UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CARRERA DE CIENCIA DE LA COMPUTACIÓN

Diseño de dispositivos nanofotónicos resilientes a errores de fabricación usando algoritmos de optimización y computación de alto desempeño

TESIS

Para optar el título profesional de Licenciado en Ciencia de la Computación

AUTOR:

José Leonidas García Gonzales

ASESOR

Jorge Luis Gonzalez Reaño

Lima - Perú

2 de mayo de 2022

Índice general

Pág.

Índice de tablas

Índice de figuras

Capítulo 1

Motivación y Contexto

1.1 Introducción

La fotónica es la ciencia que estudia la generación, detección y manipulación de la luz. Los principales beneficios que ofrece son (Shen *et al.*, 2019): (i) elevado ancho de banda en comunicaciones, (ii) bajo consumo energético, (iii) interconexiones ópticas independientes de la distancia. Asimismo, existen propuestas que aprovechan estos beneficios, por ejemplo: (i) interconexiones ópticas en centrales de datos (Shen *et al.*, 2019), (ii) redes neuronales ópticas (Shen *et al.*, 2017) e (iii) internet de las cosas (Li *et al.*, 2021).

Un problema presentado en el Top500 sistemas de computación de alto desempeño (HPC, por sus siglas en Inglés) es que desde el año 2010, el ratio entre el ancho de banda entre nodos y el poder de procesamiento por nodo ($byte/FLOP$) ha decrecido en un factor de seis. Es decir, se está llegando a un punto donde la capacidad para interconectar nodos está limitando el desempeño de sistemas HPC en programas que hacen uso extensivo de transferencias de memoria. Ante este problema, los avances en la fotónica en silicio (SiP) integrada se presenta como una de las principales alternativas de solución. Esta propuesta puede realizar interconexiones a distancias del orden de metros, manteniendo un elevado ancho de banda y bajo consumo energético (Anderson *et al.*, 2018, Shen *et al.*, 2019).

A pesar del avance en el diseño e integración de dispositivos SiP, en términos de cantidad de dispositivos por chip, la fotónica integrada aún se encuentra en la misma etapa de expansión que tenía la electrónica en los años 1970s (Lukas Chrostowski, 2010). Sin embargo, ya existen procesos de fabricación estándar en *foundries* para fabricar chips SiP,

a un precio accesible, utilizando procesos CMOS a través del *process design kit* (PDK) (Bogaerts and Chrostowski, 2018). De este modo, se facilita el desarrollo de esta tecnología.

En el área de SiP integrada, la alta densidad de fabricación es un desafío porque se requiere mantener eficiencia en el chip a nivel de sistema fotónico; por ello, se está buscando optimizar dispositivos fundamentales que lo compongan (Vuckovic, 2019). Para esto existen dos estrategias principales: (i) diseño tradicional (Huang and Ouyang, 2018, Hughes and Fan, 2016, Song and Xie, 2008) y (ii) diseño inverso (Gregory *et al.*, 2015, Malheiros-Silveira and Delalibera, 2020, Su *et al.*, 2020).

La figura 1.1 presenta una comparación de los dos tipos de diseño (tradicional e inverso) en dos dispositivos: (i) *bend-90°* y (ii) *wavelength demultiplexer* de dos canales (WDM). Como se observa en la figura 1.1, en el diseño tradicional (izquierda) se define el dispositivo con geometrías simples que permiten obtener funciones analíticas de sus propiedades físicas (Hughes and Fan, 2016, Song and Xie, 2008). Esto se realiza para poder optimizar la función obtenida a partir de los parámetros que la definan. Dicha optimización se suele ejecutar haciendo un barrido de los parámetros, con algoritmos genéticos o usando *particle swarm optimization*. Esta técnica es un enfoque simple, pero que ha obtenido buenos resultados (Su *et al.*, 2020).

Sin embargo, existen tres grandes inconvenientes con el diseño tradicional. Primero, solamente se explora una pequeña fracción de todos los posibles diseños. Segundo, por lo general no es conocido el límite de rendimiento del dispositivo (Molesky *et al.*, 2018). Tercero, al trabajar en la escala de nanómetros, existen casos como el *bend-90°* y WDM que presentan un bajo rendimiento (Su *et al.*, 2020).

Por el otro lado, en el diseño inverso, mostrado en la figura 1.1 (derecha), las geometrías resultantes no están limitadas a diseños intuitivos o regulares. Esta técnica busca hacer una mayor exploración de todos los posibles diseños. Para ello, se define geometrías arbitrarias y se usa simulaciones computacionales para determinar las propiedades

(a) *Bend* con diseño tradicional.

(b) *Bend* obtenido con diseño inverso. Extraído de (Su *et al.*, 2020).

(c) WDM con diseño tradicional.

(d) WDM obtenido con diseño inverso. Extraído de (Su *et al.*, 2020).

FIGURA 1.1: Diseños tradicionales y obtenidos a partir de diseño inverso de un *bend* y un WDM.

físicas del dispositivo, propiedades que son usadas para formular una función objetivo. Finalmente, se aplican algoritmos de optimización para encontrar valores óptimos de esta función (Molesky *et al.*, 2018, Su *et al.*, 2020). Para una descripción detallada de los distintos algoritmos de optimización comúnmente empleados, por favor revisar Campbell *et al.* (2019), Elsawy *et al.* (2020), Schneider *et al.* (2019).

