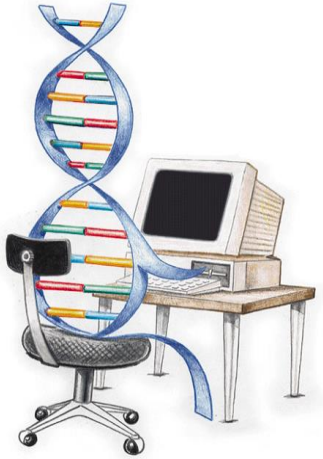
Conceptos y herramientas para la resolución de problemas de optimización multiobjetivo

Algoritmos evolutivos para optimización multiobjetivo

SERGIO NESMACHNOW (Universidad de la República, Uruguay)
DR. DIEGO ROSSIT (UNS)

Contenido



- Algoritmos evolutivos
- Algoritmos evolutivos para optimización multiobjetivo
- MOEAs no basados en dominancia de Pareto
- MOEAs basados en dominancia de Pareto
- MOEAs de primera generación: NSGA
- MOEAs de segunda generación: SPEA, SPEA-2, PAES, NSGA-II, Micro GA
- Métricas para evaluar MOEAs

Metaheurísticas

- Algoritmos aproximados para resolver problemas NP-difíciles.
- Aplicables para problemas en los cuales los algoritmos exactos no son capaces de obtener resultados de calidad en tiempos razonables.
- No garantizan encontrar soluciones óptimas, pero son muy eficaces para hallar buenas soluciones a un costo computacional reducido.
- Son muy versátiles para resolver todo tipo de problemas.
- Tienen un mecanismo de búsqueda que aporta ventajas para la resolución de problemas de optimización multiobjetivo.

Algoritmos evolutivos

- Metaheurísticas basadas en la evolución natural
  - Interpretan la naturaleza como una formidable maquinaria de resolución de problemas.
  - Aplican un mecanismo análogo a los procesos evolutivos naturales, para resolver problemas de búsqueda y optimización.
  - Trabajan con una población (de representaciones) de soluciones.
  - Principios: selección natural (aptitud), reproducción (recombinación y mutación) y diversidad genética.

Algoritmos evolutivos

- Siguen la idea de la supervivencia de los individuos más aptos, evaluando la aptitud de acuerdo al problema a resolver, mediante una función de fitness.
- La función de fitness modela al entorno, se vincula con el problema de optimización a resolver.

Algoritmos evolutivos

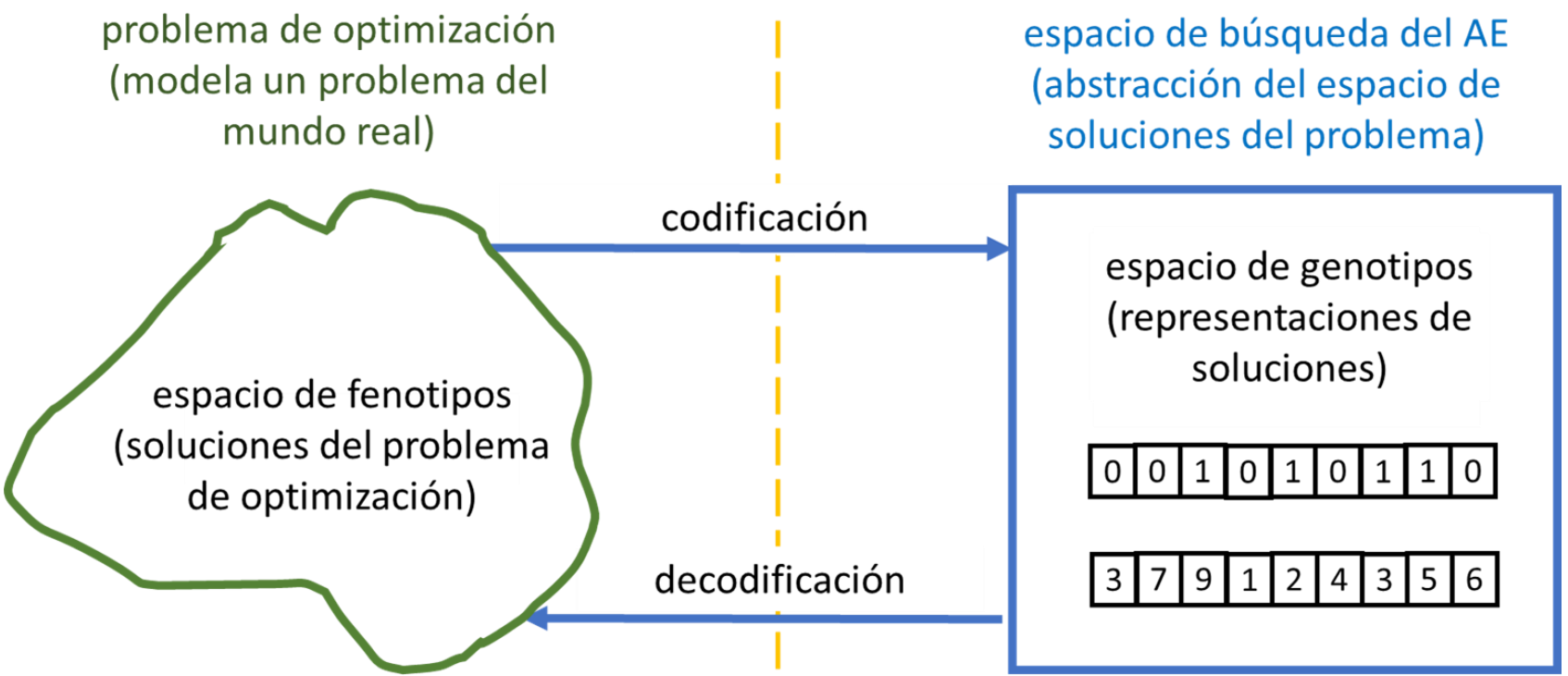
- Esquema algorítmico de un AE que trabaja sobre una población P

```
Inicializar(P(0))
generacion = 0
mientras (no CriterioParada) {
   Evaluar(P(generacion))
   Padres = Seleccion(P(generacion))
   Hijos = Operadores Evolutivos(Padres)
   NuevaPoblacion = Reemplazar(Hijos,P(generacion))
   generacion ++
   P(generacion) = NuevaPoblacion
}
retornar mejor solución encontrada
```

Algoritmos evolutivos

- Un AE trabaja sobre una población de individuos que representan soluciones potenciales al problema a resolver.
- La representación (individuo) es el genotipo, la solución el fenotipo.
- Una función de fitness evalúa las soluciones representadas por los individuos, de acuerdo a su adecuación para la resolución del problema.
- La función de fitness se aplica sobre los fenotipos, no sobre las representaciones.

Algoritmos evolutivos



Algoritmos evolutivos

- La representación es el genotipo que se corresponde con una solución al problema (fenotipo)
- Existe un proceso de codificación (y su inverso de decodificación) que permite transformar fenotipos en genotipos y viceversa
- La codificación especifica una función de correspondencia $f_C: S \rightarrow \Sigma^*$ (siendo S el espacio de soluciones del problema y Σ el alfabeto utilizado para codificar soluciones)
- La función inversa es la decodificación $f_D : \Sigma^* \rightarrow S$, que puede ser una función parcial
- La complejidad de f_C y f_D depende de las características del problema y de las variables a codificar

Algoritmos evolutivos

- La evolución consiste en un ciclo que consta de cuatro etapas:
  1. Evaluación: se asigna un valor de fitness a cada individuo.
  2. Selección: se determinan candidatos adecuados para crear la nueva generación de la población.
  3. Aplicación de los operadores evolutivos: se genera un conjunto de descendientes a partir de los individuos seleccionados, mediante operadores de variación que emulan la evolución natural.
  4. Reemplazo: mecanismo que realiza el recambio generacional de la población.

Algoritmos evolutivos

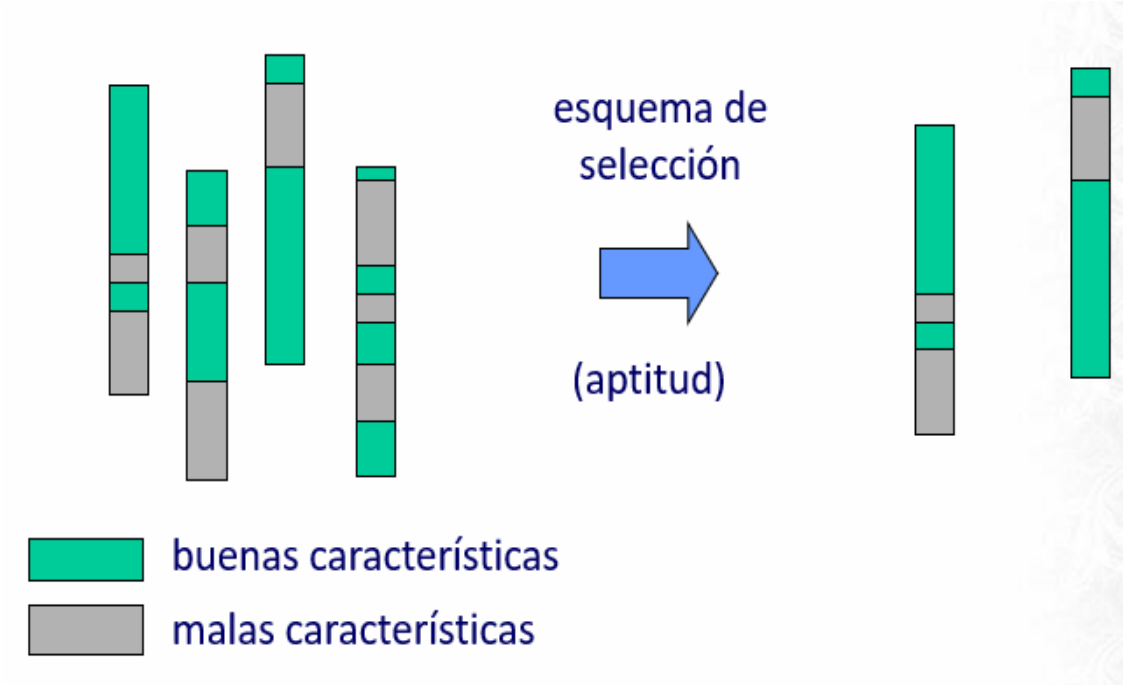
- La población se inicializa mediante un mecanismo aleatorio o guiado por heurísticas específicas
- Diversas políticas de selección y reemplazo permiten definir las características del algoritmo evolutivo
  - Privilegiar los individuos más adaptados (elitismo), aumentar la presión selectiva, incrementar la diversidad genética, etc.
- La condición de parada determina la finalización del AE
  - Número de generaciones, variación de valores de fitness, estimaciones del error cometido, etc.

