo género, edad, tamaño de nariz y labios, características de ojos y cabello, inclusión de artículos faciales, entre otros.
- Cuenta con las coordenadas de las cajas que envuelven a cada una de las caras.

- Las imágenes tienen un tamaño de 178×218 píxeles y tres canales de color en formato RGB.

Las características enumeradas implican que CelebA es utilizado para entrenar clasificadores de atributos faciales, algoritmos de identificación facial, detección facial, entre otras tareas que involucren la utilización de rostros.

En este proyecto de grado las anotaciones fueron ignoradas y se utilizan únicamente las imágenes para el entrenamiento de las GANs. En la Figura 4.2 se pueden ver algunos ejemplares del conjunto de datos y ejemplos de sus atributos.

Figura 4.2: Ejemplos del conjunto CelebA junto a algunos de sus atributos.

4.4.2. Metodología para los experimentos de la configuración paramétrica

La configuración paramétrica es una etapa previa a la evaluación de desempeño que tiene como objetivo ajustar los parámetros de los algoritmos evaluados (NSGA–II, FV–MOEA y MOEA/D). La metodología consistió en identifi-

car los parámetros más relevantes de cada algoritmo, seleccionar un subconjunto de valores de prueba para cada uno de ellos y realizar pruebas de desempeño para todas las configuraciones reportando métricas que permitan comparar y elegir la mejor configuración. Una configuración es una combinación de valores posibles para cada parámetro.

En este proyecto de grado se configuraron los siguientes parámetros: la dimensión de la grilla (2×2, 3×3, 4×4) y la función de costo (GDPP y EGAN) formando 6 configuraciones distintas para cada uno de los tres algoritmos.

La configuración paramétrica se realizó sobre el conjunto MNIST y la arquitectura a entrenar fue el perceptrón de cuatro capas con 700 neuronas descripto en el artículo original de Lipizzaner. Además, un conjunto de parámetros permanecieron fijos para todas las configuraciones, los cuales se fijaron con los mismos valores que los utilizados para el artículo original de Lipizzaner. La Tabla 4.1 muestra los parámetros que permanecieron fijos y que valor tomó cada uno.

parámetro	*valor*
tamaño de batch	100
número de iteraciones	200
tasa de aprendizaje	0.0002
tamaño de población	2
tamaño de reemplazo	1
tamaño de torneo	2

Tabla 4.1: Tabla de parámetros con valores fijos en la configuración paramétrica

Aunque en la etapa de configuración paramétrica se estudió el uso de grillas de tamaño 4×4, las pruebas preliminares con el conjunto de datos CelebA y la arquitectura de red DCGAN mostraron limitaciones de memoria en las tarjetas gráficas del entorno de ejecución. Las limitaciones fueron causadas por la arquitectura de DCGAN, que cuenta con un mayor número de parámetros que el perceptrón de cuatro capas y por las imágenes de CelebA, que tienen mayor resolución que las de MNIST, generando un perfil de uso de memoria de video mayor al soportado por la infraestructura disponible durante el entrenamiento. Las limitaciones de memoria impidieron utilizar las configuraciones con grillas de tamaño 4×4 en la etapa de evaluación de desempeño. En las figuras y tablas de la Sección 6.3 se incluyen, por completitud, los resultados de

las configuraciones con un tamaño de grilla de 4×4 aunque no fueron tomadas en cuenta para la selección de parámetros finales utilizados en la evaluación de desempeño.

Las ejecuciones de las configuraciones evaluadas demandaron entre 35 y 189 minutos para ejecutar, por lo que se optó por realizar diez ejecuciones independientes utilizando cada configuración. Se ejecutaron 180 experimentos, que demandaron 282 horas de cómputo (aproximadamente once días continuos). Debido a que los algoritmos evaluados no son deterministas, los valores reportados son las medianas de los resultados de las ejecuciones.

Se eligió para cada algoritmo la configuración dominante en las métricas reportadas de densidad y cubrimiento. En los casos donde existe un frente de soluciones no dominadas entre sí se seleccionó la configuración con el menor valor de FID entre el conjunto de configuraciones no dominadas.

4.4.3. Metodología para la evaluación de desempeño

Para la evaluación de desempeño se reportan la fidelidad y la diversidad de las imágenes generadas por los modelos obtenidos mediante cada uno de los algoritmos evaluados. Se emplea la densidad como métrica de la fidelidad de las imágenes y el cubrimiento como métrica de su diversidad. Además, se reporta el FID de las imágenes generadas.

En la etapa de evaluación de desempeño se utilizó el conjunto de datos CelebA y la arquitectura de las GANs es DCGAN. Al igual que en la configuración paramétrica, la elección de conjuntos de datos, arquitectura de las redes y los parámetros de la Tabla 4.2 fueron inspirados por las elecciones del artículo original de Lipizzaner. Esta elección permite utilizar la implementación estándar de Lipizzaner como línea de base para comparar los resultados.

Las tres configuraciones ejecutadas demandaron entre 384 y 456 minutos por ejecución, por lo que se optó por realizar diez ejecuciones independientes utilizando cada configuración. Se ejecutaron 30 experimentos, que demandaron 239 horas de cómputo (aproximadamente diez días continuos). Al igual que en la configuración paramétrica, debido a que los algoritmos evaluados no son deterministas, los valores reportados son las medianas de los resultados de las ejecuciones.

parámetro	*valor*
tamaño de batch	64
número de iteraciones	50
tasa de aprendizaje	0.00005
tamaño de población	2
tamaño de reemplazo	1
tamaño de torneo	2

Tabla 4.2: Tabla de parámetros con valores fijos en la evaluación de desempeño

Los objetivos para el análisis de la evaluación de desempeño son: comparar el desempeño de los algoritmos multiobjetivo desarrollados con la línea de base (el algoritmo de Lipizzaner monobjetivo) y comparar el desempeño de los algoritmos entre sí. Por un lado, en la comparación con la línea de base se evalúa el compromiso entre fidelidad y diversidad respecto a la solución más simple. Por el otro, en la comparación entre algoritmos se estudian las relaciones de dominancia para elegir el mejor algoritmo de acuerdo con los resultados de fidelidad y diversidad.

Para la evaluación estadística de dominancia entre las muestras de los algoritmos en cada una de las métricas se realizaron contrastes de hipótesis de normalidad para evaluar la normalidad de los resultados, para luego concluir si es necesario realizar un análisis paramétrico o no paramétrico. Por último, se realizaron pruebas de dominancia estocástica para comparar las muestras sobre las métricas de interés. Se considera un nivel de confianza del 90 % para todas las pruebas estadísticas, por lo que se consideran significativas para rechazar la hipótesis nula (normalidad de los datos) las pruebas cuyo resultado reporte p-valores menores al nivel de significancia de 0.1.

4.4.4. Metodología para la evaluación de escalabilidad

La escalabilidad de un algoritmo es la capacidad de mejorar el desempeño al utilizar recursos de cómputo adicionales para la ejecución de aplicaciones paralelas. En este proyecto de grado se eligió el tiempo de ejecución como medida de desempeño, por su simplicidad y por ser un buen indicador del esfuerzo computacional requerido.

Una forma de evaluar la escalabilidad de un algoritmo es mediante la paralelicibilidad de la métrica de desempeño. Siendo TP_1 el tiempo de ejecución

del algoritmo en un único recurso de cómputo y TP_N el tiempo de ejecución del algoritmo en N recursos de cómputo, la paralelicibilidad P de un algoritmo se define según la Ecuación 4.1.

$$P = \frac{TP_1}{TP_N} \tag{4.1}$$

Para medir la paralelicibilidad de un algoritmo se varía la cantidad de recursos de cómputo mientras la cantidad de trabajo a realizar se mantiene constante. En el caso de Lipizzaner, hay una relación uno a uno entre las celdas de la grilla (trabajo a realizar) y los núcleos de procesamiento (recursos de cómputo disponibles), por lo que el trabajo a realizar no se mantiene constante a medida que aumentan los recursos de cómputo sino que incrementa proporcionalemente a los recursos de cómputo. Dado este incremento proporcional en Lipizzaner, en este proyecto de grado se utilizó una adaptación de la paralelicibilidad. Se define la versión adaptada de paralelicibilidad para Lipizzaner P_L según la Ecuación 4.2, siendo TP_M el tiempo de ejecución de Lipizzaner con M procesos y M núcleos de procesamiento y TP_N el tiempo de ejecución de Lipizzaner con N procesos y N núcleos de procesamiento.

$$P_L = \frac{TP_M - TP_N}{M - N}, N < M \tag{4.2}$$

Por una parte, si el algoritmo a estudiar es perfectamente paralelizable, el tiempo de ejecución debe mantenerse constante al incrementar ambas variables proporcionalmente, por lo tanto $P_L = 0$. Por otra parte, si el algoritmo a estudiar tiene pasos que no pueden ser paralelizados, la pendiente será positiva $P_L > 0$.

El tiempo de ejecución de Lipizzaner se puede desglosar en tres estados: procesamiento efectivo, comunicación y ocioso. Es decir, se define el tiempo de ejecución según la Ecuación 4.3. Los experimentos de este proyecto de grado fueron ejecutados en ClusterUY, por lo que los recursos de cómputo no estaban aislados, era un entorno compartido. El tiempo de procesamiento efectivo T_{PROC} y el tiempo ocioso T_{IDLE} dependen del poder de cómputo del núcleo

de procesamiento dónde se ejecute el experimento y la carga de procesamiento presente al momento de ejecutar el experimento. El tiempo de comunicación T_{COM} se ve afectado por la carga presente en la red del entorno. Debido a que la medida de desempeño puede ser afectada por variables del entorno de ejecución, se procuró que el hardware de los núcleos de procesamiento utilizados fuera siempre el mismo, de esta forma se minimizó la diferencia entre experimentos.

$$T = T_{PROC} + T_{IDLE} + T_{COM} \tag{4.3}$$

Algunas de las variables del entorno de ejecución (ClusterUY) se vieron afectadas entre ejecuciones, por ejemplo el ancho de banda de red disponible o la utilización de los discos. Además, la cantidad de tarjetas gráficas disponibles es limitada, representando también un cuello de botella para la escalabilidad. La restricción en la cantidad de tarjetas gráficas disponibles generó variaciones en los tiempos de comunicación y ocio. Por lo tanto, se realizaron múltiples ejecuciones y se reportan sus estadísticas relevantes.

Finalmente, se corroboró que el balance de carga haya sido adecuado. Para corroborar el adecuado balance de carga se estudiaron los tiempos de inicio y final de iteración de cada proceso en la grilla. Se buscaron señales de que la carga no haya sido pareja entre los procesos o que algún proceso haya sido beneficiado por el manejador de recursos.

Capítulo 5

Implementación

La implementación de los algoritmos evaluados en este proyecto de grado para mejorar la diversidad de los resultados generados por Lipizzaner consistió en tres etapas: 1) la adaptación del esquema de distribución de Lipizzaner al paradigma de pasaje de mensajes con MPI; 2) la implementación de las funciones de costo para diversidad, E–GAN y GDPP y 3) el desarrollo de los tres algoritmos de optimización multiobjetivo, NSGA–II, FV–MOEA y MOEA/D. En este capítulo se describen las decisiones de implementación tomadas durante cada etapa y los detalles técnicos de los algoritmos desarrollados.

5.1. Adaptación del esquema de distribución

En esta sección se describen los detalles de la adaptación del esquema de distribución de Lipizzaner del modelo cliente-servidor al paradigma de pasaje de mensajes, para el cual se adoptó el estándar MPI. La adaptación fue necesaria para ejecutar los algoritmos desarrollados en ClusterUY, puesto que este utiliza middleware específico para realizar computación distribuida y no admite la utilización de implementaciones de bajo nivel como la de Lipizzaner.

5.1.1. Diseño del algoritmo con MPI

Debido a que no se cuenta con implementaciones de MPI en Python, lenguaje en el que está desarrollado Lipizzaner, fue necesario utilizar la biblioteca mpi4py como middleware entre MPI y el código en Python de Lipizzaner. mpi4py se encarga de inicializar y finalizar el contexto de MPI mediante las funciones `MPI_INIT` y `MPI_FINALIZE`, provee una interfaz orientada a objetos

para definir comunicadores y acceder a las funciones de MPI y maneja la serialización y la deserialización de objetos de Python para posibilitar su envío como mensajes de MPI utilizando la biblioteca Pickle [62].

Respecto a la serialización de las GANs, se mantuvo la codificación para discriminadores y generadores definida por el algoritmo original. Ambas redes se empaquetan en un mapa donde el discriminador está indexado por la letra "D" y el generador está indexado por la letra "G", para ser enviados en un único mensaje a los procesos vecinos. Cada proceso utiliza listas para representar poblaciones separadas de discriminadores y generadores. Se mantienen dos diccionarios de vecindad (uno para discriminadores y otro para generadores) indexados por el rango de los procesos vecinos. Los vecindarios se utilizan durante las rutinas de comunicación para llevar el registro de los rangos de MPI (vecindarios de origen) asociados a cada proceso de Lipizzaner y así tener registro del origen de los individuos. Las poblaciones se usan para correr el ciclo evolutivo.

Las principales decisiones de diseño al reimplementar Lipizzaner con MPI consistieron en: 1) basar las interacciones entre procesos en el intracomunicador de grilla cartesiana; 2) no utilizar un proceso maestro separado de los procesos que se encargan del ciclo evolutivo; 3) contar con un único hilo de ejecución en cada proceso y 4) minimizar los puntos de sincronización entre procesos.

La topología de grilla cartesiana provista por MPI facilita la identificación de procesos vecinos. Además, permite aprovechar la localidad física en los nodos de cómputo para mejorar el desempeño del envío de mensajes, dado que las comunicaciones entre nodos vecinos son las más frecuentes en el algoritmo de Lipizzaner. En la reimplementación de Lipizzaner se lanzan tantos procesos cuantas celdas se requieran en la grilla y se determina al proceso con rango cero como maestro. Aunque el proceso maestro es el responsable de la carga inicial de los datos a la memoria compartida y del ensamblaje del modelo final, a diferencia del algoritmo original de Lipizzaner el proceso maestro participa además del algoritmo evolutivo como otro proceso de la grilla.

En la figura Figura 5.1 se presenta un diagrama con el flujo de ejecución del proceso del algoritmo de Lipizzaner rediseñado. Cuando un proceso comienza su ejecución y mpi4py inicializa implícitamente el contexto de MPI, el proceso verifica si su rango es el 0 (rango que corresponde al de proceso maestro). Si el proceso tiene un rango distinto de 0 (no es el proceso maestro) espera en la primera barrera de sincronización a que los datos de entrenamiento estén

cargados en memoria y puedan ser utilizados. Si el proceso tiene rango 0 es el proceso maestro y se encarga de leer la configuración del algoritmo, cargar los datos de entrenamiento a la memoria y espera en la barrera de sincronización por los demás procesos. Una vez que todos los proceso están en la barrera de sincronización se abre la barrera para continuar.

Una vez pasada la barrera que controla la disponibilidad de los datos, cada proceso inicializa su población, la distribuye mediante la función `MPI_ALLGATHER` por un comunicador que contiene únicamente a sus vecinos y comienza la primera generación del algoritmo evolutivo. A partir de la segunda generación se inicia el flujo de migración usual: el proceso obtiene individuos inmigrantes de sus vecinos y actualiza su población, ejecuta el ciclo evolutivo para seleccionar y mutar su mejor individuo e inicia el proceso de emigración enviando el mejor individuo a sus cuatros vecinos.

Según la especificación original de Lipizzaner, los ciclos evolutivos de las distintas subpoblaciones no se sincronizan en un ningún momento. La especificación implica que la cantidad de inmigrantes al inicio de una generación no es determinista. Por lo tanto, cada proceso actualiza su población con la información disponible y procede a la ejecución del ciclo evolutivo. La inmigración se ejecuta de forma iterativa: se chequea la existencia de individuos en la cola de mensaje con la función no bloqueante `MPI_IPROBE` y si la función retorna un resultado verdadero se ejecuta la función `MPI_RECV` correspondiente para consumir el mensaje. Se repite el proceso hasta que la función `MPI_IPROBE` indique que ya no hay nuevos mensajes sobre individuos inmigrantes disponibles. Como MPI garantiza el orden de llegada de los mensajes, si existieran varios individuos provenientes del mismo vecino, la población local se actualizaría únicamente con el más reciente.

Al finalizar una generación intermedia, cada proceso envía su mejor individuo a sus cuatro vecinos con cuatro envíos no bloqueantes e independientes, utilizando la función `MPI_ISEND`. Al finalizar la última generación el mejor individuo se envía únicamente al proceso maestro mediante la función de comunicación colectiva `MPI_GATHER`. Culminado el último envío, el proceso se bloquea en una barrera final para esperar que se completen posibles envíos en curso (Una vez que uno de los procesos termina el contexto de MPI también termina e invalida todos los mensajes en curso). Una vez que todos los procesos se sincronizan en la barrera final, todos excepto el maestro finalizan su ejecución. Entonces, el proceso maestro realiza el ensamblaje del modelo

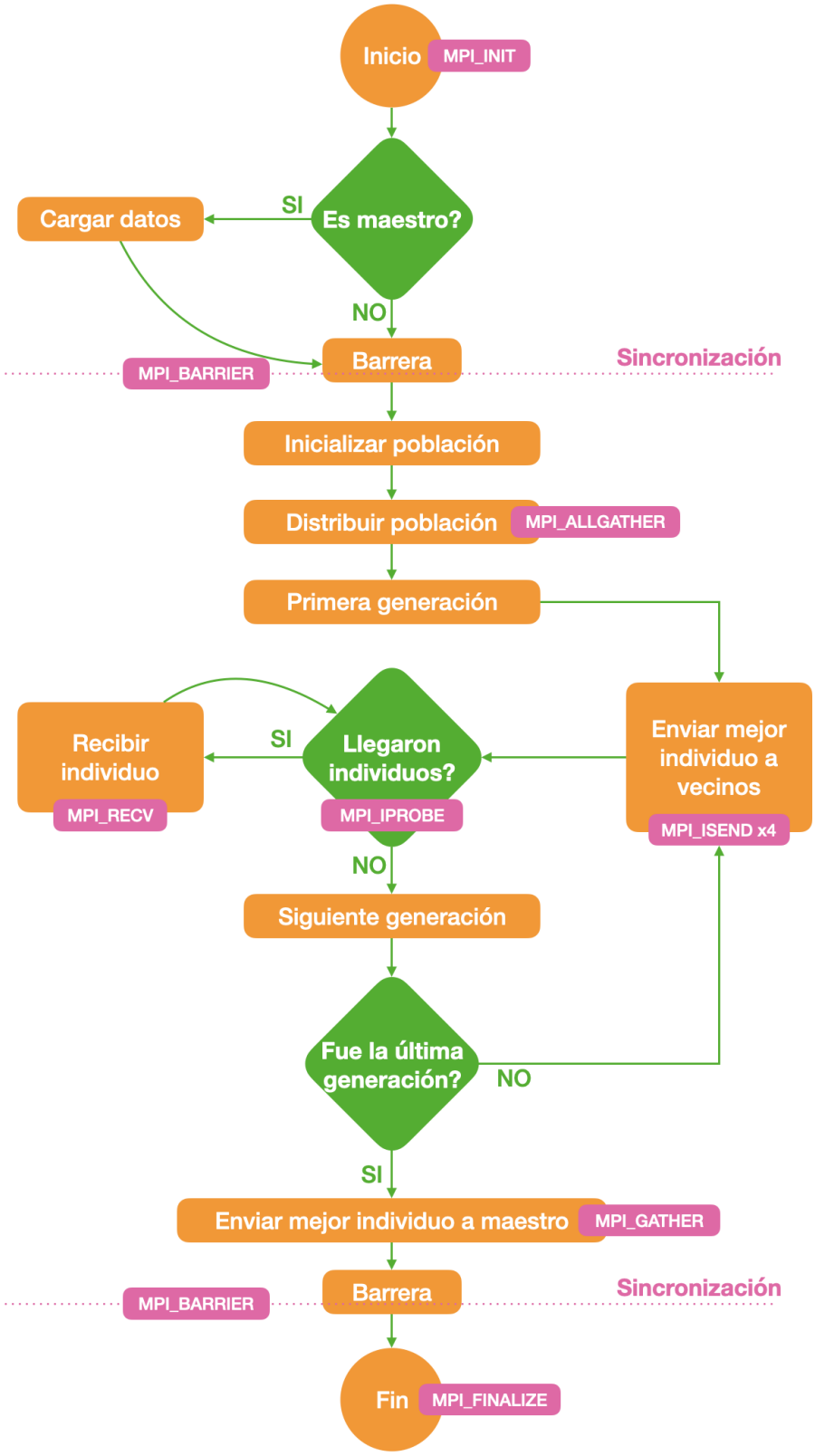


Figura 5.1: Diagrama de flujo de un proceso en el algoritmo de Lipizzaner rediseñado para distribuirse con MPI. Se incluyen etiquetas a los pasos del algoritmo en los cuales se utiliza una función de MPI. Por simplicidad, se omite la etapa de cálculo del ensamblaje que ocurre en el proceso maestro luego de la última barrera.

final utilizando los generadores finales enviados por el resto de los procesos y finaliza su ejecución.

En la Figura 5.2 se presenta un diagrama con las clases relevantes para la implementación de la nueva arquitectura del framework. En el punto de entrada, cada proceso carga el archivo de configuración y espera a que los datos sean cargados por el maestro según el diagrama de la Figura 5.1. Luego, se instancia la clase `LipizzanerWorker`, que permite iniciar el algoritmo de entrenamiento mediante el método `run`. La clase `LipizzanerWorker` se encarga por el ciclo principal de cada proceso de Lipizzaner, manteniendo la iteración actual y coordinando las operaciones de mensajería con el ciclo evolutivo.

La funcionalidad de mensajería se encapsula en la clase `Cell`, que mantiene los individuos locales, provee métodos para codificar y decodificar los mensajes e implementa los operadores de inmigración desde y emigración hacia subpoblaciones vecinas. La inmigración y la emigración se implementaron en los métodos `collect` y `distribute` respectivamente, según lo descripto en el diagrama de flujo de la Figura 5.1.

Las clases `Population` y `Neighbourhood` son colecciones de individuos que representan la población local en distintos contextos del algoritmo. Por un lado, `Population` extiende el concepto de una lista de una de individuos agregándole funcionalidad para determinar si se trata de una población de individuos con aptitud multidimensional (`is_multidimensional`), para obtener el mejor individuo de la población (`best`); y para calcular el frente de pareto (`pareto_front`). La clase `Population` modela la noción más simple y ubícua de población en Lipizzaner, por lo que se utiliza en todas las etapas del algoritmo, aunque obtiene protagonismo durante el ciclo evolutivo. Por otro lado, la `Neighbourhood` es un diccionario que modela la vecindad entre subpoblaciones, manteniendo los rangos de los procesos adyacentes en la grilla (sur, norte, este y oeste) y almacenando los individuos locales según el rango del proceso del cual emigraron. A diferencia de `Population`, la clase `Neighbourhood` representa una noción de población acoplada a la distribución del algoritmo y a las operaciones de mensajería, por lo que solamente se utiliza como auxiliar de la clase `Cell`.

