Introducción a la Computación Evolutiva

Carlos A. Coello Coello

carlos.coellocoello@ccinvestav.mx
CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Departamento de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, MEXICO

Clase 7

Variantes de la Cruza

Otra variante muy común es la de restringir los puntos de cruza a aquellas posiciones en las que los padres difieran.

A esta técnica se le conoce como **sustitución reducida** [Booker, 1987].

El objetivo es mejorar la capacidad de la cruza para producir hijos que sean distintos a sus padres.

Lashon B. Booker, "**Improving Search in Genetic Algorithms**", in Lawrence Davis (Editor), *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, San Mateo, California, USA, 1987.

Variantes de la Cruza

Otra variante interesante es la llamada **cruza con barajeo** (*shuffle crossover*) [Eshelman et al., 1989].

En este caso, se aplica un operador de permutación a una parte de las cadenas de los padres antes de efectuar la cruza.

Después de la cruza, se aplica la permutación inversa a fin de restaurar el orden original de los bits.

Larry J. Eshelman, Richard A. Caruana and J. David Schaffer, "**Biases in the Crossover Landscape**", in J. David Schaffer (Editor), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 10–19, Morgan Kaufmann Publishers, San Mateo, California, USA, 1989.

Variantes de la Cruza

Cruza con Barajeo

La **cruza con barajeo** tiene como objeto contrarrestar la tendencia de la cruza de n puntos ($n >= 1$) a causar con más frecuencia disrupción en los conjuntos de bits que están dispersos que en los que están juntos.

Variantes de la Cruza

Cruza Segmentada

Propuesta por Eshelman et al. [1989].

Es una variante de la cruza de *n* puntos en la cual el número de puntos de cruza no es constante.

Se usa una probabilidad *s* de que una subcadena tenga su extremidad derecha en una cierta posición subsecuente a su inicio.

Larry J. Eshelman, Richard A. Caruana and J. David Schaffer, "**Biases in the Crossover Landscape**", in J. David Schaffer (Editor), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 10–19, Morgan Kaufmann Publishers, San Mateo, California, USA, 1989.

Cruza Segmentada

Iniciando de la primera posición del cromosoma (a esta posición la denotaremos con i), se genera aleatoriamente un número real $q \in [0, 1]$ y un número natural j tal que:

$$i < j <= L$$

L es la longitud de la cadena.

El valor q es considerado como la probabilidad de aceptar a j como un punto de cruza.

Dependiendo de la relación entre s y q, el punto j puede o no ser aceptado. De esta forma, el número de puntos de cruza varía.

Usualmente, se acepta j como un punto de cruza si se cumple que $q <= s$.

Variantes de la Cruza

Cruzas con Varios Padres

Aunque no son comunes en los algoritmos genéticos, existen también operadores de cruza que usan varios padres. Por ejemplo:

- **Multi-parent uniform crossover** [Furuya & Haftka, 1993].
- **Diagonal crossover** [Eiben et al., 1995].
- **Scanning crossover** [Eiben et al., 1995].

H. Furuya and R. Haftka, "**Genetic algorithms for placing actuators on space structures**", in Stephanie Forrest (Editor), *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 536–542, Morgan Kaufmann Publishers, San Mateo, California, USA, 1993.

A. E. Eiben, Cees H. M. van Kemenade and Joost N. Kok, "**Orgy in the Computer: Multi-Parent Reproduction in Genetic Algorithms**", in *Proceedings of the Third European Conference on Advances in Artificial Life*, pp. 934–945, Springer-Verlag, London, UK, June, 1995.

Formas de Apareamiento

Otro punto interesante a considerar son las técnicas de apareamiento (es decir, quién puede recombinarse con quién).

A continuación revisaremos rápidamente las formas más comunes de apareamiento, de acuerdo a Goldberg [1989].

David E. Goldberg, "**Genetic Algorithms in Search, Optimization and Machine Learning**", Addison-Wesley Publishing Co., Reading, Massachusetts, USA, 1989.