Algoritmos evolutivos

- Los operadores evolutivos determinan el mecanismo de exploración del espacio de búsqueda del problema
  - Amplia gama: los más difundidos son la recombinación y la mutación
  - Determinan las diferentes variantes de AE
  - Son operadores probabilísticos (se aplican de acuerdo a probabilidades)
  - Actúan sobre los genotipos y no sobre los fenotipos de los individuos

Algoritmos evolutivos: selección

- Objetivo: mantener las características de los individuos mejor adaptados



Algoritmos evolutivos: selección

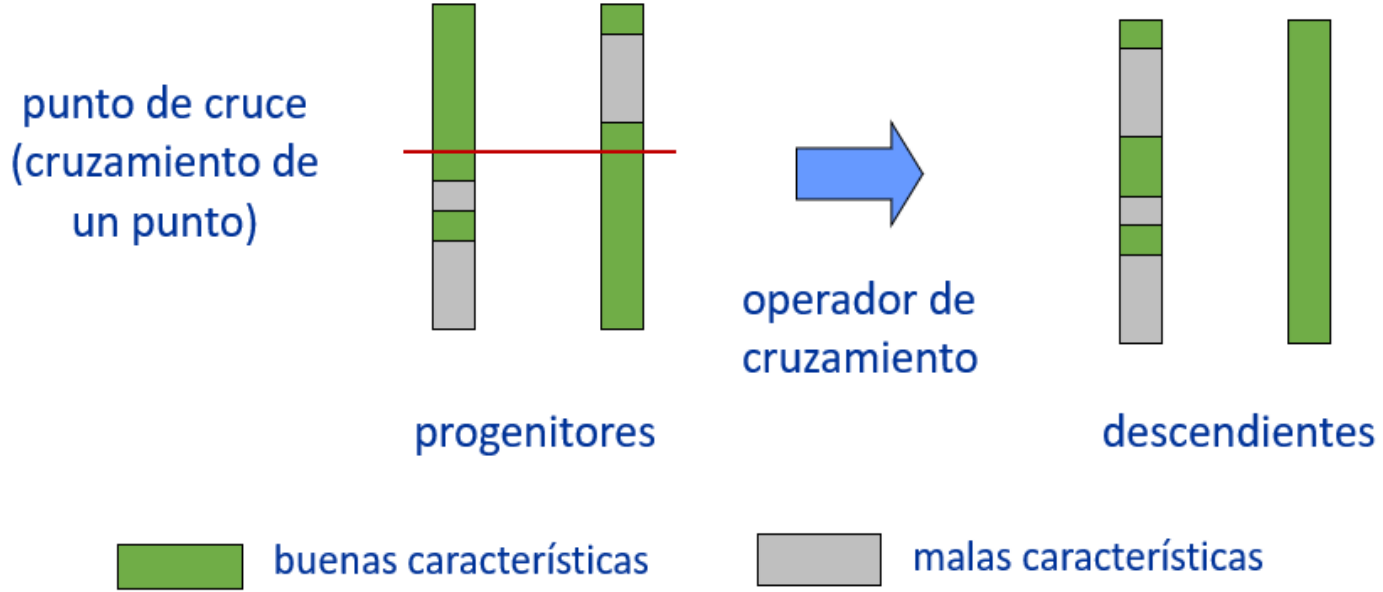
- Determina fuertemente la búsqueda (dirige al AE a explorar regiones “prometedoras” del espacio de búsqueda)
- La presión de selección es crítica para el funcionamiento del AE
  - Una presión de selección alta puede ocasionar pérdida de diversidad y convergencia prematura (en un óptimo local).
  - Una presión de selección baja puede conducir a que la búsqueda avance muy lentamente (casi de manera estocástica)
- Lo adecuado es mantener un compromiso entre la exploración del espacio de búsqueda y la explotación de buenas soluciones

Algoritmos evolutivos: selección

- Selección proporcional
  - La probabilidad de selección es proporcional al fitness relativo $ps_i = \frac{f(i)}{\sum_{j \in P} f(j)}$
  - Se implementa con la técnica de rueda de ruleta
- Selección por rango (parámetro *k*)
  - Se ordenan los individuos según su fitness y se seleccionan los *k* mejores
- Selección por torneo (parámetros *k* y *r*)
  - Se eligen *k* individuos y se seleccionan los *r* de mejor fitness
- Selección estocástica universal
  - Se implementa con una ruleta con punteros equiespaciados para eliminar el sesgo de la selección proporcional

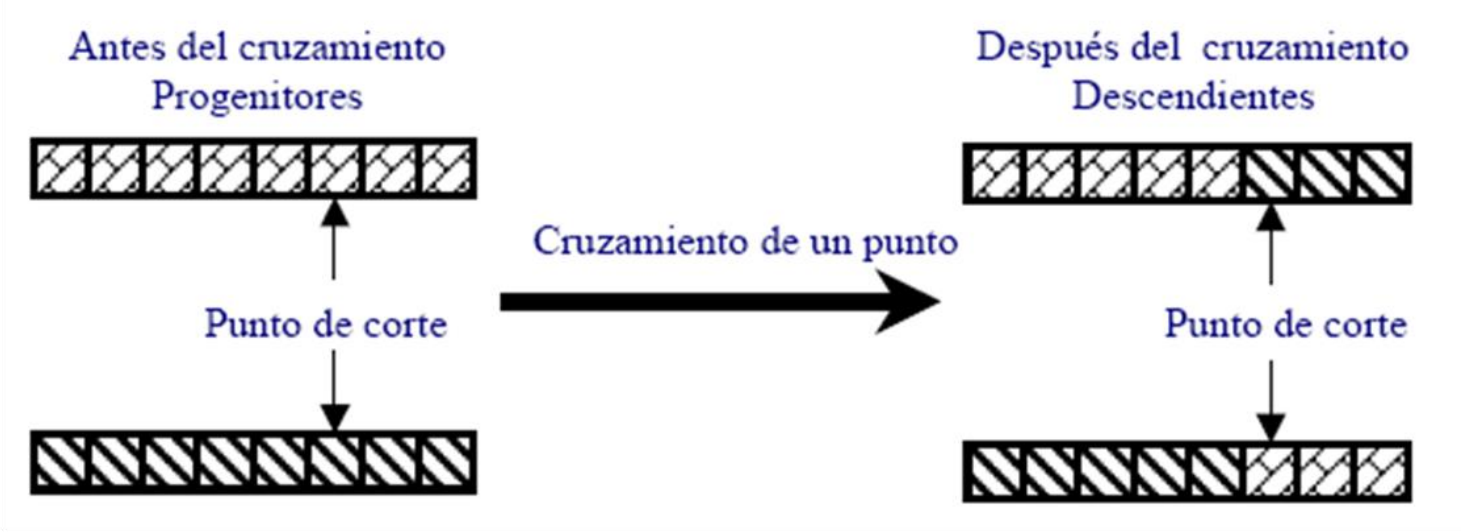
Algoritmos evolutivos: recombinación

- Objetivo: garantizar la explotación de buenas soluciones (regiones promisorias del espacio de búsqueda)



Algoritmos evolutivos: recombinación

- Cruzamiento de un punto



Algoritmos evolutivos: recombinación

- Cruzamiento de dos puntos

Algoritmos evolutivos: recombinación

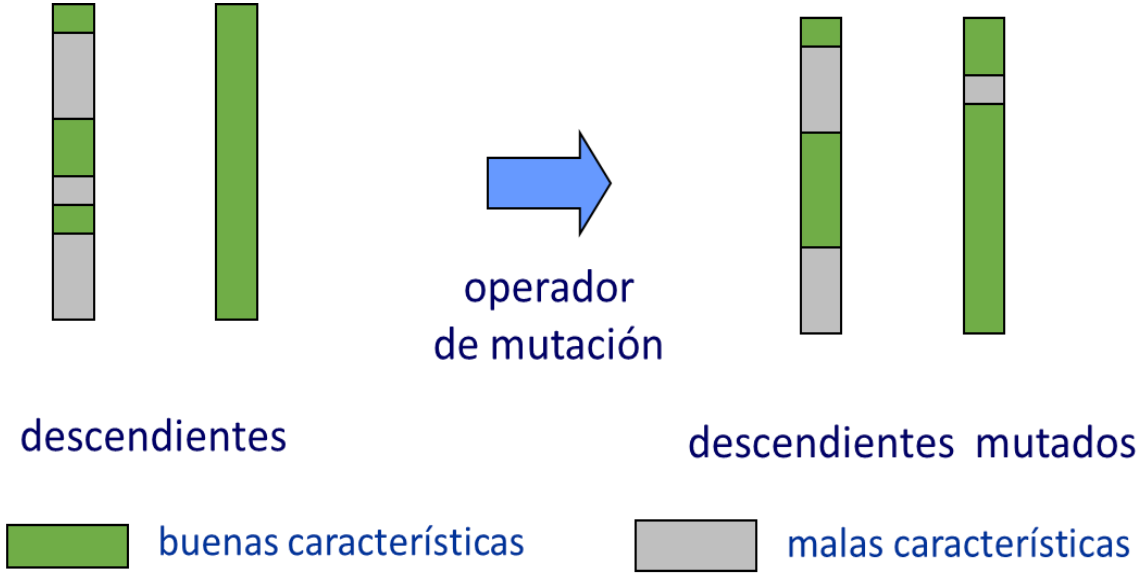
- Cruzamiento uniforme
  - Más disruptivo que cruzamiento de n puntos
  - La probabilidad de conservar grupos es independiente de la posición