El diseño inverso ha logrado conseguir mejores resultados que los obtenidos por el diseño tradicional por lo que ha ganado interés en el área de fotónica durante los últimos

20 años (Campbell *et al.*, 2019, Molesky *et al.*, 2018, Su *et al.*, 2018). Sin embargo, existen cuatro grandes desafíos con este planteamiento: (i) el espacio de búsqueda es exponencial (Vuckovic, 2019), (ii) las simulaciones computacionales son costosas en términos de tiempo y memoria (Kudyshev *et al.*, 2020), (iii) el espacio de búsqueda es no convexo (Su *et al.*, 2018) y (iv) no todos los diseños son fabricables (Su *et al.*, 2020).

Y aún cuando una cantidad considerable de investigaciones estudian dispositivos SiP (Malheiros-Silveira and Delalibera, 2020, Su *et al.*, 2018), sigue siendo necesario estudiar dispositivos SiP fundamentales para conseguir chips de SiP integrada densos y eficientes. Los avances en esta área benefiarán indirectamente al desarrollo de Ciencia de la Computación con mejoras en el *hardware* utilizado para ejecutar eficientemente, por ejemplo, programas de inteligencia artificial o de computación de alto desempeño.

De este modo, el presente trabajo se centra en estudiar dos dispositivos SiP fundamentales: (i) *bend-90°* y (ii) *wavelength demultiplexer* de dos canales. De aquí en adelante nos referiremos a estos simplemente como *bend* y WDM. Por lo tanto, el objetivo de esta tesis es aplicar el conocimiento en computación para encontrar diseños de estos dispositivos con eficiencias mayores al 90 % y resilientes a errores de fabricación, trabajando en la escala de nanómetros. Con el fin de solucionar este problema se empleará diseño inverso para encontrar geometrías con las eficiencias deseadas, para ello se realizará simulaciones computacionales con cinco algoritmos de optimización populares. De este modo, se espera obtener diseños cuyas simulaciones presenten las características señaladas y que, potencialmente, muestren propiedades similares al fabricarse.

El presente documento está organizado de la siguiente manera:

El capítulo 1 brinda una introducción al tema de investigación, describe el problema a detalle, justifica la relevancia de resolverlo, define los objetivos y señala los aportes del trabajo.

El capítulo 2 desarrolla conceptos fundamentales en fotónica necesarios para entender el resto del documento.

El capítulo 3 presenta una revisión del estado del arte en diseño inverso para optimizar un *bend* y WDM.

El capítulo 4 desbribe la metodología usada en la investigación.

Por último, el capítulo 5 muestra los resultados obtenidos con los distintos casos de estudio.

1.2 Descripción del Problema

Para poder describir el funcionamiento de un dispositivo se calcula la distribución del campo eléctrico, para ello se resuelven las ecuaciones de Maxwell (Schneider *et al.*, 2019). Una forma de encontrar la solución a estas ecuaciones en cualquier geometría es utilizando un método numérico llamado diferencias finitas en el dominio de frecuencias (FDFD) (Su *et al.*, 2020). Con este planteamiento se selecciona una región rectangular a optimizar y se la divide en $n \times m$ píxeles como si fuera una imagen, ver Figura 1.2. Luego, cada píxel se rellena con dos posibles materiales: óxido de silicio (SiO_2) o silicio (Si) (Molesky *et al.*, 2018).

El diseño inverso comienza definiendo los requerimientos del dispositivo para luego tratar de buscar entre los $2^{n \times m}$ posibles diseños algún candidato que se adapte a lo que se busca (Molesky *et al.*, 2018, Su *et al.*, 2020). Como prueba de concepto, trabajos como el de Malheiros-Silveira and Delalibera (2020) parametrizaron $2^{10 \times 10}$ posibles geometrías. Así, se presentan algunas dificultades con esta estrategia:

1. No es viable evaluar todos los posibles diseños por haber un número excesivamente elevado de ellos (Vuckovic, 2019).

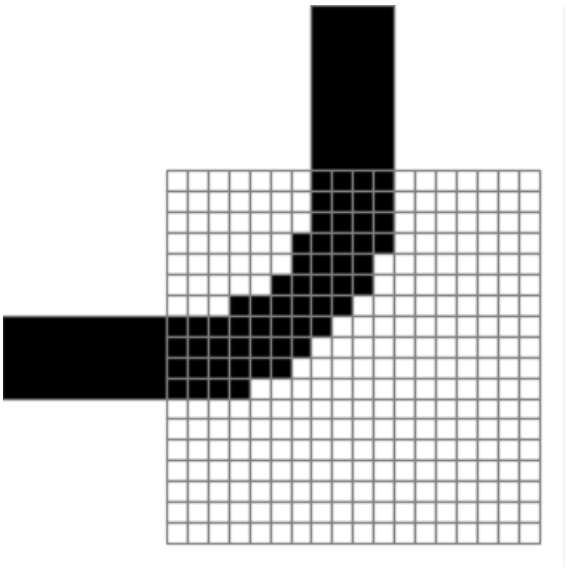

FIGURA 1.2: *Bend* con una región de diseño discretizada en 18×18 píxeles. Cada píxel negro representa la presencia de Si y cada píxel blanco de SiO_2.

2. Las simulaciones computacionales son muy costosas en términos de memoria y tiempo (Kudyshev *et al.*, 2020).

3. El espacio de búsqueda es no convexo (Su *et al.*, 2018).

4. No todos los diseños son fabricables por limitaciones físicas (Su *et al.*, 2020).

5. Cada dispositivo es una clase distinta de problema, es decir, no necesariamente funcionará la misma estrategia para cada dispositivo (Molesky *et al.*, 2018).