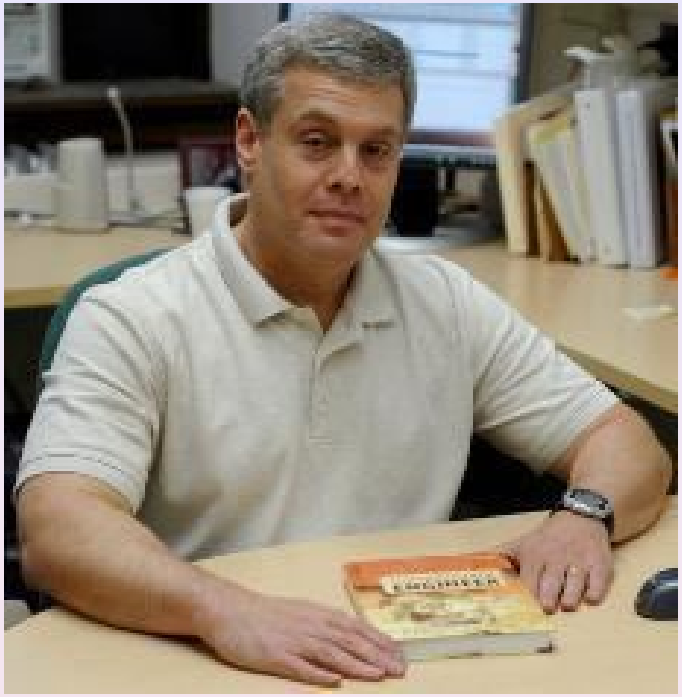
Formas de Apareamiento

- **Random Mating** (aleatorio):
  Se eligen los individuos aleatoriamente, con la misma probabilidad.
- **Inbreeding** (entre parientes):
  Se recombinan individuos similares
- **Line Breeding** (semental):
  Un solo super-individuo (aptitud alta) se recombina con una población base y sus hijos se seleccionan como padres.

Formas de Apareamiento

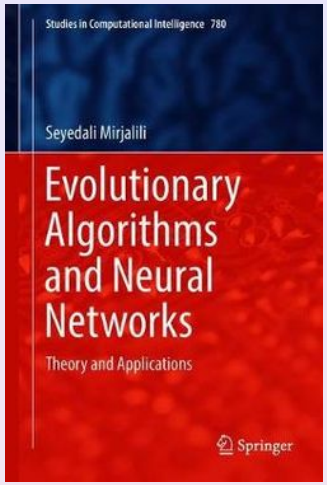
- **Outbreeding**(entre desconocidos):
  Sólo se recombinan individuos muy diferentes.
- **Self-fertilization** (auto-fertilización):
  Un individuo se recombina con sí mismo.
- **Cloning** (clonación):
  Un individuo se copia sin modificaciones

Formas de Apareamiento



- **Positive assorting mating**:
  Se recombinan individuos similares.
- **Negative assorting mating**:
  Se recombinan individuos diferentes.

Comportamiento Deseable de la Cruza



Todos los operadores de cruza descritos anteriormente, siguen el principio Mendeliano de la herencia: cada gene que tiene un hijo, es una copia de un gene heredado de alguno de sus padres.

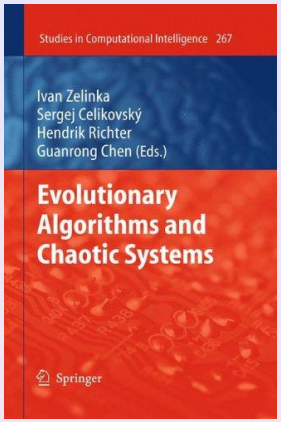
Cabe mencionar, sin embargo, que esto no tiene que ser así en computación evolutiva.

Comportamiento Deseable de la Cruza

Algunos investigadores han destacado que el énfasis de la cruza debe ser el poder generar todas las posibles combinaciones de bits (de longitud L) que hayan en el espacio de búsqueda del problema [Radcliffe, 1991].

Nicholas J. Radcliffe, "**Forma Analysis and Random Respectful Recombination**", in Richard K. Belew and Lashon B. Booker (Editors), *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 222–229, Morgan Kaufmann Publishers, San Mateo, California, USA, July 1991.

Comportamiento Deseable de la Cruza

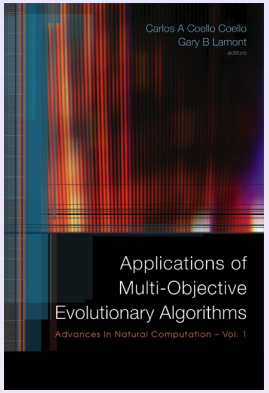


Dada una cierta representación binaria, ni la cruza de un punto, ni la de n puntos son capaces de lograr esto (generar cualquier combinación de bits posible).

La cruza uniforme, sin embargo, sí puede hacerlo.

Algunos investigadores han propuesto otras variantes de la cruza motivados por este problema.