Algoritmos evolutivos: recombinación

- Ciertas codificaciones requieren operadores de cruzamiento específicos, para conservar las características de las soluciones del problema
  - Si las soluciones deben ser ciclos, se representan por permutaciones, que no es posible cruzar directamente
- Otras codificaciones admiten operadores de cruzamiento con operaciones específicas
  - Por ejemplo, la codificación con números reales admite cruzamientos aritméticos (promedio, combinación lineal, etc)
  - En otros casos deben implementarse operadores dependientes del problema para asegurar factibilidad de soluciones o conservar significado semántico
  - Existen operadores de cruzamiento con varios padres y panmícticos

Algoritmos evolutivos: mutación

- Objetivo: introducir diversidad (aleatoriamente) en los individuos de la población, permitiendo la exploración de diferentes regiones del espacio de búsqueda



Algoritmos evolutivos: mutación

- Modificación simple de la codificación (material genético) en una posición determinada aleatoriamente
  - Sigue la analogía de la mutación en la evolución natural

Algoritmos evolutivos: mutación

- Ciertas codificaciones no admiten la mutación simple (de un solo valor), y requieren operadores de mutación específicos para conservar las características de las soluciones del problema
  - Si la solución debe ser un ciclo, cada solución está representada por una permutación, no es posible mutar un alelo en la permutación, pues un valor quedaría repetido.
- Otras codificaciones admiten operadores de mutación específicos
  - Por ejemplo, la codificación con números reales admite mutación gaussiana (o utilizando otras distribuciones).
  - En otros casos deben implementarse operadores dependientes del problema, para asegurar factibilidad de soluciones o conservar significado semántico.

Algoritmos evolutivos: función de fitness

- La función de fitness depende del problema y del criterio de optimización
- Opera directamente sobre las soluciones del problema (fenotipos)
- Debe considerar las restricciones del problema y puede definir objetivos múltiples (función vectorial, en problemas multiobjetivo) o incorporar sub-objetivos
- Puede cambiar dinámicamente a medida que un AE procede en la exploración (en problemas dinámicos)
- La función de fitness es una caja negra para un AE
  - Entrada: fenotipo
  - Salida: valor de fitness

Algoritmos evolutivos: función de fitness

- No existe una única función de fitness para un problema
  - Es necesario considerar las características del problema de optimización
  - Debe considerarse “el sentido” de la optimización. El formalismo de los AE propone maximizar el fitness, pero los problemas de optimización usualmente se plantean como una minimización de una función objetivo (costo)
  - Es necesario transformar el problema de minimización de la función objetivo a uno de maximización de la función de fitness
  - Por ejemplo, si el problema de optimización es min f(x) se pueden considerar varias alternativas para definir una función de fitness $F(x)$:

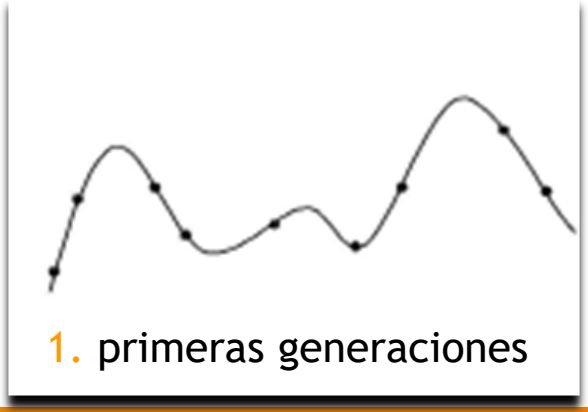
1. $F(x) = -f(x)$ (opuesto)
2. $F(x) = 1/f(x)$ (inverso)
3. $F(x) = C - f(x)$ (respecto a un costo máximo)
4. $F(x) = g(f(x))$ (genérica)

Algoritmos evolutivos: función de fitness

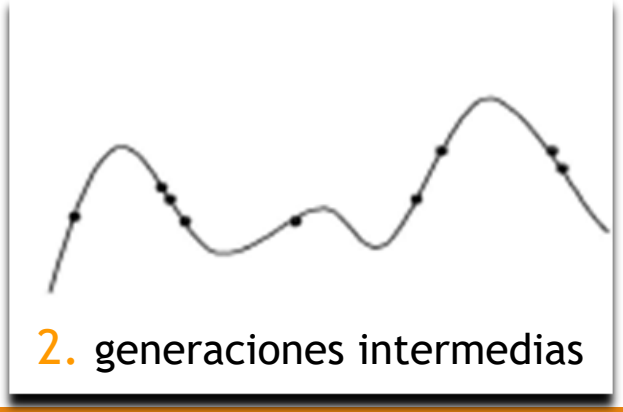
- Un posible problema es la generación de individuos no factibles durante la evolución.
- Existen cuatro enfoques para tratar los individuos no factibles:
  1. Evitarlos en la codificación. En general no es un procedimiento sencillo, complica los procesos de codificación y decodificación.
  2. Descartarlos es la opción más simple, pero conduce a la pérdida de características que podrían ser útiles para resolver el problema
  3. Penalizarlos en sus valores de fitness. Definir un modelo de penalización adecuado puede ser dificultoso y requerir estudios teóricos/empíricos.
  4. Corregirlos, mediante un procedimiento algorítmico dependiente del problema.

Algoritmos evolutivos: características

- Comportamiento típico de la búsqueda
  1. Fase temprana: distribución de individuos cuasi-aleatoria
  2. Fase intermedia: la población comienza a concentrarse alrededor de las colinas
  3. Fase final: la población se concentra en los óptimos locales



1. primeras generaciones

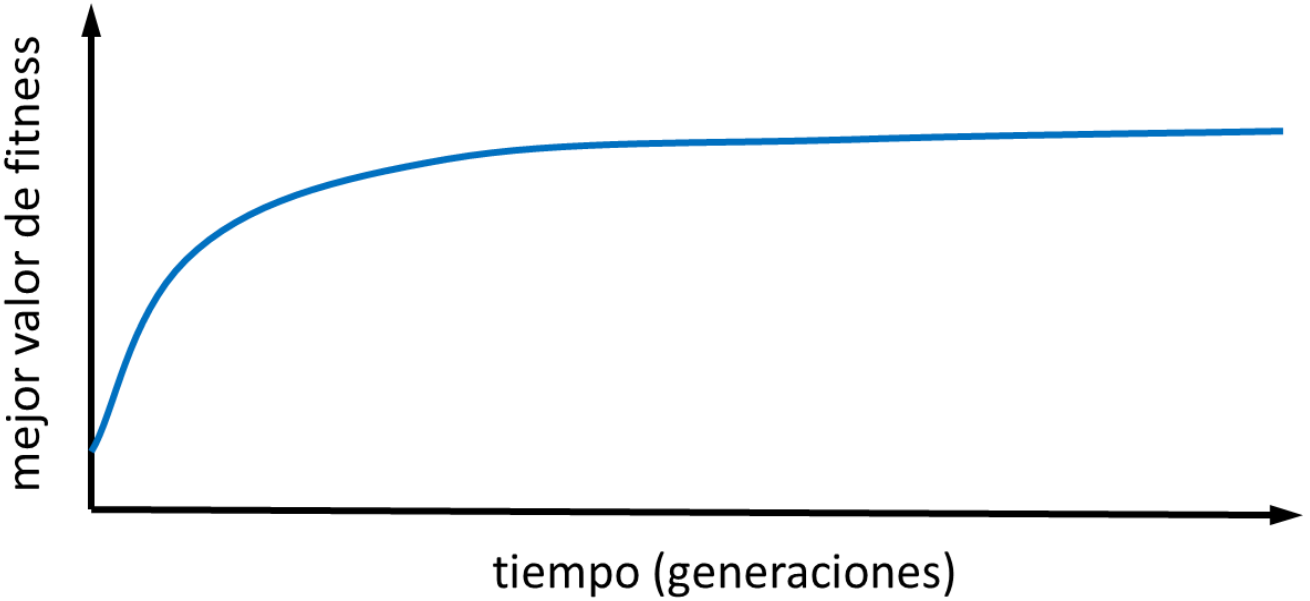


2. generaciones intermedias

3. generaciones finales

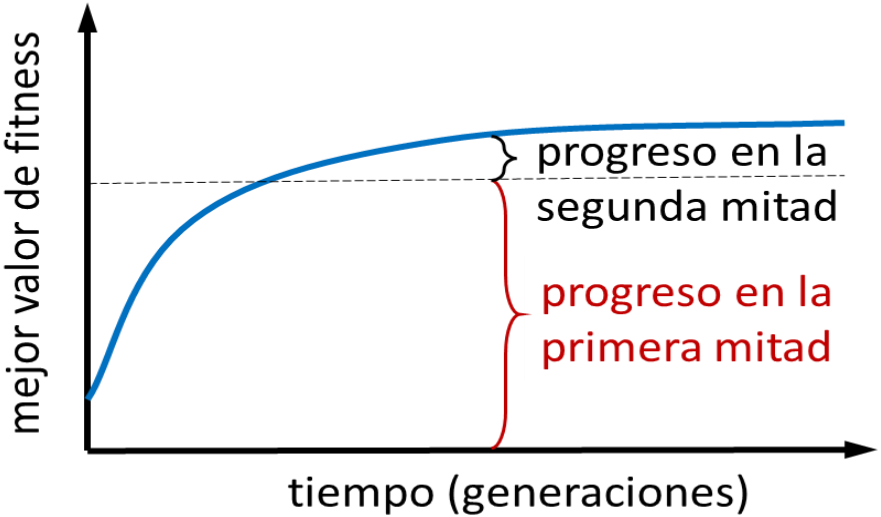
Algoritmos evolutivos: características

- Evolución del fitness (mejor, promedio)



