Diplomado de especialización de desarrollo de aplicaciones con Inteligencia Artificial

Optimización industrial con Computación Evolutiva

Sesión 4
Algoritmos Evolutivos

Dr. Soledad Espezua. Ll.
sespezua@pucp.edu.pe

Dr. Edwin Villanueva T.
evillatal@pucp.edu.pe

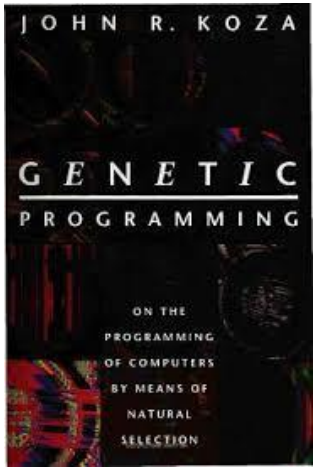
ia.inf.pucp.edu.pe

Agenda

- Programación Genetica
- Differential Evolution
- MOO - Algoritmos NSGA II
- Ejemplos de aplicación

Programación Genética (PG)

- Programación Genética (PG) puede ser vista como un caso particular de Ags que fue ampliamente difundida por J. Koza [1](1989).
- La PG es un tipo de modelo evolutivo en la cual los individuos que se encuentran dentro de la población son programas de computadora cuyo fin es realizar una tarea definida por el usuario.
  - Dichos programas son manipulados y representados como cadenas de caracteres.
- PG difiere de los AGs canónicos en la representación, operadores de reproducción y métodos de evaluación de la función de aptitud.



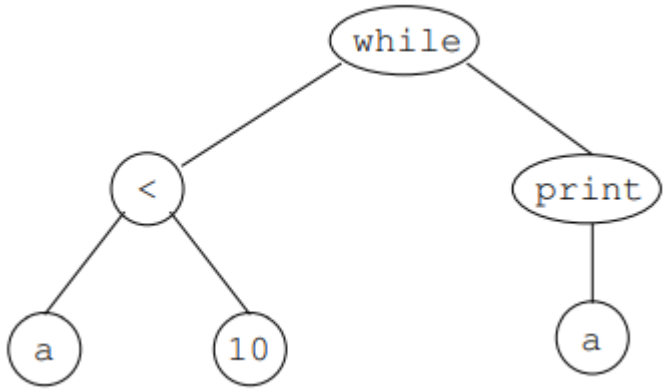
1. KOZA, J. R. Hierarchical genetic algorithms operating on population of computer programs. In: Proceedings of the 11th International Join Conference on Artificial Intelligence (IJCAI-89), Detroid, MI: Morgan Kaufmann, 1989, p. 768–774.

PG - Representación

- Las individuos se codifican como estructuras de árboles. Cada nodo del árbol contiene constantes, variables o parámetros para realizar procedimientos y funciones.
  - Cada nodo interno del árbol tiene una función como operador y cada nodo terminal tiene un operando, por lo que las expresiones matemáticas son fáciles de evolucionar y evaluar.

```
while (a <10) {
        print(a)
        a++
}
```

Segmento de Código

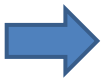


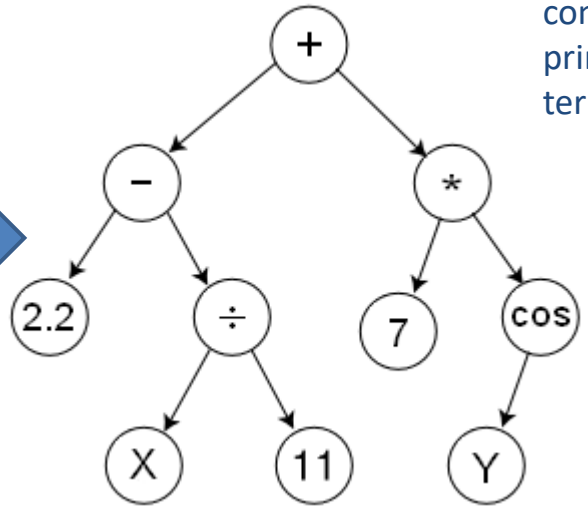
Representación en árbol

PG - Representación

Función

$$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$$





Los nodos internos contienen operaciones primarias y cada nodo terminal tiene un operando

- Se predetermina la máxima profundidad de los árboles, pero no su topología
- El tamaño, forma y contenido de los árboles puede cambiar dinámicamente durante el proceso evolutivo.

PG - Representación

- El *conjunto de nodos terminales* está compuesto por las entradas posibles al individuo, variables y constantes.

  Ejemplo: {x_1,x_2,y,a,b,1…9,etc}

- El *conjunto de nodos internos* está compuesto por operadores y funciones que pueden componer a un individuo.

  Ejemplo:

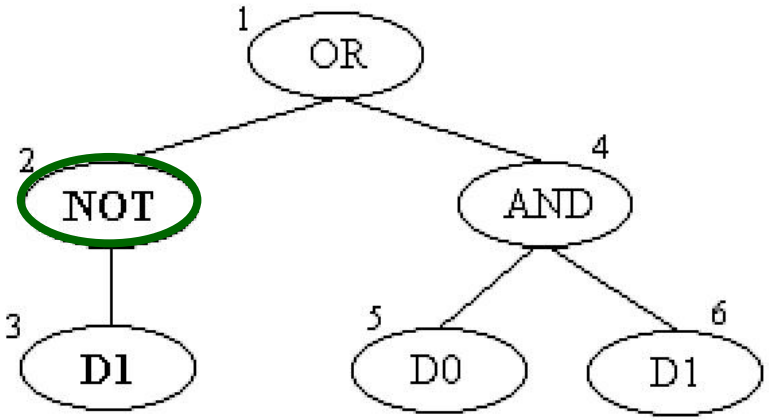
  - Operadores aritméticos (+,−,*, etc.)
  - Operadores booleanos (AND, OR, NOT)
  - Operadores condicionales (IF-THEN-ELSE)
  - Operadores de iteraciones/ recursión (DO-UNTIL, WHILE)
  - Funciones matemáticas (seno, coseno, exp, log)
  - Cualquier función específica del dominio definido

PG - Cruzamiento

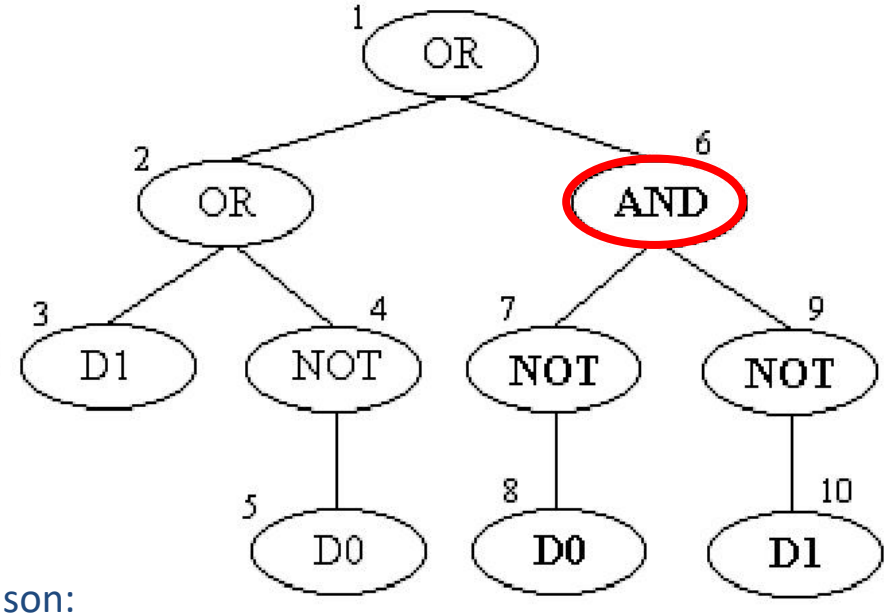
- Cruzamiento: se intercambian partes de los árboles eligiendo al azar un nodo como punto de corte en cada árbol.

Ejemplo: Punto de cruce en el Padre1 = nodo 2, y en el Padre2= nodo 6.