Comportamiento Deseable de la Cruza

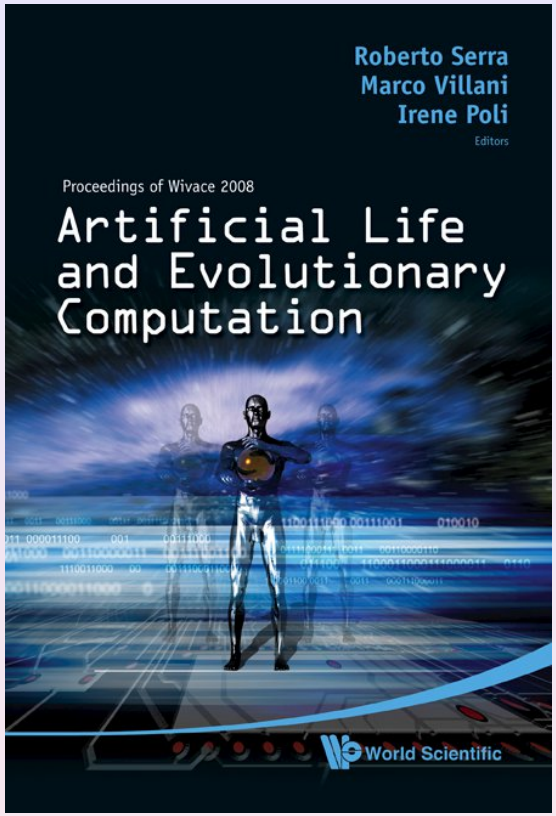


Radcliffe [1991] propuso una técnica denominada **recombinación respetuosa aleatoria**.

Según esta técnica, se genera un hijo copiando los bits en los que sus padres son idénticos, y eligiendo luego, valores al azar para llenar las posiciones siguientes.

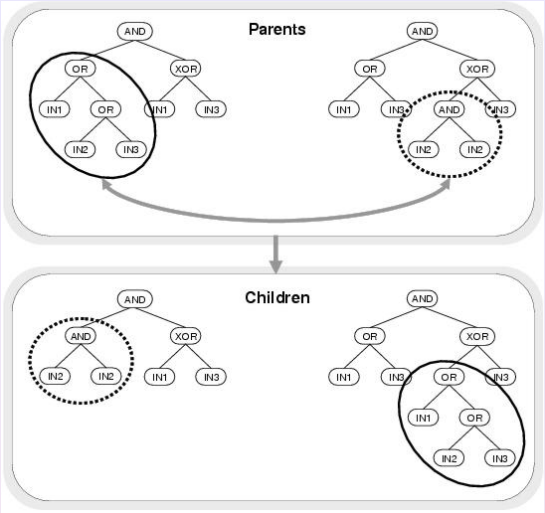
Si se usan cadenas binarias, y $Pc = 0.5$, la cruza uniforme es equivalente a esta recombinación.