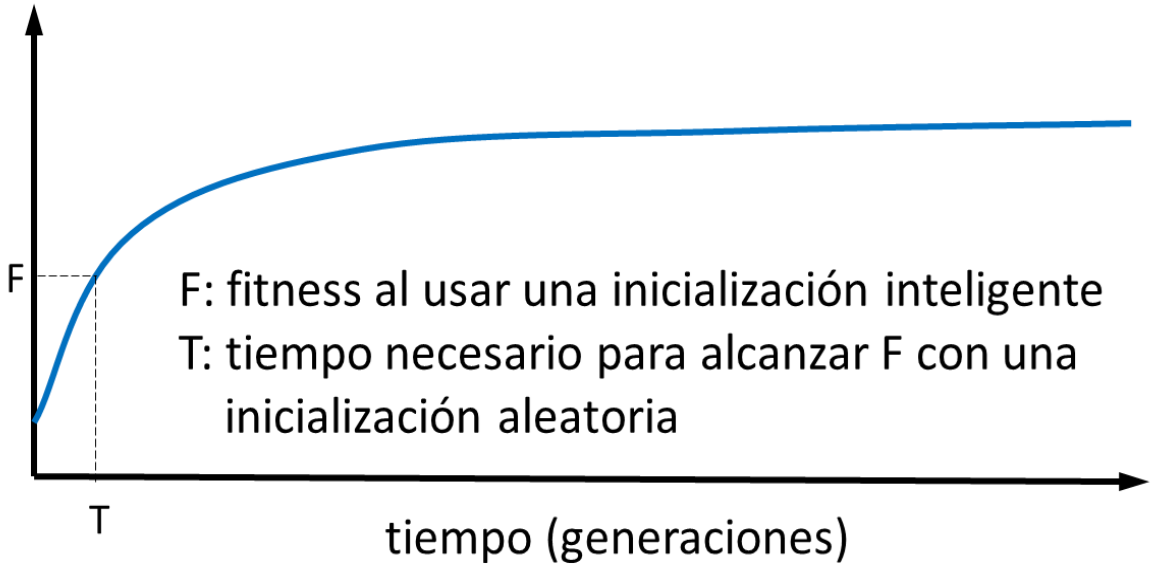
Algoritmos evolutivos: características

- Son beneficiosas las ejecuciones largas?
  - Depende del valor asignado a los últimos progresos
  - Puede ser más beneficioso hacer muchas ejecuciones cortas



Algoritmos evolutivos: características

- Son beneficiosas las inicializaciones inteligentes?
  - Depende de qué tan crítico sea calcular buenas soluciones rápidamente
  - Debe validarse sobre una variedad de instancias, para no sesgar la búsqueda



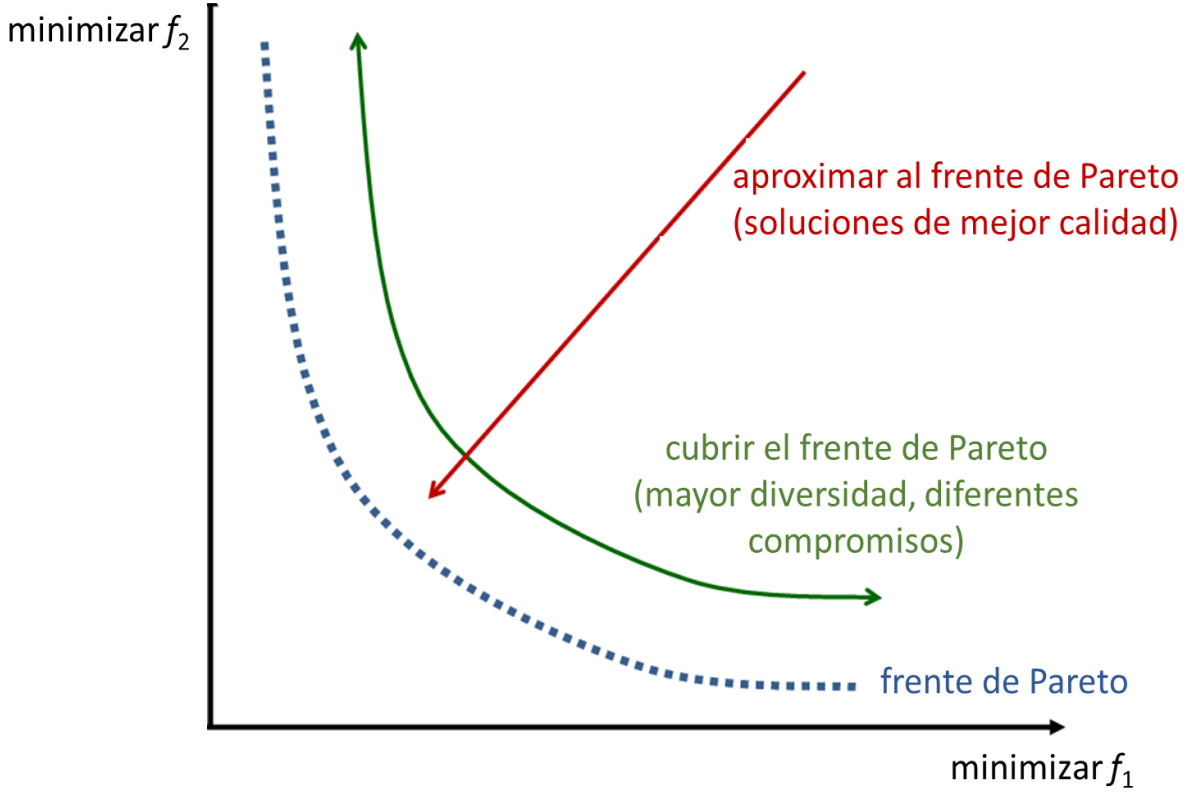
MOEAs: conceptos

- MOEA sigla (en inglés) de MultiObjective Evolutionary Algorithm.
- La primer referencia que sugirió aplicar AE a problemas de optimiización multiobjetivo la realizó Rosenberg en 1967.
- Sin embargo, el primer MOEA recién fue propuesto por Schaffer en 1984.
- Ventajas de un EA para resolver problemas multiobjetivo:
  - Al trabajar en paralelo sobre un conjunto de soluciones tienen la potencialidad de tratar problemas con múltiples objetivos, hallando en cada ejecución un conjunto de soluciones aproximadas al frente de Pareto.
  - Son menos sensibles a la forma o a la continuidad del frente de Pareto.
  - Permiten abordar problemas con espacio de soluciones de gran dimensión.

Propósitos de un MOEA

- Los propósitos de un MOEA son:
  - aproximarse al frente de Pareto del problema de optimización multiobjetivo.
  - muestrear adecuadamente el frente de Pareto, hallando alternativas que expresen diferentes compromisos entre las funciones a optimizar (que permiten realizar la toma de decisiones a posteriori).
  - tener buenos valores de eficiencia computacional (aunque en general se considera, erróneamente, un propósito secundario).

Propósitos de un MOEA



Esquema algorítmico de un MOEA

```
Inicializar(P(0))
generacion = 0
mientras (no CriterioParada) {
   Evaluar(P(generacion))
   Operador de diversidad(P(generacion))
   Asignar fitness(P(generacion))
   Padres = Seleccion(P(generacion))
   Hijos = Operadores de Reproduccion(Padres)
   NuevaPop = Reemplazar(Hijos,P(generacion))
   generacion ++
   P(generacion) = NuevaPop
}
retornar frente de Pareto
```

Esquema algorítmico de un MOEA

- Se incluyen dos operadores característicos de los MOEAs, que no aparecen en la estructura genérica de un EA: el operador de diversidad y el operador de asignación de fitness.
- El operador de diversidad aplica una técnica para evitar la convergencia a un sector del frente de Pareto.
- La asignación de fitness está orientada a brindar una mayor chance de perpetuarse a aquellos individuos con mejores características, considerando los valores de las funciones objetivo y los resultados de la métrica utilizada para evaluar la diversidad.
- Algunas propuestas de MOEAs no siguen estrictamente el esquema algorítmico presentado (incluyen variaciones menores)

Clasificación de MOEAs

- De acuerdo a su mecanismo de asignación de fitness, los MOEAs se clasifican en no basados en Pareto y basados en Pareto.
- MOEAs no basados en Pareto
  - Proponen mecanismos de asignación de fitness relativamente sencillos, que en general no reflejan la independencia entre los objetivos del problema.
  - No garantizan una correcta resolución de un problema multiobjetivo.
- MOEAs basados en Pareto
  - Utilizan explícitamente la dominancia de Pareto en el procedimiento de asignación de fitness

MOEAs no basados en Pareto

- Proponen mecanismos de asignación de fitness relativamente simples, que no consideran la dominancia de Pareto para clasificar soluciones.
- En general, no garantizan una correcta resolución del problema.
  - Calculan pocas o inclusive una única solución al problema.
  - Restringen la búsqueda a una zona/dirección del espacio de soluciones factibles del problema.
- En la práctica, sólo son adecuados para problemas con no más de dos/tres funciones objetivo y espacio de soluciones sencillo.
- Existen varias propuestas de mecanismos de asignación.