Padre1



Padre2



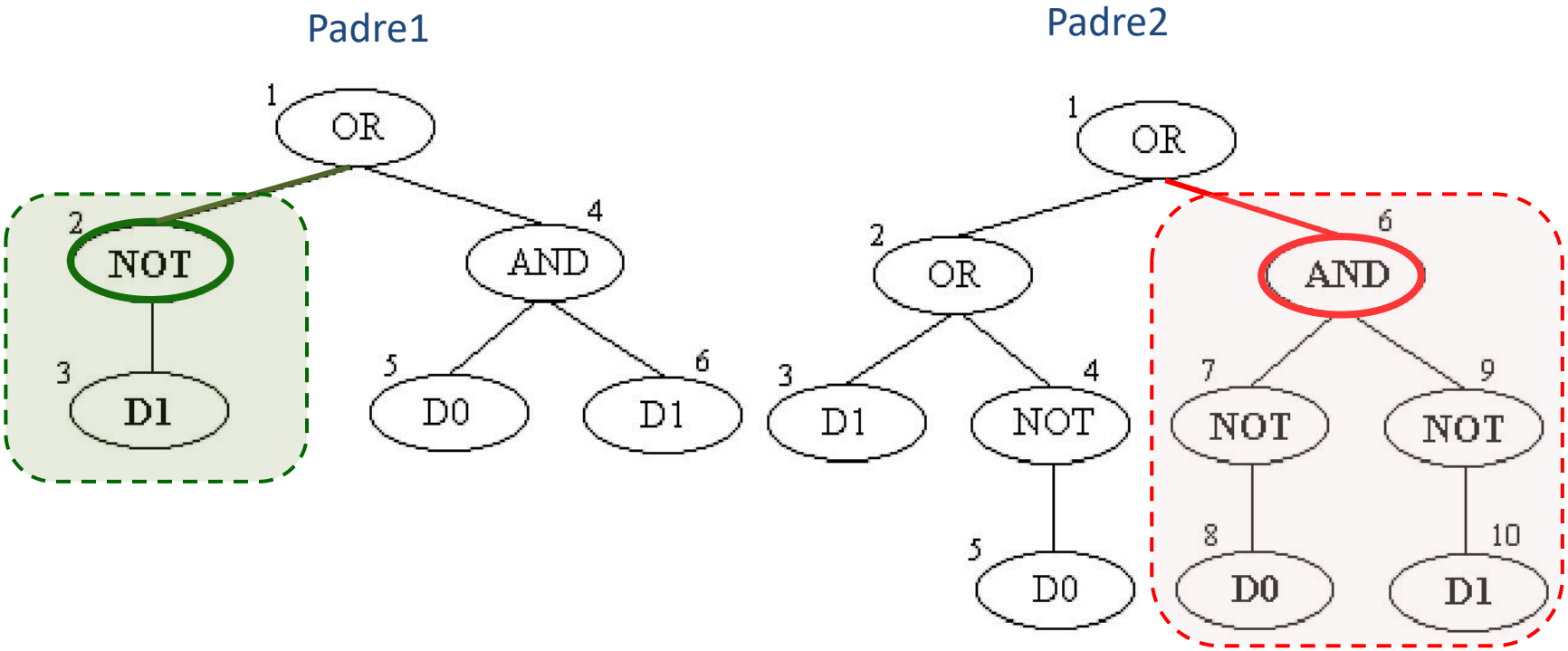
Los individuos (cromosomas) correspondientes son:

Padre1: (OR (NOT D1) (AND D0 D1))

Padre2:(OR (OR D1 (NOT D0)) (AND (NOT D0) (NOT D1)))

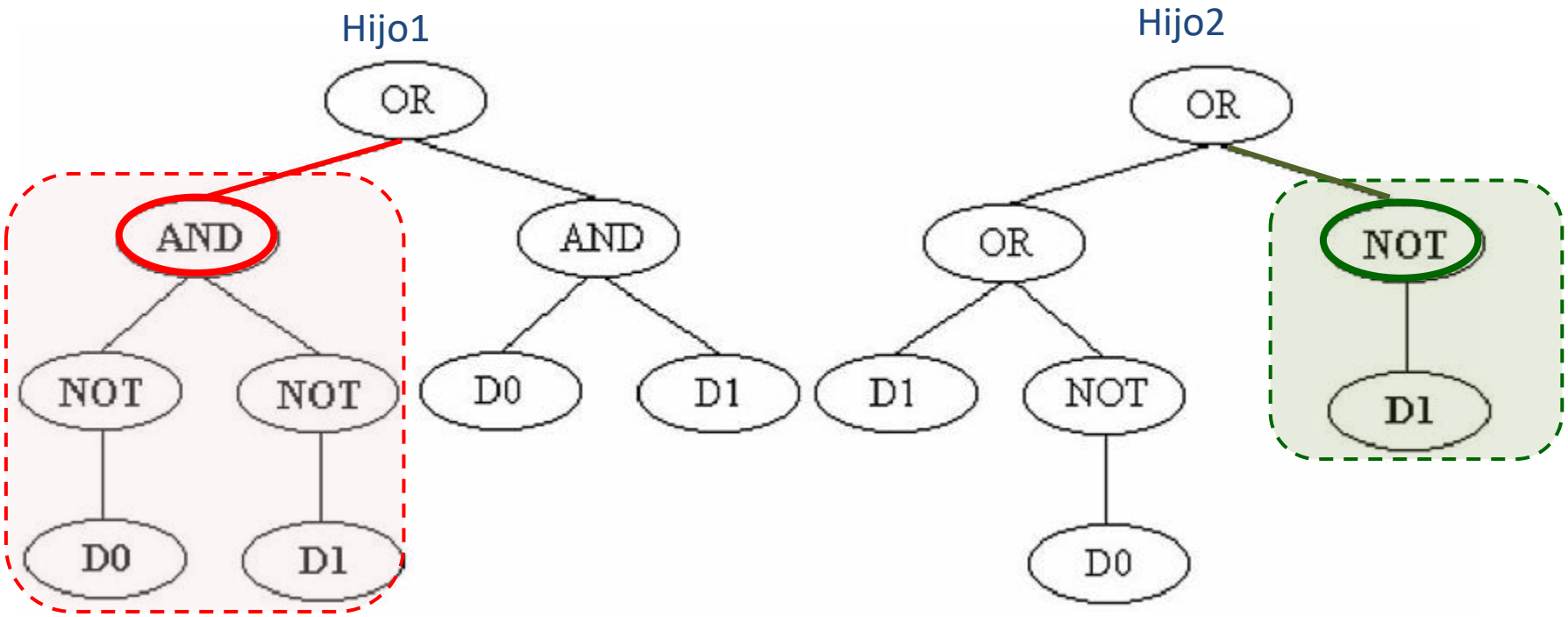
PG - Cruzamiento

- El primer hijo se produce quitándole al primer padre el fragmento indicado por el punto de cruza e insertando el fragmento (sub-árbol) correspondiente del segundo padre.
- El segundo hijo se produce de manera análoga.



PG - Cruzamiento

- Entonces los hijos resultantes serán:

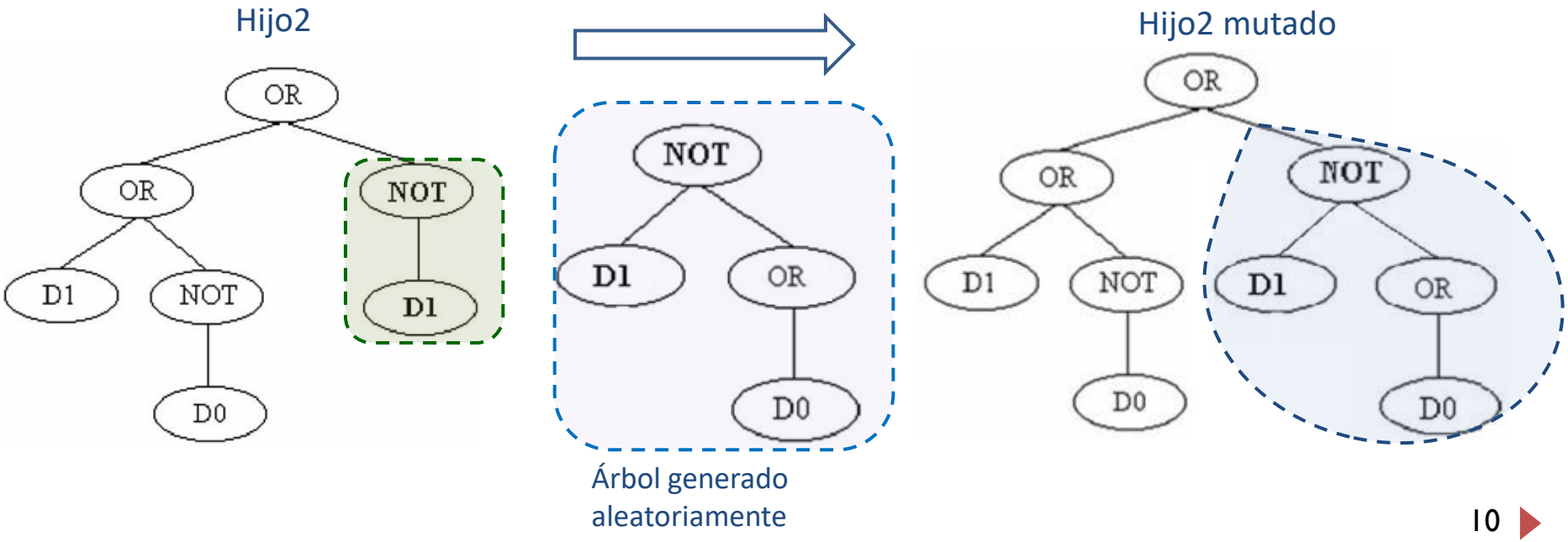


PG - Reproducción

- Mutación: actúa sobre un solo individuo, generalmente, uno resultante de un cruzamiento.
- La mutación actúa con una probabilidad colocada muy baja y que es un parámetro del algoritmo de PG.

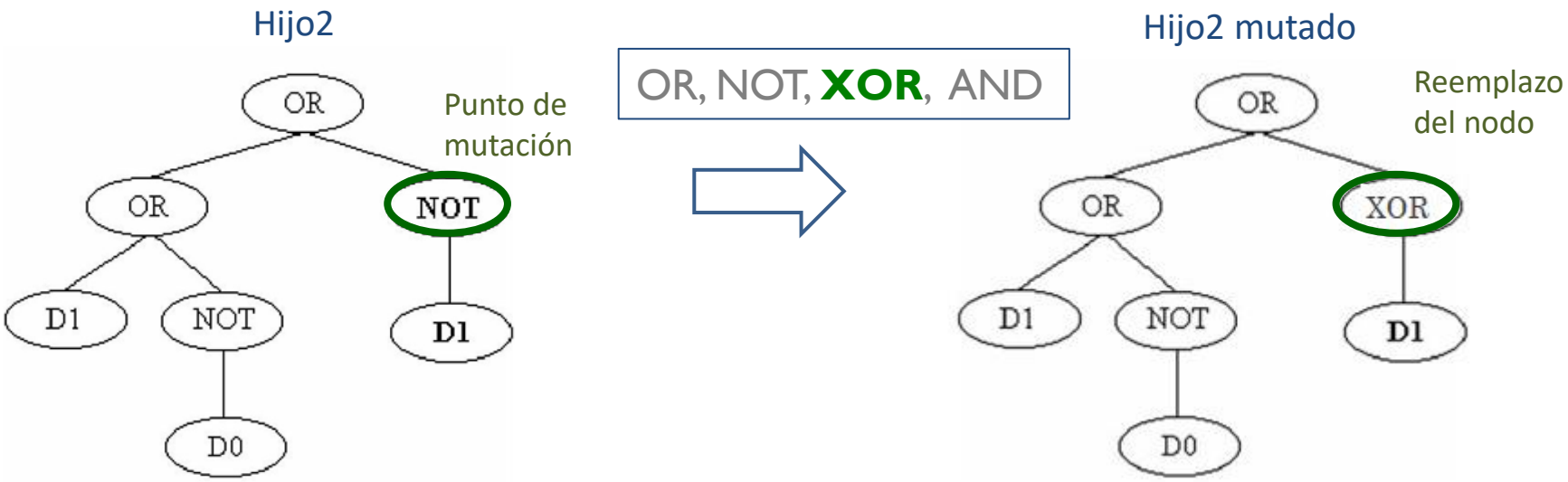
Tipos de Mutación:

1. Mutación Subárbol*:* consiste en escoger en forma aleatoria un nodo del árbol y crear bajo este, otro subárbol con restricciones de profundidad.