Cruza para Representaciones Alternativas



- Programación Genética
- Permutaciones
- Representación real

Programación Genética

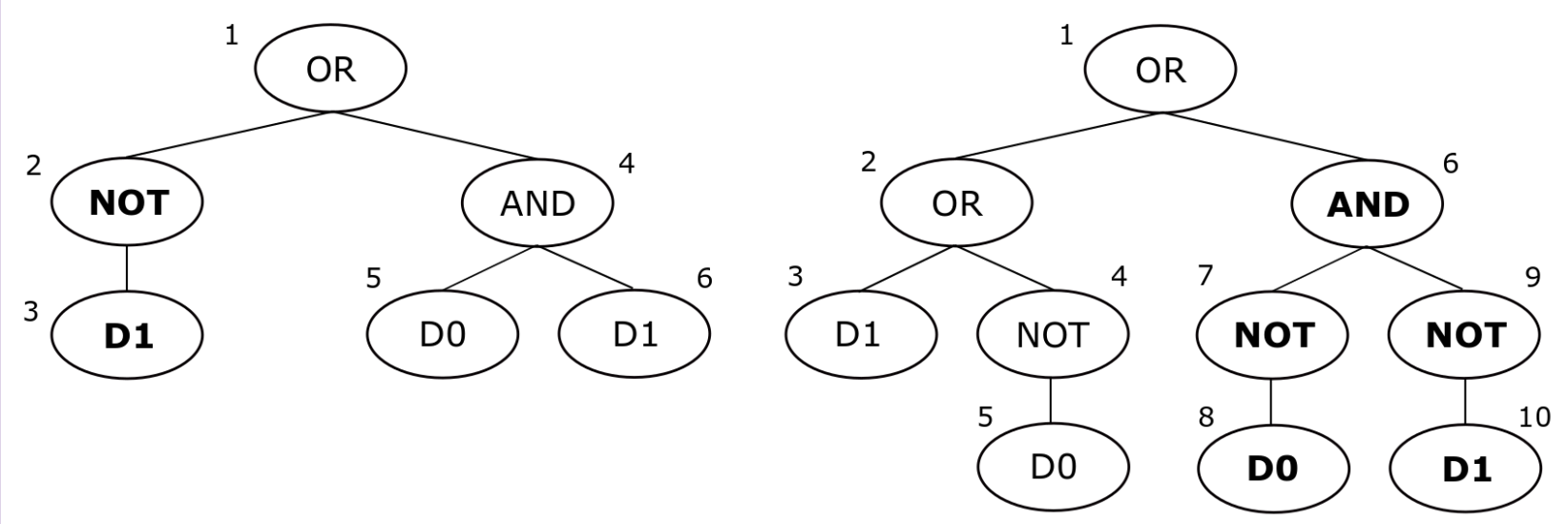


Al usarse representación de árbol, la cruza sigue funcionando de manera muy similar a la cruza convencional, sólo que en este caso, se intercambian sub-árboles entre los 2 padres.

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

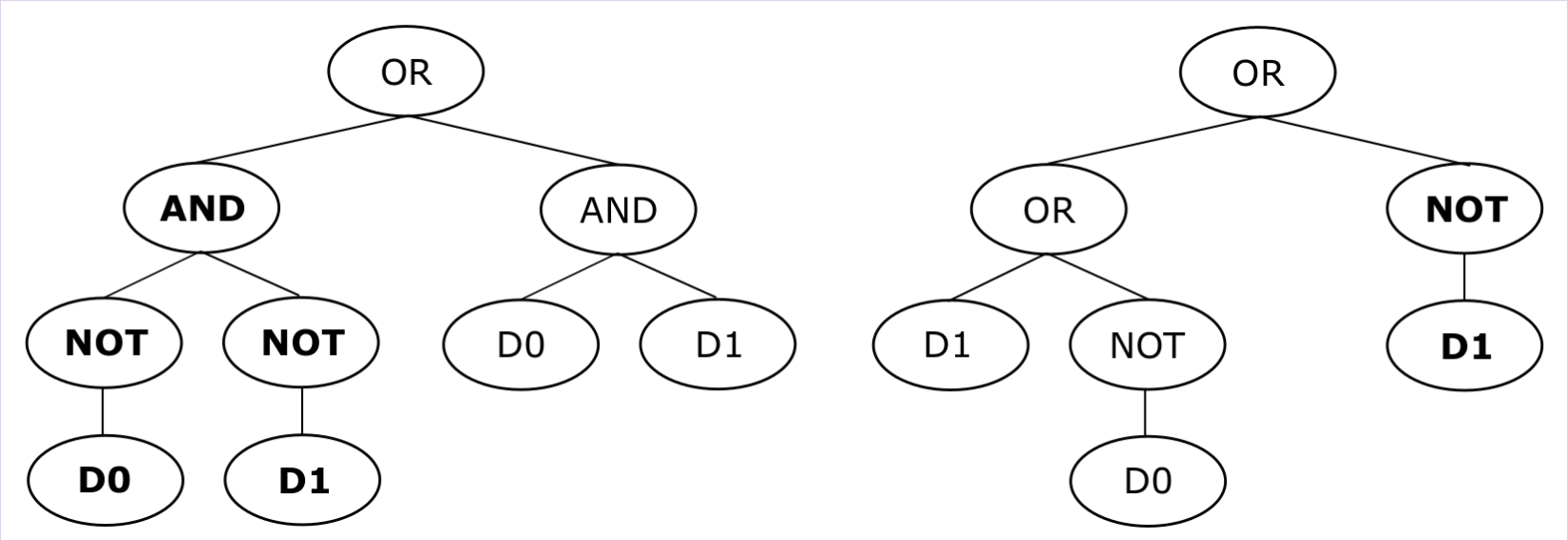
El segundo hijo se produce de manera análoga.