Además, la fabricación viene con otros desafíos, principalmente:

1. Errores de precisión (Piggott *et al.*, 2017).

2. Sensibilidad ante cambios de temperatura (Vuckovic, 2019).

Considerando las anteriores dificultades, el problema es usar diseño inverso y encontrar geometrías que muestren buen desempeño en simulaciones computacionales y que

puedan asegurar mantener un óptimo funcionamiento al ser fabricados. Este problema se estudiará para dos dispositivos nanofotónicos (i) *bend* y (ii) WDM.

1.3 Justificación

El *bend* y WDM son dispositivos SiP fundamentales que tienen aplicación directa, por ejemplo, en sistemas HPC (Shen *et al.*, 2017). Así, las mejoras de estos ayudará indirectamente al desarrollo de Ciencia de la Computación brindando, potencialmente, un mejor *hardware* para programas de inteligencia artificial, HPC, entre otros. Por otro lado, desde el punto de vista computacional, este problema es interesante porque ya hay estrategias computacionales conocidas para resolverlo, desde algoritmos evolutivos (Hansen, 2016) hasta redes neuronales (Goodfellow *et al.*, 2015) y *depth learning* (Malkiel *et al.*, 2018). Además, debido al alto costo computacional de las simulaciones (Schneider *et al.*, 2019), el trabajo requiere de computación de alto desempeño. Así, es probable que se pueda obtener buenos resultados en la investigación aplicando el conocimiento ya existente en computación.

1.4 Objetivos

- Diseñar un *bend* y WDM con eficiencias mayores al 90 % y resiliente a errores de fabricación.
    - Seleccionar una estrategia de parametrización que asegure facilidad de fabricación.
    - Definir una función objetivo que encapsule las propiedades buscadas en cada dispositivo.
    - Encontrar geometrías con valores óptimos de la función objetivo en simulaciones computacionales.

    - Encontrar geometrías resilientes a posibles errores de fabricación de dilatación o contracción.

- Comparar el desempeño y la convergencia de cinco algoritmos de optimización populares usados para optimizar dispositivos nanofotónicos.

    - Implementar el proceso de optimización de los dispositivos asegurando aprovechar los recursos CPU de un *cluster*.
    - Implementar el proceso de optimización de los dispositivos asegurando aprovechar los recursos GPU de un *cluster*.

Capítulo 2

Marco Teórico

En el presente capítulo se introducen conceptos fundamentale relacionados al diseño inverso de dispositivos fotónicos. Para ello se desarrolla seis secciones. Primero, se describen las propiedades físicas de interés de un *bend* y WDM. Segundo, se explica como parametrizar la región de diseño de estos dispositivos. Tercero, se detalla los pasos necesarios para poder simular computacionalmente los diseños realizados. Cuarto, se explica la estrategia de optimización a seguir con la parametrización señalada. Quinto, se describen tres algoritmos que se usaran en el presente trabajo: (i) algoritmos genéticos (GA), (ii) *particle swarm optimization* (CMA), (iii) *covariance matrix adaptation evolution strategy*(CMA-ES). Finalmente, se expone que transformaciones se puede aplicar dentro de la estrategia a seguir.

2.1 Dispositivos de estudio

2.1.1 *Bend*

Un *bend* es un dispositivo fotónico que se encarga de guiar un haz de ondas para que gire.

En general, al estudiar dispositivos fotónicos es de especial interés la distribución del campo eléctrico (E). Este campo se descompone en una componente transversal eléctrica (TE) y en una componente transversal magnética (TM) de acuerdo a la ecuación

$$E = E^{TE} + E^{TM}, \tag{2.1}$$

donde E^{TE} es la componente paralela al dispositivo y E^{TM} es el componente restante (Hohenester, 2020).

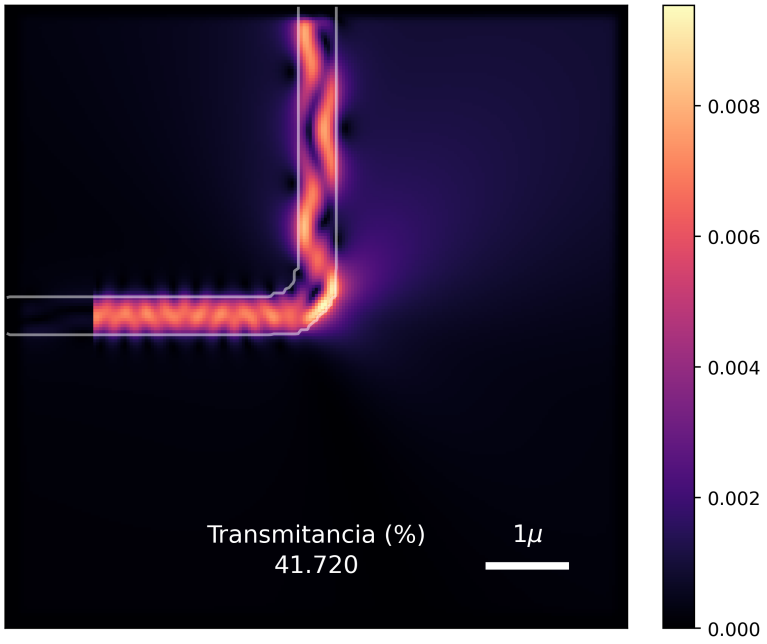

FIGURA 2.1: Intensidad de campo eléctrico (E^{TE}) para un *bend-90°* de radio interno de 0.25 μm.