PG - Reproducción

2. Mutación Puntual*:* consiste en seleccionar en forma aleatoria un nodo del árbol y luego substituirlo por otro nodo del mismo tipo (terminal o nodo interno).



PG - Reproducción

- Otros tipos de mutación:

Nombre del operador	Descripción del efecto
Permutación	Los argumentos de un nodo son permutados
Levantamiento	Nuevo individuo es generado a partir de un subárbol
Expansión	Un terminal es cambiado por un árbol generado al azar
Colapso	Subárbol es intercambiado por un terminal
Duplicación de Gen	Subárbol es reemplazado por un terminal al azar.

PG - Evaluación

- El *fitness* mide qué tan buena es una solución en la evolución simulada del PG.
- Si el objetivo es aprender una expresión simbólica, la ejecución del programa representado por el árbol, es el *fitness*. Si resuelve el problema propuesto o se acerca a la respuesta correcta, tendrá un alto valor; de lo contrario, su *fitness* será bajo.
- Si el objetivo es aprender una función matemática g, entonces una medida del *fitness* puede ser el error cuadrático medio.

$$fitness = \frac{1}{M}\sum_{i=1}^{M}(real_i - estimado_i)^2$$

donde $real_i$ es la salida real del programa y $g(i) = estimado_i$ es la salida estimada ante la entrada i.

- En general, los algoritmos de PG utilizan solo el operador de cruzamiento en el proceso de búsqueda de las mejores soluciones.

PG - Selección

- La **selección** es el proceso por el cual se escogen individuos para transmitirlos de una generación a otra.

Tipos de selección (para N individuos).

- Selección por *ruleta*:

- Repetir hasta completar una población de N individuos:
  1. Seleccionar a dos individuos padres, privilegiando los de mejor *fitness*.
  2. Aplicar cruzamiento
  3. Aplicar mutación

- Selección por *torneo*:

- Dividir aleatoriamente la población en dos grupos de N/2 individuos.
  1. Seleccionar al mejor individuo del primer grupo (padre), y al mejor individuo del segundo (madre).
  2. Aplicar cruzamiento.
  3. Aplicar mutación
  4. Los dos nuevos individuos reemplazan a los peores de cada uno de los grupos.

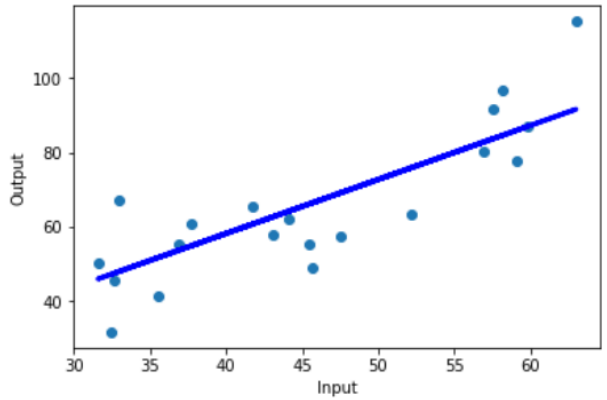
PG - Consideraciones

- Criterio de parada: por lo general un algoritmo de PG se detiene cuando el mejor individuo de la población tiene un *fitness* aceptable.
- Para resolver un problema con PG, es necesario:

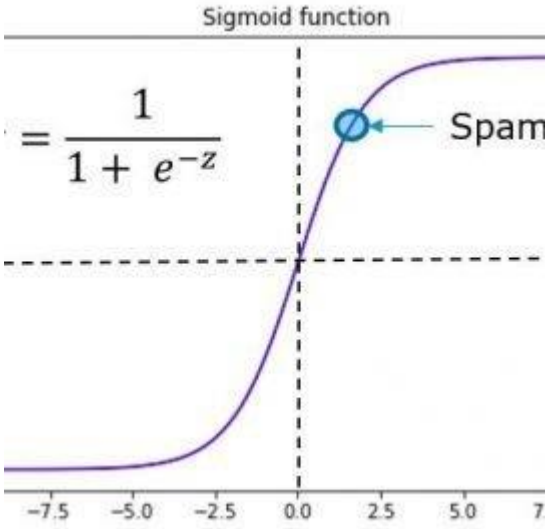
1. Definir el conjunto de nodos terminales.
2. Definir el conjunto de nodos internos (funciones).
3. Definir la función de aptitud.
4. Definir parámetros tales como tamaño de la población, tamaño máximo de un individuo, probabilidad de cruzamiento y mutación, método de selección y criterio de parada del algoritmo.

PG – Aplicación

- La principal aplicación de PG es en Regresión Simbólica.
  - Las técnicas de regresión simbólica permiten obtener expresiones matemáticas $\boldsymbol{g}$ que modelen el comportamiento de un sistema a partir de un conjunto de pares entrada-salida.
- La regresión numérica clásica necesita que la estructura de la expresión sea fijada a priori (regresión lineal, cuadrática, exponencial, logarítmica, etc.).
  - El objetivo es encontrar los coeficientes numéricos de dicha expresión usando alguna técnica de optimización numérica.



$Y = b_0 + b_1 * x$



PG – Aplicación

- La regresión simbólica permite obtener tanto la estructura de la expresión como los valores de los coeficientes de un modo automático.

Expresión simbólica

$y = a.x + b$

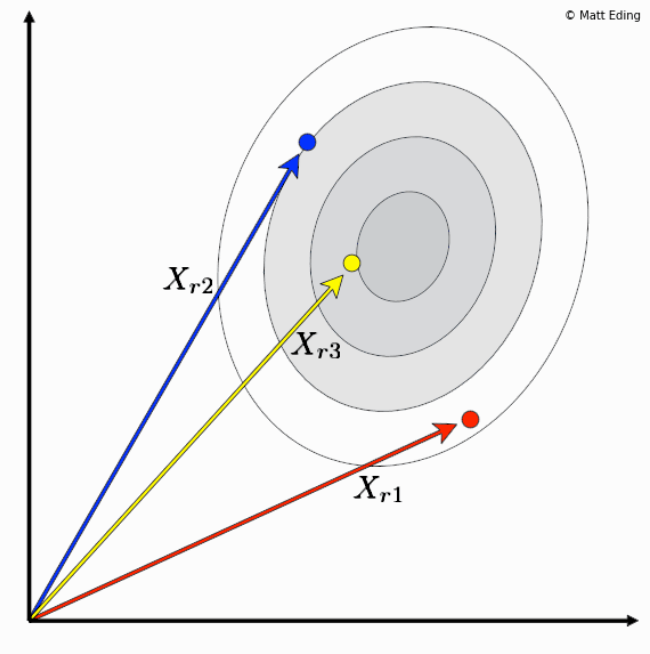
Variable independiente X	Variable dependiente Y
-1	1
-0.8	0.84
-0.6	0.76
:	:
:	:
1	3

Valores de coeficientes

a =1.197
b = 0.109

- De este modo PG, es muy útil para la identificación de sistemas no lineales.

Differential Evolution (DE)



Differential Evolution (DE)

- Fue introducida por R. Storn y K. Price en 1995[1], para optimización sobre espacios continuos, en un intento por resolver el problema de ajuste de polinomios de Chebychev.

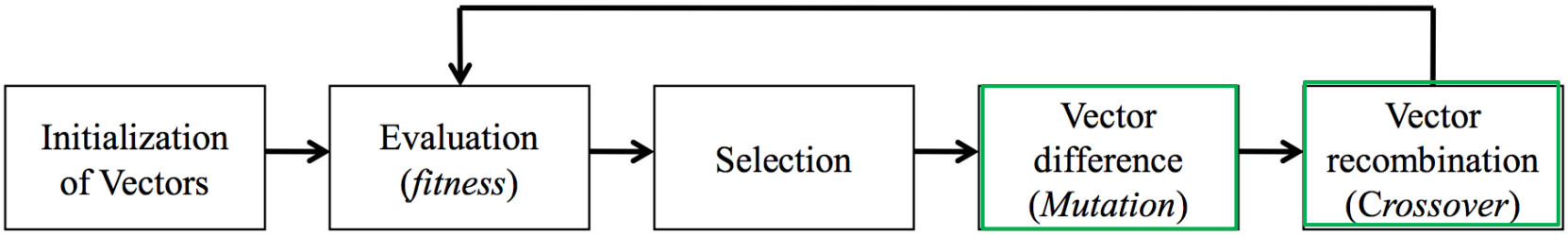
Rainer Storn [2]

Kenneth Price

1. *R. Storn and K. V. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," ICSI, USA, Technical Report TR-95-012, 1995 .*
2. *R. Storn,1997. Differential Evolution, A simple and efficiente heuristic of strategy for global optimization over continuous spaces. Journal of Global Optimization, 11 (1997) 341-359.*

DE - Características

- Es un modelo evolutivo de optimización perteneciente a los algoritmos de CE, para resolver problemas de optimización continua.
- DE comparte similitudes con los AGs tradicionales, pero no usa la codificación binaria.
  - La población en DE es un conjunto de vectores de valores reales, donde cada vector representa una posible solución al problema.
- DE comparte varias características con el ciclo básico de un AG. Después de usar selección, DE enfatiza en la mutación y continua con el cruzamiento sobre el vector mutado.