MOEAs no basados en Pareto

1. Agregación o combinación lineal de objetivos
   - El problema multiobjetivo (vectorial) es transformado en un problema de objetivo único (escalar).
   - El fitness se calcula como una suma ponderada de las funciones objetivo

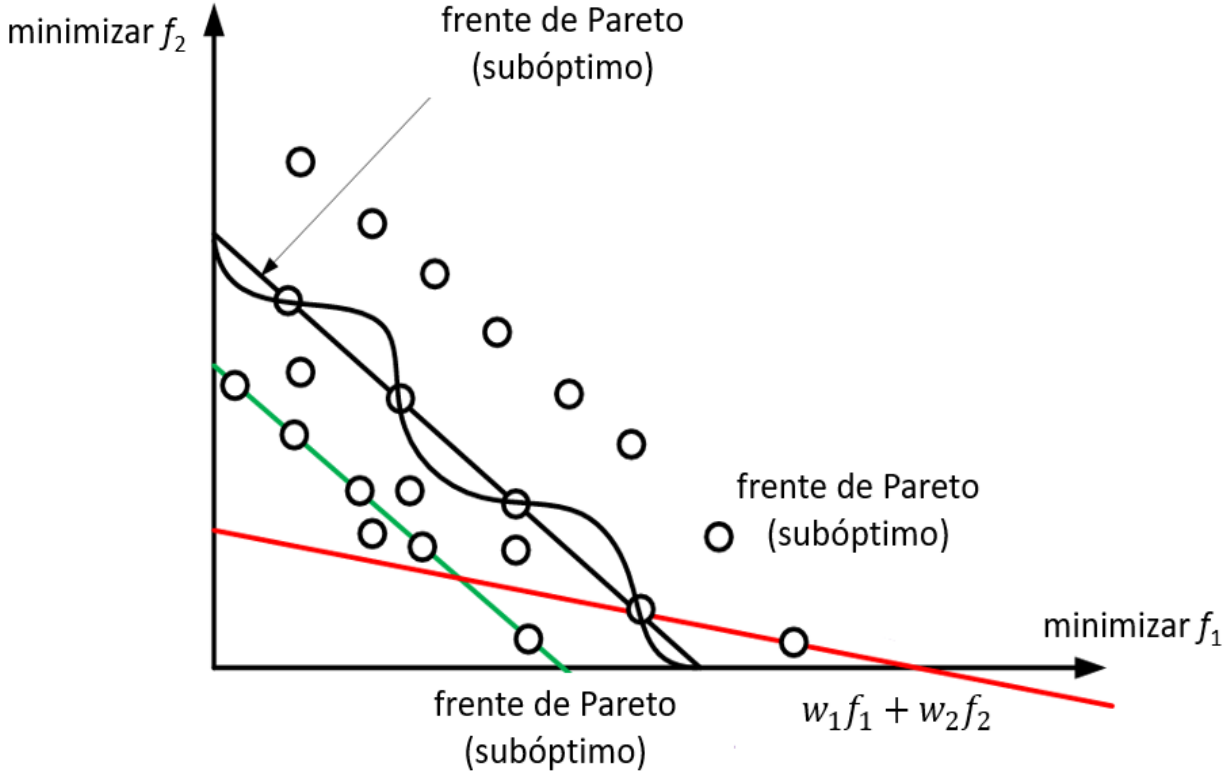
$$F(x) = \sum_{i=1}^{k} w_i \cdot f_i\,(x)$$

   - Los pesos se fijan a priori.
   - La expresión no refleja el carácter multiobjetivo del problema.

MOEAs no basados en Pareto

1. Agregación o combinación lineal de objetivos
   - Los resultados dependen fuertemente de los pesos seleccionados.
   - En general no existen criterios específicos o formales para caracterizar las regiones del espacio de búsqueda, por lo cual el AE calculará soluciones localmente óptimas.

Agregación/combinación lineal de objetivos

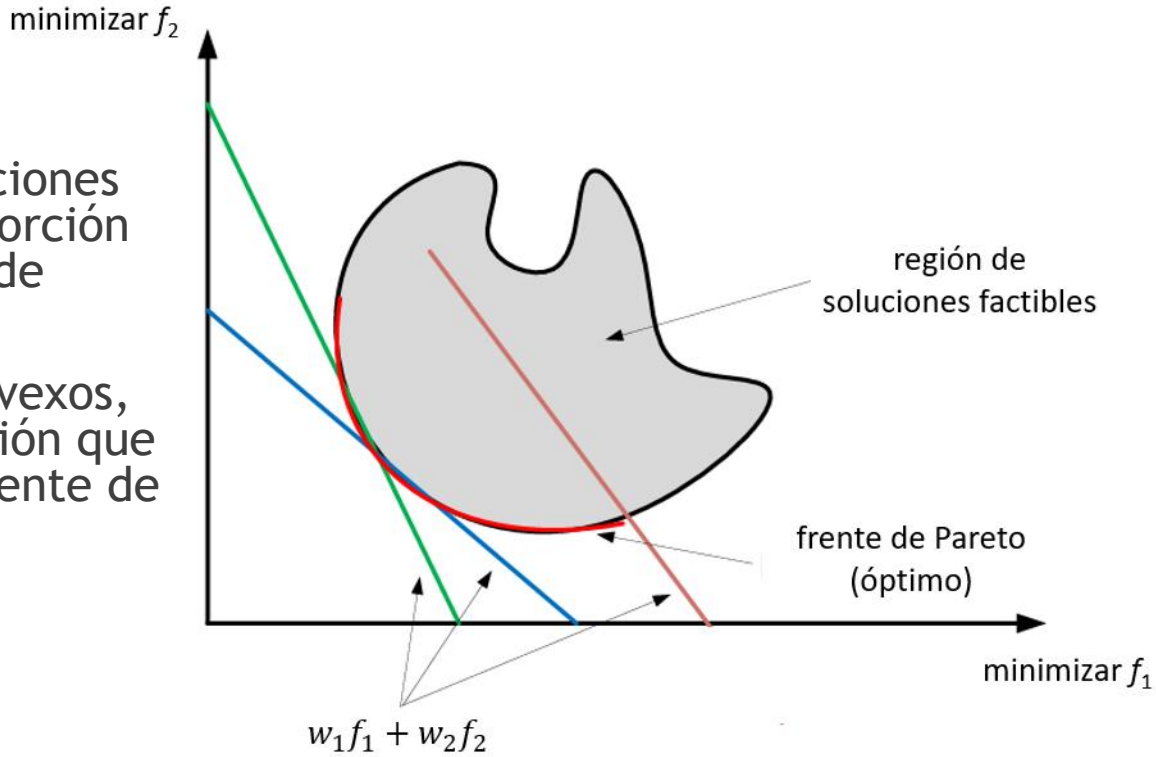


Agregación/combinación lineal de objetivos

- Para hallar múltiples soluciones es necesario resolver el mismo problema para muchos valores diferentes de los pesos w_i .
- Los pesos son factores (ficticios) para la optimización, no reflejan preferencias o importancias de los objetivos.
- El método depende de los valores de los pesos y de las escalas de los valores de las funciones objetivo.
- Es necesario escalar los valores funcionales para poder muestrear apropiadamente soluciones de compromiso.

Agregación/combinación lineal de objetivos

- No permite hallar soluciones Pareto-óptimas en la porción no convexa del frente de Pareto
- Para problemas no convexos, el AE calcula una solución que puede pertenecer al frente de Pareto [o no].



MOEAs no basados en Pareto

2. Ordenamiento lexicográfico
   - Técnica a priori, basada en el principio de “agregación por orden”
   - Las funciones objetivo se jerarquizan en orden de importancia y se definen funciones de fitness jerarquizadas correspondientemente.
   - Se resuelven los problemas en el orden de la jerarquía, agregando restricciones ficticias para las funciones objetivo resueltas previamente.
   - La idea es tratar de mejorar el fitness del subproblema que se resuelve, sin empeorar los mejores valores de las funciones objetivo ya consideradas.

MOEAs no basados en Pareto

2. Ordenamiento lexicográfico
   - Se resuelven *k* problemas de optimización multiobjetivo

$$\min f(x) = [f_1, f_2, \dots, f_k]$$
$$\text{subject to } g_j(x) \leq c_j, j = 1, 2, \dots, h$$
$$f_1 < f_2 < \dots < f_k$$

$$\min f_1(x)$$
$$\text{subject to } g_j(x) \leq c_j, j = 1, 2, \dots, h$$

sea $\hat{x}_1^*$ tal que $f_1^* = f_1(\hat{x}_1^*)$

$$\min f_2(x)$$
$$\text{subject to } g_j(x) \leq c_j, j = 1, 2, \dots, h$$
$$f_1(x) = f_1^*$$

MOEAs no basados en Pareto

2. Ordenamiento lexicográfico
   - Se aplica el procedimiento para las k funciones objetivo, de acuerdo con el orden definido para su importancia.
   - La solución final que se obtiene es $\hat{x}_k^*$ tal que $f_k^* = f_k\ (\hat{x}_k^*)$, que se considera como solución del problema de optimización multiobjetivo, de acuerdo con el ordenamiento definido.
   - Los AE aportan una ventaja importante: pueden trabajar con funciones de fitness dinámicas, que varían durante la evolución.