La intensidad de estos campos es de especial interés pues nos dan una idea del rendimiento del dispositivo. Por ejemplo, en la figura 2.1 se muestra la intensidad del campo E^{TE} cuando un haz de luz está pasándo por el *bend*. Como podemos observar, parte del campo (representando por color morado) está fuera del dispositivo. Así se visualiza de forma gráfica la pérdida de energía, lo cual es un indicador de mal rendimiento.

Por otro lado, para evaluar el desempeño de un dispositivo de forma numérica se suele calcular la transmitancia (T) como la relación entre la intensidad del haz que sale del dispositivo (I) con la intensidad con la que entra (I_0) (Su *et al.*, 2020). Esto se expresa mediante la ecuación

$$T = \frac{I}{I_0}. \tag{2.2}$$

Seguidamente, sea p los parámetros que caracterizan a un *bend*, definimos la función objetivo (f_{obj}) para este dispositivo, también conocido en el área como figura de mérito (FOM), mediante la siguiente ecuación (Su *et al.*, 2020):

$$f_{obj}(p) = max\left\{T(p)\right\}. \tag{2.3}$$

Así, dentro de todas las posibles combinaciones de los parámetros p, la equación 2.3 busca encontrar aquellos diseños con la mayor transmitancia. Distintas estrategias para definir los parámetros p se presentan en la sección de **parametrización** y los procedimientos para optimizar la función f_{obj} se presentan en la sección de **algoritmos de optimización**.

2.1.2 *Wavelength Demultiplexer* de dos canales (WDM)

Un WDM es un dispositivo fotónico que se encarga de guiar un haz de ondas de acuerdo a su longitud de onda. Así, estos suelen trabajar con dos longitudes de onda y guían las de un tipo por la guía de onda superior y las de otro tipo por la guía de onda inferior.

Similar al caso del *bend* se estudia su campo eléctrico y transmitancia para medir el rendimiento de estos dispositivos.

Basándonos en Su *et al.* (2020), conviene definir su FOM como

$$f_{obj}(p) = max\left\{g_0(p,0)^2 + (1-g_0(p,1))^2 + g_1(p,1)^2 + (1-g_1(p,0))^2\right\}, \tag{2.4}$$

donde $g_0(p,i)$ representa la transmitancia asociada a la parametrización p en la guía de onda i para una longitud de onda de $1400nm$ y $g_1(p,i)$ representa su análogo de $g_0(p,i)$ para una longitud de onda de $1550nm$.

La ecuación 2.4 busca maximizar la transmitancia por la guía de onda superior y minimizarla para la guía de onda inferior cuando se recibe una longitud de onda de $1400nm$ y lo contrario para una longitud de onda de $1550nm$.

Un dispositivo similar al WDM es un *splitter*. Este dispositivo tiene la misma geometría, pero lo que recibe lo trata de dividir en la misma proporción para ambas guías de onda (superior e inferior).

2.2 Parametrización

Tanto para el *bend* como para el WDM se define una región de diseño mediante ciertos parámetros que puedan mapear un gran conjunto de dispositivos a considerar. Una de la estrategias más populares para esta tarea es usar parametrización basada en topología, esta consiste en definir una región rectangular en forma de matriz y variar la permitividad dentro de cada celda (Molesky *et al.*, 2018).

Por ejemplo, en la figura 2.2 se ha definido una región cuadrangular de diseño que podemos ver como una imagen de 18×18 píxeles. Aquí estamos usando los píxeles negros para representar la presencia de Si, los blancos para indicar que hay SiO_2 y los grises para materiales (no necesariamente reales) que tengan una permitividad intermedia entre el Si y SiO_2. Matemáticamente expresamos esto como

$$\varepsilon(x,y) = \varepsilon_{Si} + (1 - \lambda_{x,y})\varepsilon_{SiO_2} \quad \lambda_{x,y} \in [0,1], \lambda_{x,y} \in \mathbb{R}, \tag{2.5}$$

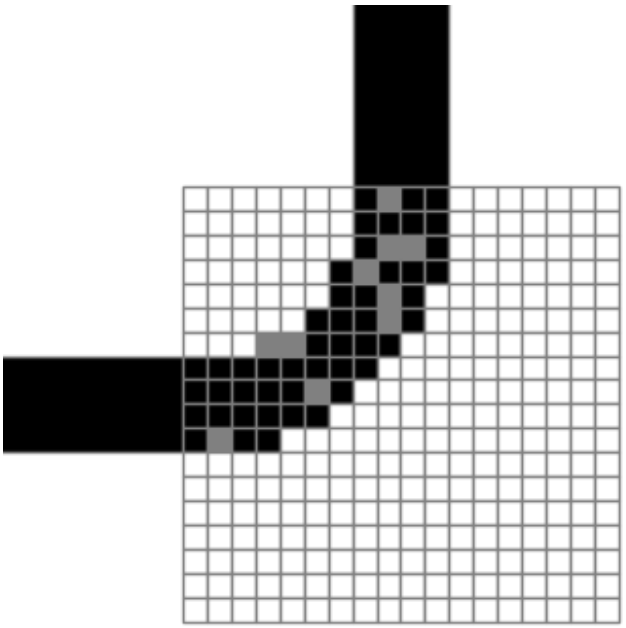

FIGURA 2.2: Parametrización por píxeles para un *bend-90°*.

donde $\varepsilon_{Si} = 3.48$ es la permitividad del Si, $\varepsilon_{SiO_2} = 1.44$ es la permitividad del $SiO2$ y $\lambda_{x,y}$ es un parámetro asociado al píxel ubicado en la fila x y columna y. Con esta ecuación se mapea el intervalo $[0, 1]$ con el intervalo $[1.44, 3.48]$. Esto se realiza para determinar la permitividad que hay en la ubicación de cada píxel y así ser capaces de simular las ecuaciones de Maxwell en el diseño. Con esta parametrización obtenemos una cantidad infinita de posibles dispositivos, mas solo nos interesan aquellos donde $\lambda_{x,y}$ es entero, pues en caso contrario un píxel se mapea a la permitividad de un material potencialemente desconocido, lo cual lo volvería infabricable.