- Presenta una forma de generar vectores, los cuales compiten con los individuos de la población actual a fin de sobrevivir.

DE - Procedimiento

- Una población $X_{n\times d}$ de tamaño "n", inicia con x_i vectores aleatorios d-dimensionales $(i = 1, \dots, d)$. La población se genera de la siguiente manera:

$$X_{n\times d} \leftarrow \cup\ (inf, sup)$$

$$x_i = inf + rand(0,1).(sup - inf)$$

donde: inf y sup son límites de acotamiento.

- Después de evaluar la población, escoger 3 vectores (x_1, x_2, x_3) aleatoriamente.
- Luego escoger aleatoriamente un vector objetivo (x_i) para ser reemplazado.

DE - Procedimiento

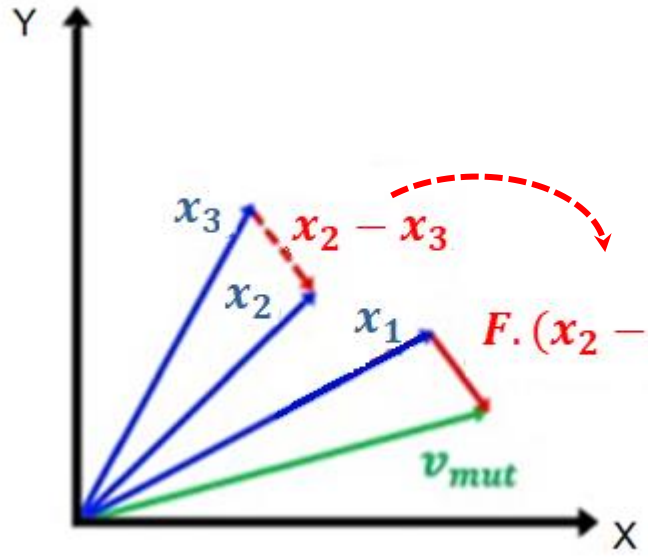
- **Mutación**

Genera un vector mutado (v_i) con el siguiente procedimiento:

$$v_i = x_1 + F.(x_2 - x_3) \qquad (1)$$

donde:

- $F \in [0,2]$ es un factor de amplificación de valor real.



$$v_{mut} = x_1 + F.(x_2 - x_3)$$

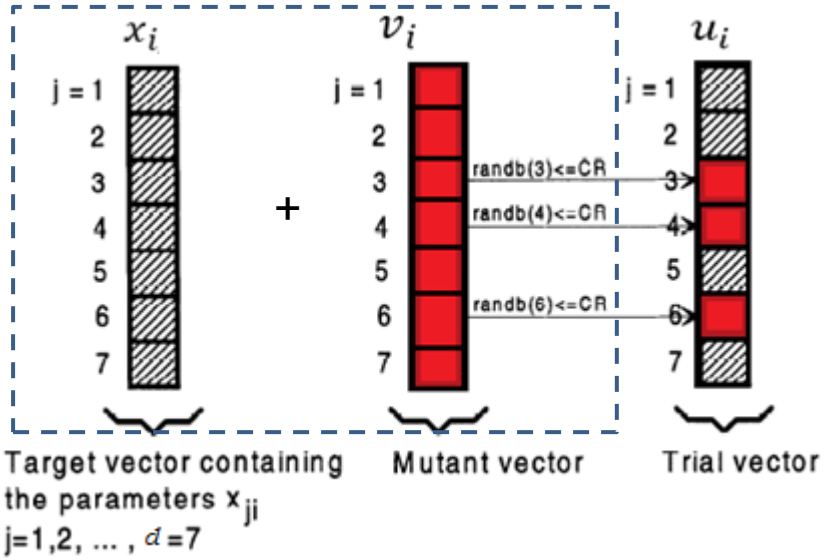
$$v_{mut} = v_i$$

DE - Procedimiento

Cruzamiento

Cruzamiento genera un vector trial (u_i) que incrementa la diversidad:

$$u_{ji} = \begin{cases} v_{ji}, & if\ (randb(j) \leq CR)\ or\ j = rnbr(i) \\ x_{ji}, & if\ (randb(j) > CR)\ or\ j \neq rnbr(i) \end{cases} \quad (2)$$



Cruzamiento uniforme

donde:

- $j = 1,2, \dots, d$ -> tamaño del vector
- $randb(j) \in [0,1]$ (real)
- CR $\in [0,1]$ taza de cruzamiento (real)
- $rnbr(i) \in [1, d]$ escoge un número aleatorio (entero)

DE - Procedimiento

- **Selección**

  Para decidir si el nuevo vector debe o no convertirse en un miembro de la generación G + 1, se compara el *fitness* del vector trial (u_i) con el *fitness* del vector objetivo x_i .

  If (u_i. fitness > x_i. fitness) (3)

  $x_i \leftarrow u_i$

DE - Pseudocódigo

Differential Evolution

Initialize the population $\boldsymbol{x}$ with randomly generated solutions
Set the weight $F \in [0, 2]$ and crossover probability $C_r \in [0, 1]$
while (stopping criterion)
  for $i = 1$ to n,
  For each $\boldsymbol{x}_i$, randomly choose 3 distinct vectors $\boldsymbol{x}_p$, $\boldsymbol{x}_r$ and $\boldsymbol{x}_r$
  Generate a new vector $\boldsymbol{v}$ by DE scheme (1)
  Generate a random index $J_r \in \{1, 2, ..., d\}$ by permutation
  Generate a randomly distributed number $r_i \in [0, 1]$
  for $j = 1$ to d,
  For each parameter $\boldsymbol{v}_{j,i}$ (jth component of $\boldsymbol{v}_i$), update

$$\boldsymbol{u}_{j,i}^{t+1} = \begin{cases} \boldsymbol{v}_{j,i}^{t+1} & \text{if } r_i \leq C_r \text{ or } j = J_r \\ \boldsymbol{x}_{j,i}^{t} & \text{if } r_i > C_r \text{ and } j \neq J_r, \end{cases} \quad (2)$$

  end
  Select and update the solution by (3)
  end
end
Post-process and output the best solution found

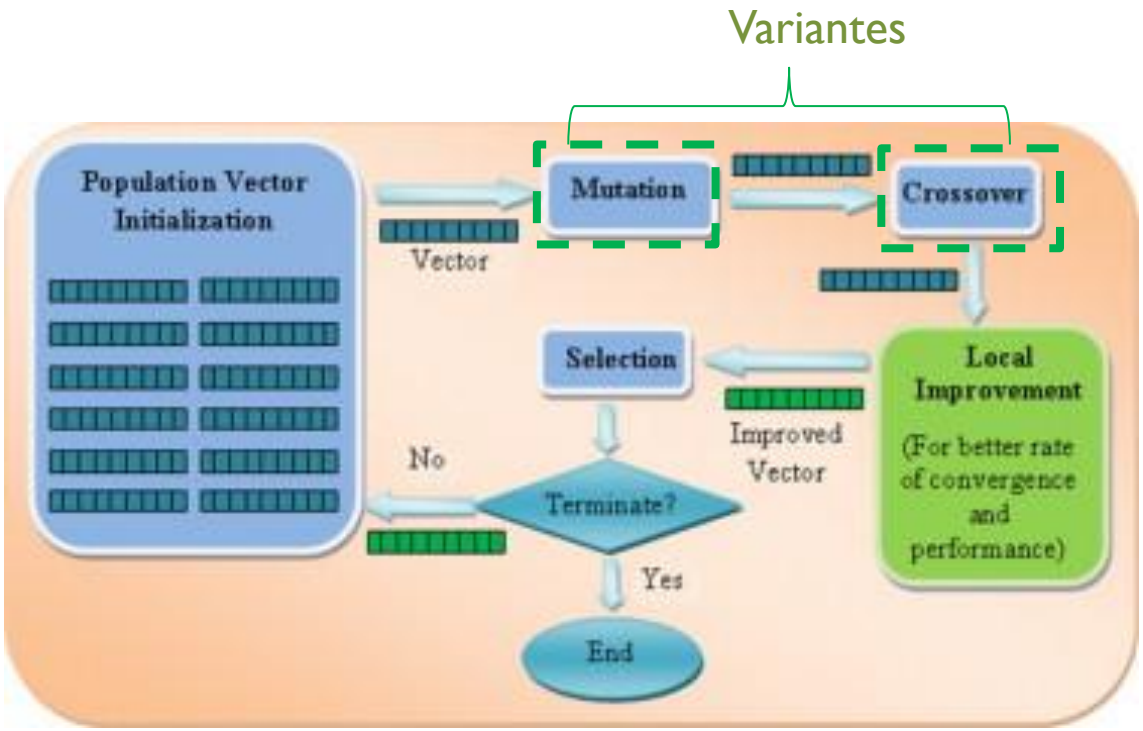
DE - Consideraciones

- Es importante notar que, incrementando ya sea el tamaño de la población o el número de parejas de soluciones usadas para calcular los valores de mutación, también se incrementa la diversidad de posibles movimientos, promoviendo la exploración del espacio de búsqueda. Sin embargo, la probabilidad de encontrar la dirección de búsqueda correcta decrece considerablemente.