Finalmente, la clase `LipizzanerTrainer` es la responsable por ejecutar la generación evolutiva (`run_generation`) sobre la población actual y retornar la población descendiente. Esta clase tiene acceso a los operadores evolutivos y encapsula su llamada a través de los métodos `_evaluate`, para calcular la

Figura 5.2: Diagrama de clases asociado a un proceso de Lipizzaner en la nueva arquitectura. La clase principal es `LipizzanerWorker`, responsable por correr algoritmo ilustrado en la Figura 5.1. Las clases `cell` es responsable por el manejo de las comunicaciones mediante MPI y `LipizzanerTrainer` se encarga de la ejecución del ciclo evolutivo. `Neighbourhood` y `Population` son clases auxiliares que modelan la noción de vecindad entre procesos y el concepto de población, respectivamente.

aptitud de cada individuo; `_select`, para seleccionar los individuos que serán mutados; `_mutate`, para generar nuevos individuos mediante mutación; y `_survival`, para determinar cuales individuos compondrán la nueva población. Además `LipizzanerTrainer` es implementa el método `optimize_generator_mixture`, responsable por ajustar los parámetros del modelo final, obtenido a partir del ensamblaje de los generadores optimizados en cada subpoblación.

5.1.2. Ejecución del algoritmo

El primer paso para ejecutar Lipizzaner es definir un archivo de configuración en formato YAML, en el cual se especifican parámetros del ciclo evolutivo, de PyTorch, de la arquitectura de la GAN, del ensamblaje del modelo final y del conjunto de datos a ser utilizado. El punto de entrada de la implementación de Lipizzaner con MPI es el archivo `main_mpi.py`, que recibe como primer parámetro el archivo de configuración del algoritmo.

La Plataforma Nacional de Supercomputación cuenta con una instalación de Simple Linux Utility for Resource Management (SLURM) [86] para el manejo de los recursos de cómputo [56]. Este manejador permite correr programas distribuidos mediante la definición y ejecución de un script con extensión `.sbatch`, en el cual se incluyen las instrucciones necesarias para preparar el entorno y realizar la ejecución. En el preámbulo del script se explicitan las opciones de ejecución, que corresponden al detalle de los recursos solicitados, a la identificación del trabajo y a la configuración de notificaciones sobre la ejecución. A seguir se definen algunas de las opciones de ejecución utilizadas para solicitar recursos a la plataforma.

nodes Cantidad de nodos a utilizar.
ntasks Cantidad de procesos a lanzar.
ntasks per node Cantidad de procesos por nodo.
mem per cpu Cantidad de memoria por CPU.
tmp Cantidad de espacio en el almacenamiento de alta velocidad a utilizar.
qos Calidad del servicio (permite pedir unidades de procesamiento gráfico).

Además, la plataforma tiene instalada una implementación de MPI, la cual está integrada con SLURM. Por lo tanto, MPI sabrá la cantidad de procesos que debe iniciar y qué nodos de cómputo puede utilizar observando directamente la configuración pasada al manejador de recursos en el script de lanzamiento.

Estando en la raíz de Lipizzaner, basta con instalar e importar las dependencias y ejecutar el comando `mpirun python main_mpi.py configuracion.yaml`.

Si se quisiera ejecutar el algoritmo en un entorno sin SLURM, debería chequearse la existencia de otro manejador de recursos y su nivel de integración con MPI. Si un manejador no estuviera disponible, entonces debe configurarse un clúster cuya información será recibida por MPI como parámetro de `mpirun`.

5.2. Implementación multiobjetivo

En esta sección se describen los detalles de implementación que fueron necesarios para extender Lipizzaner al problema de optimización multiobjetivo. Específicamente, se discuten las decisiones tomadas al implementar las nuevas funciones de costo (E–GAN y GDPP); los nuevos operadores evolutivos que componen a los algoritmos multiobjetivos elegidos (NSGA–II, FV–MOEA y MOEA/D) y al revisar la estrategia evolutiva responsable por generar el ensamblaje del modelo final.

5.2.1. Descripción general

La extensión de Lipizzaner para soportar optimización multiobjetivo se basó en tres elementos: la implementación de funciones de aptitud multidimensionales; la adaptación de los operadores evolutivos para considerar aptitud multidimensional y la implementación de nuevos operadores de supervivencia. Se optó por utilizar el patrón de diseño de estrategia [25] para encapsular distintas variantes de algoritmos y permitir que facilitar su configuración en tiempo de ejecución.

Se diseñaron tres estrategias: de aptitud, de selección y de supervivencia. Cada estrategia tiene una interfaz, implementada mediante una clase abstracta, y un conjunto de implementaciones que cubren la funcionalidad original, orientada a la optimización unidimensional y la funcionalidad extendida a la optimización multiobjetivo. Si bien este proyecto de grado se instancia en la generación de imágenes y en la optimización conjunta de la fidelidad y la diversidad de las imágenes generadas, la solución implementada es agnóstica a estos conceptos y permite añadir estrategias de aptitud, selección y supervivencia adaptadas al problema que se requiera resolver.

Los operadores de selección y de supervivencia son semejantes, puesto que ambos operan sobre poblaciones para retornar un subconjunto de individuos que serán utilizados en el paso siguiente del ciclo evolutivo. La principal diferencia entre ambos operadores es el momento en el cual se invocan en el algoritmo de Lipizzaner: la selección se utiliza luego de la evaluación para determinar cuales individuos serán sometidos al proceso de mutación y la supervivencia se realiza luego de la mutación para obtener la población final que será utilizada en la siguiente generación del ciclo evolutivo. NSGA–II, FV–MOEA y MOEA/D se implementan como operadores de supervivencia y en el caso multiobjetivo basado en Pareto el operador es el responsable por calcular el frente de soluciones no dominadas.

5.2.2. Estrategia de aptitud

La primera etapa al implementar la estrategia de aptitud multiobjetivo consistió en adaptar el algoritmo para utilizar valores de aptitud vectoriales en vez de escalares. En este contexto, la aptitud unidimensional se modela como un vector con una única componente. Luego, se implementó el patrón de estrategia según el diagrama de clases presentado en la Figura 5.3.

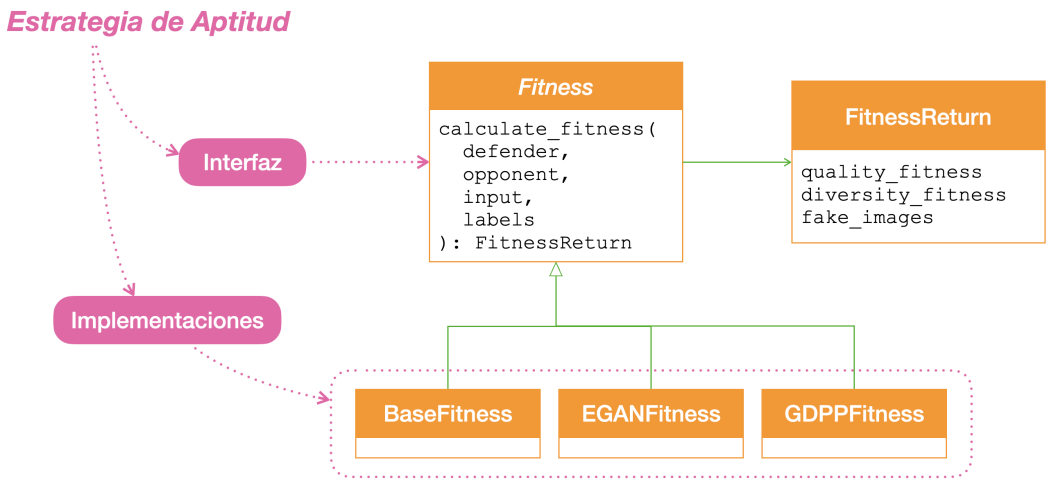


Figura 5.3: Diagrama de clases asociado a la estrategia de aptitud implementada. La clase abstracta `Fitness` corresponde a la interfaz de la estrategia. Por otro lado, `BaseFitness`, `EGANFitness` y `GDPPFitness` corresponden a las implementaciones de la estrategia. La clase `FitnessReturn` representa el valor de retorno de la función `calculate_fitness`.

La clase abstracta `Fitness` implementa la interfaz de la estrategia de aptitud, que consiste en una función `calculate_fitness` cuya entrada es un individuo (`defender`), su contraparte (`opponent`) y los datos (`input` y `labels`). Si el individuo es un generador su contraparte será un discriminador y si el individuo es un discriminador su contraparte será un generador. La salida de la función `calculate_fitness` es una terna que incluye la aptitud en fidelidad, la aptitud en diversidad y las imágenes generadas para el cálculo, en ese orden. Las dos primeras componentes de la terna de salida conforman el vector resultado de la función de aptitud.

Se desarrollaron tres implementaciones para esta estrategia: `BaseFitness`, que corresponde a la función de aptitud original de Lipizzaner (Ecuación 3.10); y `EGANFitness` y `GDPPFitness`, que implementan funciones de aptitud multiobjetivo basadas en las funciones de costo de diversidad E-GAN y GDPP, respectivamente. En ambos casos multiobjetivo, la componente de fidelidad corresponde a la función de aptitud original implementada en `BaseFitness`. Además, es posible extender la estrategia desarrollando nuevas subclases de la clase `Fitness` o de alguna de las implementaciones realizadas.

La estrategia de aptitud multiobjetivo, implementada en este proyecto de grado por `EGANFitness` o `GDPPFitness` se aplicó solamente a los generadores. A los discriminadores, por su parte, se los configuró con la estrategia de aptitud unidimensional original, correspondiente a `BaseFitness`. Además, es necesario destacar que esta estrategia es relevante para los operadores de selección y supervivencia. En el caso de la mutación, definida en Lipizzaner con base en el algoritmo usual de backpropagation, se mantienen la función de costo unidimensional original y no se utiliza la nueva estrategia de aptitud.

5.2.3. Estrategia de selección

La estrategia de selección se implementó según el diagrama de clases presentado en la Figura 5.4. La clase abstracta `BaseSelection` implementa la interfaz de la estrategia, que incluye la definición del método de invocación (`__call__`). La entrada del método de invocación es una población de individuos y su salida es otra población con los individuos seleccionados. Se desarrollaron tres implementaciones para esta estrategia: `TournamentSelection`, que implementa la selección por torneo de tamaño variable; `BinaryTournamentSelection`

que implementa la selección por torneo binario y `FidelitySelection`, que implementa la selección binaria por fidelidad.

`TournamentSelection` es una adaptación de la selección por torneo preexistente al modelo con aptitud multiobjetivo. `BinaryTournamentSelection` y `FidelitySelection` también son adaptaciones de los operadores de selección preexistente, pero incorporan además la implementación del operador de la relación de dominancia mediante los métodos `get_dominant` y `get_relation`. El método `get_dominant` recibe un par de individuos y retorna el dominante según el criterio de dominancia de Pareto (3.2), mientras que el método `get_relation` retorna la relación entre ellos mediante un valor ternario: -1 para indicar que el primer individuo es dominado; 1 para indicar que es dominante y 0 para indicar que ninguno domina al otro.

Además, se implementó la clase auxiliar `Selector`, cuyo objetivo es leer la configuración global del programa e instanciar las estrategia de selección adecuadas para generadores y discriminadores. Esta clase incorpora los métodos `select_generators` y `select_discriminators` para efectuar la selección de generadores y discriminadores, respectivamente. Esta estrategia se utiliza luego de evaluar a los individuos de la población para determinar cuales serán sometidos al proceso de mutación.

5.2.4. Estrategia de supervivencia

La estrategia de supervivencia se desarrolló según el diagrama de clases expuesto en la Figura 5.5. La clase abstracta `BaseSurvival` define la interfaz de la estrategia mediante el método de invocación (`__call__`). El método de invocación recibe una lista de individuos sobre los cuales se aplicará el operador de supervivencia y retorna una población con los individuos supervivientes.

Se desarrollaron cinco diversas implementaciones para la estrategia de supervivencia: `SimpleSurvival`, que elige los mejores individuos ordenándolos por aptitud componente a componente; `FVMOEASurvival`, que implementa un algoritmo de selección de individuos basado en FV–MOEA; `NSGAIISurvival`, que implementa un algoritmo de selección de individuos basado en NSGA–II; `WeightedSumSurvival`, que implementa un algoritmo de selección de individuos basado en MOEA/D y `ParetoFidelitySurvival`, que selecciona los mejores individuos en fidelidad entre los pertenecientes a un frente de Pareto construido obtenido con base a la aptitud multiobjetivo.

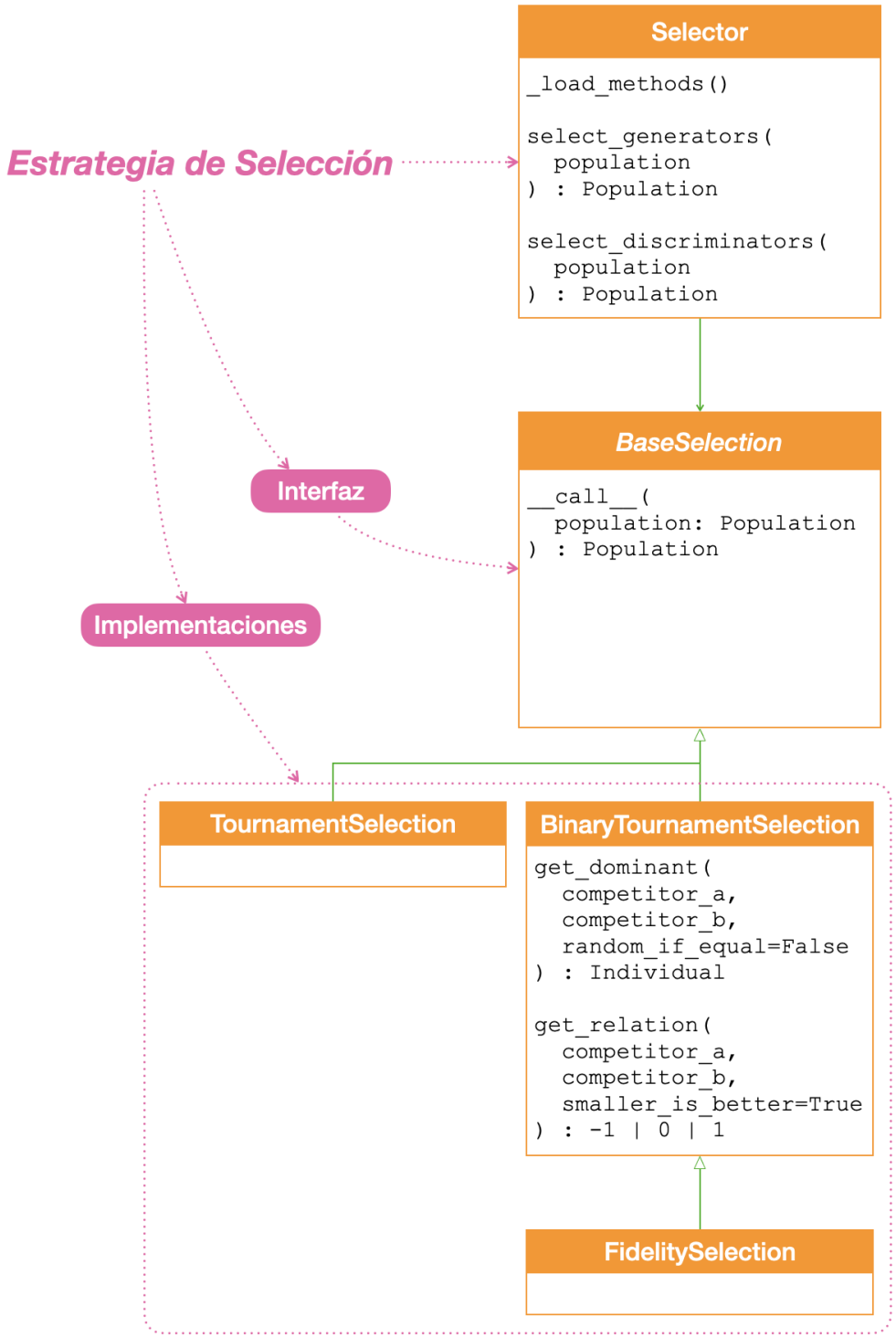


Figura 5.4: Diagrama de clases asociado a la estrategia de selección implementada. La clase abstracta `BaseSelection` corresponde a la interfaz de la estrategia. Por otro lado, `TournamentSelection`, `BinaryTournamentSelection` y `FidelitySelection` corresponden a las implementaciones de la estrategia. Se adaptaron los métodos de selección preexistentes para soportar funciones de aptitud multiobjetivo. La clase `Selector` funciona como fachada para facilitar la instanciación de la estrategia a partir del archivo de configuración provisto.

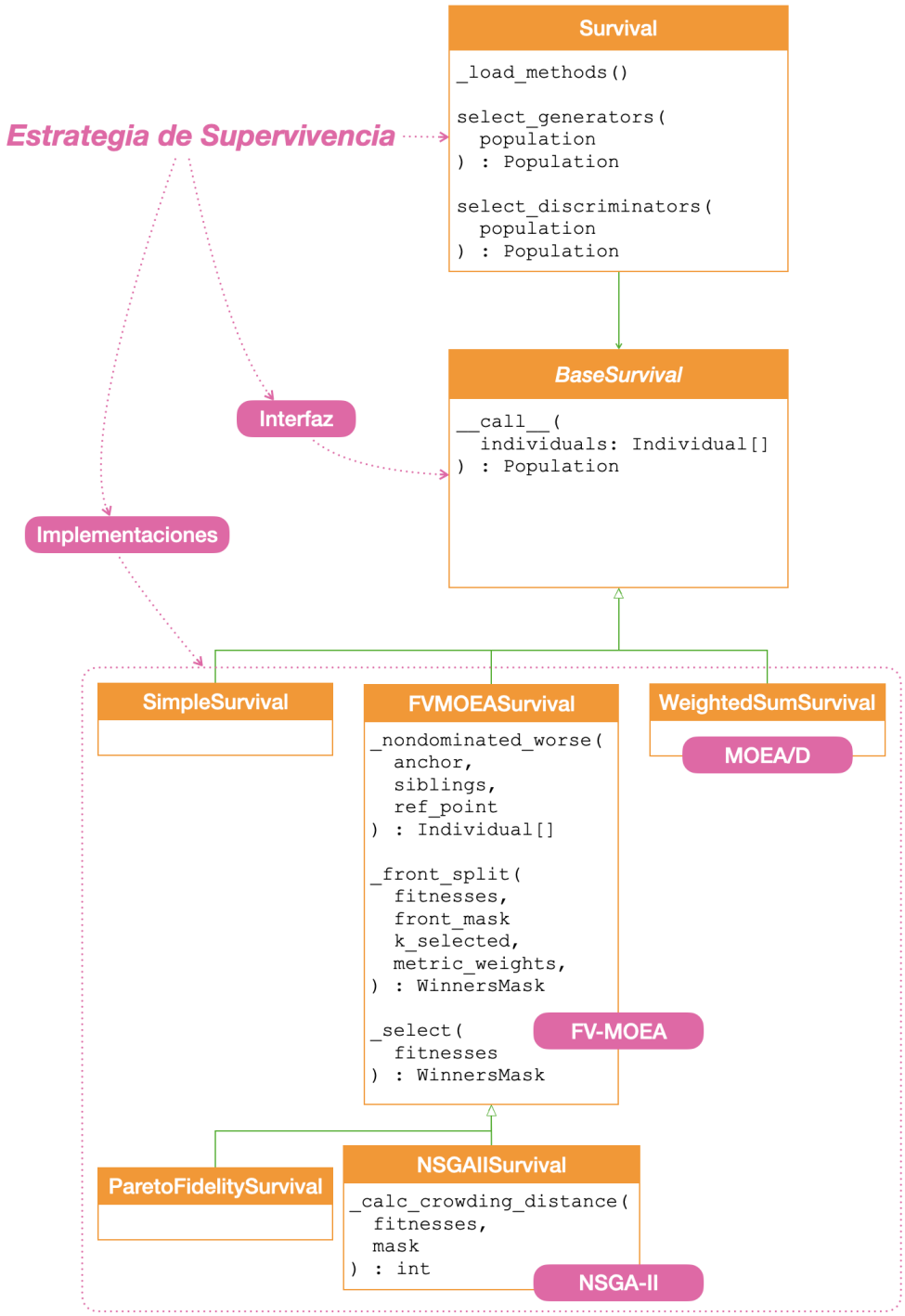


Figura 5.5: Diagrama de clases asociado a la estrategia de supervivencia implementada. La clase abstracta `BaseSurvival` corresponde a la interfaz de la estrategia. Por otro lado, `SimpleSurvival`, `ParetoFidelitySurvival`, `FVMOEASurvival`, `NSGAIISurvival` y `WeightedSum` corresponden a las implementaciones de la estrategia. Las últimas tres implementaciones corresponden, respectivamente, a FV–MOEA, NSGA–II y MOEA/D. Se adaptaron los métodos de supervivencia preexistentes para soportar funciones de aptitud multiobjetivo. La clase `Survival` funciona como fachada para facilitar la instanciación de la estrategia a partir del archivo de configuración provisto.

Además, se implementó la clase auxiliar `Survival`, cuyo objetivo es leer la configuración global del programa e instanciar las estrategia de supervivencia adecuadas para generadores y discriminadores. Esta clase incorpora los métodos `select_generators` y `select_discriminators` para efectuar la selección de generadores y discriminadores, respectivamente. Esta estrategia se utiliza luego de mutar a los individuos que corresponda, para determinar cual será la población final que ocupará el centro del vecindario en la siguiente generación del ciclo evolutivo. En el caso multiobjetivo basado en Pareto, la población final es el frente de pareto obtenido.

5.2.5. Ensamblaje del modelo final

Respecto al ensamblaje del modelo final, la única decisión de diseño tomada es relevante para los algoritmos multiobjetivo basados en Parento. Mientras que en el caso de optimización unidimensional y en MOEA/D cada proceso de Lipizzaner obtiene un único generador, el cual envía al proceso maestro para componer el modelo ensamblado, FV–MOEA y NSGA–II generan frentes de soluciones no dominadas. En este caso, se optó por utilizar los frentes de Pareto completos, por lo que el modelo final se compone potencialmente por más de un generador por proceso. Una vez obtenidas todas las redes relevantes, el proceso maestro realiza la optimización de los parámetros del ensamblaje siguiendo el algoritmo original definido en Lipizzaner.

Capítulo 6

Evaluación experimental

En este capítulo se presentan los resultados de la evaluación experimental de los algoritmos implementados (NSGA–II, FV–MOEA y MOEA/D), realizada aplicando la metodología de trabajo descrita en el Capítulo 4.

6.1. Descripción general

La primera etapa del proceso de evaluación consistió en la configuración paramétrica de los tres algoritmos, con el objetivo de estudiar la incidencia de la función de costo (GDPP o E–GAN) y el tamaño de la grilla (2×2, 3×3 o 4×4 celdas) en la ejecución de cada algoritmo. La configuración paramétrica se realizó entrenando la población de GANs sobre el conjunto de datos MNIST y comparando el desempeño obtenido con cada combinación de parámetros. Debido a que la evaluación de modelos generativos es un problema difícil para el que no existe un consenso establecido [8] y con la intención de fundamentar las decisiones tomadas durante todo el proceso, también se aprovechó esta etapa para estudiar las relaciones entre las métricas de evaluación. Una vez ajustados los parámetros de los algoritmos implementados, se ejecutaron sobre otro conjunto de datos, CelebA y se compararon entre sí y con el algoritmo de referencia, el Lipizzaner mono-objetivo original.

La organización de este capítulo refleja el proceso realizado. Preliminarmente, se discute sobre las relaciones entre métricas de evaluación con base a sus dependencias lineales. Las dependencias lineales se hallan mediante los coeficientes de correlación de Pearson entre métricas y contribuyen a la fundamentación de aquellas decisiones tomadas en el proceso de evaluación que

involucran la adopción de alguna métrica específica como criterio de selección o desempate. Luego se presentan los resultados de la configuración paramétrica, discutiendo los resultados separadamente para cada algoritmo. En el análisis se incluye el estudio de tiempos de ejecución y escalabilidad de los algoritmos implementados. Finalmente, se presentan los resultados sobre el dataset CelebA y se realiza la comparación de desempeño.