2.3 Simulación

Una vez tenemos definido un dispositivo con regiones fijas (guías de onda) y una región de diseño (descrita por la parametrización), es necesario incorporar tres elementos adicionales (Su *et al.*, 2020):

- **Fuente:** Suele representarse como un cuadrado en el eje XZ o YZ. Simula la emición de un haz de ondas por el diseño.
- **Monitores:** Suelen representarse como cuadrados en el eje XZ o YZ. Capturan información en su ubicación (e.g. valores del campo eléctrico).
- **PML:** Representan las condiciones de frontera en la simulación de las ecuaciones de Maxwell. Se utilizan para limitar el espacio donde se deberá realizar las simulaciones computacionales.

Dos librerías de Python de código abierto que permiten hacer las simulaciones de las ecuaciones de Maxwell en diseños como los descritos son MEEP (Oskooi *et al.*, 2010) y SPINS (Su *et al.*, 2020). Una evaluación cualitativa de sus funcionalidades puede ser vista en la tabla 2.1.

Librería	Usabilidad	Eficiencia	Bugs	Funcionalidad
MEEP	Difícil	Alta	Elevados	Extensa
SPINS	Moderada	Moderada-Alta	Pocos	Básica-necesaria

TABLA 2.1: Evaluación cualitativa de las librerías MEEP y SPINS.

Ambas librerías utilizan métodos finitos para discretizar el espacio de simulación y permiten realizar simulaciones en 2D y 3D. Particularmente, podemos definir un parámetro dx para indicar que tan preciso serán nuestros resultados. En general, cuanto más pequeño sea el valor de dx los resultados serán más precisos, pero la cantidad de memoría y el tiempo de simulación se incrementarán considerablemente.

Adicionalmente, hay dos diferencias importantes entre estas librerías. Primero, en una simulación SPINS puede obtener resultados en solo una frecuencia mientras que MEEP puede obtener resultados en un rango de frecuencias de forma eficiente. Segundo, MEEP realiza simulaciones en 3D utilizando MPI para aprovechar recursos *multi-cores*, por otro lado SPINS permite usar uno o más GPUs para realizar estos cálculos.

2.4 Estrategia de optimización

Como fue descrito en la sección de **parametrización**, podemos tener diseños que no puedan ser fabricados. Para evitar esto se necesita obtener que $\lambda_{x,y}$ de la ecuación 2.5 tenga valores enteros. Sin embargo, al optimizar la función f_{obj} no podemos asegurar esta condición. Ante esta dificultad, Su *et al.* (2020) trabaja en dos etapas:

1. **Optimización continua**

   En esta etapa se optimiza la función f_{obj} sin imponer ninguna restricción. En la siguiente sección, **algoritmos de optimización**, se detallará algoritmos que suelen usarse para este fin.

2. **Optimización discreta**

   Se utiliza el resultado de la optimización continua como punto inicial para el algoritmo de optimización que se escoja. Luego, se trabaja por iteraciones. En cada iteración se aplica una transformación a cada diseño antes de evaluar su FOM. Esta transformación se escoge de tal manera que asegure que una parametrización vaya convergiendo a un diseño fabricable, es decir, a tener $\lambda_{x,y} = 0$ o $\lambda_{x,y} = 1$, más detalles en la sección **transformaciones**. Así, la idea de realizarlo por iteraciones es para ir discretizando nuestro diseño de forma suave e intentando mantener un buen valor del FOM. Por ello es crucial usar el resultado de una iteración como punto inicial de la próxima

2.5 Algoritmos de optimización

Como observamos en la anterior sección, es necesario escoger algún algoritmo que nos permita optimizar la función f_{obj}. Por este motivo, se presentan los siguientes tres algoritmos que serán usados en el presente trabajo.

2.5.1 *Algoritmos genéticos* (GA)

Como se describe en el algoritmo 1 (Kochenderfer and Wheeler, 2019), la idea es comenzar generando una población (*population*) de n individuos representados por p parámetros, línea 1. Los siguientes tres pasos se ejecutan por k iteraciones. Primero, se realiza un proceso de selección para obtener los mejores individuos (*parents*), línea 3. Segundo, los seleccionados se encargan de producir la nueva generación (*children*), línea 4. Tercero, la nueva generación muta obteniendo nuevas características, línea 5.

Algorithm 1: Estructura de un algoritmo genético

```
1 population = generate_population(n, p)
2 for t = 0; t < k; t++ do
3     parents = select(population)
4     children = crossover(population, parents)
5     population = mutation(children)
```

Entrando en más detalle, para nuestro caso tenemos:

- $generate\_population(n, p)$: retorna n vectores de dimensión p con valores aleatorios en $U(0, 1)$.
- $select(population)$: retorna $number\_selected\_GA$ individuos de acuerdo a la probabilidad $prob_i$ dada por la ecuación

$$prob_i = \frac{f_{obj}^{(i)} - min(f_{obj})}{\sum_j (f_{obj}^{(j)} - min(f_{obj}))}, \tag{2.6}$$

  donde $f_{obj}^{(i)}$ está asociado con el $i-$ésimo individuo.
- $crossover(population, parents)$: retorna n vectores de dimensión p. El $i-$ésimo vector es la combinación de dos padres aleatorios pa_i y pb_i seleccionados de $parents$

y su $j-$ésimo parámetro es escogido con igual probabilidad entre el $j-$ésimo parámetro de $parents[pa_i]$ y $parents[pb_i]$.