MOEAs no basados en Pareto

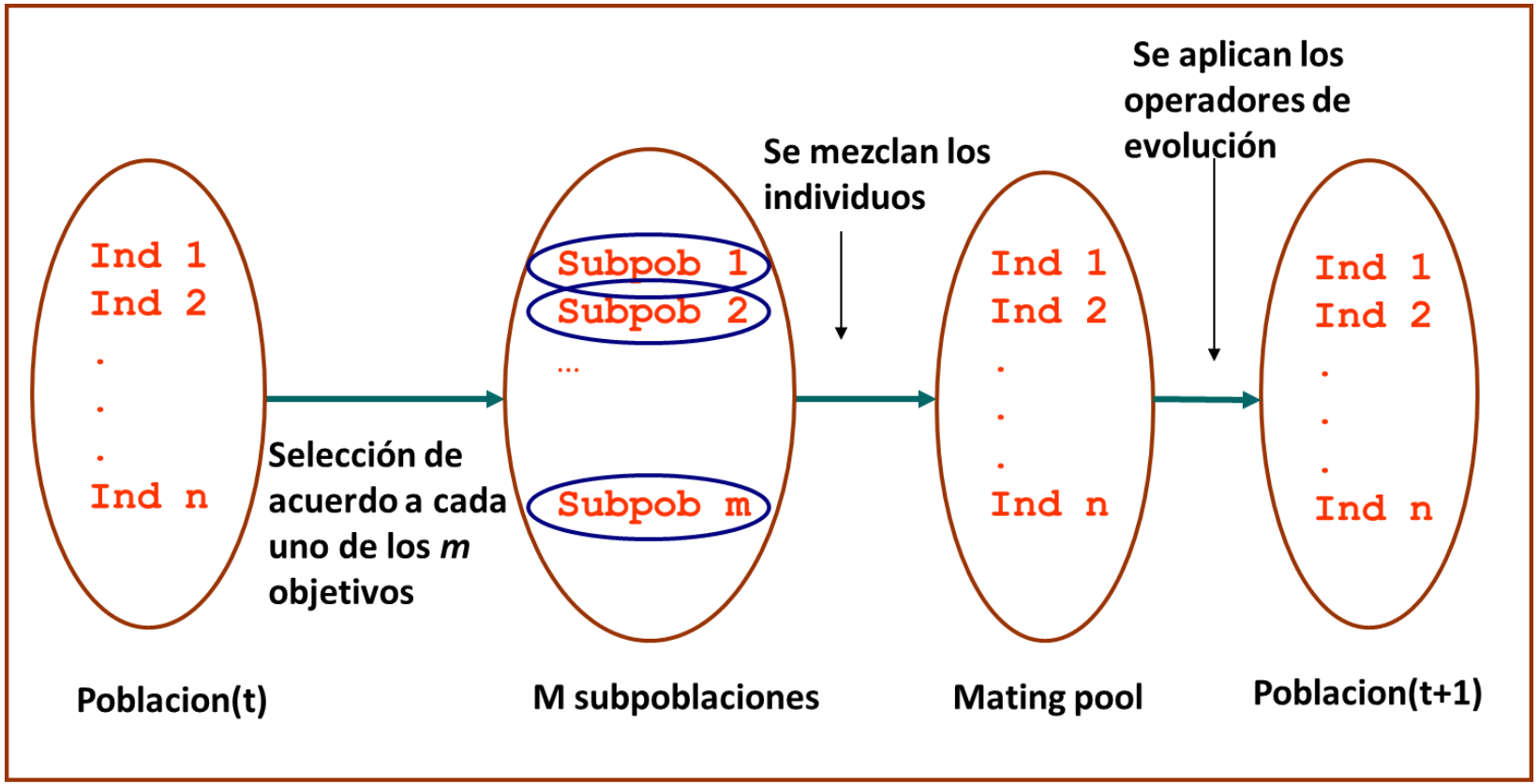
2. Ordenamiento lexicográfico
   - Decidir el orden apropiado para las funciones objetivo es un problema.
   - Las soluciones calculadas varían si se considera un orden diferente de las funciones obetivo.
   - Seleccionar aleatoriamente el orden de las funciones objetivos no es un enfoque razonable.
   - Un enfoque trivial requeriría intentar con las $k!$ permutaciones posibles de las k funciones objetivo, que es prohibitivo computacionalmente.

El método calcula múltiples soluciones no dominadas/óptimos de Pareto, pero está sesgado por la exploración definida por el orden considerado para las funciones.

MOEAs no basados en Pareto

- David Schaffer realizó la primer propuesta de MOEA en 1984: el Vector Evaluated Genetic Algorithm (VEGA).
- La idea consiste en incorporar una estrategia de selección especial, que considera los objetivos múltiples en un AE simple.
- Para un problema con m objetivos se generan m subpoblaciones de tamaño #poblacion/m.
- Los individuos son seleccionados de acuerdo a cada uno de los objetivos.
- El mecanismo de selección utilizado se basa en usar soluciones localmente dominadas en cada generación.
- No utiliza dominancia de Pareto.

MOEAs no basados en Pareto: VEGA



MOEAs no basados en Pareto: VEGA

- Si bien se tienen en cuenta las distintas funciones a optimizar, el mecanismo de búsqueda es unidimensional.
- Sin embargo surge un problema relacionado con la especialización:
- Los individuos se especializan en optimizar $f_1,\ldots, f_m$ por separado.
- Un individuo que tenga buenos valores de compromiso pero que no optimiza en particular ninguna de las funciones objetivo $f_1,\ldots, f_m$ no sobrevive a la selección.
- Es contraproducente para hallar los óptimos de Pareto y para muestrear el frente de Pareto.

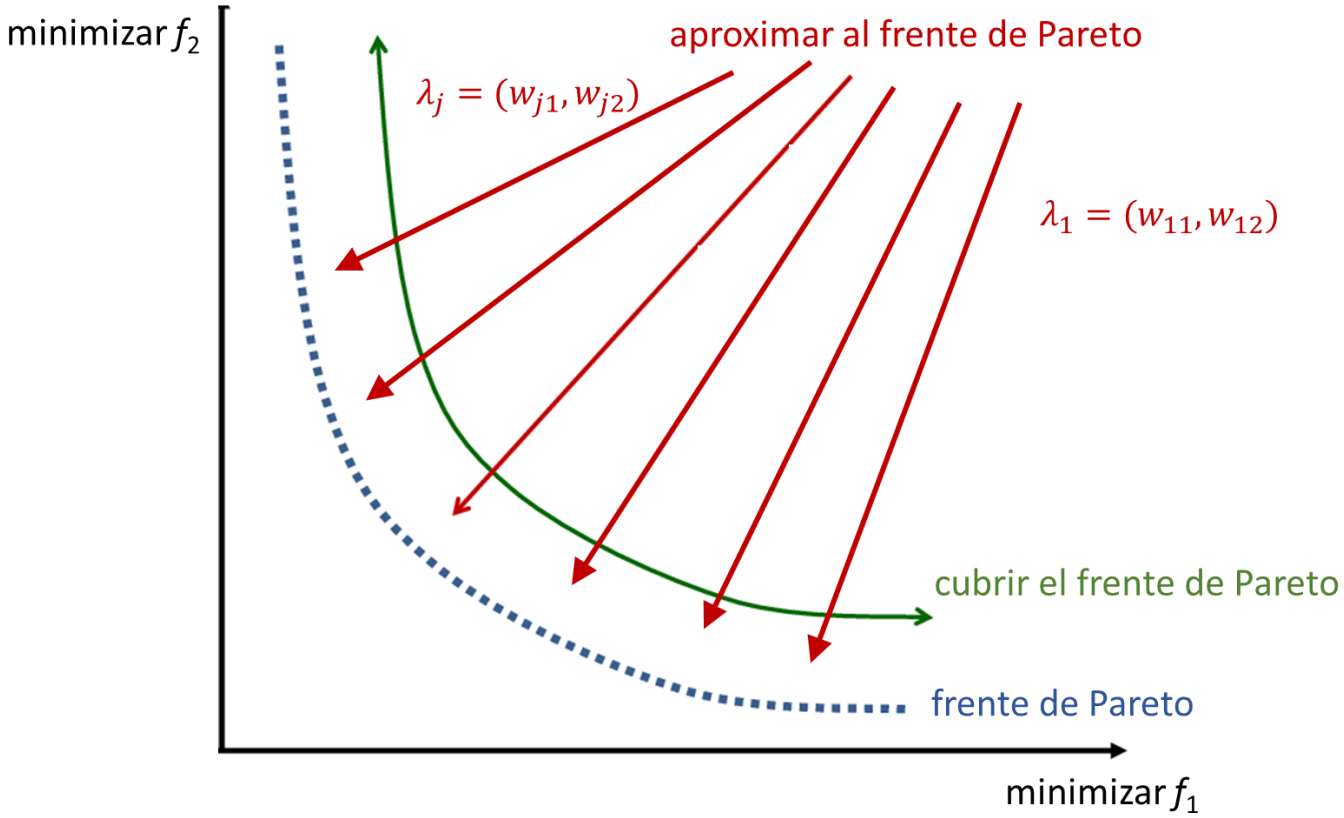
MOEAs no basados en Pareto: MOEA/D

- Aplica descomposición de dominio en el espacio de pesos (ponderaciones) de una función de agregación lineal.
- Utiliza el enfoque de Tchebycheff (no excluyente).
- El problema j-ésimo es $g^{te}(x|\lambda^j, z^*) = \max_{1 \leq i \leq m} \{\lambda_i^j |f_i(x) - z_i^*|\}$
- $\vec{\lambda}$ es el vector de pesos, z_i^* es el nadir, usado como referencia

$$z_i^* = \max\{f_i(x)|x \in \Omega\}$$

- La evolución considera al problema i-ésimo y su(s) vecino(s) cercano(s), definidos por los valores de los pesos λ_j.
- Aplica los operadores evolutivos a los mejores individuos según λ_i y λ_j.
- Las soluciones no dominadas se almacenan en una población secundaria.