Ejemplo: Dados los 2 padres siguientes

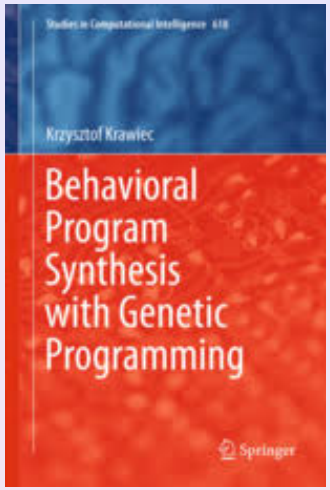


Las expresiones *S* de estos 2 padres son:
(OR (NOT D1) (AND D0 D1)) y
(OR (OR D1 (NOT D0)) (AND (NOT D0) (NOT D1)))

Si el punto de cruza del primer padre es **2**, y el del segundo es **6**, entonces los hijos resultantes serán:



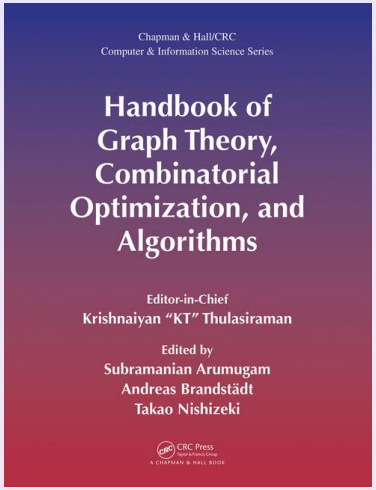
Programación Genética



Observaciones:

- Tipicamente, los 2 padres serán de tamaños distintos.
- Los padres son seleccionados mediante alguna de las técnicas que vimos antes.
- Suele limitarse la profundidad máxima de un árbol.
- La raíz es un punto de cruza válido.

Cruzas para Permutaciones

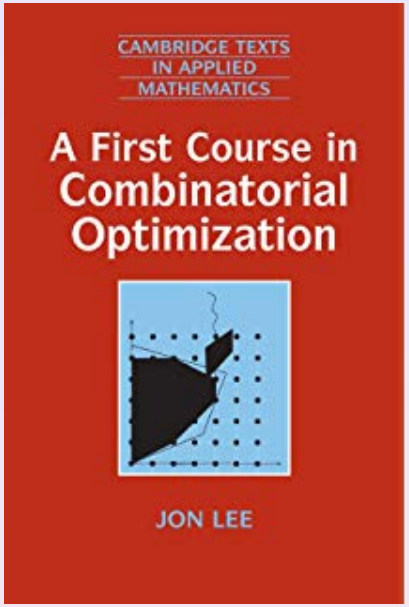


Al efectuar cruza entre 2 cadenas que usan representación de permutaciones, los hijos invariablemente serán no válidos.

La representación de permutaciones se usa frecuentemente en problemas de optimización combinatoria, como el del viajero.

Ejemplo de una permutación: 1 2 3 4 5 6 7 8 9

Cruzas para Permutaciones



- Order Crossover
- Partially Mapped Crossover
- Position-Based Crossover
- Order-Based Crossover
- Cycle Crossover
- Otros

Cruzas para Permutaciones

Order Crossover (OX)

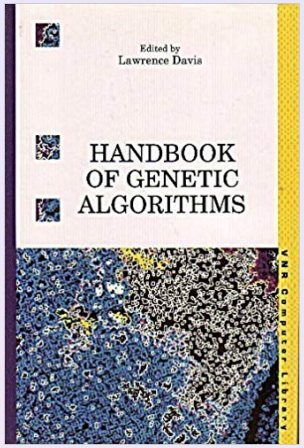
Fue propuesta por Davis [1985].

Algoritmo (los padres son **P1** y **P2**)

1. Seleccionar (aleatoriamente) una sub-cadena de **P1**.
2. Producir un hijo copiando la sub-cadena en las posiciones correspondientes a **P1**.
3. Las posiciones restantes se dejan en blanco.

Lawrence Davis, "**Job Shop Scheduling with Genetic Algorithms**", in John J. Grefenstette (Editor), *Proceedings of the First International Conference on Genetic Algorithms*, pp. 136–140, Lawrence Erlbaum Associates, Hillsdale, New Jersey, USA, July 1985.

Cruzas para Permutaciones



Order Crossover (OX) (Algoritmo Parte 2)

4. Borrar los valores que ya se encuentren en la sub-cadena de **P2**. La secuencia resultante contiene los valores faltantes.
5. Colocar los valores en posiciones no conocidas del hijo de izquierda a derecha.
6. Para obtener el segundo hijo, se repiten los pasos del 1 al 5, pero tomando ahora la sub-cadena de **P2**.