- $mutation(children)$: retorna lo que recibe, pero a cada atributo de cada individuo se le agrega un valor en $U(-range\_GA, range\_GA)$ con probabilidad de 1 sobre 2.

2.5.2 *Particle Swarm Optimization* (PSO)

Podemos pensar este algoritmo como un caso especial del algoritmo 1 (Kochenderfer and Wheeler, 2019, Prosopio-Galarza *et al.*, 2019). La idea es visualizar el $i-$ésimo individuo como una partícula definida por: (i) su posición $x^{(i)}$ (el vector p-dimensional asociado al $i-$ésimo individuo), (ii) su velocidad $\nu^{(i)}$ (un número real) y (iii) la mejor posición encontrada hasta el momento.

Cada particula acumula velocidad en una dirección favorable dada por: (i) la mejor posición encontrada hasta el momento por ella y (ii) la mejor posición encontrada por la población completa. Como consecuencia, los individuos se pueden mover independientemente de perturbaciones locales. Adicionalmente, agregando caminos aleatorios las particulas incorporan comportamientos impredecibles que puede permitirles encontrar potenciales mejores direcciones.

Entrando en más detalle, siguiendo la estructura del algoritmo 1, tenemos:

- $generate\_population(n, p)$: retorna n particulas con cada uno de sus parámetros tomando valores aleatorios en $U(0, 1)$.
- $select(population)$: retorna la particula con el mejor f_{obj}
- $crossover(population, parents)$: retorna la población luego de aplicar

$$x^{(i)} \leftarrow x^{(i)} + \nu^{(i)} \tag{2.7}$$

$$\nu^{(i)} \leftarrow \omega\nu^{(i)} + c_1 r_1 \left(x_b^{(i)} - x^{(i)}\right) + c_2 r_2 \left(x_b - x^{(i)}\right), \tag{2.8}$$

donde x_b es la mejor posición encontrada globalmente, ω representa la tendencia de la particula de conservar su velocidad actual, c_1 y c_2 cuantifica la atracción relativa de $x_b^{(i)}$ y x_b respectivamente, y $r_1, r_2 \in U(0,1)$ representan el comportamiento impredecible.

2.5.3 *Covariance Matrix Adapatation Evolution Strategy* (CMA-ES)

La idea general de esta estrategia evolutiva, mostrada en el algoritmo 2, es mantener: (i) un vector μ p-dimensional, (ii) una matriz Σ y (iii) un número σ para ir generando n puntos aleatorios de una distribución $\mathcal{N}(\mu,\, \sigma^2\Sigma)$. Donde n es la cantidad de individuos a considerar definidos por p parámetros.

Tomar puntos de esta distribución limita el espacio de búsqueda a una hiperelipse. Luego, el algoritmo evalua puntos en esta región limitada. Usando los valores obtenidos, se puede decidir entre: (i) mover la hiperelipse a otra región del espacio de búsqueda (ii) expandir or reducir la región cubierta por la distribución. El algoritmo de CMA-ES trabaja iterativamente sobre esta idea hasta que la hiperelipse termina casi degenerándose en un punto, potencialmente un óptimo. Para una descripción más detallada del algoritmo, revisar (Hansen, 2016, Kochenderfer and Wheeler, 2019).

Entrando en más detalla del algoritmo 2 y considerando variables globales por simplicidad, podemos resumir el procedimiento en cinco pasos. En la línea 1 simplemente se repite las siguientes líneas por k iteraciones. En la iteración t, comenzamos con la línea 2 generando n puntos p-dimensionales x_i de la distribución $\mathcal{N}(\mu,\, \sigma^2\Sigma)$, donde estos son

Algorithm 2: CMA-ES

```
1 for t = 0; t < k; t++ do
2     sample() // Obtener n puntos de N(μ, σ²Σ)
3     update() // Ecuación 2.9
4     control() // Ecuación 2.10
5     adapt() // Ecuación 2.12
```

ordenados descendentemente de acuerdo al valor de f_{obj}. En la línea 3 actualizamos la media μ usando una promedio ponderado dado por

$$\mu^{(t+1)} \leftarrow \sum_{i=1}^{n} w_i x_i \tag{2.9}$$

,

donde w_i son fijos y escogidos de tal manera que proporcionen mayor contribución a los puntos con meyor f_{obj}. Esto permite mover la media μ en una dirección favorable.

Seguidamente, se necesita actualizar σ para expandir o reducir la hiperelipse en la siguiente iteración. Por este motivo, la línea 4 controla este valor mediante las ecuaciones

$$\sigma^{(t+1)} \leftarrow \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\underbrace{\left(\frac{||p_\sigma||}{\mathbb{E}||\mathcal{N}(0,\mathbf{I})||} - 1\right)}_{\text{evolution path comparison}}\right) \tag{2.10}$$

$$\mathbb{E}||\mathcal{N}(0,\mathbf{I})|| = \sqrt{2}\left(\frac{\Gamma\left(\frac{p+1}{2}\right)}{\Gamma\left(\frac{p}{2}\right)}\right) \tag{2.11}$$

donde p_σ es una variable que acumula los pasos llevados, $c_\sigma \in [0, 1]$ es una variable que determina el tiempo acumulado para p_σ y $d_\sigma \approx 1$ es un parámetro que determina el ratio de posibilidad de cambio de $\sigma^{(t+1)}$. La principal parte de la ecuación 2.10 es el término *evolution path comparison*, aquí se compara el tamaño de p_σ con su tamaño esperado bajo selección aleatoria. De esta comparación podemos controlar si el valor de σ debe incrementarse, disminuirse o permanecer igual.