6.2. Análisis de las métricas de evaluación

Esta sección presenta el análisis de los resultados obtenidos en la configuración paramétrica. En esta etapa se estudian las relaciones entre las métricas de evaluación, dado que las técnicas de evaluación de modelos generativos y, consecuentemente, las métricas de evaluación utilizadas en este proyecto de grado, no están consolidadas. Las métricas consideradas son las presentadas en el Capítulo 3: FID; TVD; precisión y exhaustividad; y densidad y cubrimiento. La métrica más popular en la evaluación de modelos generativos de imágenes es FID, aunque en este proyecto de grado existe una inclinación teórica por el uso de densidad y cubrimiento para poder evaluar los modelos en las dimensiones de fidelidad y diversidad de los resultados.

En este proyecto de grado se plantean algoritmos para mejorar la diversidad de los datos que generan los modelos entrenados por Lipizzaner. Por este motivo, se adoptan medidas de fidelidad independientes, que permitan estudiar ambas dimensiones por separado. Al ser un problema donde las métricas conocidas están poco consolidadas se considera al coeficiente de correlación de Pearson como una buena primera aproximación para estudiar la hipótesis de independencia.

En la Tabla 6.1 se reporta el coeficiente de correlación de Pearson para cada par de métricas, considerando los resultados obtenidos durante la etapa de configuración paramétrica. Este coeficiente indica la existencia de relaciones lineales entre variables. El módulo del coeficiente (en el intervalo $[0, 1]$) indica el grado de relación y el signo indica la dirección de la relación.

En la Figura 6.1 se presenta el mapa de correlaciones entre las métricas estudiadas. El gráfico es una representación de la matriz de correlación entre las variables reportadas, en el cual los colores expresan el signo del coeficiente de correlación lineal (verde para positivo, naranja para negativo) y la intensidad de los colores representa el módulo de la correlación (un color más claro

	FID	TVD	Cubrimiento	Densidad	Precisión	Exhaustividad
FID	1.00	0.97	-0.94	0.82	0.74	-0.84
TVD	0.97	1.00	-0.91	0.83	0.77	-0.77
Cubrimiento	-0.94	-0.91	1.00	-0.74	-0.69	0.86
Densidad	0.82	0.83	-0.74	1.00	0.76	-0.78
Presición	0.74	0.77	-0.69	0.76	1.00	-0.62
Exhaustividad	-0.84	-0.77	0.86	-0.78	-0.62	1.00

Tabla 6.1: Matriz de correlaciones entre las métricas de evaluación. Se estudian dos familias de métricas, las enfocadas en la fidelidad: FID, densidad y Precisión; y las enfocadas en diversidad: TVD, cubrimiento y Exhaustividad. Para calcular estos valores, se utilizaron los datos de todas las ejecuciones realizadas en el proceso de configuración parametrica.

indica menor correlación entre las variables). Para el cálculo de la matriz de correlación se consideraron todos los datos disponibles, sin discriminarlos por algoritmo o configuración.

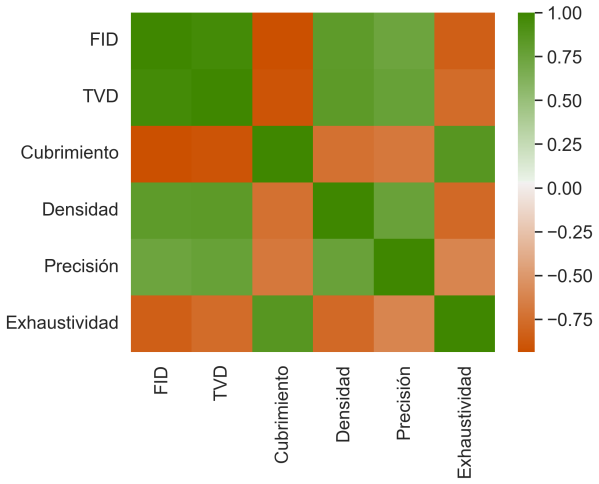
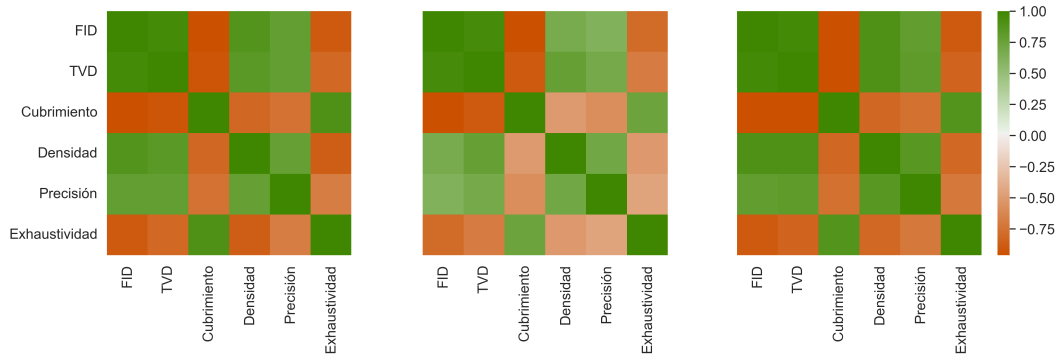


Figura 6.1: Mapa de correlaciones entre métricas de evaluación para los resultados de la configuración paramétrica. El signo de los coeficientes de las correlaciones se expresa según los colores, denotando el verde una correlación positiva y el naranja una correlación negativa. El módulo de la correlación está expresada por la intensidad de los colores, dónde más claro (o menos intenso) significa menor correlación. Los datos numéricos están expuestos en la Tabla 6.1.

Los resultados reportados en la Tabla 6.1 y en el mapa de correlaciones de la Figura 6.1 demuestran que las métricas presentan alta correlación en general. Específicamente, FID, TVD, densidad y precisión presentan correlaciones altas positivas entre sí (entre 0.76 y 0.97) y negativas con la exhaustividad y el cubrimiento (entre -0.62 y -0.97). La densidad y la precisión son las variables

que presentan menor correlación con el resto, destacando las bajas correlaciones de la precisión con la densidad y con la exhaustividad. Asimismo, la densidad presenta una correlación moderada con el cubrimiento.

En la Figura 6.2 se presentan los mapas de correlaciones calculados con los datos discriminados por algoritmo. Los valores numéricos se encuentran en las Tablas 2.1, 2.2 y 2.3 del Apéndice 2. Al visualizar las correlaciones de esta manera y compararlas con las presentadas en la Tabla 6.1, se mantienen los signos de las correlaciones entre pares de variables, pero varían sus módulos. Los resultados de NSGA–II (Figura 6.2a) y MOEA/D (Figura 6.2c) soportan conclusiones semejantes a las mencionadas en relación a la Figura 6.1 sobre los coeficientes obtenidos sin discriminar por algoritmo. En el caso de FV–MOEA (Figura 6.2b) se atenuan las correlaciones entre densidad y precisión con el resto de las variables. En especial, se destaca la baja correlación entre densidad y cubrimiento para el algoritmo FV–MOEA.



(a) NSGA–II **(b)** FV–MOEA **(c)** MOEA/D

Figura 6.2: Mapas de correlaciones entre métricas de evaluación para los resultados de la configuración paramétrica para cada algoritmo. El signo de los coeficientes de las correlaciones se expresa según los colores, denotando el verde una correlación positiva y el naranja una correlación negativa. El módulo de la correlación está expresada por la intensidad de los colores, dónde más claro (o menos intenso) significa menor correlación. Los datos numéricos están expuestos en las Tablas 2.1, 2.2 y 2.3 del Apéndice 2.

El análisis de la correlación entre las métricas de evaluación provee información para clasificar las métricas en dos familias, una familia se enfoca en la medición de la fidelidad y la otra en la medición de la diversidad de los datos generados. La fidelidad y la diversidad se consideran dos dimensiones independientes en la evaluación de datos generados. A priori, FID, densidad y precisión pertenecen a la familia de métricas de fidelidad y TVD, cubrimiento

y exhaustividad pertenecen a la familia de métricas enfocadas en la diversidad. Ambas familias de métricas intentan medir conceptos inherentemente subjetivos y es posible que algunas de las métricas evaluadas no describan únicamente su dimensión asociada. Por ejemplo, FID mide la fidelidad de los resultados, pero para dos conjuntos similarmente fieles favorecerá a aquel con mayor diversidad.

En líneas generales, los resultados muestran la correlación positiva entre variables pertenecientes a la misma dimensión y negativa entre variables pertenecientes a dimensiones distintas. Se favorece, entonces, la idea de que en la práctica es difícil optimizar alguna de las dimensiones sin incurrir en penalizaciones en la otra, aunque en la teoría sea posible. La excepción a esta observación es TVD, que presenta correlación positiva con las métricas de fidelidad y negativa con las métricas de diversidad. La correlación se debe a que cubrimiento y exhaustividad se maximizan, mientras que TVD se minimiza.

Además, la precisión y la exhaustividad presentaron menor correlación entre sí que la densidad y el cubrimiento. No obstante, la correlación entre densidad y cubrimiento resultó baja en relación a los otros pares de métricas. Dado el análisis de las correlaciones observadas entre las métricas y teniendo en cuenta que las métricas densidad y cubrimiento son una reformulación de las métricas precisión y exhaustividad (Subsección 3.7.1), se concluye que densidad y cubrimiento son las métricas más apropiadas para el análisis de resultados. Además, se decidió mantener a FID como principal métrica candidata a criterio de desempate debido a su amplia adopción en la literatura.

6.3. Configuración paramétrica

En esta sección se exponen los resultados del proceo de configuración paramétrica. Primeramente, se realiza una descripción general del análisis de configuración realizado. Luego, se presentan y discuten los resultados para NSGA–II, FV–MOEA y MOEA/D.

6.3.1. Conceptos generales del análisis de parámetros

El objetivo de la configuración paramétricas es estudiar la incidencia de la función de aptitud y el tamaño de la grilla en el desempeño de los algoritmos para elegir los valores que serán utilizados en el resto de la evaluación

experimental. Se consideran dos funciones de aptitud, GDPP y E–GAN, y tres tamaños de grilla, 2×2, 3×3 y 4×4. Siendo f una función de aptitud y T un tamaño de grilla, se denomina "configuración" a una combinación (f, T) específica, por lo que se cuenta con seis configuraciones para cada algoritmo. Se analizaron los resultados de diez ejecuciones por configuración y algoritmo en función de las métricas seleccionadas en la Sección 6.2.

6.3.2. Análisis de parámetros de NSGA–II

Esta sección describe el análisis de configuración paramétrica para NSGA–II. En la Figura 6.3 se presenta un gráfico de dispersión con las seis configuraciones en el plano densidad×cubrimiento. Los valores reportados corresponden a las medianas de los resultados obtenidos en las diez ejecuciones de cada configuración. Se codificaron las funciones de aptitud con colores (verde para E–GAN y naranja para GDPP) y los tamaños de la grilla con formas (círculo para 2×2, cuadrado para 3×3 y cruz para 4×4).

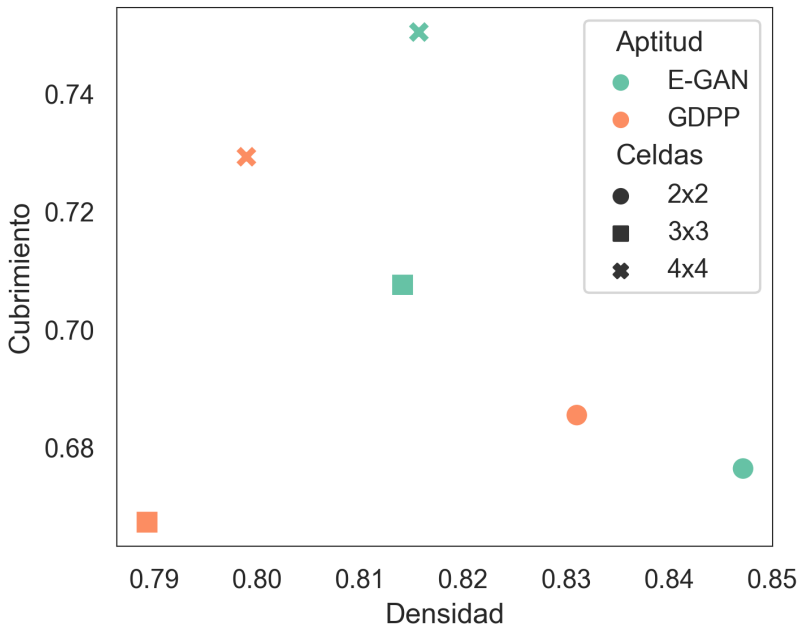


Figura 6.3: Representación de los resultados de la configuración paramétrica para NSGA–II en el plano densidad×cubrimiento. Se presentan las medianas de los resultados obtenidos por configuración, cuyos valores numéricos están disponibles en la Tabla 6.2. En verde y en naranja se indican los puntos correspondientes a configuraciones con aptitud E–GAN y GDPP, respectivamente. Las formas de los puntos expresan los tamaños de grilla. Círculo, cuadrado y cruz expresan los tamaños 2×2, 3×3 y 4×4, respectivamente.

Al considerar todas las configuraciones, se identifican como configuraciones no dominadas (E–GAN, 4×4), (GDPP, 2×2) y (E–GAN, 2×2). Sin embargo, al excluir las configuraciones con tamaño de grilla 4×4, el frente no dominado fue compuesto por (E–GAN, 3×3), (GDPP, 2×2) y (E–GAN, 2×2). En el resto del análisis no se consideran las configuraciones con grillas de dimensión 4, por los motivos explicados en el Capítulo 4.

En la Tabla 6.2 se presentan los resultados de densidad, cubrimiento y FID para los resultados de NSGA–II en cada configuración. Los valores reportados son la mediana y el IQR de cada métrica. Además, se indica la dirección en la cual debe observarse cada métrica al comparar (en densidad y cubrimiento mayores valores indican mejores resultados, mientras que en FID menores valores indican mejores resultados y se destacan en negritas las mejores configuraciones en cada métrica).

Configuración		Densidad ↑		Cubrimiento ↑		FID ↓	
Celdas	Aptitud	m	IQR	m	IQR	m	IQR
2×2	E–GAN	**0.866**	0.069	0.655	0.114	175.600	124.018
	GDPP	0.831	0.062	0.685	0.029	143.993	43.412
3×3	E–GAN	0.814	0.046	**0.708**	0.036	**116.025**	53.390
	GDPP	0.789	0.022	0.667	0.054	141.942	53.329
4×4	E–GAN	0.816	0.034	0.751	0.043	87.231	23.390
	GDPP	0.799	0.031	0.729	0.090	89.422	111.209

Tabla 6.2: Resultados de las configuraciones en NSGA–II. Se reporta la mediana m y el rango intercuartílico IQR de densidad, cubrimiento y FID para cada configuración, en muestras correspondientes a 10 ejecuciones. Las configuraciones son pares (función de aptitud, tamaño de la grilla). Se expresa el tamaño de la grilla como la cantidad de celdas que la componen. Las funciones de aptitud consideradas son E–GAN y GDPP. Los tamaños de grilla son 2×2, 3×3 y 4×4 celdas. Las configuraciones con tamaños de grilla 4×4 se reportan separadas por no ser consideradas en la evaluación de desempeño.

Respecto a la densidad, la configuración que obtuvo la mayor mediana fue (E–GAN, 2×2). La configuración que obtuvo mejor cubrimiento mediano fue (E–GAN, 3×3), con un IQR tres veces menor que (E–GAN, 2×2). Se destaca que (E–GAN, 2×2) obtuvo una densidad media 6.4 % mayor que (E–GAN, 3×3), mientras que (E–GAN, 3×3) obtuvo un cubrimiento 8.1 % mayor que (E–GAN, 2×2), por lo que se identificó empate y se recurrió al FID para resolverlo. En este aspecto, la mejor configuración entre todas las consideradas

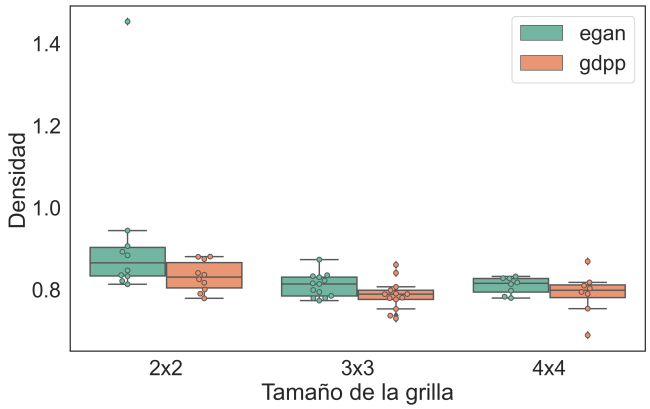
fue (E–GAN, 3×3), con FID mediano 33.9 % menor e IQR 56.9 % menor que los presentados por (E–GAN, 2×2).

En la Figura 6.4 se presentan los gráficos de caja asociados a los resultados alcanzados por cada configuración para NSGA–II. Los gráficos proveen una visualización alternativa sobre los datos presentados en la Tabla 6.2 y facilitan el análisis de la incidencia del tamaño de la grilla en los resultados.

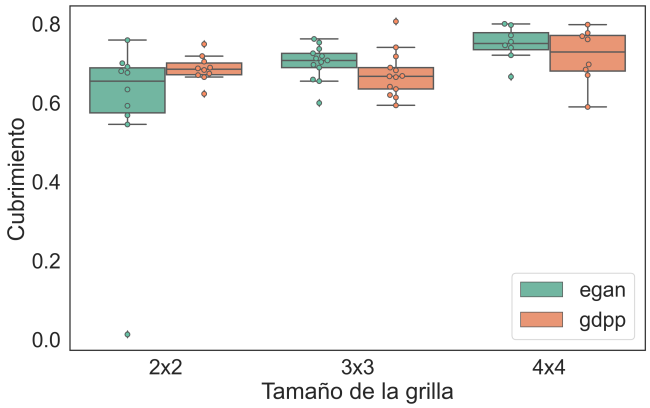
Los gráficos de la Figura 6.4 indican que al utilizar E–GAN como función de aptitud, aumentar el tamaño de la grilla mejoró el desempeño en las tres métricas de evaluación. Al considerar GDPP como función de aptitud, se observaron buenos desempeños en fidelidad con grillas de tamaño 3×3. En el caso de FID se observó una variabilidad considerablemente mayor para grillas de tamaño 4×4, a pesar de que la mediana indicó mejores resultados.

El comportamiento del cubrimiento fue similar al de la densidad cuando se considera E–GAN como la función de aptitud: mejores valores de cubrimiento estuvieron correlacionados con un aumento en el tamaño de la grilla. Sin embargo, al observar los resultados con GDPP como la función de aptitud los resultados de cubrimiento no estuvieron correlacionados al tamaño de la grilla dado que las grillas de 2×2 y 4×4 presentaron mejor cubrimiento que la de 3×3.

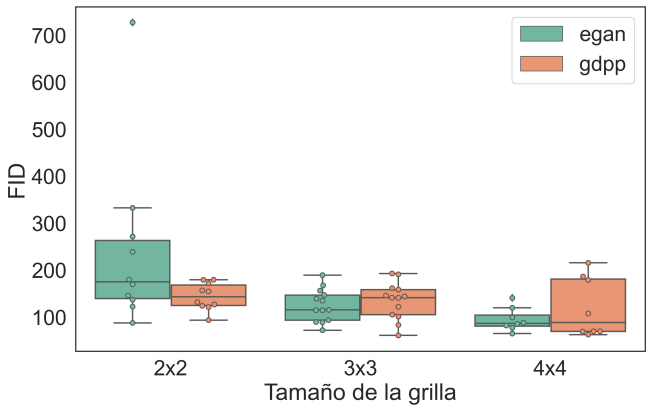
En general, se observaron mejoras de desempeño al aumentar el tamaño de la grilla, aunque los resultados indicaron que, considerando todas las métricas de evaluación, la configuración (E–GAN, 3×3) era la más adecuada de las estudiadas para NSGA–II.



(a) Densidad



(b) Cubrimiento



(c) FID

Figura 6.4: Gráficos de cajas de los resultados obtenidos para NSGA–II. Se reportan cubrimiento, densidad y FID discriminados por los parámetros de la configuración: la función de aptitud (E–GAN en verde y GDPP en naranja) y por el tamaño de la grilla (2×2, 3×3 o 4×4 celdas, en el eje horizontal).

6.3.3. Análisis de parámetros de FV–MOEA

Esta sección describe el análisis de configuración paramétrica para FV–MOEA. En la Figura 6.5 se presenta un gráfico de dispersión con las seis configuraciones en el plano densidad×cubrimiento. Los valores reportados corresponden a las medianas de los resultados obtenidos en las diez ejecuciones de cada configuración. Se codificaron las funciones de aptitud con colores (verde para E–GAN y naranja para GDPP) y los tamaños de la grilla con formas (círculo para 2×2, cuadrado para 3×3 y cruz para 4×4).

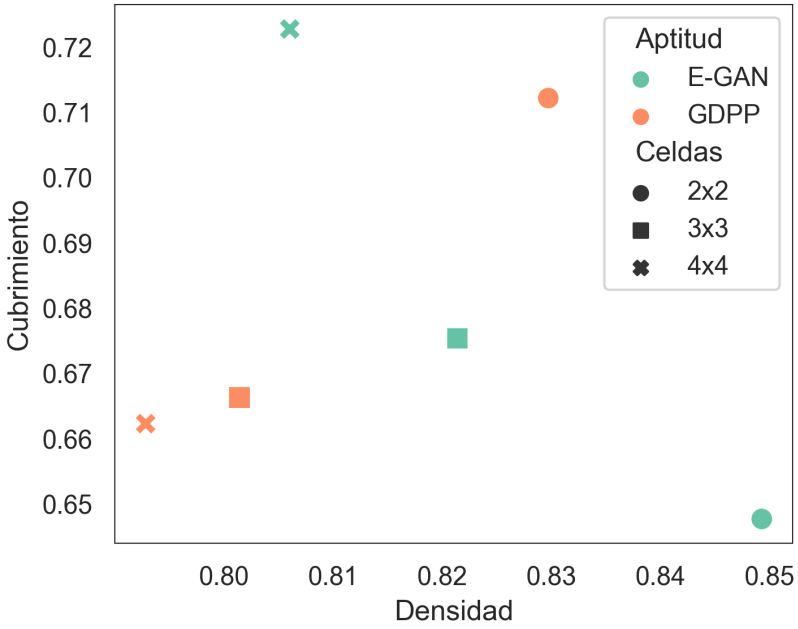


Figura 6.5: Representación de los resultados de la configuración paramétrica para FV–MOEA en el plano densidad×cubrimiento. Se presentan las medianas de los resultados obtenidos por configuración, cuyos valores numéricos están disponibles en la Tabla 6.3. En verde y en naranja se indican los puntos correspondientes a configuraciones con aptitud E–GAN y GDPP, respectivamente. Las formas de los puntos expresan los tamaños de grilla. Círculo, cuadrado y cruz expresan los tamaños 2×2, 3×3 y 4×4, respectivamente.

Al considerar todas las configuraciones, se identifican como configuraciones no dominadas (E–GAN, 4×4), (GDPP, 2×2) y (E–GAN, 2×2), al igual que lo obtenido para NSGA–II. Sin embargo, al excluir las configuraciones con tamaño de grilla 4×4, el frente no dominado fue compuesto por (GDPP, 2×2) y (E–GAN, 2×2). En el resto del análisis no se consideran las configuraciones con grillas de dimensión 4, por los motivos explicados en el Capítulo 4.