MOEAs no basados en Pareto: MOEA/D



MOEAs basados en Pareto

- Utilizan explícitamente la dominancia de Pareto para asignar el fitness.
- Suelen utilizar mecanismos para la preservación de la diversidad.
  - Técnicas usualmente usadas en AE para problemas multimodales.
- Mecanismos para la preservación de la diversidad usados frecuentemente:

1. Fitness sharing
   - Aplicado en el espacio de soluciones o en el espacio de valores funcionales.
   - Depende fuertemente de la función de distancia utilizada.

Fitness sharing

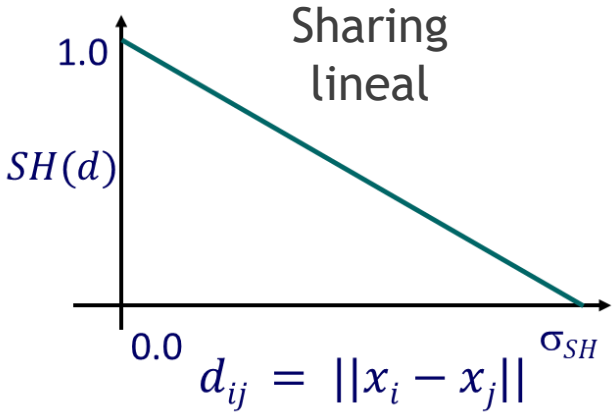
- La idea consiste en decrementar el fitness de las zonas muy representadas del espacio de búsqueda.
- Se determina un grado de vecindad entre individuos sumando para cada uno de ellos los valores de una función S (función de Sharing) correspondientes a cada vecino.
- Se trabaja con valores del fitness reducido f_R:

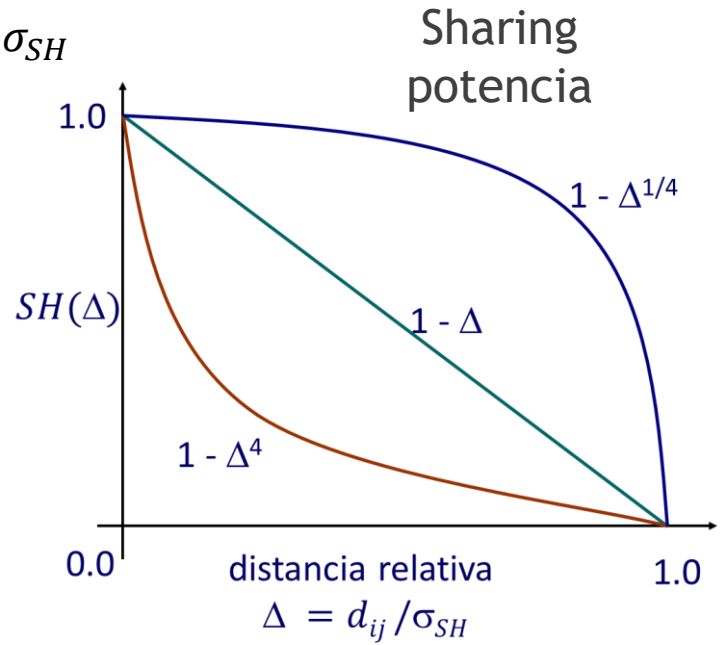
$$f_R(x_i) = \frac{f(x_i)}{\sum_j SH(d(x_i, x_j))}$$

Fitness sharing

- La función de sharing toma valores cercanos a 1 para individuos vecinos o cercanos:

$$SH\left(d(x_i, x_j)\right) = \begin{cases} 1 - \left(\dfrac{d(x_i, x_j)}{\sigma_{SH}}\right) & \text{si } d(x_i, x_j) \leq \sigma_{SH} \\ 0 & \text{sino} \end{cases}$$





MOEAs basados en Pareto

- Mecanismos para la preservación de la diversidad usados frecuentemente:

2. Restricción al cruzamiento

- Impide el cruzamiento de individuos muy similares (cuya distancia sea menor que un valor σ_{MATING})

3. Crowding

- No tiene dependencia paramétrica

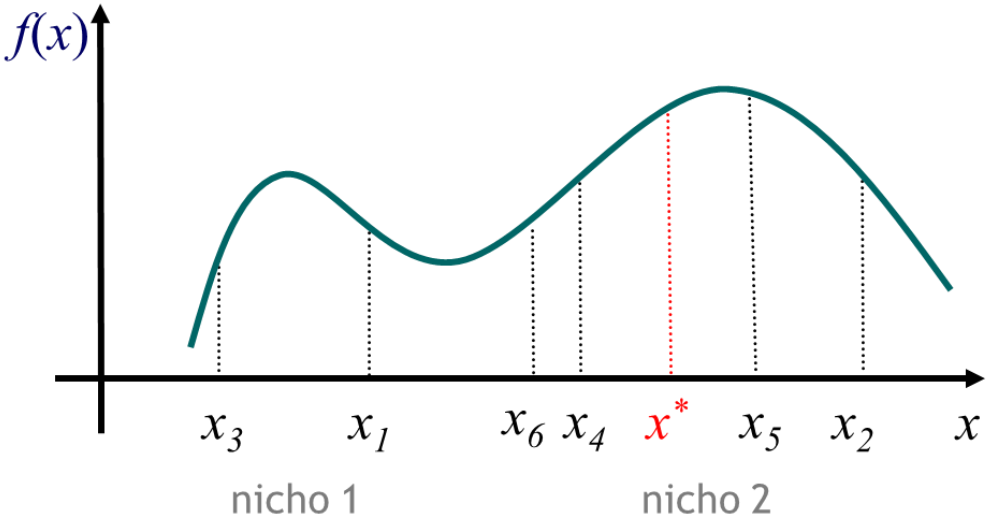
4. Otros mecanismos

- Isolation by distance, para mantener individuos “alejados”

Crowding

- Identifica nichos y un nuevo individuo reemplaza al individuo más similar a si mismo existente en la población

- x^* reemplazará a algún individuo del nicho 2 (aleatorio o determinista)
- Puede aplicarse en el espacio de valores objetivo
- Mantiene la diversidad de soluciones



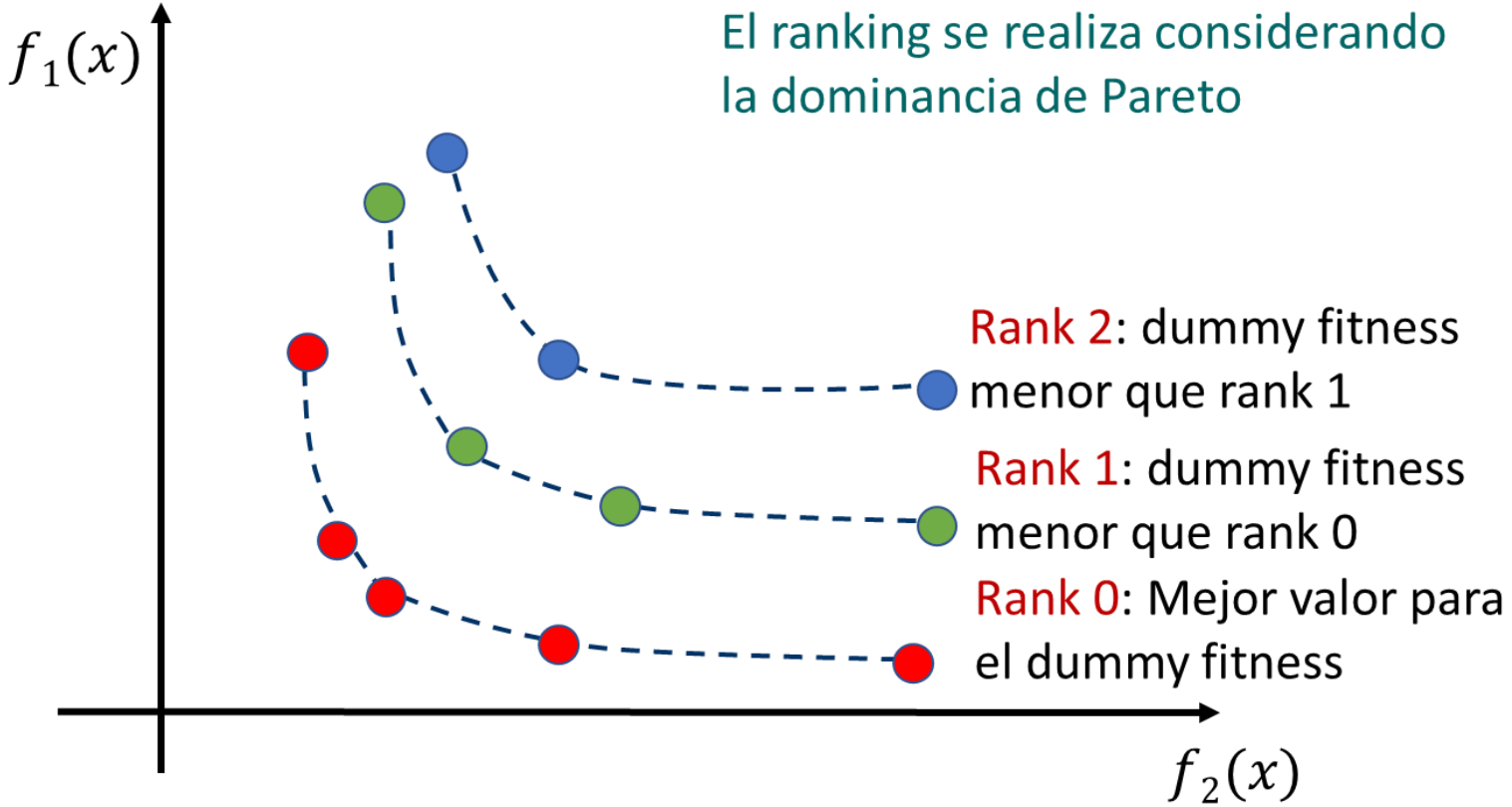
MOEAs de primera generación

- Primera etapa de propuestas de MOEAs (período 1994–2000).
- Se caracterizaron por:
  - la utilización de dominancia de Pareto.
  - el empleo de nichos y sharing para proveer diversidad.
  - en general no utilizaban elitismo.
- Algoritmos representativos de esta generación:
  - NSGA
  - NPGA
  - MOGA

MOEAs de primera generación: NSGA

- Nondominated Sorting Genetic Algorithm, propuesto por Shrnivas y Deb
- Se caracteriza por una asignación de fitness por rangos de dominancia.
- No trabaja con valores de las funciones objetivo, sino con un fitness ficticio constante (dummy fitness) que se establece a partir de la posición en el ranking de dominancia.
- Se sigue un esquema de asignación de fitness por frentes:
  - Se asigna el fitness para los individuos no dominados de la población.
  - Se remueven los individuos no dominados de la población.
  - Se repite el proceso hasta que no queden más individuos.
- Utiliza fitness sharing sobre el dummy fitness.