Finalmente, en la línea 5 cambiamos Σ a una dirección favorable usando

$$\Sigma^{(t+1)} = \overbrace{\left(1 - c_1 c_c (1 - h_\sigma)(2 - c_c) - c_1 - c_\mu\right)\Sigma^{(t)}}^{\text{cumulative update}} + \underbrace{c_1 p_\Sigma p_\Sigma^T}_{\text{rank-one update}} + \underbrace{c_\mu \sum_{i=1}^{n} w_i' \delta^{(i)} \left(\delta^{(i)}\right)^T}_{\text{rank-}\mu\text{ update}} \quad (2.12)$$

donde $c_\mu \leq 1$ es el radio de aprendizaje para el término *rank-μ update*, $c_1 \leq 1 - c_\mu$ es el radio de aprendizaje para el término *rank-one update*, $c_c \in [0, 1]$ es el radio de aprendizaje para el término *cumulative update*, h_σ es la evaluación bajo la función unitaria usado para actualizar apropiadamente el camino evolutivo, p_Σ es un vector acumulativo usado para actualizar la matriz de covarianza, w_i' son los coeficientes de ponderación modificados y $\delta^{(i)}$ son las desviaciones seleccionadas.

En la ecuación 2.12, el primer término (*cumulative update*) mantiene información de la anterior matriz de convarianza. El segundo término (*rank-one update*) permite expandir la distribución en una dirección favorable. El tercer término (*rank-μ update*) incrementa la búsqueda en espacios donde es probable encontrar buenas soluciones. La combinación de estos tres términos actualiza Σ de tal manera que mueva la hiperelipse en una dirección favorable.

2.6 Transformaciones

La aplicación de transformaciones a un diseño se realiza con el fin de obtener dispositivos que se puedan fabricar con mayor facilidad (Su *et al.*, 2020).

Basándonos en Zhang *et al.* (2021), una manera de asegurar que los parámetros p describan un diseño más fácil de fabricar es aplicacando la función $s(p)$ descrita como

$$s(p) = \frac{\tanh(\beta \times \eta) + \tanh(\beta \times (p - \eta))}{\tanh(\beta \times \eta) + \tanh(\beta \times (1 - \eta))}, \tag{2.13}$$

donde $\eta = 0.5$ y β comienza con un valor de 1 y va incrementándose exponencialmente en cada iteración. Como se observa en la figura 2.3, la ecuación 2.13 se encarga de ir haciendo converger los valores de la parametrización a 0 o 1 de acuerdo a cual esté más cercano. Conforme aumenta el valor de β esta convergencia es más rápida.

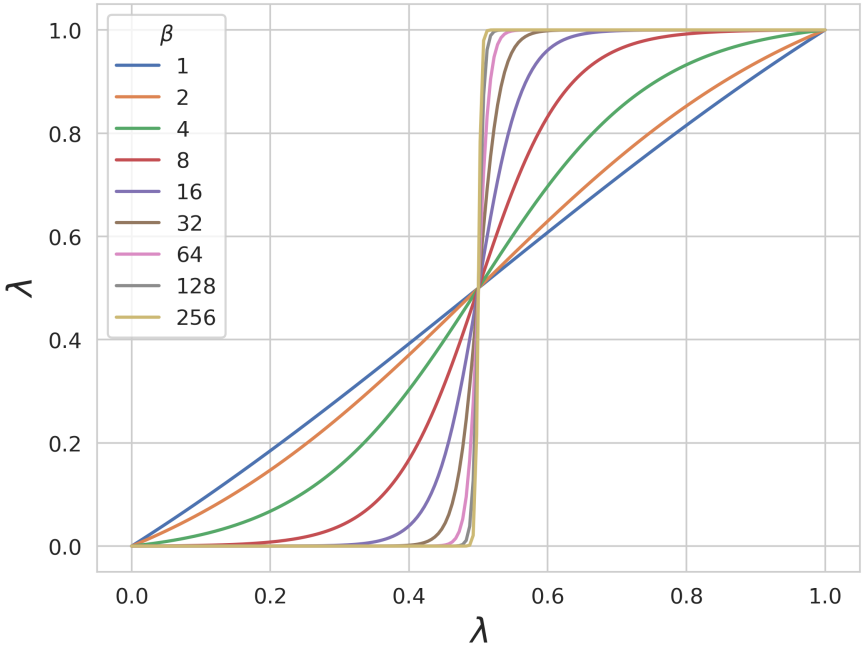

FIGURA 2.3: Función de discretización con $\eta = 0.5$ y distintos valores de β.

REFERENCIAS BIBLIOGRÁFICAS

Anderson, E., González, J., Gazman, A., Azevedo, R., and Bergman, K. (2018). Optically connected and reconfigurable gpu architecture for optimized peer-to-peer access. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '18, page 257–258, New York, NY, USA. Association for Computing Machinery.

Bogaerts, W. and Chrostowski, L. (2018). Silicon photonics circuit design: Methods, tools and challenges. *Laser & Photonics Reviews*, 12(4):1700237.

Campbell, S. D., Sell, D., Jenkins, R. P., Whiting, E. B., Fan, J. A., and Werner, D. H. (2019). Review of numerical optimization techniques for meta-device design [invited]. *Opt. Mater. Express*, 9(4):1842–1863.