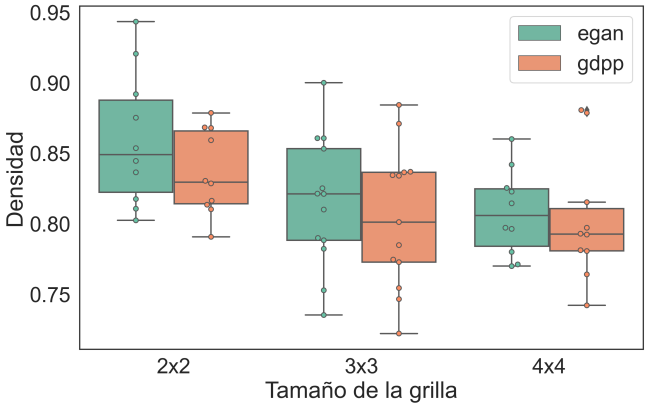
En la Tabla 6.3 se presentan los resultados de densidad, cubrimiento y FID para los resultados de FV–MOEA en cada configuración. Los valores reportados son la mediana y el IQR de cada métrica. Además, se indica la dirección en la cual debe observarse cada métrica al comparar (en densidad y cubrimiento mayores valores indican mejores resultados, mientras que en FID menores valores indican mejores resultados, y se destacan en negritas las mejores configuraciones en cada métrica).

Configuración		Densidad ↑		Cubrimiento ↑		FID ↓	
Celdas	Aptitud	m	IQR	m	IQR	m	IQR
2×2	E–GAN	**0.849**	0.065	0.648	0.105	157.748	103.458
	GDPP	0.830	0.052	**0.712**	0.029	**120.328**	25.723
3×3	E–GAN	0.821	0.065	0.675	0.073	159.126	100.384
	GDPP	0.801	0.064	0.666	0.085	127.002	57.213
4×4	E–GAN	0.806	0.041	0.723	0.067	102.690	38.637
	GDPP	0.793	0.030	0.662	0.099	130.798	72.450

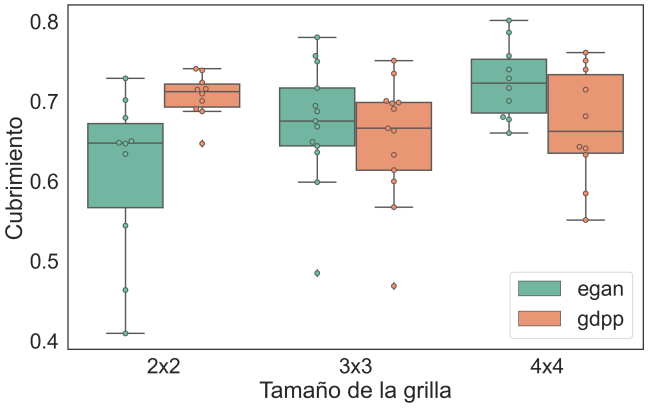
Tabla 6.3: Resultados de las configuraciones en FV–MOEA. Se reporta la mediana m y el rango intercuartílico IQR de densidad, cubrimiento y FID para cada configuración, en muestras correspondientes a 10 ejecuciones. Las configuraciones son pares (función de aptitud, tamaño de la grilla). Se expresa el tamaño de la grilla como la cantidad de celdas que la componen. Las funciones de aptitud consideradas son E–GAN y GDPP. Los tamaños de grilla son 2×2, 3×3 y 4×4 celdas. Las configuraciones con tamaños de grilla 4×4 se reportan separadas por no ser consideradas en la evaluación de desempeño.

Sobre la densidad, la configuración (E–GAN, 2×2) fue la que alcanzó mejores resultados, seguida de (GDPP, 2×2) que presentó una mediana 2,24 % menor que la primera, e IQR 20 % menor que el resto de las configuraciones. Asimismo, (GDPP, 2×2) fue la configuración que alcanzó mejores valores en cubrimiento, con una mediana 9,88 % mayor e IQR 72,38 % menor que (E–GAN, 2×2). Además, (GDPP, 2×2) fue la métrica que presentó menor FID, con mediana 23,72 % menor e IQR 75,13 % menor que (E–GAN, 2×2). En general, se destacó el buen desempeño y la robustez de (GDPP, 2×2) en todas las métricas, presentando buenos resultados y baja variabilidad incluso al considerar las configuraciones con tamaño de grilla 4×4.

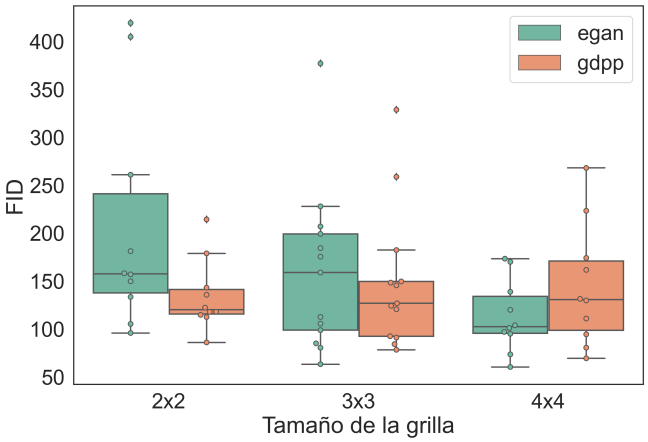
En la Figura 6.6 se presentan los gráficos de caja asociados a los resultados alcanzados por cada configuración para FV–MOEA. Los gráficos proveen una visualización alternativa sobre los datos presentados en la Tabla 6.3.



(a) Densidad



(b) Cubrimiento



(c) FID

Figura 6.6: Gráficos de cajas de los resultados obtenidos para FV–MOEA. Se reportan cubrimiento, densidad y FID discriminados por los parámetros de la configuración: la función de aptitud (E–GAN en verde y GDPP en naranja) y por el tamaño de la grilla (2×2, 3×3 o 4×4 celdas, en el eje horizontal).

Los diagramas de caja de la Figura 6.8 presentan las relaciones entre métricas y parámetros así como comparar de forma global las distintas configuraciones. Los resultados indicaron un descenso en densidad (Figura 6.12b) a medida que se aumentó el tamaño de la grilla. No obstante, las configuraciones con grilla 4×4 son las que presentaron menor variabilidad, mientras que aquellas con grillas de tamaño 3×3 presentaron mayor variabilidad. Asimismo, todos los tamaños de grilla con E–GAN como función de aptitud obtuvieron mejores resultados que los mismos tamaños con GDPP. Por otro lado, respecto al cubrimiento (Figura 6.6b) no se observaron tendencias claras entre los resultados obtenidos y los parámetros, aunque se destacó la baja variabilidad de la configuración (E–GAN, 2×2) respecto a las demás.

La Figura 6.6c presenta un decrecimiento en densidad para las configuraciones con E–GAN como función de aptitud al aumentar el tamaño de la grilla. Esta observación no es válida para GDPP, en cuyo caso las distribuciones del FID están centradas en la misma zona de la recta real. Sin embargo, sí es posible observar un aumento en la variabilidad de los resultados de las configuraciones con GDPP al aumentar el tamaño de la grilla, tendencia inversa a la observada en las configuraciones con E–GAN.

En general, los resultados indicaron que entre las configuraciones estudiadas, (GDPP, 2×2) era la más adecuada para continuar el análisis de FV–MOEA. Esto se debió a su buen desempeño relativo en densidad, a su destacable desempeño en cubrimiento y FID y a la baja variabilidad de sus resultado en comparación con los obtenidos por las otras configuraciones.

6.3.4. Análisis de parámetros de MOEA/D

Esta sección describe el análisis de configuración paramétrica para MOEA/D. En la Figura 6.7 se presenta un gráfico de dispersión con las seis configuraciones en el plano densidad×cubrimiento. Los valores reportados corresponden a las medianas de los resultados obtenidos en las diez ejecuciones de cada configuración. Se codificaron las funciones de aptitud con colores (verde para E–GAN y naranja para GDPP) y los tamaños de la grilla con formas (círculo para 2×2, cuadrado para 3×3 y cruz para 4×4).

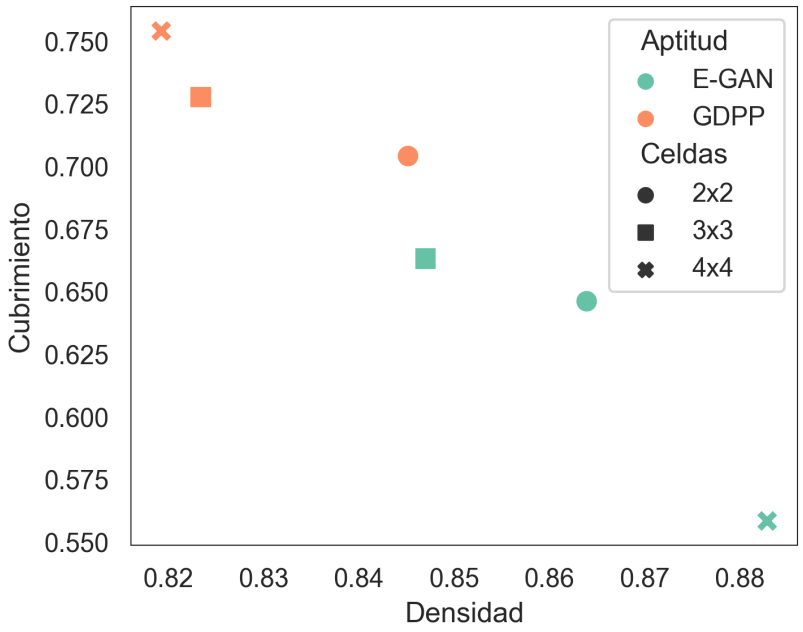


Figura 6.7: Representación de los resultados de la configuración paramétrica para MOEA/D en el plano densidad×cubrimiento. Se presentan las medianas de los resultados obtenidos por configuración, cuyos valores numéricos están disponibles en la Tabla 6.4. En verde y en naranja se indican los puntos correspondientes a configuraciones con aptitud E–GAN y GDPP, respectivamente. Las formas de los puntos expresan los tamaños de grilla. Círculo, cuadrado y cruz expresan los tamaños 2×2, 3×3 y 4×4, respectivamente.

Al considerar todas las configuraciones, se identifican como configuraciones no dominadas (GDPP, 4×4), (GDPP, 2×2), (E–GAN, 2×2) y (E–GAN, 4×4). Sin embargo, al excluir las configuraciones con tamaño de grilla 4×4 el frente no dominado fue compuesto por (GDPP, 3×3), (GDPP, 2×2) y (E–GAN, 2×2). En el resto del análisis no se consideran las configuraciones con grillas de dimensión 4, por los motivos explicados en el Capítulo 4.

En la Tabla 6.4 se presentan los resultados de densidad, cubrimiento y FID para los resultados de FV–MOEA en cada configuración. Los valores reportados son la mediana y el IQR de cada métrica. Además, se indica la dirección en la cual debe observarse cada métrica al comparar (en densidad y cubrimiento mayores valores indican mejores resultados, mientras que en FID menores valores indican mejores resultados, y se destacan en negritas las mejores configuraciones en cada métrica).

Configuración		Densidad ↑		Cubrimiento ↑		FID ↓	
Celdas	Aptitud	m	IQR	m	IQR	m	IQR
2×2	E–GAN	**0.864**	0.151	0.646	0.161	184.476	230.309
	GDPP	0.845	0.041	0.704	0.072	157.572	76.076
3×3	E–GAN	0.847	0.054	0.663	0.054	165.069	71.237
	GDPP	0.823	0.035	**0.728**	0.058	**107.542**	53.670
4×4	E–GAN	0.883	0.127	0.558	0.193	304.225	235.748
	GDPP	0.819	0.018	0.754	0.038	88.882	32.314

Tabla 6.4: Resultados de las configuraciones en MOEA/D. Se reporta la mediana m y el rango intercuartílico IQR de densidad, cubrimiento y FID para cada configuración, en muestras correspondientes a 10 ejecuciones. Las configuraciones son pares (función de aptitud, tamaño de la grilla). Se expresa el tamaño de la grilla como la cantidad de celdas que la componen. Las funciones de aptitud consideradas son E–GAN y GDPP. Los tamaños de grilla son 2×2, 3×3 y 4×4 celdas. Las configuraciones con tamaños de grilla 4×4 se reportan separadas por no ser consideradas en la evaluación de desempeño.

Respecto a la densidad, (E–GAN, 2×2) alcanzó el resultado con mejor mediana y la mayor variabilidad. En comparación con las otras configuraciones no dominadas, esta configuración presentó una mediana 2,24 % mayor que (GDPP, 2×2) y 4,98 % mayor que (GDPP, 3×3). Sin embargo, (E–GAN, 2×2) presentó IQR 72,85 % mayor que (GDPP, 2×2) Y 76,82 % mayor que (GDPP, 3×3). Se destaca que (GDPP, 3×3) fue la configuración que presentó menor variabilidad en densidad, con un IQR 14,63 % menor que (GDPP, 2×2).

Sobre el cubrimiento, (GDPP, 3×3) presentó el mayor valor de todos y la menor variabilidad entre las configuraciones no dominadas. (GDPP, 3×3) obtuvo una mediana 12,69 % mayor que (E–GAN, 2×2) y 3,41 % mayor que (GDPP, 2×2). El IQR observado fue 63,98 % menor que el de (E–GAN, 2×2) y 19,44 % menor que el de (GDPP, 2×2).

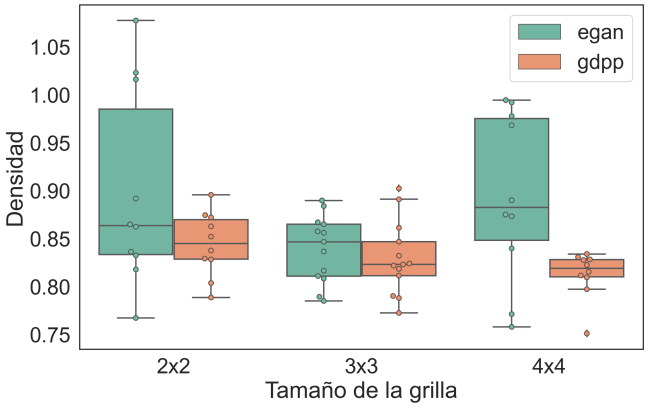
Respecto al FID, (GDPP, 3×3) fue la métrica que obtuvo la mejor mediana y la menor variabilidad. En este caso, esta configuración obtuvo mediana 41,70 % menor que (E–GAN, 2×2) y 31,75 % menor que (GDPP, 2×2). Además, los resultados obtenidos con (GDPP, 3×3) presentaron variabilidad 76,70 % menor que (E–GAN, 2×2) y 29,45 % menor que (GDPP, 2×2).

En la Figura 6.8 se presentan los gráficos de caja asociados a los resultados alcanzados por cada configuración para MOEA/D. Los gráficos proveen una visualización alternativa sobre los datos presentados en la Tabla 6.4 y facilitan la comparación de las distribuciones de los resultados obtenidos para cada configuración.

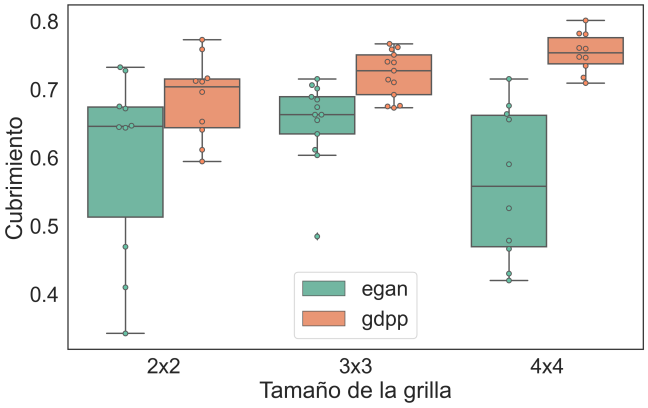
Al considerar las configuraciones con E–GAN como función de aptitud resaltaron las altas variabilidades en los tamaños de grilla 2×2 y 4×4, considerablemente mayores a las presentadas por los resultados obtenidos con grillas de tamaño 3×3. Además, se observaron peores medianas en densidad y mejores medianas en cubrimiento y FID para las grillas de tamaño 3×3. Respecto a las configuraciones con GDPP como función de aptitud, se observaron tendencias decrecientes en densidad y FID y creciente en cubrimiento al aumentar el tamaño de la grilla. Estas tendencias implicaron que el desempeño en fidelidad de MOEA/D se redujo con grillas mayores, mientras que, en contrapartida, aumentó el desempeño en diversidad.

En general, se destacó la baja variabilidad de las configuraciones con GDPP en comparación a las que tienen E–GAN como función de aptitud. Sin embargo, en grillas de tamaño 3×3 ambas funciones de aptitud alcanzaron variabilidades semejantes. Los resultados también indicaron que las configuraciones con E–GAN presentaron mejores resultados medianos de densidad y peores resultados medianos de cubrimiento y FID en comparación a sus contrapartes con GDPP.

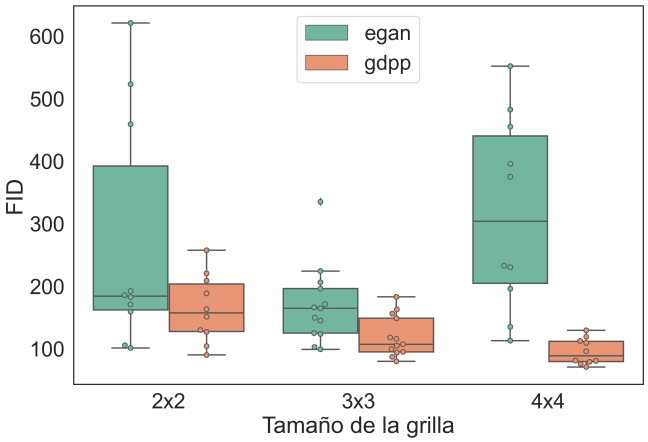
Finalmente, se concluyó que (GDPP, 3×3) era la más adecuada de las configuraciones estudiadas para continuar el análisis de MOEA/D. Esta conclusión se debe a que presentó resultados considerablemente superiores en cubrimiento y FID, compensando su inferioridad moderada en densidad. Su robustez, analizada mediante el IQR, en comparación con las otras configuraciones no dominadas, fue un factor favorable a la selección de (GDPP, 3×3)



(a) Densidad



(b) Cubrimiento



(c) FID

Figura 6.8: Gráficos de cajas de los resultados obtenidos para MOEA/D. Se reportan cubrimiento, densidad y FID discriminados por los parámetros de la configuración: la función de aptitud (E–GAN en verde y GDPP en naranja) y por el tamaño de la grilla (2×2, 3×3 o 4×4 celdas, en el eje horizontal).

6.3.5. Conclusiones del análisis de parámetros

En esta sección se recopilan las conclusiones obtenidas mediante los análisis de configuración paramétrica de cada algoritmo y se comparan brevemente los resultados entre algoritmos.

Los resultados de NSGA–II y FV–MOEA tuvieron menor dispersión que MOEA/D para las tres métricas estudiadas. Al analizar los gráficos de caja que reportan los resultados de las métricas para cada algoritmo analizado se concluye que NSGA–II y FV–MOEA tuvieron un IQR dos veces menor en las tres métricas reportadas que MOEA/D. En cuanto a las funciones de costo,

GDPP tuvo menor dispersión que E-GAN. Para la mayoría de experimentos realizados, GDPP demostró tener entre dos y diez veces menor dispersión que E-GAN. Una menor dispersión no viene acompañada necesariamente de un mejor desempeño. Sin embargo, una baja dispersión puede ser un atributo deseable ya que describe a un algoritmo más robusto y permite tener más confianza en resultados conseguidos en una cantidad baja de experimentos.

E-GAN produjo consistentemente mejores resultados de densidad. E-GAN obtuvo un mejor desempeño que GDPP en densidad para todos los experimentos. Un mayor desempeño en densidad implica que se logran generar imágenes de mejor calidad (mas parecidas a las reales) pero no da garantía de su diversidad.

No se observó una correlación entre el tamaño de la grilla y las métricas. Contrariamente a lo que esperaba el equipo de trabajo y a lo reportado en el articulo de Lipizzaner, no se observó una correlación entre el tamaño de grilla y el desempeño. A partir de los datos obtenidos mediante la configuración paramétrica no se puede asegurar que aumentar el tamaño de grilla este relacionado a mejores los resultados en las métricas para todos los casos. Que no exista correlación entre el tamaño de la grilla y los resultados de las métricas puede deberse a los algoritmos de reemplazo introducidos y a no utilizar mutaciones especificas para introducir o mejorar la diversidad.

Finalmente, las configuraciones elegidas para la evaluación de desempeño fueron:

- Para NSGA–II se seleccionó E–GAN y tamaño de grilla 3×3.
- Para FV–MOEA se seleccionó GDPP y tamaño de grilla 2×2.
- Para MOEA/D se seleccionó GDPP y tamaño de grilla 3×3.

En la Figura 6.9 se presentan dígitos generados por los ensambles de GANs entrenadas sobre MNIST con las tres configuraciones seleccionadas. Visualmente se observó que los generadores no están colapsados (generan ejemplares de todas las clases). Se observó que la mayoría de los dígitos son reconocibles visualmente en los resultados de cualquiera de los tres algoritmos, aunque resulta difícil clasificar visualmente algunos de ellos.

(a) FV–MOEA **(b)** MOEA/D **(c)** NSGA–II

Figura 6.9: Ejemplares de MNIST generados por los ensambles ganadores de cada algoritmo de reemplazo.

6.4. Evaluación de desempeño

En esta sección se presentan los resultados de la evaluación de desempeño sobre el conjunto de datos CelebA. Para el análisis se utilizan los valores de los parámetros elegidos considerando los resultados de los experimentos de configuración paramétrica para cada uno de los MOEAs implementados: NSGA–II, FV–MOEA y MOEA/D. Asimismo, se realiza una comparación del mejor MOEA con la línea de base: el algoritmo original de Lipizzaner. Al aumentar la complejidad del conjunto de datos utilizado, fue necesario también aumentar la complejidad de las redes neuronales entrenadas en este proceso. En este caso, se utilizan redes convoluciones, tal como se describió en la metodología de trabajo, descrita en la Sección 4.4.

6.4.1. Metodología de evaluación de desempeño

Para la evaluación de desempeño se contó con resultados calculados para diez ejecuciones independientes de cada métrica de evaluación (FID, densidad y cubrimiento) para cada uno de los algoritmos implementados (NSGA–II, FV-MOEA y MOEA/D). El criterio para la elección de estas métricas es el mismo que el utilizado en la configuración paramétrica, que fue descripto en la Sección 6.2 sobre el análisis de las métricas de evaluación.

El análisis estadístico para soportar la evaluación de desempeño se condujo según la siguiente metodología. Inicialmente, se estudió la normalidad de los resultados obtenidos, con el fin de decidir si es posible estudiarlas mediante un análisis paramétrico, tal como se reporta en la Subsección 6.4.2. Luego, se compararon los resultados de los MOEAs utilizando métodos compatibles con las conclusiones del paso anterior, como se reporta en la Subsección 6.4.3. Finalmente, se compararon los mejores algoritmos con la línea de base, como se reporta en la Subsección 6.4.4.

Las preguntas que se deseaban responder con este análisis estadístico eran: ¿alguno de los tres MOEAs estudiados tiene un mejor desempeño según alguna métrica?; ¿es posible elegir un mejor MOEA teniendo en cuenta la globalidad de los resultados? y ¿alguno de los MOEAs implementados supera en desempeño a la línea de base según las métricas de evaluación consideradas?