MOEAs de primera generación: NSGA



Líneas de investigación

- La aplicación de los MOEAs de primera generación abrió las siguientes preguntas en la comunidad científica:
  - ¿es la agregación efectivamente una mala idea?
  - ¿es posible mejorar la eficiencia computacional y los resultados?
  - ¿es posible mantener la diversidad sin utilizar nichos?
- Algunos aspectos importantes que no fueron abordados en la etapa conocida como “primera generación”:
  - la fundamentación teórica de los MOEAs
  - métricas para la evaluación de resultados
  - funciones estándar para análisis y validación

MOEAs de segunda generación

- Segunda etapa de propuestas de MOEAs (de 2000 a la actualidad).
- La característica fundamental es la incorporación de elitismo a los algoritmos.
- Dos mecanismos principales para la incorporación de elitismo:
  - mediante una selección (μ+λ), por ejemplo en NSGA-II
  - mediante la utilización de una población externa (que almacena soluciones no dominadas), por ejemplo en SPEA y SPEA-2
- Algoritmos representativos de esta generación:
  - NSGA–II, SPEA, SPEA-2, PAES, NPGA–2 y MICRO-GA

MOEAs de segunda generación

- El esquema que utiliza población externa ha sido exitosamente implementado por varios MOEAs.
- Algunos aspectos clave a considerar en este enfoque son:
  - La interacción entre la población y la población externa.
  - Los criterios para realizar las inserciones de elementos en la población externa.
  - Las acciones a tomar cuando se completa la población externa.

MOEAs de segunda generación: SPEA

- Strength Pareto Evolutionary Algorithm (SPEA), propuesto por Zitzler y Thiele.
- Utiliza una población externa de soluciones no dominadas obtenidas previamente.
- Se basa en el concepto de fuerza (strength) que tiene un rol similar al rango de dominancia en otros MOEAs. La fuerza de una solución no dominada es proporcional al número de individuos que domina.
- La asignación de fitness se basa en la medida de "fuerza"

$$S(i) = dominated(i)/(\#pop + 1)$$

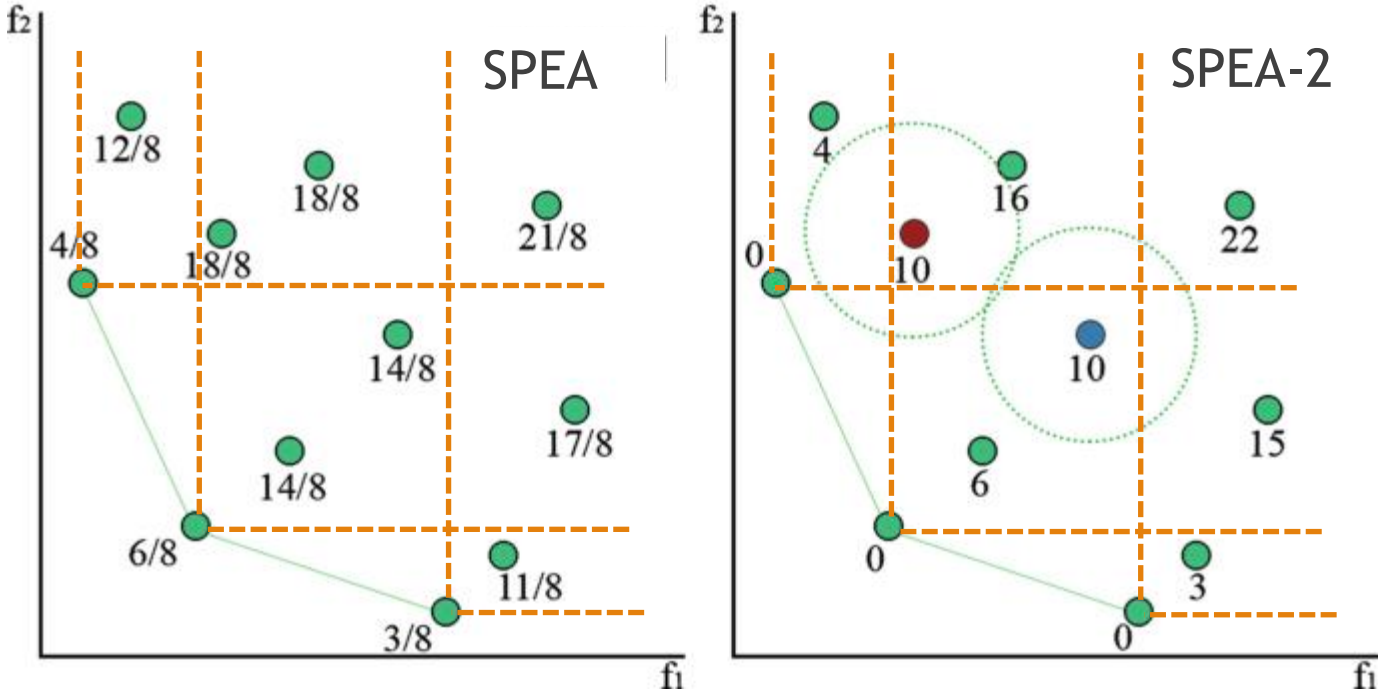
- Utiliza clustering (densidad) para mantener la diversidad.
- El fitness se minimiza.

MOEAs de segunda generación: SPEA-2

- SPEA-2 es una mejora al algoritmo SPEA original
- Presenta tres diferencias sustanciales con SPEA:
  - La asignación de fitness considera la cantidad de individuos dominados por cada solución (como SPEA), pero además toma en cuenta la cantidad de individuos que la dominan.
  - Para mantener diversidad utiliza una técnica de estimación de densidad de individuos vecinos.
  - Utiliza un esquema de truncamiento de la población externa que le garantiza la preservación de soluciones de frontera (bordes).

MOEAs de segunda generación: SPEA y SPEA-2

- Asignación de fitness



MOEAs de segunda generación: PAES

- Pareto Archived Evolution Strategy (PAES), propuesto por Knowles
- Es uno de los métodos más simples, ya que incorpora el elitismo mediante una selección (1+1).
- Utiliza un archivo histórico de individuos no dominados.
- Realiza una división del espacio de funciones objetivo aplicada en forma recursiva, generando una grilla adaptativa.
- Para preservar la diversidad utiliza crowding, evaluando las soluciones en cada uno de los cuadrantes de la grilla.

MOEAs de segunda generación: NSGA-II

- Versión mejorada de NSGA.
- No utiliza población secundaria.
- Incorpora elitismo mediante un esquema de selección (μ+λ).
- El chequeo de dominancia fue mejorado para aumentar la eficiencia computacional.
- En lugar de la técnica de sharing de NSGA, utiliza un mecanismo de crowding que no requiere parámetros.
- En general funciona mejor para representación real que para representación binaria.
- Es uno de los MOEAs que en la práctica ha mostrado mejores resultados.

MOEAs de segunda generación: micro-GA

- Micro Genetic Algorithm for Multiobjective Optimization, propuesto por Toscano y Coello.
- Se caracteriza por utilizar una población muy pequeña e incorporar un mecanismo de reinicialización cuando se alcanza la convergencia.
- Utiliza dos poblaciones, una reemplazable y una no reemplazable con la cual se maneja la dominancia y la diversidad.
- La incorporación del elitismo se basa en la idea de mantener soluciones no dominadas pero con una memoria que se actualiza a plazos.
- Utiliza la grilla adaptativa de PAES.

Líneas de investigación

- En la segunda generación se realizaron avances en la fundamentación teórica de estas técnicas.
- Se realizaron múltiples propuestas de problemas de prueba estándar, dentro de los que destacan:
  - ZDT (Ziztler, Deb, Thiele): proponen un mecanismo para generar problemas de prueba. Diseñan 6 problemas que fueron siguiendo su metodología (ZDT1 a ZDT6).
  - Otros problemas: Ozyczka, Golinski, Viennet.
- Se avanzó en propuestas para mejorar el desempeño computacional (MOEAs paralelos).
- Se propusieron métricas para la evaluación objetiva y sistemática de los resultados obtenidos.

Implementación

- En Python: biblioteca DEAP.
- Biblioteca para computación evolutiva, para prototipo y desarrollo simple de algoritmos de optimización.
- Incluye algoritmos evolutivos para optimización multiobjetivo:
  - NSGA-II
  - Covariance Matrix Adaptation Evolution Strategy
- Notebook de ejemplo:
  https://drive.google.com/drive/u/1/folders/1g3T6B3-_CA5AgZgK4ZCQzjucmngSXasZ