Order Crossover (OX) Ejemplo (Parte 1)

$$P1 = 9\ \ 8\ \ 4\ \ 5\ \ 6\ \ \ 7\ \ 1\ \ 2\ \ 3\ \ 10$$

$$P2 = 8\ \ 7\ \ 1\ \ 2\ \ 3\ \ 10\ \ 9\ \ 5\ \ 4\ \ 6$$

Sub-cadena elegida: 5 6 7 1 (de **P1**)

Primer hijo:

$$H1 = X\ \ X\ \ X\ \ 5\ \ 6\ \ 7\ \ 1\ \ X\ \ X\ \ X$$

Order Crossover (OX) Ejemplo (Parte 2)

Borrar de **P2** la sub-cadena tomada de **P1**:

$$P2' = 8 \;\; X \;\; X \;\; 2 \;\; 3 \;\; 10 \;\; 9 \;\; X \;\; 4 \;\; X$$

Determinar los valores faltantes de **H1** sustituyendo (de izquierda a derecha) los valores que aparecen en **P2'**:

$$H1 = 8 \;\; 2 \;\; 3 \;\; 5 \;\; 6 \;\; 7 \;\; 1 \;\; 10 \;\; 9 \;\; 4$$

Para obtener **H2**, el procedimiento es similar, aunque ahora la sub-cadena se tomará de **P2** y la sustitución se hará a partir de **P1'**.

Cruzas para Permutaciones

Partially Mapped Crossover (PMX)

Fue propuesta por Goldberg y Lingle [1985] y tiene ciertas similitudes con OX.

Partially Mapped Crossover (Algoritmo Parte 1)

1. Elegir aleatoriamente dos puntos de cruza.
2. Intercambiar estos 2 segmentos en los hijos que se generan (como la cruza de 2 puntos convencional).

David E. Goldberg and Robet Lingle, Jr., "**Alleles, Loci, and the Traveling Salesman Problem**", in John J. Grefenstette (Editor), *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pp. 154–159, Lawrence Erlbaum Associates, Hillsdale, New Jersey, USA, 1985.

Cruzas para Permutaciones

Partially Mapped Crossover (Algoritmo Parte 2)

3. El resto de las cadenas que conforman los hijos se obtienen haciendo mapeos entre los 2 padres:
   a) Si un valor no está contenido en el segmento intercambiado, permanece igual.
   b) Si está contenido en el segmento intercambiado, entonces se sustituye por el valor que tenga dicho segmento en el otro padre.

Partially Mapped Crossover (Ejemplo Parte 1)

Dados:

$$P1 = 9\ 8\ 4 \mid 5\ 6\ 7 \mid 1\ 3\ 2\ 10$$

$$P2 = 8\ 7\ 1 \mid 2\ 3\ 10 \mid 9\ 5\ 4\ 6$$

Los hijos son:

$$H1 = X\ X\ X \mid 2\ 3\ 10 \mid X\ X\ X\ X$$

$$H2 = X\ X\ X \mid 5\ 6\ 7 \mid X\ X\ X\ X$$

Partially Mapped Crossover (Ejemplo Parte 2)

Para completar **H1** y **H2**, copiamos primero los valores que no están en el segmento intercambiado:

$$H1 = 9\ 8\ 4 \mid 2\ 3\ 10 \mid 1\ X\ X\ X$$

$$H2 = 8\ X\ 1 \mid 5\ 6\ 7 \mid 9\ X\ 4\ X$$

Ahora mapeamos los valores restantes:

$$H1 = 9\ 8\ 4\ 2\ 3\ 10\ 1\ 6\ 5\ 7$$

$$H2 = 8\ 10\ 1\ 5\ 6\ 7\ 9\ 2\ 4\ 3$$

Cruzas para Permutaciones

Position-based Crossover

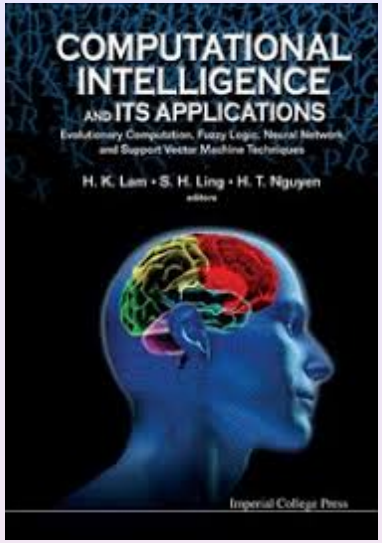
Propuesta por Syswerda [1991] como una adaptación de la cruza uniforme para permutaciones.

Algoritmo (Parte 1)

1. Seleccionar (al azar) un conjunto de posiciones de **P1** (no necesariamente consecutivas).
2. Producir un hijo borrando de **P1** todos los valores, excepto aquéllos que hayan sido seleccionados en el paso anterior.

Gilbert Syswerda, "**Schedule Optimization using Genetic Algorithms**, in Lawrence Davis (Editor), *Handbook of Genetic Algorithms*, Chapter 21, pp. 332–349, Van Nostrand Reinhold, New York, USA, 1991.