6.4.2. Análisis de normalidad de los resultados

Para estudiar la normalidad de los resultados, se aplicaron los contrastes de hipótesis de D'Agostino-Pearson, Anderson-Darling y de Shapiro-Wilk y se discutieron los resultados. Se realizaron también los gráficos cuantil-cuantil, o gráficos Q-Q, entre los cuantiles empíricos de los resultados estandarizados y los teóricos de una distribución normal estándar para aportar a la discusión de los resultados. Se adoptaron tres contrastes de normalidad debido a que consideran distintas características para establecer evidencia a favor de la normalidad, aumentando la robustez del análisis. El criterio de decisión para aceptar la hipótesis es por votación entre los tres contrastes. Los gráficos Q-Q, por otro lado, son una herramienta gráfica útil para visualizar los desvíos de la normalidad y diagnosticar qué tan generales son, o si se deben a valores atípicos.

En la Figura 6.10 se presentan los gráficos Q-Q para cada uno de los algoritmos estudiados en las tres métricas consideradas. La principal referencia en estos gráficos es la recta de la identidad: cuanto mejor alineados estén los puntos con ella, más parecidos son los datos a los provenientes de una distribución normal estándar.

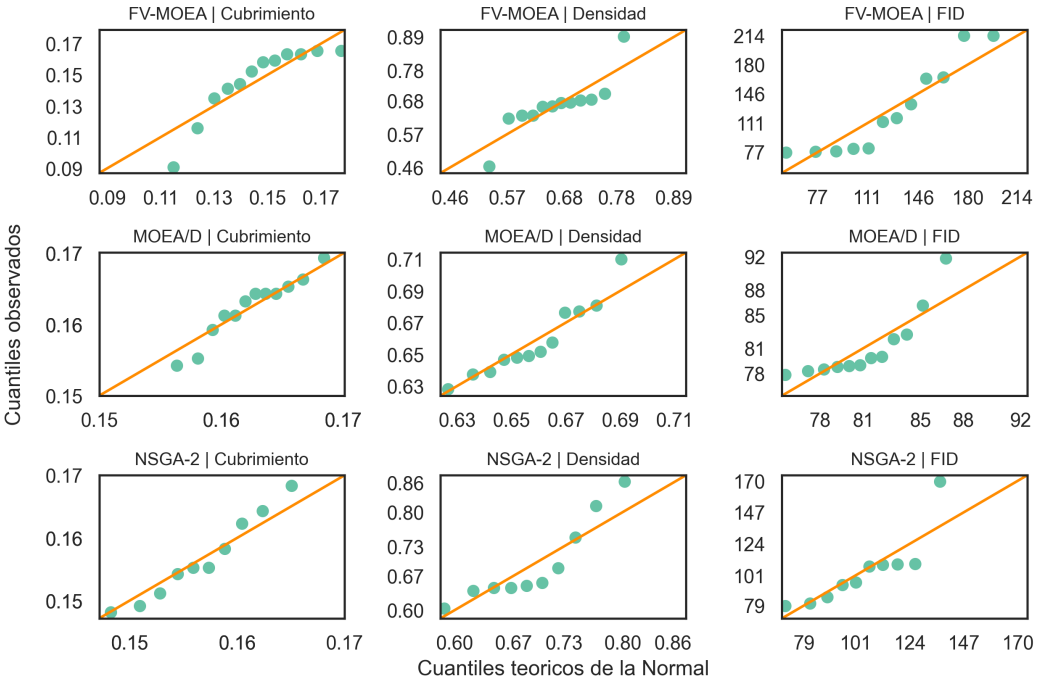


Figura 6.10: Gráficos Q-Q para las métricas de evaluación calculadas sobre los resultados de cada algoritmo en CelebA. En verde se presentan los resultados obtenidos en el plano de los cuantiles teóricos de una distribución normal de referencia y los cuantiles observados. La recta de la identidad, en naranja, sirve como referencia gráfica para diagnosticar la aproximación de los resultados a la distribución de referencia. Si los datos están alineados sobre esta recta, se deduce que se distribuyen de manera similar a una distribución normal. La distribución de referencia tiene media y desvío estándar iguales a los empíricos.

En la Tabla 6.5 se presentan los resultados de los contrastes de hipótesis para cada algoritmo y para cada métrica. Según lo definido en la metodología de trabajo, descrita en la Capítulo 4, se determinó un nivel de confianza del 90 %, por lo que se consideraron significativas para rechazar la hipótesis nula (normalidad de los datos) aquellos experimentos para las cuales se hayan obtenido p-valores menores al nivel de significancia de 0.1.

Algoritmo	Métrica	p-valores		
		Anderson-Darling	Shapiro-Wilk	D'Agostino-Pearson
NSGA–II	FID	*0.02*	*0.01*	*0.00*
	densidad	*0.02*	*0.04*	0.22
	cubrimiento	0.70	0.67	0.69
FV–MOEA	FID	*0.06*	*0.04*	0.41
	densidad	*0.00*	*0.01*	*0.04*
	cubrimiento	*0.02*	*0.02*	*0.02*
MOEA/D	FID	*0.00*	*0.00*	*0.00*
	densidad	0.21	0.26	0.24
	cubrimiento	0.31	0.47	0.58

Tabla 6.5: Resumen de los contrastes de hipótesis de normalidad sobre los resultados de cada algoritmo sobre CelebA. Se reportan p-valores para los tests de Anderson-Darling, Shapiro-Wilk y D'Agostino-Pearson sobre los resultados obtenidos en cada métrica de interés: FID, densidad y cubrimiento. En itálicas, se destacan aquellos resultados significativos para rechazar la normalidad, considerando un nivel de significación de 0.1 o, equivalentemente, un nivel de confianza del 90 %.

En el caso de NSGA–II, la normalidad de los resultados de FID se rechazó por unanimidad, mientras que la normalidad de los resultados de densidad se rechazó por mayoría. No se detectó evidencia en contra de la normalidad de los datos de cubrimiento, dado que los p-valores observados no solamente son mayores a 0.1, sino que son relativamente altos. Estos resultados son coherentes con los gráficos de la Figura 6.10, en la cual se muestra que los datos de cubrimiento se disponen relativamente bien en la recta identidad, mientras que en las otras métricas se muestran otras relaciones (no lineales) entre los cuantiles.

En el caso de FV–MOEA, los resultados de los tests sugieren rechazar la normalidad de los resultados de todas las métricas. Esta conclusión se tomó por unanimidad para densidad y cubrimiento. Para FID se rechazó la normalidad por mayoría. El rechazo de la normalidad para los resultados de FV–MOEA en las tres métricas consideradas es coherente con lo indicado en la Figura 6.10, en la cual se muestran relaciones entre cuantiles que se alejan claramente de la linealidad.

En el caso de MOEA/D, se rechazó la normalidad de los resultados de

FID por unanimidad con p-valores prácticamente cercanos a cero. Respecto a densidad, no fue posible rechazar la hipótesis de normalidad. En la Figura 6.10 se identifica que en en los datos de densidad para MOEA/D hay un punto muy alejado de la identidad, a la vez que el resto de los puntos están muy cercanos a ella, lo que explica los resultados obtenidos. En la Figura 6.10 se muestra que los datos de cubrimiento para MOEA/D están alineados sobre la identidad, lo que es coherente con los p-valores obtenidos.

En conclusión, se puede rechazar la hipótesis de normalidad para alguno de los algoritmos en todas las métricas, por lo que no es posible realizar un análisis estadístico paramétrico para la comparación de los resultados. Por este motivo, se adoptan los experimentos no paramétricos de Kruskall-Wallis y de Kolmogorov-Smirnov para el análisis de las distribuciones de resultados, escogidas por los motivos descriptos en los fundamentos teóricos comentados en Subsección 3.7.3.

6.4.3. Desempeño y comparación entre algoritmos

En esta sección se reportan los resultados obtenidos para cada MOEA en las tres métricas de evaluación consideradas: FID, densidad y cubrimiento. El reporte se enfoca en la comparación de los algoritmos desarrollados, con el objetivo de estudiar en qué aspectos destaca cada uno y si es posible elegir alguno que sea superior globalmente.

En la Figura 6.11 se presenta el gráfico de dispersión sobre el plano densidad×cubrimiento de todas las ejecuciones realizadas, se discriminan los resultados según su algoritmo de reemplazo: en azul FV–MOEA, en naranja MOEA/D y en verde NSGA–II.

El gráfico de la Figura 6.11 muestra una gran dispersión de los resultados de FV–MOEA en ambos ejes, mientras que NSGA–II presenta poca variabilidad en cubrimiento y alta dispersión en densidad. Los resultados de MOEA/D, por otro lado, muestran poca variabilidad en ambos ejes, dado que están concentrados en valores medianos de densidad y altos de cubrimiento.

En la Tabla 6.6 se presenta las mediana m y el IQR de las métricas de evaluación para cada algoritmo. Se optó por reportar m debido a su robustez frente a valores atípicos y reportar IQR por la naturaleza no paramétrica del análisis de comparación realizado. Los datos de los rangos intercuartílicos son coherentes con lo reportado en el gráfico de dispersión de la Figura 6.11, puesto

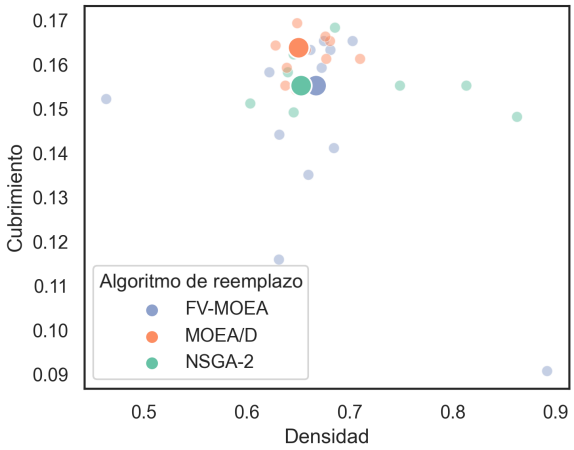


Figura 6.11: Representación de los resultados sobre el conjunto de datos CelebA para cada algoritmo en el plano densidad×cubrimiento. Los puntos pequeños corresponden a los resultados obtenidos con las ejecuciones y los puntos grandes/círculos son las medianas. Se discriminan los resultados según su algoritmo de reemplazo: en azul FV–MOEA, en naranja MOEA/D y en verde NSGA–II.

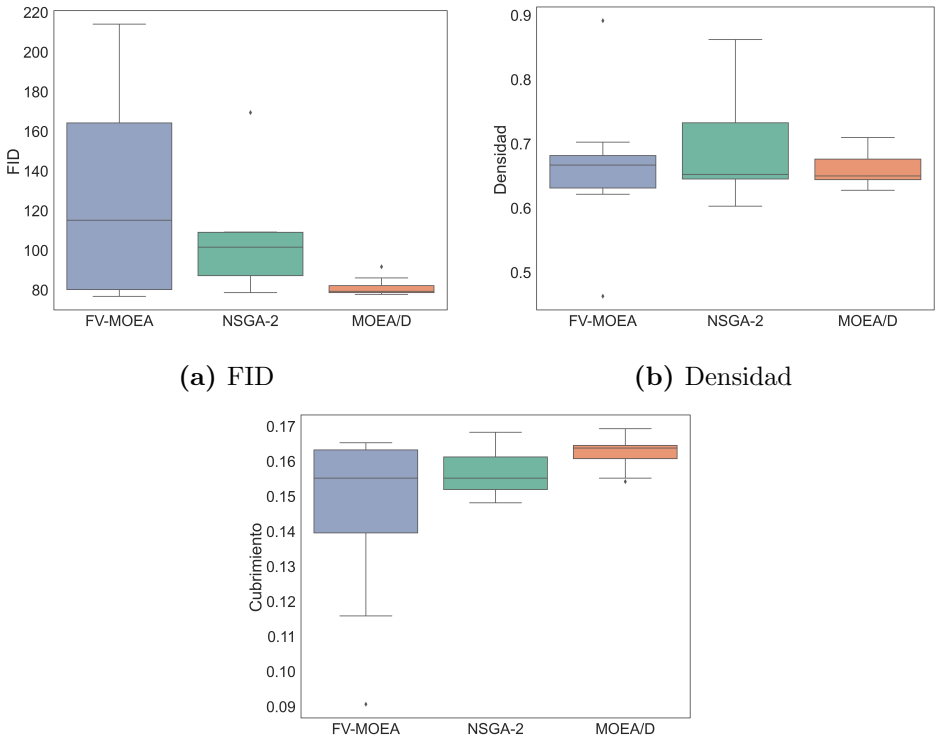
que MOEA/D presentó valores de IQR bajos en densidad y cubrimiento, FV–MOEA presentó valores de IQR moderados en densidad y relativamente altos en cubrimiento y NSGA–II presentó valores de IQR altos en densidad y bajos en cubrimiento. El IQR de MOEA/D en FID fue notoriamente inferior al de los otros algoritmos, con una diferencia de aproximadamente 80 unidades con FV–MOEA y de 18 unidades con NSGA–II, lo cual implica una mayor robustez de MOEA/D según este criterio.

Algoritmo	Densidad ↑		Cubrimiento ↑		FID ↓	
	m	IQR	m	IQR	m	IQR
FV–MOEA	**0.667**	0.051	0.155	0.024	115.229	83.996
MOEA/D	0.650	0.032	**0.164**	0.004	**79.418**	3.540
NSGA–II	0.653	0.087	0.155	0.009	101.654	21.847

Tabla 6.6: Resumen de los resultados de las tres funciones de reemplazo FV–MOEA, NSGA–II y MOEA/D. Se reportan, para cada caso, la mediana m y el IQR de las métricas de evaluación de interés: FID, densidad y cubrimiento.

Respecto a las medianas de los resultados, la Tabla 6.6 indica que FV–MOEA presentó la mayor mediana en densidad, con poca diferencia con los otros algoritmos. MOEA/D presentó valores levemente mayores que el resto en

cubrimiento. No obstante, MOEA/D destacó por su desempeño en FID, donde presentó una mediana 31 % menor que FV–MOEA y 22 % menor que NSGA–II. Al considerar que interesa maximizar densidad y cubrimiento y minimizar FID, los resultados son favorables a MOEA/D.



(a) FID **(b)** Densidad

(c) Cubrimiento

Figura 6.12: Gráficos de cajas de los resultados obtenidos para el conjunto de datos CelebA. Se reportan cubrimiento, densidad y FID discriminados por el algoritmo de reemplazo (FV–MOEA, NSGA–II o MOEA/D) en el eje horizontal.

Para la comparación formal entre algoritmos, dado que no se verificó la normalidad de los resultados obtenidos, se optó por realizar un análisis no paramétrico basado en los contrastes de hipótesis de Kruskall-Wallis y de Kolmogorov-Smirnov. Con el contraste de Kruskall-Wallis se estudia la hipótesis de que los datos provienen de la misma distribución y con el de Kolmogorov-Smirnov se estudia la hipótesis de dominancia estocástica entre muestras. Los resultados se complementan con gráficos de las funciones de distribución acumulada empírica y de la densidad estimada de los resultados.

En la Tabla 6.7 se reportan los resultados del contraste de Kruskall-Wallis sobre cada algoritmo y para cada métrica. El primer contraste tiene como hipótesis nula que los resultados de los tres algoritmos provienen de la misma

distribución. En el caso de FID y cubrimiento, hay evidencia suficiente para rechazar esta afirmación. En densidad, los resultados no permiten rechazar la afirmación. Se realizó el contraste de Kruskall-Wallis entre pares, con el objetivo de estudiar más a fondo las relaciones de igualdad entre muestras. Los resultados de los contrastes entre pares que la conclusión para densidad se mantuvo en todos los casos, lo que coincide con la Figura 6.12b, cuyos diagramas de caja evidencian el solapamiento entre las distribuciones de los tres algoritmos en la métrica densidad.

Respecto a FID y cubrimiento, hay fuerte evidencia en contra de la igualdad de los resultados obtenidos entre MOEA/D y NSGA–II y entre MOEA/D y FV–MOEA. Sin embargo, los resultados muestran poca evidencia a favor de la desigualdad de los resultados de NSGA–II y FV–MOEA en todas las métricas. Este resultado resalta las similitudes de desempeño en los resultados de estos algoritmos.

Hipótesis	p-valores		
	FID	Densidad	Cubrimiento
NSGA–II $\sim$ FV–MOEA $\sim$ MOEA/D	*0.03*	0.82	*0.04*
NSGA–II $\sim$ FV–MOEA	0.55	0.79	0.53
NSGA–II $\sim$ MOEA/D	*0.00*	0.55	*0.04*
MOEA/D $\sim$ FV–MOEA	*0.09*	0.69	*0.03*

Tabla 6.7: Resumen de los contrastes de hipótesis de igualdad de distribuciones sobre los resultados de cada algoritmo sobre CelebA. Se reportan p-valores para los contrastes de Kruskall-Wallis sobre los resultados obtenidos en cada métrica de interés: FID, densidad y cubrimiento.

En la Tabla 6.8 se presentan los p-valores del contraste de Kolmogorov-Smirnov. Se reportan valores para el contraste de dos colas (identificado por el símbolo $\sim$), cuyo objetivo es verificar la bondad de ajuste entre dos muestras, y para el contraste de una cola (identificado por el símbolo $\succ_{(1)}$), cuya hipótesis nula es que existe una relación de dominancia estocástica entre los algoritmos comparados. Nuevamente, se reportan resultados del contraste para cada par de algoritmos y en el caso de dos colas se verifican resultados similares a los del contraste de Kruskall-Wallis. No obstante, se destaca el p-valor de 0.11

obtenido para la igualdad en FID de NSGA–II y FV–MOEA, que resultó casi significativo para el nivel de confianza de 90 % estipulado en la descripción metodológica.

Hipótesis			p-valores		
			FID	Densidad	Cubrimiento
NSGA–II	$\sim$	FV–MOEA	0.11	0.71	0.23
NSGA–II	$\sim$	MOEA/D	*0.00*	0.56	*0.06*
FV–MOEA	$\sim$	MOEA/D	*0.03*	0.54	*0.10*
FV–MOEA	$\succ_{(1)}$	NSGA–II	0.43	0.48	0.12
MOEA/D	$\succ_{(1)}$	NSGA–II	*0.00*	0.28	0.95
NSGA–II	$\succ_{(1)}$	FV–MOEA	*0.06*	0.37	0.74
MOEA/D	$\succ_{(1)}$	FV–MOEA	*0.02*	0.27	1.00
NSGA–II	$\succ_{(1)}$	MOEA/D	1.00	0.71	*0.03*
FV–MOEA	$\succ_{(1)}$	MOEA/D	0.73	0.48	*0.05*

Tabla 6.8: p-valores obtenidos al realizar el contraste de hipótesis de Kolmogorov-Smirnov sobre los resultados de cada algoritmo en CelebA. Se reportan resultados para el contraste de una cola (identificado por el símbolo $\sim$), cuya hipótesis nula es que ambas muestras provienen de la misma distribución, y para el contraste de cola superior (identificado por el símbolo $\succ_{(1)}$), cuya hipótesis nula es que la muestra de la izquierda domina estocásticamente a la muestra de la derecha.

Los resultados obtenidos para la comparación del FID sugieren una fuerte evidencia en contra de la dominancia estocástica de MOEA/D sobre FV–MOEA (p-valor de 0.02) y NSGA–II (p-valor de 0.00). Respecto a cubrimiento, se rechazó la dominancia de NSGA–II y FV–MOEA sobre MOEA/D, con p-valores de 0.03 y 0.05 respectivamente. Dado que interesa minimizar FID y maximizar cubrimiento, se interpreta que MOEA/D presentó resultados superiores a los de los otros algoritmos.

Respecto a NSGA–II y FV–MOEA, solo es posible rechazar la hipótesis de que NSGA–II domina a FV–MOEA en FID. Sin embargo, se obtuvo un p-valor de 0.12, relativamente bajo para el nivel de significación estipulado, para la hipótesis de que FV–MOEA domina a NSGA–II en cubrimiento. La no dominancia de FV–MOEA sobre NSGA–II en cubrimiento se muestra en la Figura 6.12c con la baja dispersión de los resultados de NSGA–II en comparación a los de FV–MOEA.

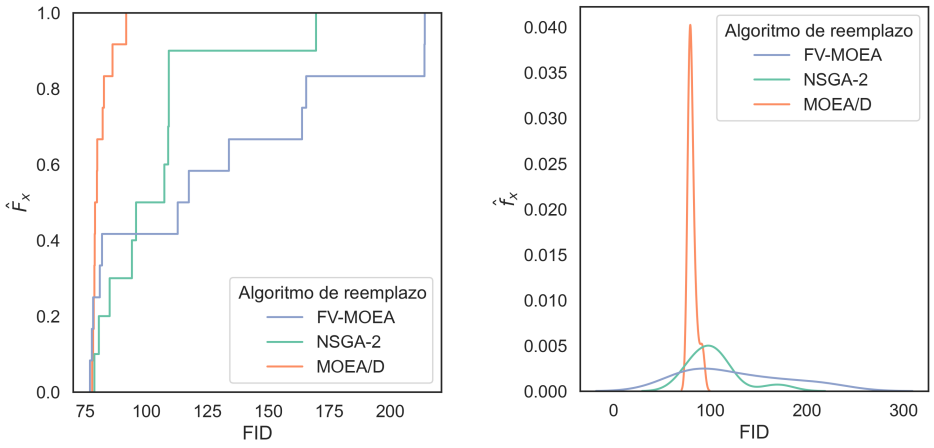
En la Figura 6.13 se presentan las gráficas de la función de distribución acumulada empírica y de la densidad estimada de cada muestra. Las gráficas presentan los resultados obtenidos de los contrastes de Kolmogorov-Smirnov permitiendo diagnosticar gráficamente la dominancia estocástica. En el caso de FID, los resultados demostraron que MOEA/D acumuló prácticamente toda su probabilidad en regiones más bajas del soporte, mientras que FV–MOEA y NSGA–II presentaron más dispersión y se encontraron localizadas en regiones más altas.

En densidad, los resultados indicaron que las tres distribuciones se encontraron localizadas en la misma zona del soporte y que acumularon probabilidad de manera semejante, puesto que presentaron dispersión similar. En cubrimiento la distribución de MOEA/D se encontró localizada en zonas más altas del soporte, lo que apoyó la hipótesis de que este algoritmo domina a los demás en este criterio. En todos los casos, MOEA/D contó con una baja dispersión en relación al resto de los algoritmos estudiados, lo que se intepretó como que MOEA/D es el más robusto de los algoritmos comparados.

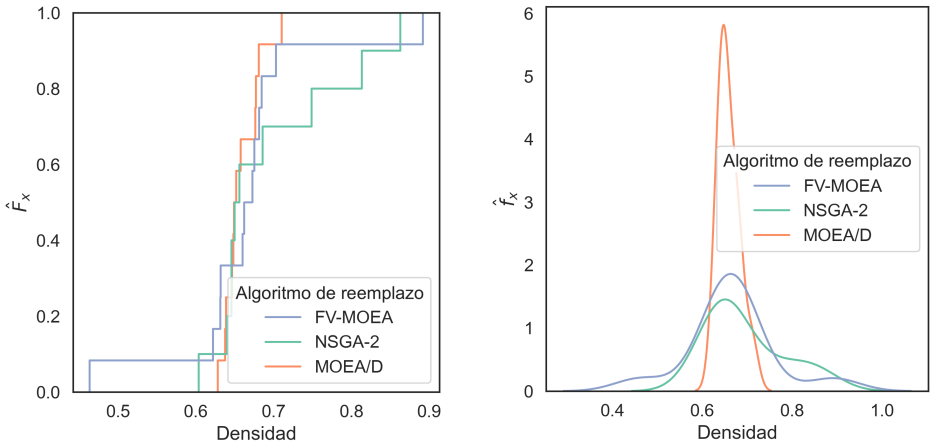
A modo de recapitulación, se realizó una comparación preliminar entre algoritmos analizando los diagramas de dispersión de los resultados, los diagramas de cajas, las medianas y los rangos intercuartílicos. La comparación permitió identificar la superioridad de MOEA/D en FID y cubrimiento, así como la baja dispersión de sus resultados.