Elsawy, M. M., Lanteri, S., Duvigneau, R., Fan, J. A., and Genevet, P. (2020). Numerical Optimization Methods for Metasurfaces. *Laser and Photonics Reviews*, 14(10):1–17.

Goodfellow, I. J., Vinyals, O., and Saxe, A. M. (2015). Qualitatively characterizing neural network optimization problems. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Gregory, M. D., Martin, S. V., and Werner, D. H. (2015). Improved Electromagnetics Optimization. *IEEE Antennas and Propagation Magazine*, 57(june):48–59.

Hansen, N. (2016). The CMA Evolution Strategy: A Tutorial.

Hohenester, U. (2020). *Nano and Quantum Optics: An Introduction to Basic Principles and Theory*. Graduate Texts in Physics. Springer International Publishing, 1st ed. 2020 edition.

Huang, H. and Ouyang, Z. (2018). General method for eliminating wave reflection in 2d photonic crystal waveguides by introducing extra scatterers based on interference cancellation of waves. *Optics Communications*, 406:260–270. Optoelectronics and Photonics Based on Two-dimensional Materials.

Hughes, T. W. and Fan, S. (2016). Plasmonic circuit theory for multiresonant light funneling to a single spatial hot spot. *Nano Letters*, 16(9):5764–5769.

Kochenderfer, M. J. and Wheeler, T. A. (2019). *Algorithms for Optimization*. The MIT Press.

Kudyshev, Z. A., Kildishev, A. V., Shalaev, V. M., and Boltasseva, A. (2020). Machine learning-assisted global optimization of photonic devices. *Nanophotonics*, 10(1):371–383.

Li, N., Ho, C. P., Wang, I.-T., Pitchappa, P., Fu, Y. H., Zhu, Y., and Lee, L. Y. T. (2021). Spectral imaging and spectral lidar systems: moving toward compact nanophotonics-based sensing. *Nanophotonics*, 10(5):1437–1467.

Lukas Chrostowski (2010). *Silicon Photonics Design: From Device to System*.

Malheiros-Silveira, G. N. and Delalibera, F. G. (2020). Inverse design of photonic structures using an artificial bee colony algorithm. *Applied Optics*, 59(13):4171.

Malkiel, I., Mrejen, M., Nagler, A., Arieli, U., Wolf, L., and Suchowski, H. (2018). Plasmonic nanostructure design and characterization via Deep Learning. *Light: Science and Applications*, 7(1).

Molesky, S., Lin, Z., Piggott, A. Y., Jin, W., Vucković, J., and Rodriguez, A. W. (2018). Inverse design in nanophotonics. *Nature Photonics*, 12(11):659–670.

Oskooi, A. F., Roundy, D., Ibanescu, M., Bermel, P., Joannopoulos, J. D., and Johnson, S. G. (2010). Meep: A flexible free-software package for electromagnetic simulations by the FDTD method. *Computer Physics Communications*, 181(3):687–702.

Piggott, A. Y., Petykiewicz, J., Su, L., and Vučković, J. (2017). Fabrication-constrained nanophotonic inverse design. *Scientific Reports*, 7(1):1–7.

Prosopio-Galarza, R., De La Cruz-Coronado, J., Hernandez-Figueroa, H. E., and Rubio-Noriega, R. (2019). Comparison between optimization techniques for Y-junction devices in SOI substrates. *Proceedings of the 2019 IEEE 26th International Conference on Electronics, Electrical Engineering and Computing, INTERCON 2019*, pages 1–4.

Schneider, P. I., Garcia Santiago, X., Soltwisch, V., Hammerschmidt, M., Burger, S., and Rockstuhl, C. (2019). Benchmarking Five Global Optimization Approaches for Nano-optical Shape Optimization and Parameter Reconstruction. *ACS Photonics*, 6(11):2726–2733.

Shen, Y., Harris, N. C., Skirlo, S., Prabhu, M., Baehr-Jones, T., Hochberg, M., Sun, X., Zhao, S., Larochelle, H., Englund, D., and Soljačić, M. (2017). Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 11(7):441–446.

Shen, Y., Meng, X., Cheng, Q., Rumley, S., Abrams, N., Gazman, A., Manzhosov, E., Glick, M. S., and Bergman, K. (2019). Silicon photonics for extreme scale systems. *J. Lightwave Technol.*, 37(2):245–259.

Song, W. and Xie, K. (2008). Optimal design of a multi-mode interference splitter based on SOI. *Optoelectronics Letters*, 4(2):92–95.

Su, L., Piggott, A. Y., Sapra, N. V., Petykiewicz, J., and Vučković, J. (2018). Inverse Design and Demonstration of a Compact on-Chip Narrowband Three-Channel Wavelength Demultiplexer. *ACS Photonics*, 5(2):301–305.

Su, L., Vercruysse, D., Skarda, J., Sapra, N. V., Petykiewicz, J. A., and Vučković, J. (2020). Nanophotonic inverse design with SPINS: Software architecture and practical considerations. *Applied Physics Reviews*, 7(1).

Vuckovic, J. (2019). From inverse design to implementation of practical (quantum) photonics (Conference Presentation). In Soci, C., Sheldon, M. T., and Agio, M., editors, *Quantum Nanophotonic Materials, Devices, and Systems 2019*, volume 11091. International Society for Optics and Photonics, SPIE.

Zhang, J., Bi, S., and Zhang, G. (2021). A directional Gaussian smoothing optimization method for computational inverse design in nanophotonics. *Materials and Design*, 197:109213.