Cruzas para Permutaciones



Position-based Crossover (Algoritmo Parte 2)

3. Borrar los valores seleccionados de **P2**. La secuencia resultante de valores se usará para completar el hijo.
4. Colocar en el hijo los valores faltantes de izquierda a derecha, de acuerdo a la secuencia de **P2**.
5. Repetir los pasos del 1 al 4, pero tomando ahora la secuencia de **P2**.

Position-based Crossover (Ejemplo Parte 1)

Dados:

$$P1 = 9\ 8\ 4\ 5\ 6\ 7\ 1\ 2\ 3\ 10$$

$$P2 = 8\ 7\ 1\ 2\ 3\ 10\ 9\ 5\ 4\ 6$$

Valores elegidos de $P1$: 8 6 2 10

Producir un hijo:

$$H1 = X\ 8\ X\ X\ 6\ X\ X\ 2\ X\ 10$$

Position-based Crossover (Ejemplo Parte 2)

Borrar de **P2** la secuencia usada para **H1**:

$$P2' = X\ 7\ 1\ X\ 3\ X\ 9\ 5\ 4\ X$$

Sustituir de izquierda a derecha los valores que aparecen en **P2'**:

$$H1 = 7\ 8\ 1\ 3\ 6\ 9\ 5\ 2\ 4\ 10$$

Para obtener **H2**, el procedimiento es similar, pero la secuencia se toma ahora de **P2** y la sustitución se hace a partir de **P1'**.

Cruzas para Permutaciones

Order-based Crossover

Fue propuesta por Syswerda [1991] como una ligera variante de **Position-based Crossover** en la que se cambia el orden de los pasos del algoritmo.

Algoritmo

1. En este caso, primero seleccionamos una serie de valores de **P1**.
2. Luego, removemos de **P2** esos valores.
3. A continuación generamos un hijo a partir de **P2'**.
4. Finalmente, completamos el hijo con los valores de la secuencia obtenida de **P1** (insertada de izquierda a derecha en el orden impuesto por **P1**).

Gilbert Syswerda, "**Schedule Optimization using Genetic Algorithms**, in Lawrence Davis (Editor), *Handbook of Genetic Algorithms*, Chapter 21, pp. 332–349, Van Nostrand Reinhold, New York, USA, 1991.

Order-based Crossover (Ejemplo (Parte 1))

$$P1 = 9\ 8\ 4\ 5\ 6\ 7\ 1\ 2\ 3\ 10$$

$$P2 = 8\ 7\ 1\ 2\ 3\ 10\ 9\ 5\ 4\ 6$$

Valores elegidos de **P1**: 8 6 2 10

Removamos de **P2** estos valores:

$$P2' = X\ 7\ 1\ X\ 3\ X\ 9\ 5\ 4\ X$$

Order-based Crossover (Ejemplo (Parte 2))

Producir un hijo:

$$H1 = X\ 7\ 1\ X\ 3\ X\ 9\ 5\ 4\ X$$

Insertamos ahora la secuencia elegida de **P1**:

$$H1 = 8\ 7\ 1\ 6\ 3\ 2\ 9\ 5\ 4\ 10$$

Cruzas para Permutaciones

Cycle Crossover (CX)

Propuesta por Oliver, Smith y Holland [1987].

Es similar a la **Position-based Crossover**, porque toma algunos valores de un padre y selecciona los restantes del otro.

La principal diferencia es que los valores tomados del primer padre no se seleccionan al azar, sino de acuerdo a ciertas reglas específicas.

I.M. Oliver, D.J. Smith and J.R.C. Holland, "**A Study of Permutation Crossover Operators on the Traveling Salesman Problem**", in John J. Grefenstette (Editor), *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 224–230, Lawrence Erlbaum, Hillsdale, New Jersey, USA, July 1987.

Cycle Crossover (CX) (Algoritmo)

1. Encontrar un ciclo que se define mediante las posiciones correspondientes de los valores entre los padres.
2. Copiar los valores de **P1** que sean parte del ciclo.
3. Borrar de **P2** los valores que estén en el ciclo.
4. Rellenar el hijo con los valores restantes de **P2** (sustituyendo de izquierda a derecha).
5. Repetir los pasos del 1 al 4, usando ahora **P2**.

Cycle Crossover (CX)

El elemento clave de esta técnica es saber cómo encontrar un **ciclo**.