Luego, se utilizó el contraste de hipótesis de Kruskall-Wallis para explorar la igualdad entre muestras. La igualdad de muestras se rechazó para FID y cubrimiento, pero no para densidad. No haber rechazado la hipótesis de igualdad de muestras para densidad se interpretó como evidencia de la no dominancia de ninguno de los algoritmos respecto a densidad. Se utilizó el contraste de Kolmogorov-Smirnov para estudiar las relaciones de dominancia estocástica entre variables. Las principales conclusiones del análisis fueron que NSGA–II y FV–MOEA no dominaron a MOEA/D en cubrimiento y que MOEA/D no dominó a NSGA–II y FV–MOEA en FID.

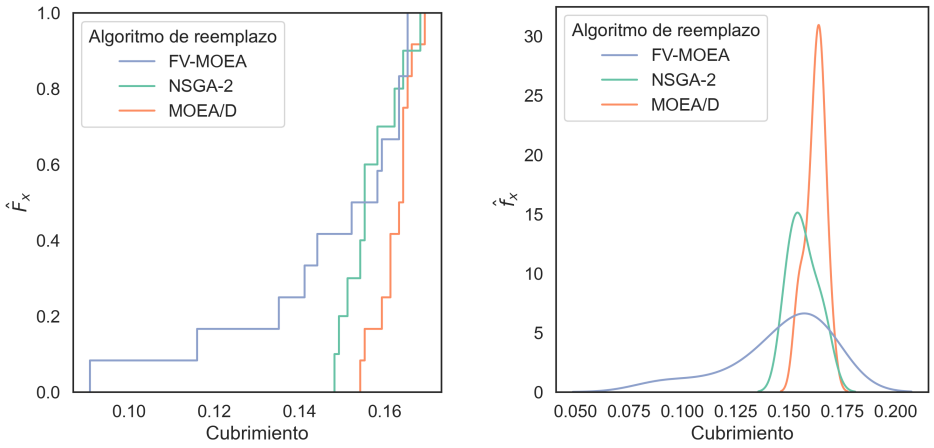
Finalmente, todas las conclusiones se apoyaron en la visualización de las funciones de distribución acumulada empírica y de densidad estimada de los resultados. Se culminó el análisis comparativo concluyendo la superioridad global de MOEA/D según los resultados obtenidos y los criterios considerados.



(a) $\hat{F}_x$ - FID **(b)** $\hat{f}_x$ - FID



(c) $\hat{F}_x$ - densidad **(d)** $\hat{f}_x$ - densidad



(e) $\hat{F}_x$ - cubrimiento **(f)** $\hat{f}_x$ - cubrimiento

Figura 6.13: Gráficos de la función de distribución acumulada empírica $\hat{F}_x$ y de la función de densidad empírica $\hat{f}_x$ para cada métrica de interés según los resultados sobre CelebA, discriminadas por algoritmo.

6.4.4. Comparación con la línea de base

En esta sección se compara el algoritmo que presentó mejor desempeño global, MOEA/D, con el algoritmo original de Lipizzaner, orientado a la optimización mono-objetivo. El análisis adoptado es análogo al utilizado para comparar a los MOEAS entre sí. Se utilizan herramientas gráficas y resúmenes estadísticos para obtener una idea preliminar sobre la comparación de desempeños. Luego se formaliza el análisis con análisis estadísticos no paramétricos y se verifican los resultados intermedios con los resultados de la función de distribución acumulada empírica y la función de densidad estimada de cada muestra.

En la Figura 6.14 se presenta el diagrama de dispersión de los resultados de MOEA/D y de Lipizzaner en el plano densidad×cubrimiento. Se destacaron la alta dispersión de los resultados del algoritmo base en ambas dimensiones, aunque la mayoría de los resultados se concentraron en la misma zona que los resultados de MOEA/D. Los resultados presentados en la Tabla 6.9 indicaron, al igual que la gráfica, una mayor dispersión de los resultados de Lipizzaner dado que MOEA/D presentó valores de IQR 81 % menores en densidad, 88 % menores en cubrimiento y 96 % menores en FID.

Algoritmo	Densidad ↑		Cubrimiento ↑		FID ↓	
	m	IQR	m	IQR	m	IQR
Lipizzaner base	**0.736**	0.172	0.160	0.032	106.370	80.101
MOEA/D	0.650	0.032	**0.164**	0.004	**79.418**	3.540

Tabla 6.9: Resumen de los resultados de MOEA/D y del algoritmo base sobre el conjunto de datos CelebA. Se reportan la mediana m y el IQR de las métricas de evaluación de interés: FID, densidad y cubrimiento.

Respecto a las medianas de las distribuciones de los resultados, el algoritmo base presentó una mediana 12 % mayor en densidad y MOEA/D una mediana 25 % menor en FID. Ambos algoritmos obtuvieron resultados con medianas semejantes en cubrimiento. Los IQR y las medianas de los resultados de cada algoritmo se presentan en los diagramas de caja de la Figura 6.15. Los diagramas sugieren la superioridad de MOEA/D en todas las métricas, aunque menos clara en los casos de densidad y cubrimiento.

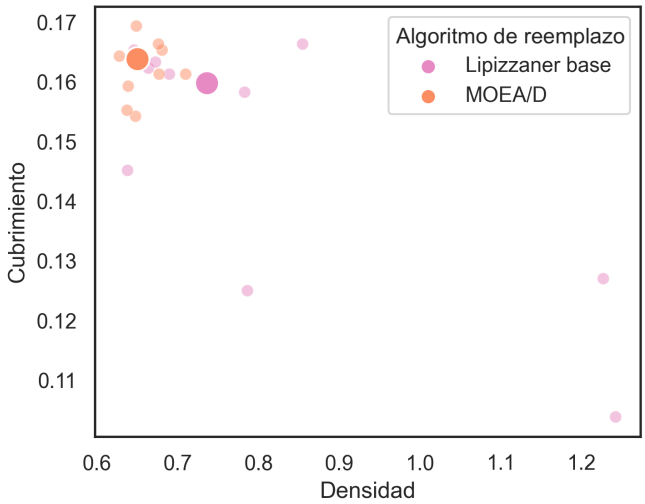
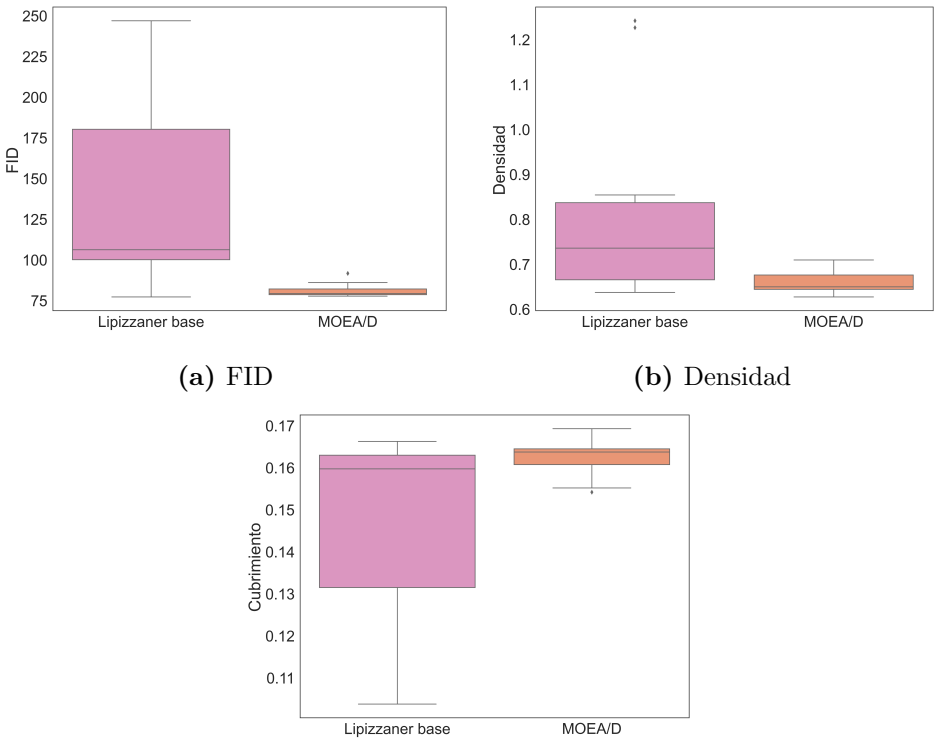


Figura 6.14: Representación de los resultados sobre el conjunto de datos CelebA para MOEA/D y Lipizzaner en el plano densidad×cubrimiento. Los puntos pequeños corresponden a los resultados obtenidos con las ejecuciones y los círculos grandes corresponden a las medianas por algoritmo de reemplazo. En naranja y rosado se indican los puntos correspondientes a MOEA/D y Lipizzaner, respectivamente.



(a) FID **(b)** Densidad

(c) Cubrimiento

Figura 6.15: Gráficos de cajas de los resultados obtenidos para el conjunto de datos CelebA. Se reportan cubrimiento, densidad y FID para MOEA/D y la linea base.

En la Tabla 6.10 se reportan los p-valores de los contrastes de hipótesis realizados para comparar los resultados de MOEA/D con los del algoritmo base. Los resultados indicaron una fuerte evidencia en contra de la hipótesis de igualdad de muestras para las tres métricas, puesto que los contrastes de Kruskall-Wallis y Kolmogorov-Smirnov resultaron significativos al 90 % de confianza para los casos de FID y densidad, y presentaron p-valores relativamente bajos para el cubrimiento.

Al realizar los contrastes de Kolmogorov-Smirnov de una cola, se obtuvieron resultados que permitieron rechazar las hipótesis de dominancia estocástica de MOEA/D sobre Lipizzaner en FID y densidad. Entonces, se interpretó que MOEA/D es superior en FID e inferior en densidad cuando se compara con el algoritmo base. Respecto al cubrimiento, no fue posible rechazar la hipótesis de que Lipizzaner dominó a MOEA/D, aunque el p-valor de 0.14, relativamente bajo para el nivel de significación de 0.1, es indicador de que la evidencia a favor de esta hipótesis fue débil.

Hipótesis	p-valores		
	FID	Densidad	Cubrimiento
MOEA/D $\sim$ Lipizzaner base (KW)	*0.01*	*0.03*	0.12
MOEA/D $\sim$ Lipizzaner base (KS)	*0.00*	*0.07*	0.28
MOEA/D $\prec_{(1)}$ Lipizzaner base (KS)	0.84	1.00	0.14
MOEA/D $\succ_{(1)}$ Lipizzaner base (KS)	*0.00*	*0.04*	1.00

Tabla 6.10: Resumen de los contrastes de hipótesis de igualdad y dominancia estocástica de distribuciones sobre los resultados de MOEA/D y el algoritmo base sobre CelebA. Se reportan p-valores para los contrastes de Kruskall-Wallis (KW) y de Kolmogorov-Smirnov (KS) sobre los resultados obtenidos en cada métrica de interés: FID, densidad y cubrimiento

En la Figura 6.16 se presentan los gráficos de la función de distribución acumulada empírica y de la función de densidad estimada para los resultados de Lipizzaner y MOEA/D en las tres métricas de interés. Las funciones de distribución acumulada empírica evidencian que MOEA/D fue estocásticamente menor que Lipizzaner en FID y densidad, soportando las conclusiones sobre la dominancia estocástica de Lipizzaner sobre MOEA/D según Kolmogorov-Smirnov.

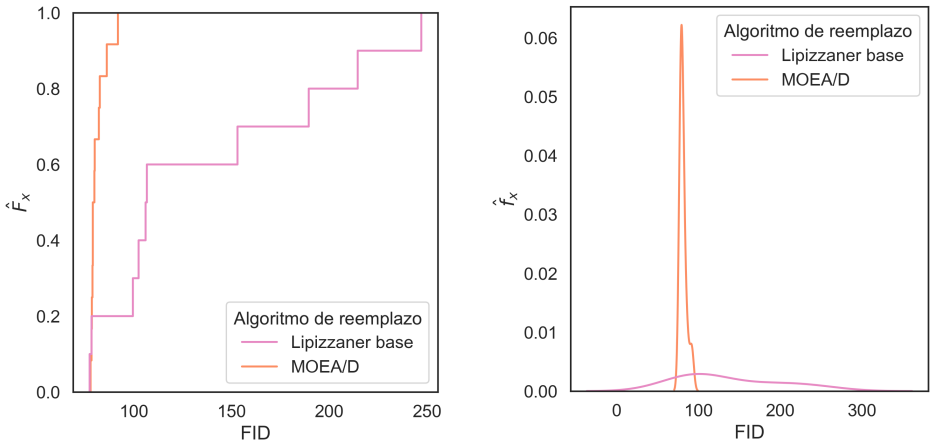
En las funciones de distribución acumulada para el cubrimiento (Figura 6.16e) los resultados indican que Lippizzaner es estocásticamente mejor que MOEA/D según el criterio de orden estocástico. Un leve solapamiento entre las distribuciones acumuladas explica el p-valor bajo, pero insuficiente para rechazar la hipótesis de dominancia de Lipizzaner sobre MOEA/D.

Las funciones de densidad estimada en la Figura 6.16 indican nuevamente la baja dispersión de los resultados de MOEA/D respecto a los resultados de Lipizzaner. Asimismo, los resultados de FID y densidad de MOEA/D concentraron su probabilidad en zonas más bajas del soporte que Lipizzaner. Respecto al cubrimiento, ambas distribuciones se localizaron en la misma zona del soporte, aunque los resultados para MOEA/D acumularon más probabilidad en las zonas altas del soporte.

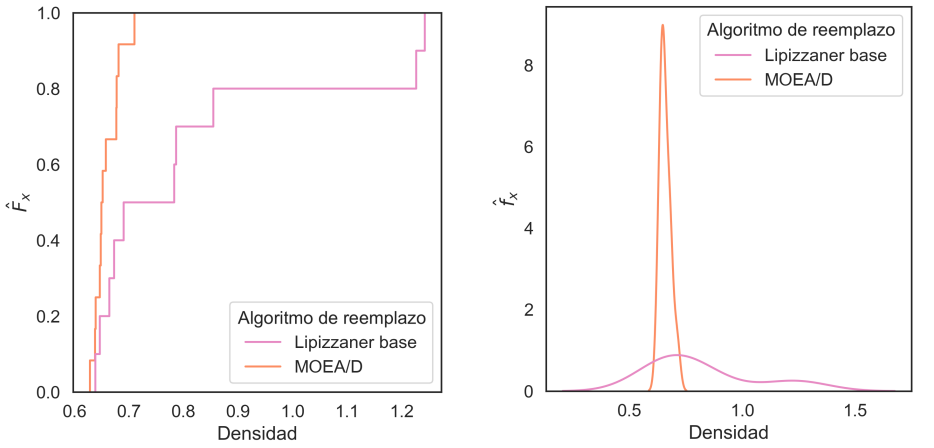
La Figura 6.17 presenta imágenes generadas por los cuatro modelos entrenados en CelebA: Lipizzaner base (Figura 6.17a); MOEA/D (Figura 6.17b); FV–MOEA (Figura 6.17c) y NSGA–II (Figura 6.17d). Las imágenes no presentan claras diferencias visuales entre los algoritmos en calidad o diversidad.

6.4.5. Conclusiones de la evaluación de desempeño

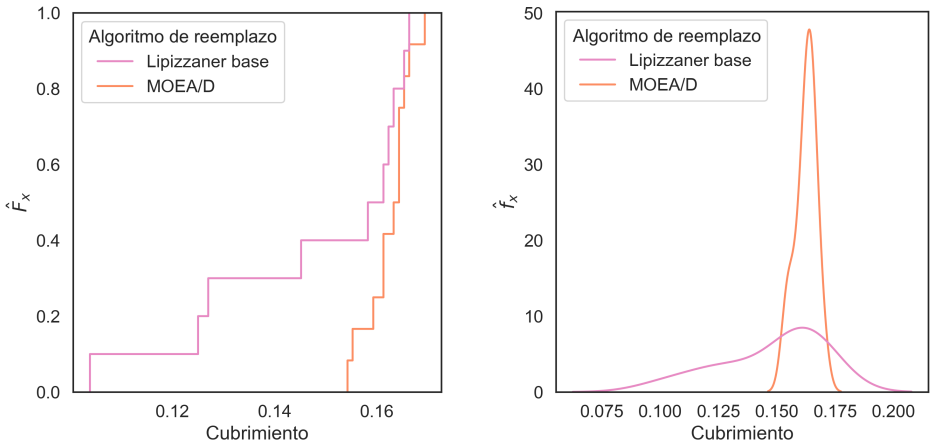
A partir de los resultados reportados se concluyó que MOEA/D es superior a Lipizzaner en FID y cubrimiento, mientras que Lipizzaner es superior a MOEA/D en densidad. Esta conclusión implica que la introducción de la diversidad en el proceso de entrenamiento de GANs favoreció a la diversidad de las imágenes generadas por los modelos obtenidos, aunque perjudicó su fidelidad. Se destaca que el uso de conjuntos de datos diversos también favorece a la minimización de FID, una métrica utilizada para evaluar la fidelidad. En la Figura 6.18 se presenta un mapa de calor con las medianas de los resultados obtenidos para cada métrica y cada algoritmo. Los datos se presentan normalizados en el intervalo [0,1] y sintetizan las conclusiones alcanzadas con el análisis estadístico formal: MOEA/D presenta los mejores resultados en cubrimiento y FID, mientras que Lipizzaner presenta los mejores resultados en densidad.



(a) $\hat{F}_x$ - FID **(b)** $\hat{f}_x$ - FID



(c) $\hat{F}_x$ - densidad **(d)** $\hat{f}_x$ - densidad



(e) $\hat{F}_x$ - cubrimiento **(f)** $\hat{f}_x$ - cubrimiento

Figura 6.16: Gráficos de la función de distribución acumulada empírica $\hat{F}_x$ y de la función de densidad empírica $\hat{f}_x$ para cada métrica de interés según los resultados sobre CelebA, discriminadas por algoritmo.

(a) Lipizzaner base **(b)** MOEA/D

(c) FV–MOEA **(d)** NSGA–II

Figura 6.17: Ejemplares de CelebA generados por los ensambles ganadores de cada algoritmo de reemplazo y el de la linea base.

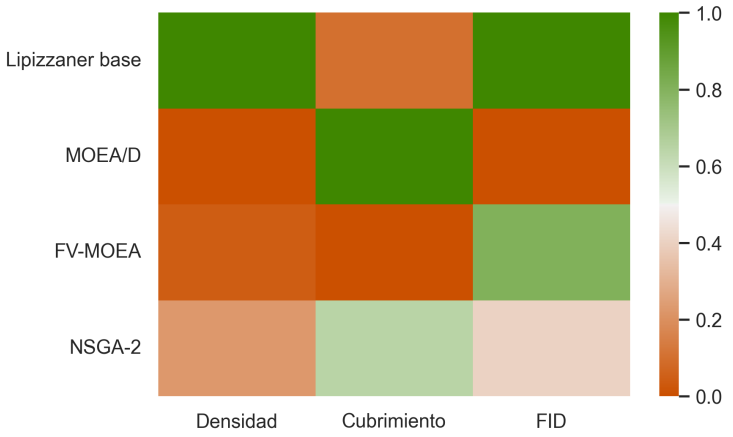


Figura 6.18: Mapa de calor entre los algoritmos y las métricas de evaluación para los resultados de la evaluación de desempeño. Los datos están normalizados por métrica entre 0 y 1. El valor de los datos se expresa según los colores, denotando el verde un valor próximo a 1 y el naranja un valor próximo a 0. Los datos numéricos están expuestos en la Tabla 2.4 del Apéndice 2.

6.5. Evaluación de escalabilidad y tiempo

En esta sección se analiza la escalabilidad de Lipizzaner luego de la migración a MPI. La Subsección 6.5.1 describe el análisis comparativo entre los distintos parámetros evaluados durante la etapa de experimentación y cómo inciden en el tiempo total de ejecución de Lipizzaner. Luego, la Subsección 6.5.2 reporta el análisis de escalabilidad de Lipizzaner. En la última sección se estudia el balance de carga entre los procesos encargados de ejecutar cada una de las celdas de Lipizzaner.

6.5.1. Influencia de los parámetros en el tiempo total de ejecución de Lipizzaner

La evaluación de escalabilidad se realizó sobre el mismo conjunto de experimentos que la configuración paramétrica. Antes de la evaluación, se realizó un análisis del impacto del algoritmo de reemplazo y la medida de diversidad en el tiempo total de ejecución de Lipizzaner.

En la Figura 6.19 se muestra la distribución de tiempos de ejecución para las dos medidas de diversidad estudiadas. Los resultados indican que GDPP es estrictamente mayor en tiempo total de ejecución para todos los tamaños de grilla.

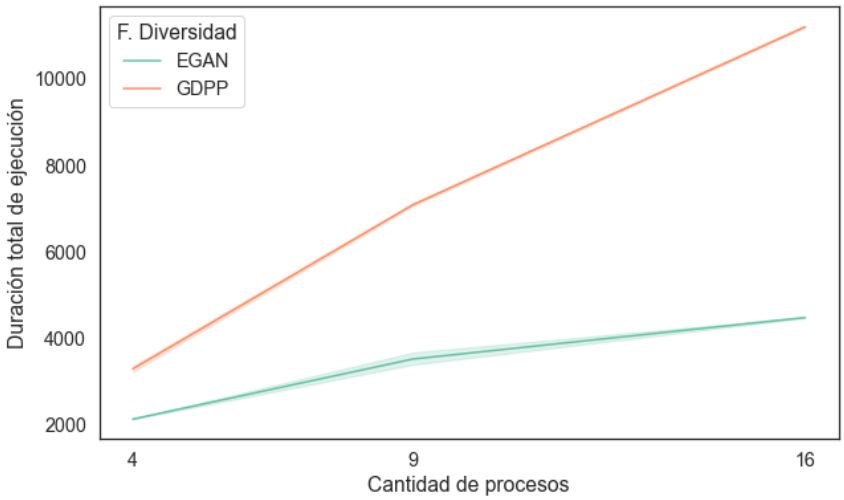


Figura 6.19: Distribuciones de tiempo total de ejecución para las diferentes medidas de diversidad.

Las distribuciones de los tiempos totales de ejecución de los tres algoritmos de reemplazo se muestran en la Figura 6.20. Las tres distribuciones coinciden en mediana y varianza, indicando que el algoritmo de reemplazo tiene baja incidencia en el tiempo total de ejecución.

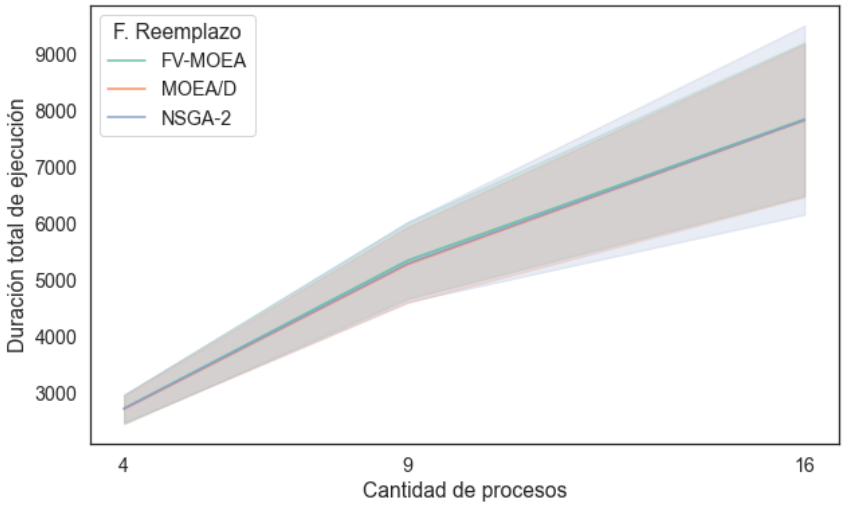


Figura 6.20: Distribuciones de tiempo total de ejecución para los diferentes algoritmos de reemplazo.