  - Así, el balance entre el tamaño de la población y el número vectores seleccionados para realizar mutación, determina la eficiencia del algoritmo.

DE - Consideraciones

- Otro factor importante en DE es la selección de las **variantes**(un esquema de **diferentes nomenclaturas para DE**).
  - Las diferencias de una variante a otra son la forma en la que los individuos son seleccionados para calcular el vector de **mutación** y el operador de **cruzamiento**.



DE - Variantes

1. **“DE/rand/1/bin”, “DE/rand/1/exp”, “DE/best/1/bin” y “DE/best/1/exp”**

- “DE”: significa *Differential Evolution*
- “rand”: seleccionar todos los individuos para calcular la mutación de forma aleatoria
- “1” : es el número de pares de soluciones escogidas
- “best”: usar la mejor solución en la población además de las aleatorias.

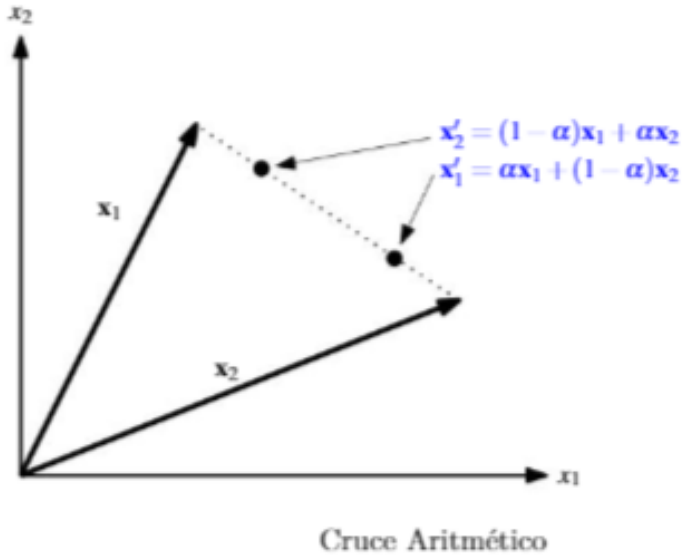
Usa cruzamiento discreto, siempre sobre dos individuos, el padre original y el vector de mutación. Variantes:

- “bin”: binomial, cada variable del hijo se toma en cada ocasión de uno de los dos padres, con base en el valor del parámetro “CR”.
- “exp”: exponencial, cada variable del hijo se toma del primer padre hasta que un número aleatorio rebasa el valor de “CR”.

DE - Variantes

2. **"DE/current-to-rand/1" y la "DE/current-to-best/1"**

Usa cruzamiento aritmético y a diferencia del discreto, es invariante a la rotación.



Cruce Aritmético

$z_i = \alpha\, x_i + (1 - \alpha)\, y_i$, donde $0 \leq \alpha \leq 1$

- "current-to-rand": selecciona los individuos para la mutación de forma aleatoria
- "current-to-best": usa la mejor solución en la población, además de soluciones aleatorias.

DE - Variantes

3. **“DE/rand/2/dir”**

   Incorpora información de la función objetivo a los operadores de mutación y cruzamiento.

   El objetivo de este enfoque es guiar la búsqueda a zonas prometedoras más rápido que en la DE tradicional.

   “2” : numero de pares de soluciones elegidas. Los autores argumentan que se obtienen los mejores resultados cuando eligen 2.

4. **“DE/current-to-rand/1/bin”**

   Usa una combinación de cruzamiento discreto- aritmético.

DE - Aplicaciones

- DE es uno de los campos más activos en el desarrollo de algoritmos evolutivos para optimización continua.

Control Systems and Robotics	***Bioinformatics***
System identification	*Gene regulatory networks*
Optimal control problems	*Micro-array data analysis*
Controller design and tuning	*Protein folding*
Aircraft control	*Bioprocess optimization*
Nonlinear system control	***Artificial Neural Networks***
Simultaneous localization and modeling problem	*Training of feed-forward ANNs*
Robot motion planning and navigation	*Training of wavelet neural networks*
Cartesian robot control	*Training of B-Spline neural networks*
Multi-sensor data fusion	***Chemical Engineering***
Pattern Recognition and Image Processing	*Chemical process synthesis and design*
Data clustering	*Phase equilibrium and phase study*
Pixel clustering and region based image segmentation	*Parameter estimation of chemical process*
Feature extraction	
Image registration and	
enhancement	
Image Watermarking	

Bibliografía DE

- Raúl Benítez, Gerard Escudero, Samir Kanaan. Inteligencia artificial avanzada.
- R. Storn,1997. Differential Evolution, A simple and efficiente heuristic of strategy for global optimization over continuous spaces. Journal of Global Optimization, 11 (1997) 341-359.
- R. Storn and K. V. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," ICSI, USA, Technical Report TR-95-012, 1995 [Online]. Available: http://icsi.berkeley.edu/~storn/litera.html

Review DE:

- Swagatam Das, Sankha Subhra Mullick, P.N. Suganthan. Recent advances in differential evolution – An updated survey. Swarm and Evolutionary Computation, Volume 27, 2016, Pages 1-30.
- Guohua Wu, Rammohan Mallipeddi, P.N. Suganthan, Rui Wang, Huangke Chen. Differential evolution with multi-population based ensemble of mutation strategies. Information Sciences, Volume 329, 2016, Pages 329-345.

Optimización Multiobjetivo (MOO)

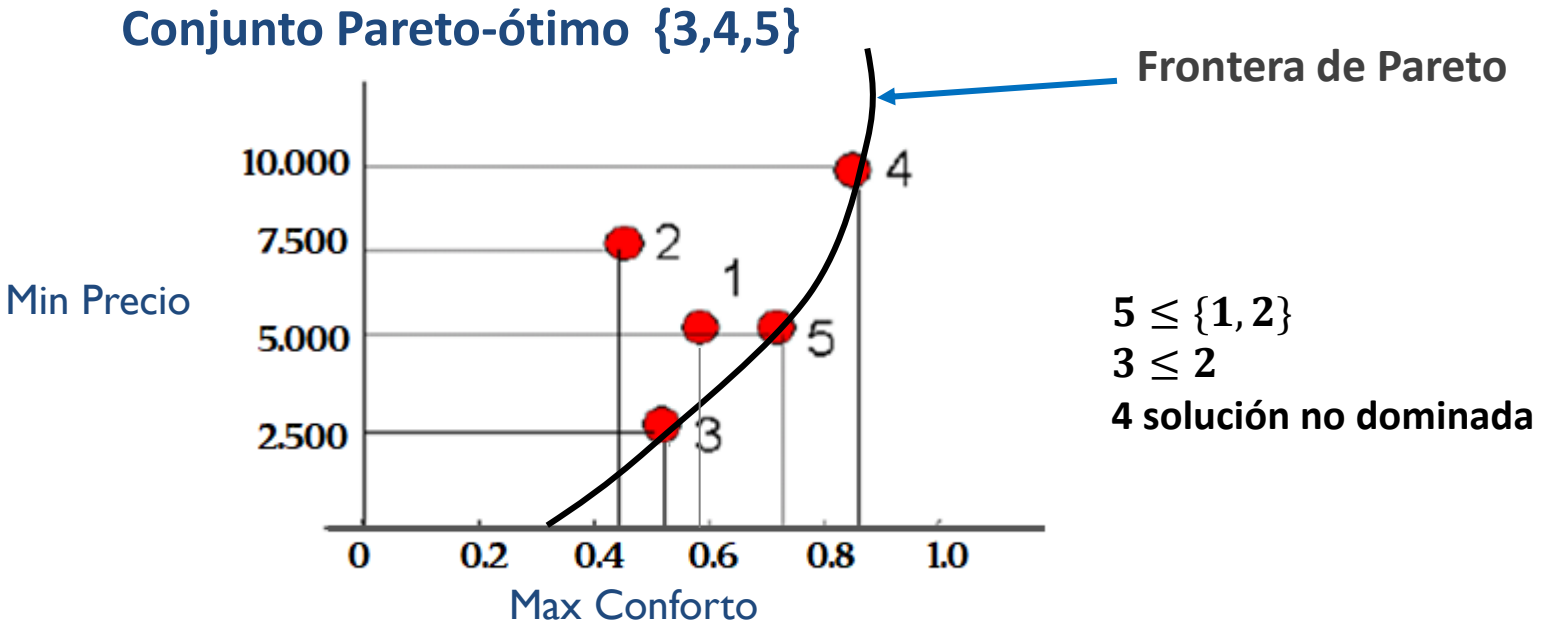
Definición [Andrzej Osyczka]

"Es encontrar un vector de variables de decisión, que satisface restricciones y optimiza un vector de funciones cuyos elementos representan funciones objetivas. Estas funciones generalmente están en conflicto entre sí.

Entonces, "optimizar" significa encontrar una solución de todas las funciones objetivas que proporcione valores aceptables para el desarrollador ".

MOO - Problema Multiobjetivo

- Un problema multiobjetivo tiene un conjunto de soluciones eficientes que no pueden ser consideradas diferentes entre sí. Este conjunto de soluciones se le denomina Frontera de Pareto.

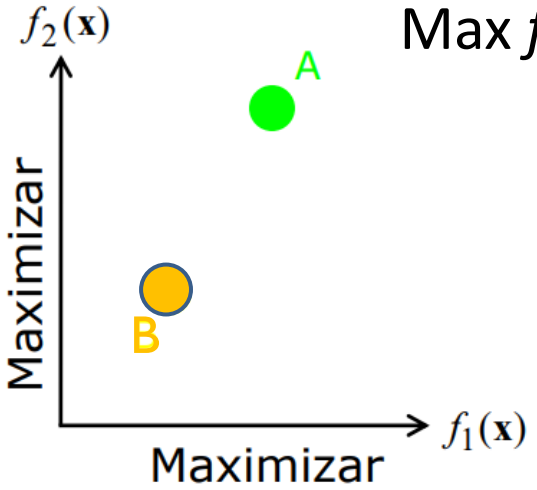


$\mathbf{5} \leq \{\mathbf{1}, \mathbf{2}\}$
$\mathbf{3} \leq \mathbf{2}$
4 solución no dominada

Una solución es Pareto-óptima cuando es *no dominada* (es al menos tan buena como las otras en todos los objetivos o es mejor en al menos uno de ellos).