El tiempo total de ejecución es la métrica que se analiza en la sección de análisis de escalabilidad y balance de carga. Dado que el tiempo total de ejecución se ve significativamente alterado por la medida de diversidad elegida

para el experimento, en las siguientes secciones se agrupan los experimentos según la medida de diversidad utilizada y los resultados se analizan independientemente.

6.5.2. Resultados del análisis de escalabilidad

Para el análisis de escalabilidad se evaluó la interacción entre el tiempo total de ejecución y el tamaño de la grilla. La Figura 6.21 presenta una aproximación de la relación de dependencia entre el tamaño de la grilla y el tiempo total de ejecución utilizando un modelo de regresión lineal. Se observó en la Figura 6.21a que el modelo de regresión lineal no se ajustó precisamente a los datos, ya que presentó un R^2 de 0.860. Sin embargo, al ajustar el modelo de regresión lineal únicamente sobre los experimentos que utilizan GDPP (Figura 6.21b) el modelo presentó un R^2 de 0.996, por lo que el modelo de GDPP consiguió un mejor ajuste del modelo a los datos y se utilizó este para medir la paralelicibilidad P_L.

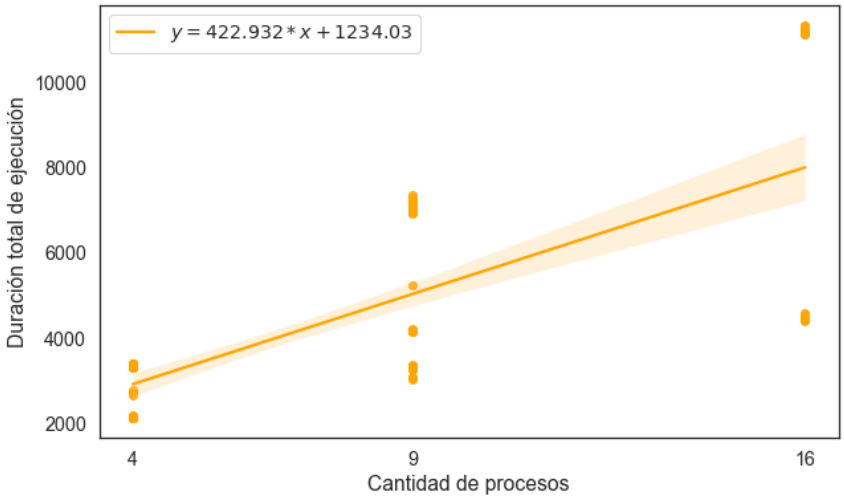
Como se definió en la Subsección 4.4.4, la evaluación de escalabilidad se realizó midiendo la paralelicibilidad P_L. P_L es el valor de la pendiente de la recta ajustada en la Figura 6.21b definida por la Ecuación 6.1. El valor de pendiente resultó mayor a cero como era esperable. El crecimiento lineal de los tiempos de ejecución se debe a que no todos los pasos de Lipizzaner son paralelizables. Por ejemplo, el paso final del algoritmo donde el proceso maestro recibe todas las redes de la grilla para optimizar los pesos del ensamble, es un paso no paralelizable. Además, el hardware puede ser una limitante para incrementar el tamaño de la grilla, ya que algunos recursos compartidos como las tarjetas gráficas, el acceso al disco compartido y la red son un cuello de botella, dado que no incrementan su velocidad, capacidad y disponibilidad.

$$y = 653,974 * x + 909,906 \tag{6.1}$$

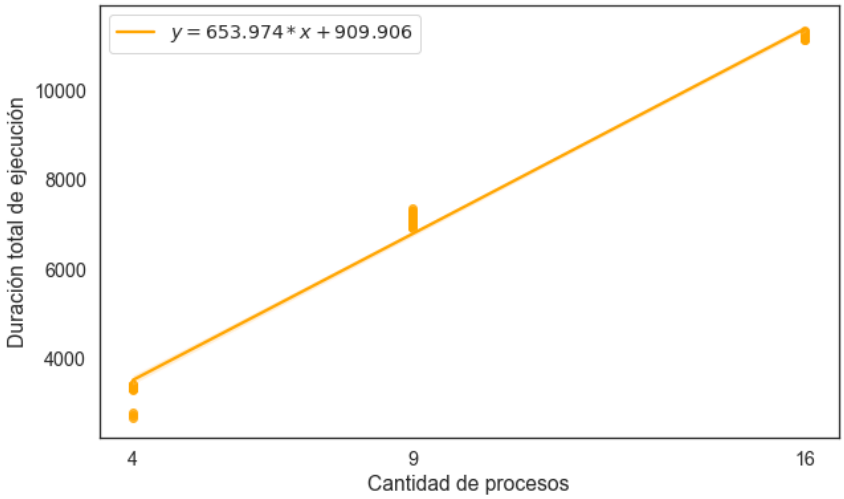
6.5.3. Balance de carga y asignación de recursos

El objetivo de esta subsección es reportar los resultados del análisis de la distribución de la carga y la asignación de recursos entre procesos.

La carga de trabajo en Lipizzaner se distribuye asignando la misma cantidad de iteraciones de entrenamiento de las redes a todos los procesos pertenecientes a una grilla de Lipizzaner. Además, cada proceso trabaja sobre



(a) Ajustado en todos los experimentos.



(b) Ajustado en los experimentos que utilizan GDPP.

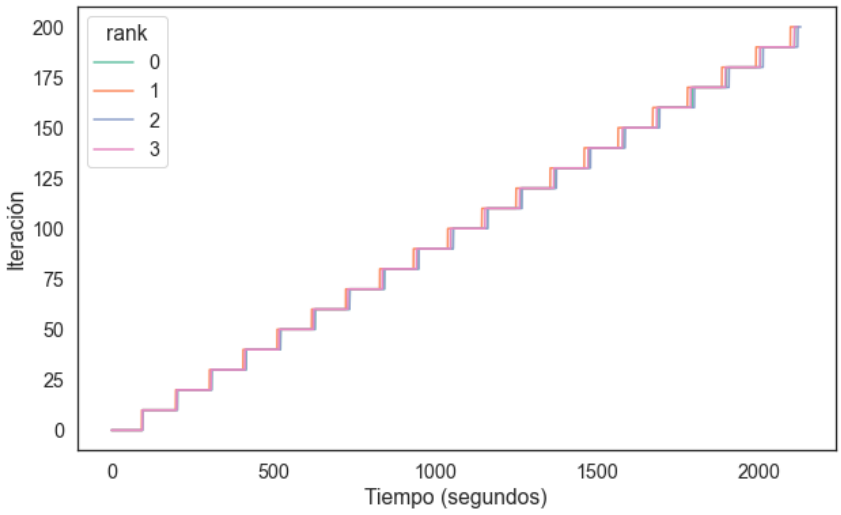
Figura 6.21: Aproximación de la relación de dependencia entre la cantidad de procesos y el tiempo total de ejecución utilizando un modelo de regresión lineal. Los datos utilizados son los obtenidos durante la configuración paramétrica.

una porción del conjunto de datos utilizado, que es del mismo tamaño para cada proceso. Si el algoritmo a ejecutar, la cantidad de iteraciones y los datos son iguales entre procesos, se puede garantizar que la carga entre procesos es equitativa.

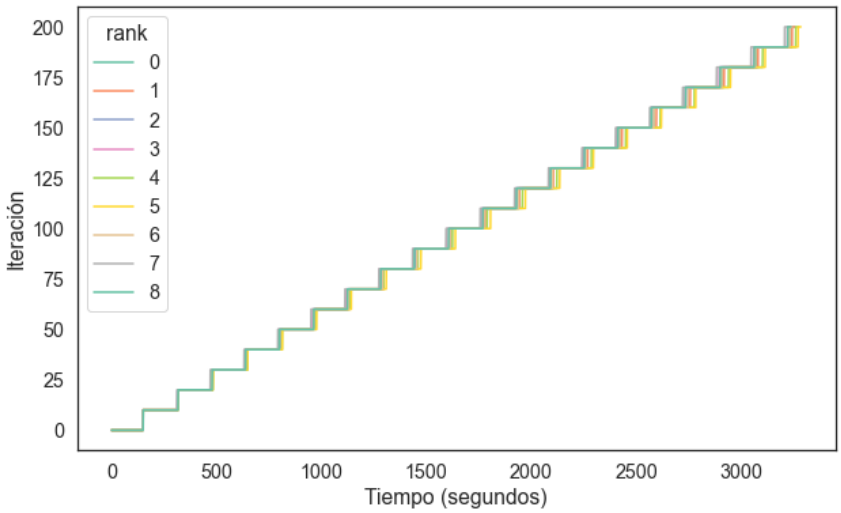
En la Figura 6.22 se presenta la duración de cada iteración para cada uno de los procesos pertenecientes a una grilla de Lipizzaner. Las figuras 6.22a y 6.22b indican que los procesos se mantuvieron en la misma o a una iteración de distancia durante todo el entrenamiento. Sin embargo, en la figura 6.22c se muestra que el proceso 10 finalizó su ejecución mientras el proceso 9 aún tenía dos iteraciones por procesar. La diferencia entre los finales de las iteraciones se incrementa conforme avanza el tiempo para el caso de la grilla de 4×4.

En la Tabla 6.11 se reporta la media y varianza en los tiempos de iteración para cada proceso. La varianza en la duración de las iteraciones fue baja, menor a un 15 % del promedio en todos los casos, lo que indica una asignación equitativa de los recursos entre los procesos.

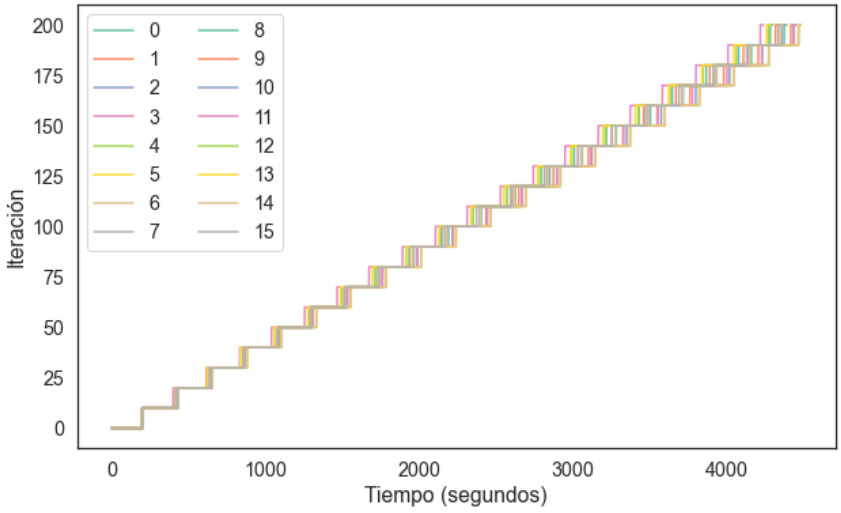
En conclusión, en los experimentos realizados no se detectaron patologías en la distribución de la carga o en la asignación de recursos. Una línea interesante de trabajo futuro es explorar como se comporta la diferencia entre los tiempos finales de iteración para tamaños de grilla más grandes, ya que si la diferencia se incrementara podría resultar en problemas de entrenamiento.



(a) Grilla de 2×2



(b) Grilla de 3×3



(c) Grilla de 4×4

Figura 6.22: Avance de las iteraciones en cada proceso para los tres tamaños de grilla evaluados.

Tamaño de grilla	Función de aptitud	Reemplazo	Promedio (s)	Desvío estándar (s)
2×2	E–GAN	FV–MOEA	10.39	0.23
		MOEA/D	10.38	0.17
		NSGA–II	10.37	0.18
	GDPP	FV–MOEA	16.10	1.46
		MOEA/D	16.00	1.50
		NSGA–II	16.07	1.46
3×3	E–GAN	FV–MOEA	17.25	2.85
		MOEA/D	16.84	1.95
		NSGA–II	16.86	1.90
	GDPP	FV–MOEA	32.34	3.76
		MOEA/D	32.16	3.63
		NSGA–II	32.38	3.63
4×4	E–GAN	FV–MOEA	21.48	0.86
		MOEA/D	21.37	0.85
		NSGA–II	21.46	0.83
	GDPP	FV–MOEA	54.92	4.00
		MOEA/D	54.81	3.89
		NSGA–II	54.83	3.88

Tabla 6.11: Promedios y desvíos estándar de tiempos de iteración.

Capítulo 7

Conclusiones y trabajo futuro

En este capítulo se recapitula el trabajo realizado en el proyecto de grado y se presentan las conclusiones derivadas del mismo. Además, se describen líneas de trabajo futuro identificadas para mejorar la solución desarrollada o profundizar en aspectos específicos de la misma.

7.1. Conclusiones

En el proyecto se plantearon diversos objetivos relacionados a la inclusión explícita de la diversidad en el algoritmo de entrenamiento de Lipizzaner, un framework coevolutivo para entrenamiento distribuido de redes generativas antagónicas. En especial, el proyecto se enfocó en la extensión del framework para soportar el paradagiama de optimización multiobjetivo, con el fin de optimizar conjuntamente la fidelidad y la diversidad de los datos generados por los modelos entrenados. Además, con el fin de realizar el análisis experimental sobre el Centro Nacional de Supercomputación, fue necesario modificar el middleware utilizado para distribuir el algoritmo, adoptando el paradigma de pasajes de mensajes con MPI.

El análisis de la literatura científica sobre optimización multiobjetivo, algoritmos coevolutivos y algoritmos evolutivos multiobjetivo permitió orientar el trabajo a la extensión de Lipizzaner mediante la implementación de tres algoritmos de optimización multiobjetivo: NSGA–II, FV–MOEA y MOEA/D. Estos algoritmos son representantes de las tres clases de MOEAs predominantes: algoritmos basados en Pareto, algoritmos basados en indicadores y algoritmos basados en decomposición, respectivamente.

La implementación se organizó en cuatro etapas: la reimplementación del esquema de distribución del framework con MPI, la adaptación del algoritmo para el uso de funciones de aptitud vectoriales en lugar de escalares, la implementación de las funciones de costo de diversidad y la implementación de los MOEAs seleccionados. Se generaron nuevas estructuras para que la funcionalidad de mensajería entre procesos y el ciclo evolutivo puedan manejarse independientemente y se utilizó el patrón de estrategia para facilitar la configuración y la extensión del framework. Concretamente, los tres MOEAs implementados se introdujeron como operadores de supervivencia, responsables por elegir los individuos pertenecientes a la siguiente generación del ciclo evolutivo.

De la nueva implementación de Lipizzaner se destacan tres fortalezas: i) la mejor separación de responsabilidades, una vez que el mecanismo de comunicación entre procesos se desacopló completamente de la lógica de computación evolutiva; ii) la mejor modularización, dado que se reimplementaron los operadores evolutivos y las funciones de aptitud siguiendo el patrón de diseño de estrategia; iii) y la generalidad en la implementación multiobjetivo, ya que el nuevo algoritmo es agnóstico al problema de diversidad de los datos abordado en este proyecto.

Para abordar el problema de la diversidad de los datos generados por los modelos entrenados en Lipizzaner.fue necesario estudiar funciones de costo de diversidad para GANs que pudieran configurarse como objetivo en el algoritmo de entrenamiento. A partir del estudio se decidió implementar y utilizar nuevas funciones de aptitud basadas en las funciones E–GAN y GDPP. Estas funciones de aptitud son bidimensionales: su primera componente se define como la función de costo en fidelidad usual en el entrenamiento de GANs y su segunda componente se define como alguna de las funciones de costo en diversidad escogidas.

El análisis experimental se enfocó en estudiar el compromiso entre fidelidad y diversidad obtenido al optimizar ambas características durante el entrenamiento de un modelo con la implementación multiobjetivo de Lipizzaner. Con este fin, se redujo el problema a la generación de imágenes y se estudiaron métricas de evaluación de modelos generativos en conjuntos de imágenes. Se estudiaron tres métricas de fidelidad (precisión, densidad y FID) y tres métricas de diversidad (exhaustividad, cubrimiento y TVD). Se optó por analizar los resultados mediante las métricas densidad y cubrimiento, por ser teóricamente

independientes y medir fidelidad y diversidad de manera aislada. Asimismo, se analizó el FID de los datos generados, debido a su popularidad en la literatura y a su utilidad como métrica de desempate.

También se investigaron contrastes de hipótesis útiles para formalizar el análisis estadístico de los resultados y complementar las evaluaciones gráficas. Los contrastes adoptados fueron los de D'Agostino-Pearson, Anderson-Darling y Shapiro-Wilk para estudiar la normalidad de los datos y los de Kruskall-Wallis y Kolmogorov-Smirnov para comparar las muestras de datos. La comparación de muestras se realizó con base en el concepto de dominancia estocástica, cuya definición e interpretación se estudiaron previamente para apoyar el análisis experimental. Una vez establecidas las herramientas de evaluación, se realizó el proceso de evaluación experimental en tres etapas: análisis de configuración paramétrica, comparación entre MOEAs y comparación con la versión original de Lipizzaner.

En el proceso de configuración paramétrica se estudió la incidencia de la función de aptitud y de la dimensión de la grilla en las métricas de evaluación seleccionadas. Se consideraron las dos funciones de aptitud implementadas con E–GAN y GDPP y tres tamaños de grilla: 2×2, 3×3 y 4×4. Se realizaron diez ejecuciones sobre el conjunto de datos MNIST para cada una de las seis configuraciones posibles y para cada uno de los tres algoritmos evaluados. Los resultados de esta etapa se analizaron utilizando resúmenes estadísticos y herramientas gráficas. Además, se utilizaron estos resultados para realizar un análisis empírico sobre las seis métricas de evaluación generadas y un estudio de la escalabilidad del nuevo sistema de distribución.

En el análisis de las métricas de evaluación se identificaron las correlaciones entre ellas y se fundamentó empíricamente la elección de densidad y cubrimiento como métricas de evaluación, puesto que presentaron baja correlación entre sí y correlaciones con las otras métricas coherentes con los conceptos de fidelidad y diversidad. En el análisis de escalabilidad se concluyó que durante el ciclo evolutivo la cantidad de trabajo entre procesos estuvo bien balanceada. Sin embargo, el tiempo necesario para finalizar el entrenamiento creció linealmente con el tamaño de la grilla. Este aumento puede deberse al ensamblaje del modelo final, que no está paralelizado y requiere trabajo proporcional a la cantidad de subpoblaciones configuradas. Se identificó, además, que las ejecuciones configuradas con GDPP requirieron mayor tiempo de ejecución que las configuradas con E–GAN.

En los resultados del análisis de configuración paramétrica las configuraciones con GDPP presentaron menos dispersión, por lo que se consideran más robustos, pero las configuraciones con E–GAN alcanzaron mejores resultados en densidad. No se identificó una relación clara entre tamaño de la grilla y alguna de las métricas para FV–MOEA y NSGA–II. Sin embargo, MOEA–D con GDPP presentó mejor desempeño en cubrimiento y FID a medida que se aumentó el tamaño de la grilla. Mediante la observación de densidad y cubrimiento y el desempate por FID fue posible seleccionar las configuraciones (E–GAN, 3×3) para NSGA–II, (GDPP, 2×2) para FV–MOEA y (GDPP, 3×3) para MOEA/D. En algunos casos, se obtuvieron mejores compromisos entre fidelidad y diversidad por configuraciones con grillas de tamaño 4×4, aunque éstas no se consideraron. Una vez finalizados los experimentos de configuración paramétrica, se procedió a utilizar las configuraciones seleccionadas para la comparación entre MOEAs.

Para comparar los algoritmos desarrollados se realizaron diez ejecuciones de cada uno de ellos sobre el conjunto de datos CelebA. Además de los resúmenes estadísticos y las herramientas gráficas utilizadas en los experimentos de configuración paramétrica, el análisis se basó en el estudio de la normalidad de los resultados obtenidos y de las relaciones de dominancia estocástica entre muestras. Para observar la normalidad de los datos se observaron gráficos Cuantil-Cuantil y se realizó una votación entre los contrastes de hipótesis de D'Agostino-Pearson, Anderson-Darling y Shapiro-Wilk. Para obtener evidencia de la dominancia estocástica entre muestras, que puede extrapolarse como una comparación de desempeño entre algoritmos, se utilizaron los contrastes de Kruskall-Wallis y de Kolmogorov-Smirnov. El análisis de dominancia estocástica se apoyó también en la comparación gráfica de las funciones de densidad y de distribución acumulada empíricas de las muestras obtenidas. Para la evaluación de todos los contrastes de hipótesis se consideró un nivel de confianza del 90 %.

En la etapa de comparación entre MOEAs se obtuvo evidencia a favor de la dominancia en cubrimiento de MOEA/D frente a los otros algoritmos, lo que indica que este algoritmo es el que presentó más diversidad en sus resultados. Respecto a la densidad, no se identificaron relaciones de dominancia entre algoritmos, por lo que la evidencia resulta insuficiente para concluir la superioridad en fidelidad de alguno. Respecto al FID, los resultados indicaron que MOEA/D no domina a FV–MOEA ni a NSGA–II en esta métrica, lo que

implica que MOEA/D presenta mejores valores de FID que los otros algoritmos. Se concluyó, entonces, la superioridad de MOEA/D frente a FV–MOEA y NSGA–II en dos de las tres métricas de interés, por lo que se procedió a comparar el desempeño de este algoritmo con la versión original de Lipizzaner.

Para comparar MOEA/D con la versión original de Lipizzaner se realizaron diez ejecuciones de Lipizzaner original sobre CelebA y se procedió de forma análoga a la comparación entre MOEAs. Se obtuvo evidencia a favor de la superioridad en cubrimiento y FID de MOEA/D sobre Lipizzaner original. Sin embargo, Lipizzaner original resultó superior a MOEA/D en densidad. Estos resultados indican que la introducción de optimización multiobjetivo a Lipizzaner mediante MOEA/D mejoró el desempeño en diversidad pero redujo el desempeño en fidelidad. Además, se destacó la robustez de MOEA/D frente a NSGA-II, FV–MOEA y Lipizzaner original durante todo el proceso de evaluación, dado que presentó los resultados con menor variabilidad.

En síntesis, en este proyecto de grado se desarrolló una nueva versión del framework Lipizzaner, construida sobre el estándar de computación distribuida MPI y capaz de realizar optimización multiobjetivo en el entrenamiento de redes neuronales antagónicas. La nueva versión fue utilizada para introducir explícitamente la optimización de diversidad en el entrenamiento de GANs y logró obtener modelos que generan conjuntos de imágenes más diversos a costa de un compromiso razonable en fidelidad. En general, se concluyó que la evidencia obtenida es favorable a la utilización de MOEAs, específicamente los basados en descomposición, para mejorar el entrenamiento de GANs. Los resultados alcanzados motivan a la continuidad de esta línea de investigación, dado que son promisores y se identificaron diversas posibilidades de mejora para la solución desarrollada.

7.2. Trabajo futuro

En esta sección se comentan algunas de las líneas de trabajo futuro que surgieron durante la realización del proyecto de grado. Las líneas de trabajo futuro se clasifican en tres categorías: posibles aplicaciones de la versión modificada del framework Lipizzaner, experimentos interesantes que quedaron fuera del alcance del proyecto y otras extensiones al framework Lipizzaner.

Durante el desarrollo del proyecto se realizó la extensión del framework Lipizzaner para dar soporte a la optimización de funciones de aptitud enfocadas

en diversidad, considerando múltiples objetivos. Sin embargo, el soporte fue añadido únicamente para la etapa de selección de los individuos (las redes) y no para la etapa de mutación (entrenamiento y propagación hacia atrás de las redes). Por lo tanto, una extensión interesante consiste en agregar a Lipizzaner la capacidad de realizar la propagación hacia atrás tomando en cuenta las funciones de aptitud de diversidad, incentivando a que las redes también se optimicen en la dirección de la diversidad. Esta idea ya fue evaluada por el artículo de Elfeki y col. [22], quienes obtuvieron resultados positivos. Para incluir a las funciones de aptitud en el proceso de propagación hacia atrás, una opción es realizar la propagación hacia atrás para cada una de las funciones de aptitud, generando múltiples individuos que luego pueden ser incluidos en el proceso de selección. Otra alternativa más simple de implementar es realizar la propagación hacia atrás de las redes sobre una combinación lineal de la función de aptitud de calidad y diversidad (aunque se pierde el enfoque multiobjetivo explícito).

Otra línea de desarrollo interesante es ampliar los experimentos de configuración paramétrica con una mayor cantidad de parámetros y valores. En este proyecto se restringió el espacio de búsqueda de forma que el tiempo de ejecución de los experimentos sea viable en el marco del proyecto de grado. Para restringir el espacio de búsqueda se seleccionaron los parámetros más interesantes, tamaño de grilla y función de costo de diversidad. Sin embargo, muchos otros parámetros pueden influenciar el desempeño del framework, como el tamaño de población, el número de épocas y las configuraciones del algoritmo de selección. Además, el espacio de búsqueda de los dos parámetros elegidos puede ser ampliado mediante la utilización de tamaños de grilla mayores y la incorporación de nuevas medidas de diversidad.

Siguiendo la línea de extender Lipizzaner, también es interesante explorar posibles mejoras en el algoritmo de optimización de los pesos del ensamble de modelos. El proceso de optimización de los pesos del ensamble es un algoritmo evolutivo de objetivo único que optimiza el FID. Es interesante desarrollar una extensión que utilice un algoritmo multiobjetivo para la optimización de los pesos del ensamble, consiguiendo un conjunto de pesos del ensamble que genere resultados no dominados según diversidad y calidad.

Finalmente, para evaluar la utilidad de la extensión del framework desarrollado es de interés crear datos sintéticos a partir de los ensambles de GANs y entrenar modelos predictivos utilizando los datos sintéticos generados. El

entrenamiento de modelos predictivos con datos sintéticos se realiza mediante las técnicas de aumentación de conjuntos de datos mencionadas en la Subsección 3.5.3. Un estudio comparando modelos entrenados con aumentación de datos (mediante los ensambles de Lipizzaner extendido) contra modelos entrenados sin la aumentación permitiría evaluar si la mejora en cubrimiento obtenida sobre el framework original se traduce en mejores conjuntos de datos aumentados para entrenar modelos predictivos.

Referencias bibliográficas

[1] Agarap A. "Deep learning using rectified linear units (ReLU)". En: *arXiv preprint* arXiv:1803.08375 (2018). Accedido en mayo 2022.

[2] Anderson T. y Darling D. "Asymptotic Theory of Certain "Goodness of Fit" Criteria Based on Stochastic Processes". En: *The Annals of Mathematical Statistics* 23.2 (1952), págs. 193-212.

[3] Anscombre F. y Glynn W. "Distribution of the kurtosis statistic for normal samples". En: *Biometrika* 70.1 (1983), págs. 227-234.

[4] Arjovsky M. y Bottou L. "Towards Principled Methods for Training Generative Adversarial Networks". En: *arXiv preprint* arXiv:1701.04862 (2017). Accedido en mayo 2022.

[5] Arora S. y Zhang Y. "Do GANs actually learn the distribution? An empirical study". En: *arXiv preprint* arXiv:1706.08224 (2017). Accedido en mayo 2022.

[6] Beume N., Naujoks B. y Emmerich M. "SMS-EMOA: Multiobjective selection based on dominated hypervolume". En: *European Journal of Operational Research* 181.3 (2007), págs. 1653-1669.

[7] Bhagat V. y Bhaumik S. "Data Augmentation using Generative Adversarial Networks for Pneumonia classification in chest Xrays". En: *Fifth International Conference on Image Information Processing*. 2019.

[8] Borji A. "Pros and cons of GAN evaluation measures". En: *Computer Vision and Image Understanding* 179 (2019), págs. 41-65.

[9] Chawla N. y col. "SMOTE: Synthetic Minority Over-sampling Technique". En: *Journal of Artificial Intelligence Research* 16 (2002), págs. 321-357.

[10] Chen Y., Lv Y. y Wang F. "Traffic Flow Imputation Using Parallel Data and Generative Adversarial Networks". En: *IEEE Transactions on Intelligent Transportation Systems* 21.4 (2020), págs. 1624-1630.

[11] Cohen J. Paul, Morrison P. y Dao L. "COVID-19 image data collection". En: *arXiv preprint* arXiv:2003.11597 (2020). Accedido en mayo 2022.

[12] D'Agostino R. y Belanger A. "A Suggestion for Using Powerful and Informative Tests of Normality". En: *The American Statistician* 44.4 (1990), pág. 316.

[13] D'Agostino R. y Pearson E. "Tests for Departure from Normality. Empirical Results for the Distributions of b_2 and $sqrt(b_1)$". En: *Biometrika* 60.3 (1973), pág. 613.

[14] Dalcin L. y Fang Y. "mpi4py: Status Update After 12 Years of Development". En: *Computing in Science and Engineering* 23.4 (2021), págs. 47-54.

[15] Deb K. y col. "A fast and elitist multiobjective genetic algorithm: NSGA-II". En: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), págs. 182-197.

[16] Deng Li. "The mnist database of handwritten digit images for machine learning research". En: *IEEE Signal Processing Magazine* 29.6 (2012), págs. 141-142.

[17] Denton E. y col. "Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks". En: *arXiv preprint* arXiv/1506.05751 (2015). Accedido en mayo 2022.

[18] Dimitrakopoulos P., Sfikas G. y Nikou C. "ISING-GAN: Annotated Data Augmentation with a Spatially Constrained Generative Adversarial Network". En: *IEEE 17th International Symposium on Biomedical Imaging.* 2020.

[19] Divine G. y col. "The Wilcoxon–Mann–Whitney Procedure Fails as a Test of Medians". En: *The American Statistician* 72.3 (2018), págs. 278-286.

[20] Donahue C. y col. "Semantically Decomposing the Latent Spaces of Generative Adversarial Networks". En: *arXiv preprint* arXiv:1705.07904 (2018). Accedido en mayo 2022.

[21] Dumoulin V. y Visin F. "A guide to convolution arithmetic for deep learning". En: *arXiv preprint* arXiv:1603.07285 (2018).

[22] Elfeki M. y col. "GDPP: Learning Diverse Generations Using Determinantal Point Process". En: *arXiv preprint* arXiv:1812.00068 (2019). Accedido en mayo 2022.

[23] Farias L. y col. "MOEA/D with Uniformly Randomly Adaptive Weights". En: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2018, págs. 641-648.

[24] Foster D. *Generative deep learning : teaching machines to paint, write, compose, and play*. O'Reilly Media, 2019.

[25] Gamma E. y col. *Design Patterns*. Addison-Wesley Longman Publishing Co. Inc., 1995.

[26] Goldberg D. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[27] Goodfellow I., Bengio Y. y Courville A. *Deep Learning*. MIT Press, 2016.

[28] Goodfellow I. y col. "Generative Adversarial Nets". En: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. 2014, págs. 2672-2680.

[29] Gropp W., Lusk E. y Skjellum A. *Using MPI*. Scientific and Engineering Computation. London, England: MIT Press, 1995.

[30] Guan Shuyue y Loew Murray. "Breast cancer detection using synthetic mammograms from generative adversarial networks in convolutional neural networks". En: *14th International Workshop on Breast Imaging*. Vol. 10718. SPIE, 2018, pág. 107180.

[31] Haibo H. y col. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning". En: *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. 2008, págs. 1322-1328.

[32] Haradal S., Hayashi H. y Uchida S. "Biosignal Data Augmentation Based on Generative Adversarial Networks". En: *40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2018.

[33] Hart A. “Mann-Whitney test is not just a test of medians: differences in spread can be important”. En: *BMJ* 323.7309 (2001), págs. 391-393.

[34] Hase N. y col. “Data augmentation for intra-class imbalance with generative adversarial network”. En: *14th International Conference on Quality Control by Artificial Vision*. Vol. 11172. 2019, págs. 34-41.

[35] He K. y col. “Deep Residual Learning for Image Recognition”. En: *arXiv preprint* arXiv:1512.03385 (2015). Accedido en mayo 2022.

[36] Heathcote A. y col. “Distribution-free tests of stochastic dominance for small samples”. En: *Journal of Mathematical Psychology* 54.5 (2010), págs. 454-463.

[37] Heusel M. y col. “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. En: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, págs. 6629-6640.

[38] Hu H., Tan T. y Qian Y. “Generative Adversarial Networks Based Data Augmentation for Noise Robust Speech Recognition”. En: *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2018.

[39] Hua Y. y col. “A Survey of Evolutionary Algorithms for Multi-Objective Optimization Problems With Irregular Pareto Fronts”. En: *IEEE/CAA Journal of Automatica Sinica* 8 (2021), págs. 303-318.

[40] Jiang S. y col. “A Simple and Fast Hypervolume Indicator-Based Multiobjective Evolutionary Algorithm”. En: *IEEE Transactions on Cybernetics* 45.10 (2015), págs. 2202-2213.

[41] Knowles J. y Corne D. “The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation”. En: vol. 1. 1999.

[42] Kruskal W. y Wallis W. “Use of Ranks in One-Criterion Variance Analysis”. En: *Journal of the American Statistical Association* 47.260 (1952), págs. 583-621.

[43] Kulesza A. “Determinantal Point Processes for Machine Learning”. En: *Foundations and Trends in Machine Learning* 5.2-3 (2012), págs. 123-286.

[44] Lan T. y col. “Arrhythmias Classification Using Short-Time Fourier Transform and GAN Based Data Augmentation”. En: *42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society*. 2020.

[45] Lavinas Y. y col. “MOEA/D with Random Partial Update Strategy”. En: *IEEE Congress on Evolutionary Computation* (2020), págs. 1-8.

[46] LeCun Y. y col. “Backpropagation Applied to Handwritten Zip Code Recognition”. En: *Neural Computation* 1.4 (1989), págs. 541-551.

[47] Lim S. y col. “DOPING: Generative Data Augmentation for Unsupervised Anomaly Detection with GAN”. En: *IEEE International Conference on Data Mining*. 2018, págs. 1122-1127.

[48] Lin Y. y Chung I. “Medical Data Augmentation Using Generative Adversarial Networks : X-ray Image Generation for Transfer Learning of Hip Fracture Detection”. En: *International Conference on Technologies and Applications of Artificial Intelligence*. 2019.

[49] Liqaa M. y Saud J. “DCGAN for Handling Imbalanced Malaria Dataset based on Over-Sampling Technique and using CNN”. En: *Medico-Legal Update* 20 (2020), págs. 1079-1085.

[50] Liu Z. y col. “Deep Learning Face Attributes in the Wild”. En: *Proceedings of International Conference on Computer Vision*. 2015.

[51] Mann H. y Whitney D. “On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other”. En: *The Annals of Mathematical Statistics* 18.1 (1947), págs. 50-60.

[52] Mariani G. y col. “BAGAN: Data Augmentation with Balancing GAN”. En: *arXiv preprint* arXiv:1803.09655 (2018).

[53] Mirza M. y Osindero S. “Conditional Generative Adversarial Nets”. En: *CoRR* abs/1411.1784 (2014).

[54] Muramatsu C. y col. “Improving breast mass classification by shared data with domain transformation using a generative adversarial network”. En: *Computers in Biology and Medicine* 119 (2020), pág. 103698.

[55] Naeem M. y col. “Reliable Fidelity and Diversity Metrics for Generative Models”. En: (2020).

[56] Nesmachnow S. e Iturriaga S. “Cluster-UY: Collaborative Scientific High Performance Computing in Uruguay”. En: *Communications in Computer and Information Science*. 2019, págs. 188-202.

[57] Pearson E. “Note on Tests for Normality”. En: *Biometrika* 22.3-4 (1931), págs. 423-424.

[58] Perrakis S. “Stochastic Dominance: Introduction”. En: *Stochastic Dominance Option Pricing*. 2019, págs. 1-17.

[59] Al-Qerem A. “An efficient machine-learning model based on data augmentation for pain intensity recognition”. En: *Egyptian Informatics Journal* 21.4 (2020), págs. 241-257.

[60] Qi Y. y col. “MOEA/D with adaptive weight adjustment”. En: *Evolutionary computation* 22 (2013).

[61] Radford A., Metz L. y Chintala S. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. En: *arXiv preprint* arXiv:1511.06434 (2016). Accedido en mayo 2022.

[62] Rossum G. Van. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[63] Ruthotto L. y Haber E. “An introduction to deep generative modeling”. En: *GAMM-Mitteilungen* 44 (2021).

[64] Sajjadi M. y col. “Assessing Generative Models via Precision and Recall”. En: NIPS’18. Curran Associates Inc., 2018, págs. 5234-5243.

[65] Sampath V. y col. “A survey on generative adversarial networks for imbalance problems in computer vision tasks”. En: *Journal of Big Data 8, 27* (2021).

[66] Schaffer J. “Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms (Artificial Intelligence, Optimization, Adaptation, Pattern Recognition)”. Tesis doct. 1984.

[67] Schmiedlechner T. y col. “Lipizzaner: A System That Scales Robust Generative Adversarial Network Training”. En: *arXiv preprint* arXiv:1811.12843 (2018). Accedido en mayo 2022.

[68] Schmiedlechner T. y col. “Towards Distributed Coevolutionary GANs”. En: *arXiv preprint* arXiv:1807.08194 (2018). Accedido en mayo 2022.

[69] Sethia A., Patel R. y Raut P. "Data Augmentation using Generative models for Credit Card Fraud Detection". En: *4th International Conference on Computing Communication and Automation*. 2018.

[70] Shaked M. y Shanthikumar G. "Univariate Stochastic Orders". En: *Stochastic Orders*. 2007, págs. 3-79.

[71] Shapiro S. y Wilk M. "An analysis of variance test for normality (complete samples)". En: *Biometrika* 52.3-4 (1965), págs. 591-611.

[72] Sheng P., Yang Z. y Qian Y. "GANs for Children: A Generative Data Augmentation Strategy for Children Speech Recognition". En: *IEEE Automatic Speech Recognition and Understanding Workshop*. 2019.

[73] Sheng P. y col. "Data Augmentation using Conditional Generative Adversarial Networks for Robust Speech Recognition". En: *11th International Symposium on Chinese Spoken Language Processing*. 2018, págs. 121-125.

[74] Shorten C. y Khoshgoftaar T. "A survey on Image Data Augmentation for Deep Learning". En: *Journal of Big Data* 6.1 (2019), págs. 1-45.

[75] Srinivas N. y Deb K. "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms". En: *Evolutionary Computation* 2.3 (1994), págs. 221-248.

[76] Toutouh J., Esteban M. y Nesmachnow S. "Parallel/Distributed Generative Adversarial Neural Networks for Data Augmentation of COVID-19 Training Images". En: *Communications in Computer and Information Science*. 2021, págs. 162-177.

[77] Virtanen P. y col. "SciPy 1.0: fundamental algorithms for scientific computing in Python". En: *Nature Methods* 17.3 (2020), págs. 261-272.

[78] Waheed A. y col. "CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection". En: *IEEE Access* 8 (2020), págs. 91916-91923.

[79] Waheed A. y col. "CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection". En: *IEEE Access* 8 (2020), págs. 91916-91923.

[80] Wang C. y col. "Evolutionary Generative Adversarial Networks". En: *Transactions on Evolutionary Computation* 23.6 (2019), págs. 921-934.

[81] Wang G. y col. “Generative Adversarial Network (GAN) Based Data Augmentation for Palmprint Recognition”. En: *Digital Image Computing: Techniques and Applications*. 2018.

[82] Wilcoxon F. “Individual Comparisons by Ranking Methods”. En: *Biometrics Bulletin* 1.6 (1945), pág. 80.

[83] Yang Y. y col. “Generative Adversarial Networks based X-vector Augmentation for Robust Probabilistic Linear Discriminant Analysis in Speaker Verification”. En: *11th International Symposium on Chinese Spoken Language Processing*. 2018.

[84] Yangru H. y col. “Towards Imbalanced Image Classification: A Generative Adversarial Network Ensemble Learning Method”. En: *IEEE Access* 8 (2020), págs. 88399-88409.

[85] Yi L. y Mak M. “Adversarial Data Augmentation Network for Speech Emotion Recognition”. En: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. 2019.

[86] Yoo A., Jette M. y Grondona M. “SLURM: Simple Linux Utility for Resource Management”. En: *Job Scheduling Strategies for Parallel Processing*. 2003, págs. 44-60.

[87] Yu P. “Cone convexity, cone extreme points, and nondominated solutions in decision problems with multiobjectives”. En: *Journal of Optimization Theory and Applications* 14.3 (1974), págs. 319-377.

[88] Yue Z. y col. “Data Augmentation of X-Ray Images in Baggage Inspection Based on Generative Adversarial Networks”. En: *IEEE Access* 8 (2020), págs. 86536-86544.

[89] Zeiler M. y Fergus R. “Visualizing and Understanding Convolutional Networks”. En: *arXiv preprint* arXiv:1311.2901 (2013). Accedido en mayo 2022.

[90] Zhang Q. y Li H. “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition”. En: *IEEE Transactions on Evolutionary Computation* 11.6 (2007), págs. 712-731.

[91] Zhang W. y col. “Machinery fault diagnosis with imbalanced data using deep generative adversarial networks”. En: *Measurement* 152 (2020), pág. 107377.

[92] Zhu J. y col. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. En: *IEEE International Conference on Computer Vision*. 2017, págs. 2242-2251.

[93] Zhuang Z. y col. “Machine-Learning-based Alarm Prediction with GANs-based Self-Optimizing Data Augmentation in Large-Scale Optical Transport Networks”. En: *International Conference on Computing, Networking and Communications*. 2020, págs. 294-298.

[94] Zitzler E. *Evolutionary algorithms for multiobjective optimization: methods and applications*. Shaker Verlag, 1999.

[95] Zitzler E., Laumanns M. y Thiele L. “SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization”. En: *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*. International Center for Numerical Methods in Engineering, 2001, págs. 95-100.

APÉNDICES

Apéndice 1

Bibliotecas utilizadas

1.1. Bibliotecas para aprendizaje profundo

La biblioteca Lipizzaner fue desarollada en el lenguaje Python[1] utilizando el framework de código abierto para desarrollo de aprendizaje automático PyTorch[2]. Con este framework se definen las arquitecturas de las redes neuronales así como las operaciones de entrenamiento sobre las mismas en unidades de procesamiento gráfico. En específico, las transformaciones sobre las imágenes y algunos de los conjuntos de datos se obtienen desde el paquete torchvision[3].

1.2. Bibliotecas para comunicación

La comunicación entre vecindarios fue originalmente desarrollada en Flask[4] creando un servidor para cada vecindario y disponibilizando una API REST. Mientras tanto, el manejo de recursos se realizaba manualmente externamente desde el centro de cómputo utilizado. La adaptación para utilizar Message Passing Interface (MPI) y poder manejar tanto la asignación de recursos como la comunicación entre vecindarios se desarrolló utilizando la biblioteca MPI4Py[5] con el fin de encapsular toda la lógica en único lenguaje.

[1] https://www.python.org/
[2] https://pytorch.org/
[3] https://pytorch.org/vision/
[4] https://flask.palletsprojects.com/
[5] https://mpi4py.readthedocs.io/

1.3. Bibliotecas para análisis y manipulación de datos

Tanto en las etapas de prototipado de soluciones, implementaciones finales y en el posterior análisis de datos se utilizaron las bibliotecas NumPy[6] y Pandas[7] para el manejo de datos.

1.4. Bibliotecas para visualización de datos

Todas las figuras presentando resultados en este trabajo fueron trazadas utilizando las bibliotecas Matplotlib[1] y Seaborn[2].

[6]https://numpy.org/
[7]https://pandas.pydata.org/
[1]https://matplotlib.org/
[2]https://seaborn.pydata.org/

Apéndice 2

Resultados experimentales

	FID	TVD	Cubrimiento	Densidad	Precisión	Exhaustividad
FID	1.00	0.97	-0.96	0.88	0.79	-0.90
TVD	0.97	1.00	-0.93	0.83	0.79	-0.81
Cubrimiento	-0.96	-0.93	1.00	-0.82	-0.75	0.91
Densidad	0.88	0.83	-0.82	1.00	0.77	-0.87
Precisión	0.79	0.79	-0.75	0.77	1.00	-0.68
Exhaustividad	-0.90	-0.81	0.91	-0.87	-0.68	1.00

Tabla 2.1: Matriz de correlaciones entre las métricas de evaluación. Se estudian dos familias de métricas, las enfocadas en la fidelidad: FID, Densidad y Precisión; y las enfocadas en diversidad: TVD, Cubrimiento y Exhaustividad. Para calcular estos valores, se utilizaron los datos de las ejecuciones realizadas en el proceso de configuración parametrica con el algoritmo NSGA-2.

	FID	TVD	Cubrimiento	Densidad	Precisión	Exhaustividad
FID	1.00	0.96	-0.92	0.68	0.63	-0.77
TVD	0.96	1.00	-0.86	0.78	0.70	-0.66
Cubrimiento	-0.92	-0.86	1.00	-0.48	-0.55	0.75
Densidad	0.68	0.78	-0.48	1.00	0.72	-0.48
Precisión	0.63	0.70	-0.55	0.72	1.00	-0.42
Exhaustividad	-0.77	-0.66	0.75	-0.48	-0.42	1.00

Tabla 2.2: Matriz de correlaciones entre las métricas de evaluación. Se estudian dos familias de métricas, las enfocadas en la fidelidad: FID, Densidad y Precisión; y las enfocadas en diversidad: TVD, Cubrimiento y Exhaustividad. Para calcular estos valores, se utilizaron los datos de las ejecuciones realizadas en el proceso de configuración parametrica con el algoritmo FV-MOEA.

	FID	TVD	Cubrimiento	Densidad	Precisión	Exhaustividad
FID	1.00	0.98	-0.96	0.90	0.80	-0.89
TVD	0.98	1.00	-0.95	0.91	0.82	-0.84
Cubrimiento	-0.96	-0.95	1.00	-0.82	-0.76	0.88
Densidad	0.90	0.91	-0.82	1.00	0.85	-0.81
Precisión	0.80	0.82	-0.76	0.85	1.00	-0.72
Exhaustividad	-0.89	-0.84	0.88	-0.81	-0.72	1.00

Tabla 2.3: Matriz de correlaciones entre las métricas de evaluación. Se estudian dos familias de métricas, las enfocadas en la fidelidad: FID, Densidad y Precisión; y las enfocadas en diversidad: TVD, Cubrimiento y Exhaustividad. Para calcular estos valores, se utilizaron los datos de las ejecuciones realizadas en el proceso de configuración parametrica con el algoritmo MOEA/D.

	Densidad	Cubrimiento	FID
Lipizzaner base	1.00	0.10	1.00
MOEA/D	0.00	1.00	0.00
FV-MOEA	0.04	0.00	0.81
NSGA-2	0.23	0.65	0.40

Tabla 2.4: Resultados de los experimentos de cada algoritmo y Lipizzaner base para las métricas de evaluación de la evaluación de desempeño. Los datos están normalizados por métrica entre 0 y 1.