MOO – Concepto de dominancia

- Una solución x* ∈ $\mathfrak{I}$ es Pareto-óptima cuando es *no dominada* por ninguna otra solución.

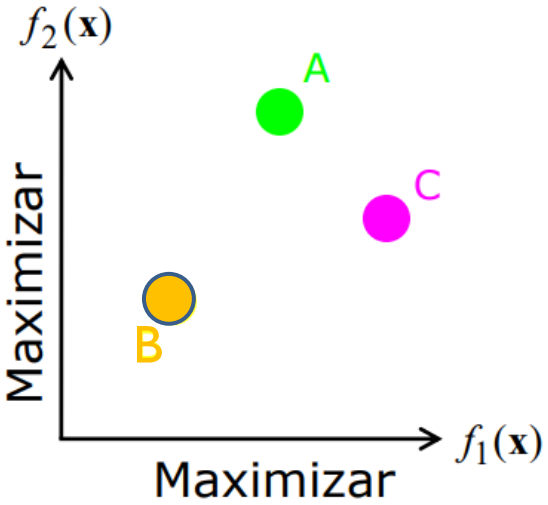


Max $f(x) = (f_1(x), f_2(x))$

A domina a B

B es dominada por A

(A es mejor que B)



A y C son no dominadas entre sí

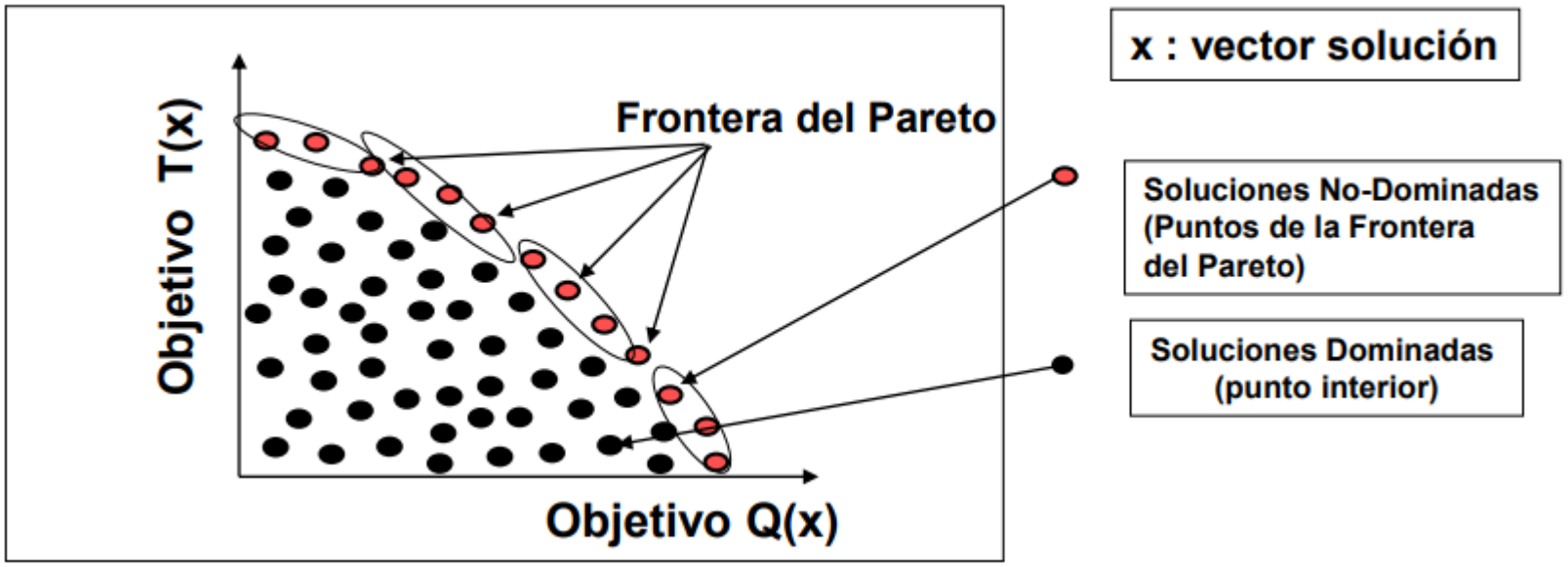
(ninguna domina a la otra)

Las dos dominan a B

MOO – Concepto de dominancia

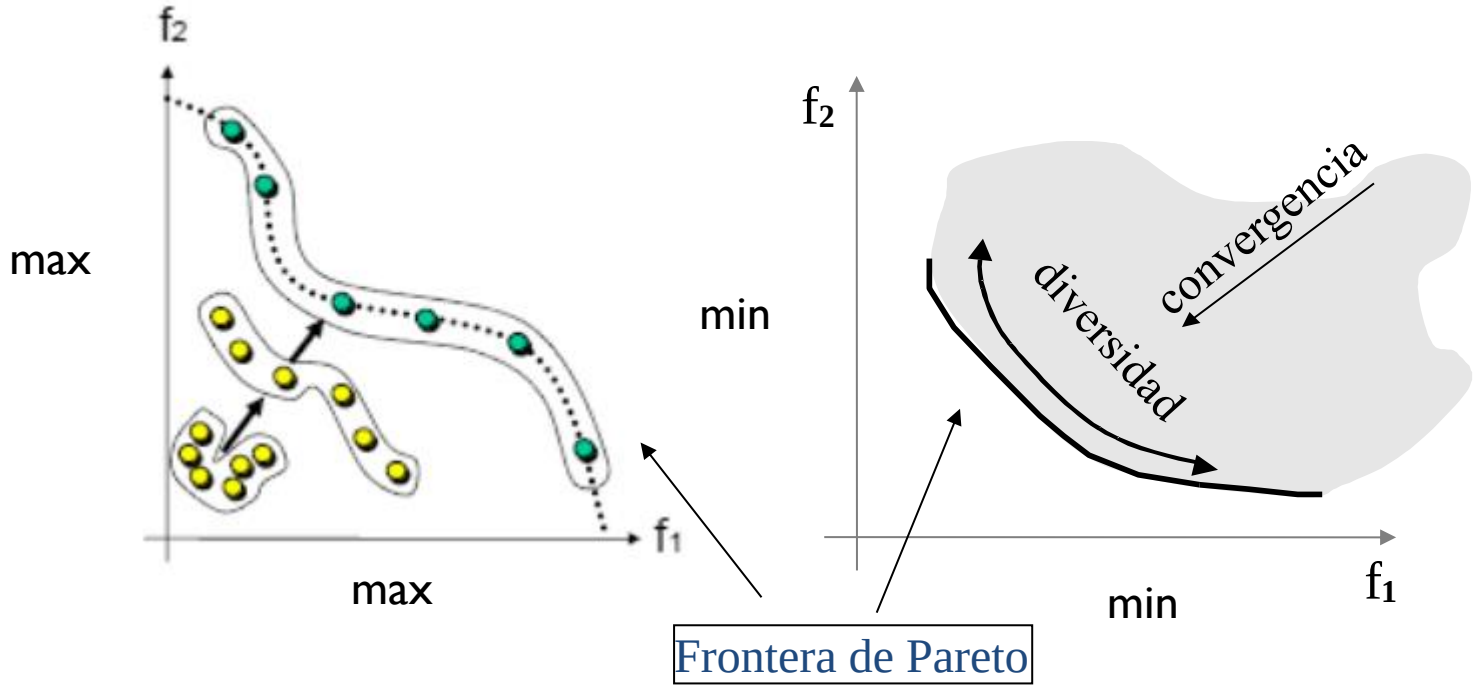
- Para que una solución domine a otra, esta necesita ser estrictamente mejor en al menos un objetivo, y no peor en ninguno de ellos.
- El conjunto de todas las soluciones *no dominadas* $x^* \in \mathfrak{I}$ es el conjunto Pareto-óptimo y compone la solución óptima del problema multiobjetivo.
- La ***frontera de Pareto*** consta de los *fitness (valores de las funciones objetivo f(x*)*), de cada vector x* del conjunto de Pareto-óptimos.

Ejemplo: Frontera del Pareto para [Max Q(x), Max T(x)]



Objetivo de MOO

- Encontrar el conjunto de soluciones que estén lo más próximo posible de la Frontera de Pareto.
- Encontrar el conjunto de soluciones con la mayor diversidad posible.



NSGA II

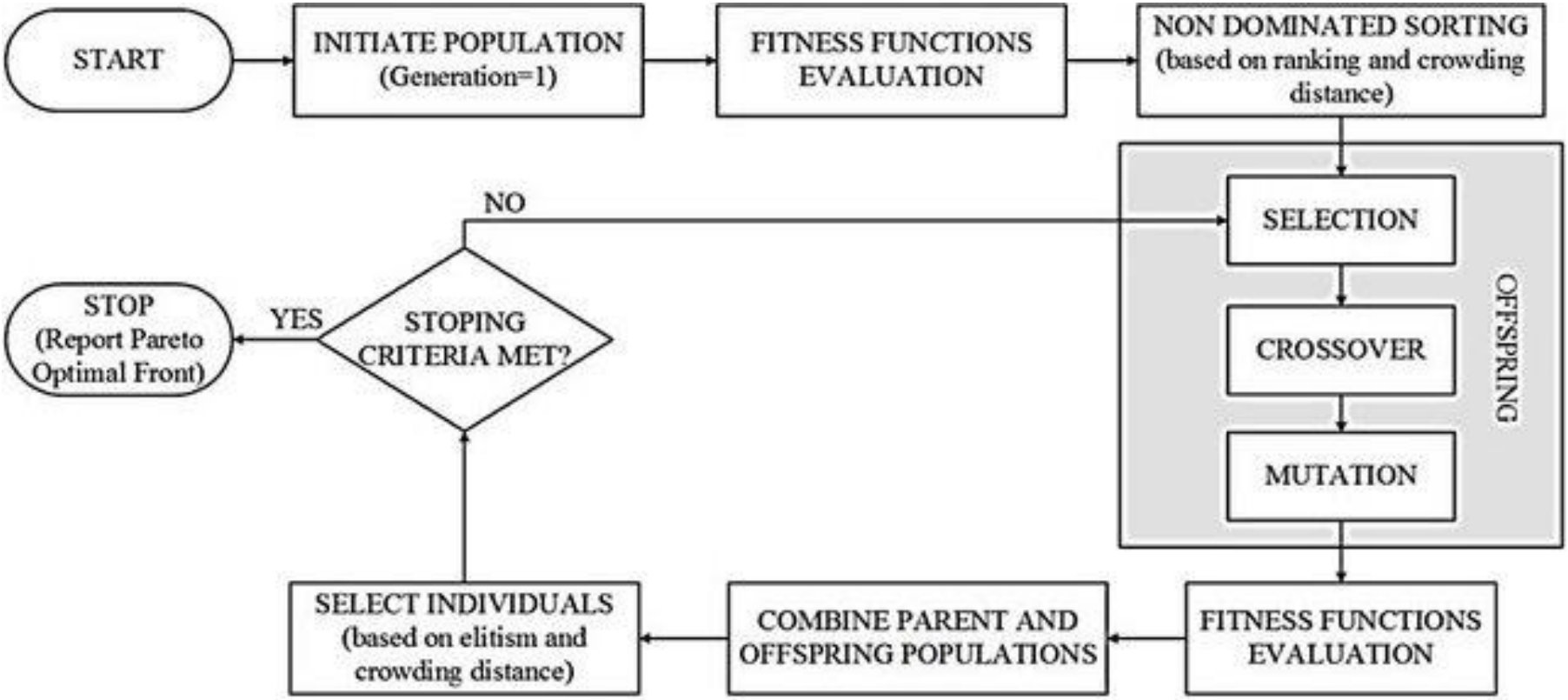
Nondominated Sorting Genetic Algorithm (NSGA-II)

NSGA-II [1], fue propuesto por Kalyanmoy Deb y sus estudiantes. Es una versión mejorada del NSGA[2] y tiene las siguientes características:

- Combina la población actual con la siguiente población, conservando las mejores soluciones de ambas.
- Utiliza un operador de distancia (*crowding distance)* que no requiere parámetros.
- Usa elitismo, que lo hace mucho más eficiente (computacionalmente) que NSGA (descarta soluciones no-dominadas en la iteración actual).

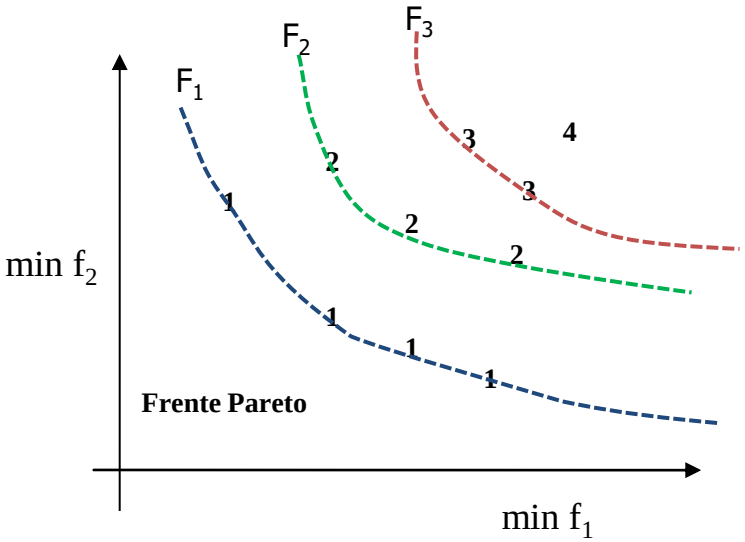
1. DEB, K.; AGRAWAL, S.; PRATAB, A.; MEYARIVAN, T. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.
2. SRINIVAS, N.; DEB, K. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation, v. 2, n. 3, p. 221–248, 1994.

NSGA II


START
INITIATE POPULATION (Generation=1)
FITNESS FUNCTIONS EVALUATION
NON DOMINATED SORTING (based on ranking and crowding distance)
SELECTION
CROSSOVER
MUTATION
OFFSPRING
FITNESS FUNCTIONS EVALUATION
COMBINE PARENT AND OFFSPRING POPULATIONS
SELECT INDIVIDUALS (based on elitism and crowding distance)
STOPING CRITERIA MET?
NO
YES
STOP (Report Pareto Optimal Front)

NSGA II - Asignación del *fitness*

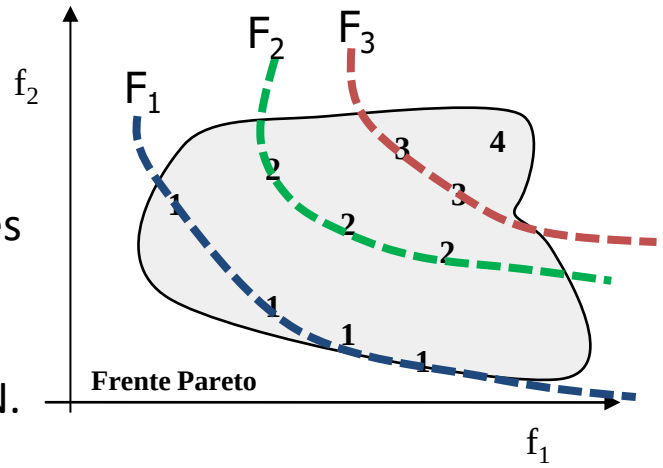
- Identificar los individuos no dominados de una población y colocarles el mismo *fitness*.
  - Esto implica que todas las soluciones en dicha categoría tienen la misma probabilidad de reproducirse.
- Remover los individuos ya asignados a un categoría (*rank*) e identificar un nuevo conjunto de soluciones no dominadas.
  - A este nuevo grupo se le asigna el mismo *fitness* pero mayor que el rank anterior.
- El proceso continua hasta que todos los individuos de la población son asignados a una categoria.



NSGA II- Ordenamiento por no dominancia

NSGA

- Seleccionar N individuos divididos en categorías, de la siguiente manera:
  - todas las soluciones no dominadas pertenecientes al ranking 1,
  - las restantes siguientes soluciones no dominadas pertenecientes al ranking 2
  - Y continuar así hasta completar N.



- Después , realizar cruzamiento y mutación hasta que se tenga una nueva población de N individuos.
- Luego repetir el ciclo, combinando padres e hijos usando **elitismo**.

NSGA II

- Para distinguir soluciones con un mismo ranking se calcula la distancia de amontonamiento (*crowding distance*).

NSGA II - Crowding distance

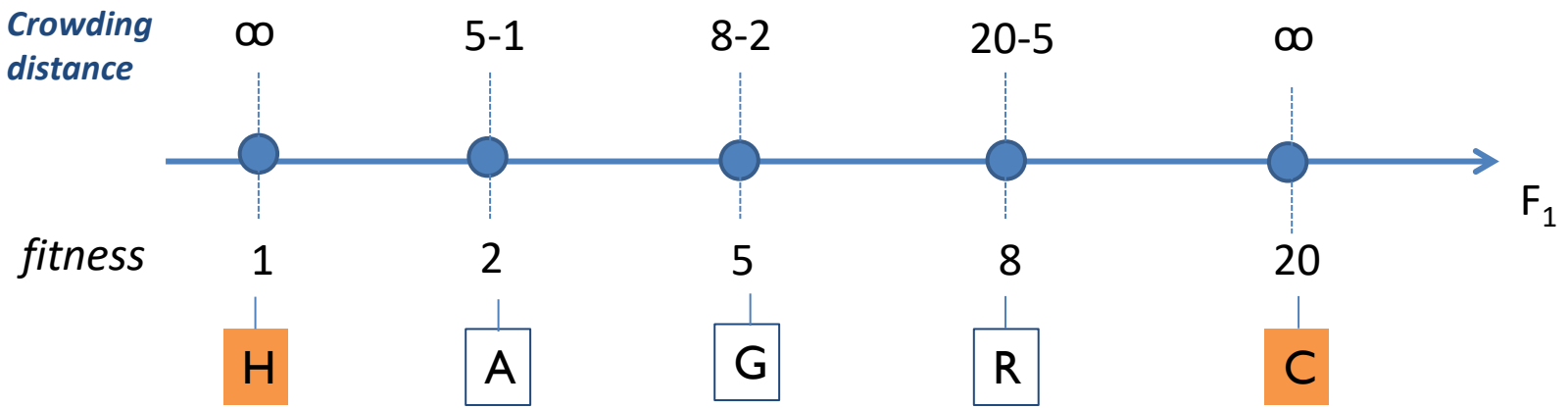
Es una medida de "densidad de las soluciones que rodean una solución particular en la población". Pasos:

- Primero se ordena la población según el valor (*fitness*) de cada función objetivo en orden ascendente.
- A los primeros y últimos individuos en el rank se les asigna:

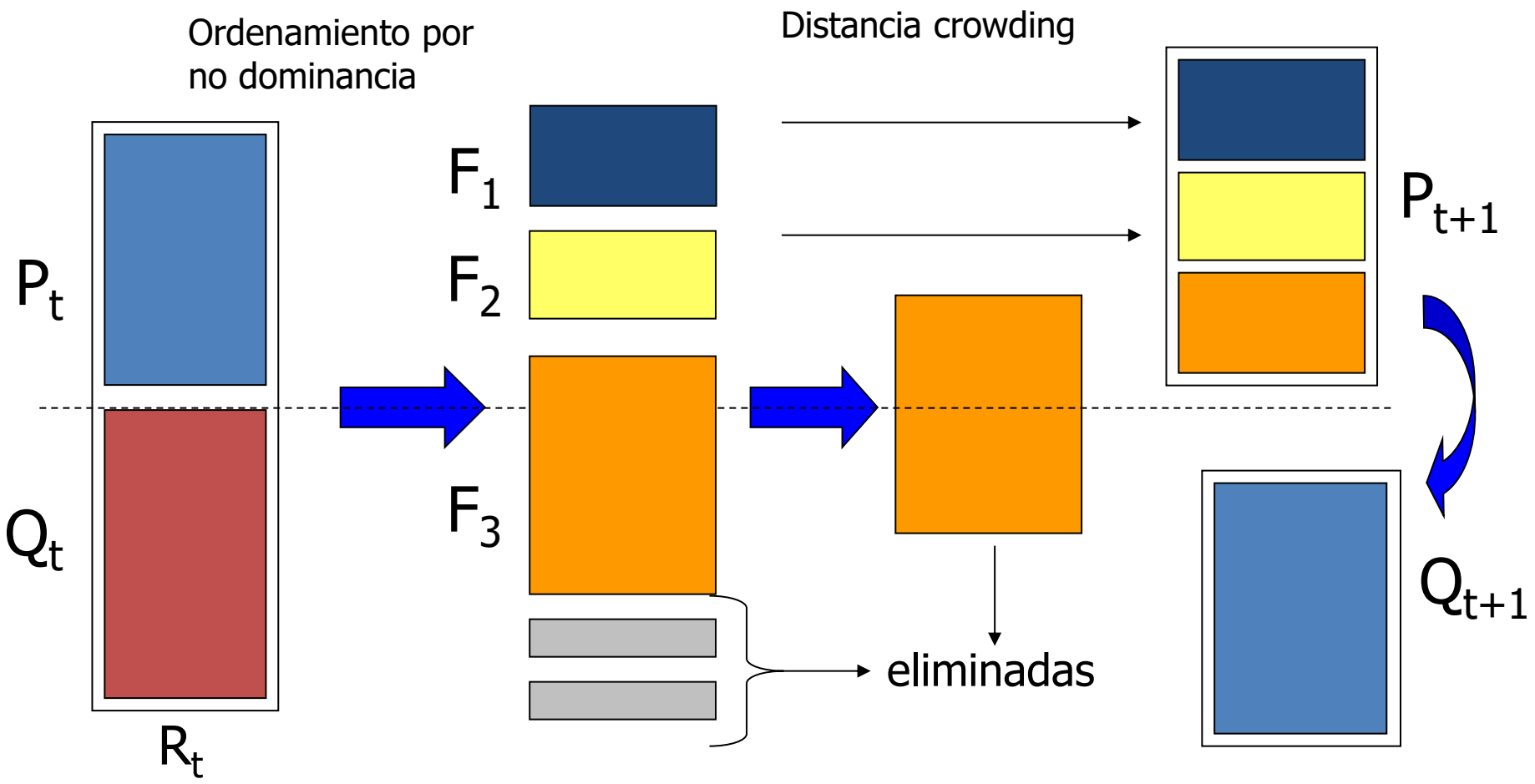
  Crowding distance = infinity

- En los demás individuos, la distancia *crowding* se calcula como la diferencia de los *fitness* de los 2 vecinos más cercanos.

Ejemplo: Para el objetivo F_1



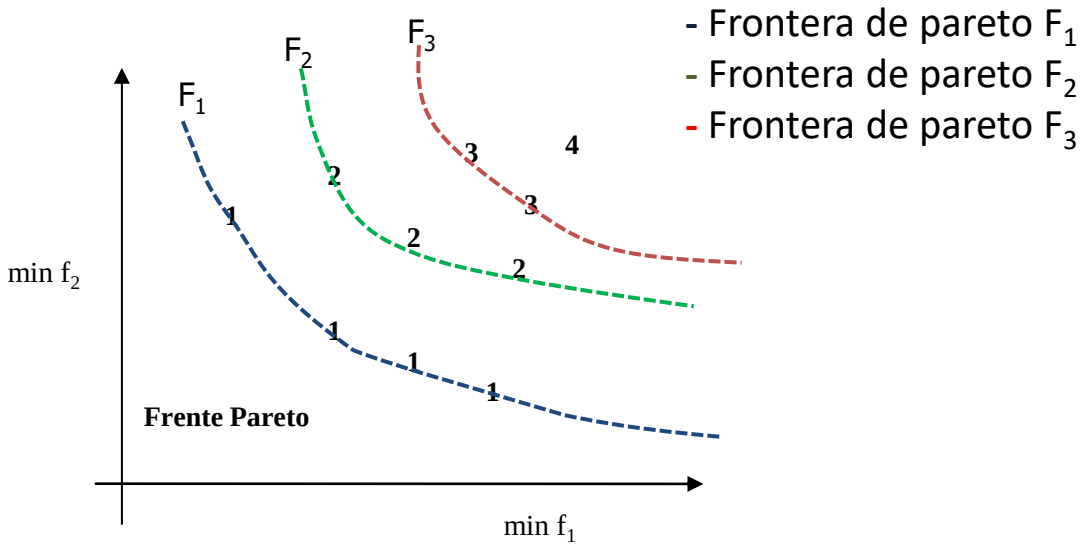
NSGA II



Combina los padres ($\boldsymbol{P}_t$) e hijos ($\boldsymbol{Q}_t$) para formar $\boldsymbol{R}_t = \boldsymbol{P}_t \cup \boldsymbol{Q}_t$, lo que garantiza elitismo.

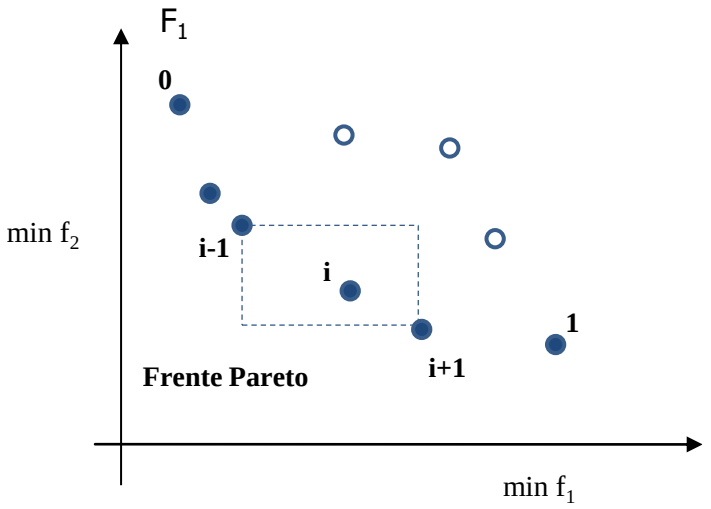
NSGA II - Procedimiento

1. Generar aleatoriamente una población P_t de tamaño N.
2. Calcular las funciones objetivo de la primera generación.
3. Ordenar la población P_t por ranking de no dominancia de pareto.
   - Ordenar población en los diferentes frentes (F_1,F_2, F_3,…,etc, donde 1 es el mejor nivel, luego 2 y así sucesivamente) según el nivel de no dominancia.



NSGA II - Procedimiento

4. Seleccionar los individuos por torneo de dos.
   - Escoger 2 individuos y eligir al mejor con respecto al ranking de no dominancia.
   - Los individuos en la misma frontera de no dominancia son comparados usando la distancia crowding.



Los puntos obscuros son individuos en la misma frontera de no dominancia.

NSGA II - Procedimiento

5. Usar los operadores de reproducción (cruzamiento y mutación) para generar la siguiente generación Q_t de tamaño N.
6. Evaluar las funciones objetivo de la nueva generación.
7. Combinar la población de padres P_t e hijos Q_t para formar R_t de tamaño 2N.
8. Ordenar la población por ranking de no dominancia de pareto.
9. Crear la siguiente generación seleccionando por elitismo los mejores individuos de la población R_t.
   - El criterio de selección es escoger primero los mejores individuos con respecto al ranking de no dominancia y luego los individuos resultantes de la distancia crowding.
10. El procedimiento termina cuando se alcanza el máximo numero de generaciones

Algoritmos Evolutivos Multiobjetivo

Métodos populares en la literatura:

Sigla	Nombre del Modelo
VEGA	*Vector Evaluated Genetic Algorithm*
WBGA	*Weight Based Genetic Algorithm*
MOGA	*Multiple Objective Genetic Algorithm*
NSGA	*Non-Dominated Sorting Genetic Algorithm*
NPGA	*Niched-Pareto Genetic Algorithm*
PPES	*Predator-Prey Evolution Strategy*
REMOE	*A Rudoph's Elitist Multi-Objective Evolutionay Algorithm*
NSGA-II	*Elitist Non-Dominated Sorting Genetic Algorithm*
SPEA, SPEA-2	*Strenght Pareto Evolutionary Algorithm 1 y 2*
TGA T	*Thermodynamical Genetic Algorithm*
PAES	*Pareto-Archived Evolutionary Strategy*
MONGA -I,MONGA - II	*Multi-Objective Messy Genetic Algorithm*
PESA-I, PESA-II	*Pareto Envelope-Base Selection Algorithm*

Ejemplo