Es más fácil explicar este concepto con un ejemplo:

$$P1 = h\ k\ c\ e\ f\ d\ b\ l\ a\ \ i\ g\ j$$

$$P2 = a\ b\ c\ d\ e\ f\ g\ h\ i\ \ j\ k\ l$$

Posiciones:

1 2 3 4 5 6 7 8 9 10 11 12

Cycle Crossover (Ejemplo de un ciclo Parte 1)

Tomamos al azar una posición de cualquier padre. En este caso, tomaremos la posición **1**:

Los elementos **(h,a)** pertenecen al ciclo **1**.

Si observamos a **P1** y **P2**, veremos que "**h**" y "**a**" aparecen también en las posiciones **8** y **9**.

Por tanto, el ciclo incluye ahora las posiciones **(1,8,9)** y se agregan los elementos "**i**" y "**l**". O sea, que los elementos del ciclo son: **(h,a,i,l)**.

Cycle Crossover (Ejemplo de un ciclo Parte 2)

Vemos ahora que “**i**” y “**l**” aparecen en las posiciones **10** y **12**.

Por tanto, el ciclo ahora incluye las posiciones **(1,8,9,10,12)**, y se agrega el elemento “**j**”.

Vemos ahora que “**j**” ya no se encuentra en ninguna otra posición, por lo que el ciclo se termina.

Cycle Crossover (Ejemplo Parte 1)

$$P1 = 8\ 11\ 3\ 5\ 6\ 4\ 2\ 12\ 1\ 9\ \ 7\ \ 10$$

$$P2 = 1\ \ 2\ 3\ 4\ 5\ 6\ 7\ \ 8\ 9\ 10\ 11\ 12$$

Posiciones:

$$1\ 2\ 3\ 4\ 5\ 6\ 7\ \ 8\ 9\ 10\ 11\ 12$$

Tomemos la posición **1** para iniciar el ciclo: Los elementos del ciclo serán entonces **(1,8)**.

Cycle Crossover (Ejemplo Parte 2)

Pero **1** y **8** también aparecen en las posiciones **9** y **8**. Por lo tanto, el ciclo ahora incluye los elementos (1,8,12).

Pero **12** aparece también en la posición **12**. Por lo tanto, el ciclo ahora incluye los elementos **(1,8,12,10)**.

Pero **10** aparece también en la posición **10**. Por lo tanto, el ciclo ahora incluye los elementos **(1,8,12,10,9)**.

Ya no hay nuevos elementos qué agregar, por lo que se concluye el ciclo.

Cycle Crossover (Ejemplo Parte 3)

Para generar al primer hijo, tomamos a **P1**, removiéndole los elementos que no sean parte del ciclo:

$$H1 = 8\ X\ X\ X\ X\ X\ X\ 12\ 1\ 9\ X\ 10$$

Remover de **P2** los valores del ciclo:

$$P2' = X\ 2\ 3\ 4\ 5\ 6\ 7\ \ X\ X\ X\ 11\ X$$

Rellenar **H1** usando los valores restantes de **P2'**.

$$H1 = 8\ 2\ 3\ 4\ 5\ 6\ 7\ 12\ 1\ 9\ \ 11\ 10$$

Cruzas para Permutaciones

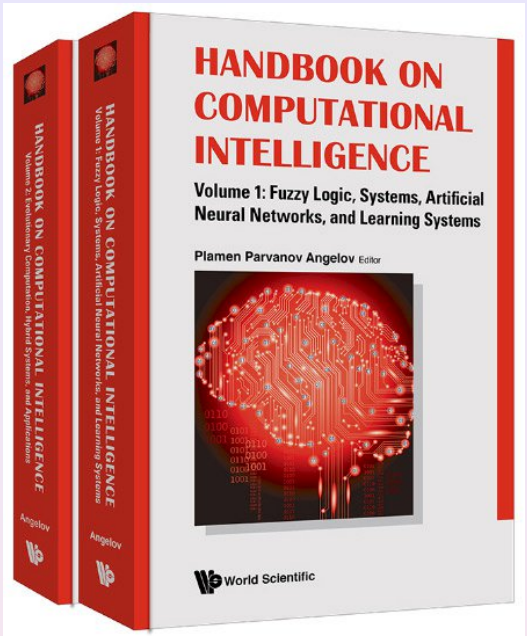
Hay muchas otras propuestas de operadores de cruza para permutaciones. Por ejemplo:

- Fox y McMahon [1991] propusieron un operador de intersección que extrae características comunes de ambos padres.
- Blanton y Wainwright [1993] propusieron un operador llamado **merge crossover**, que permite incorporar preferencias en una cierta ruta a la hora de efectuar la cruza.

B.R. Fox and M.B. McMahon, "**Genetic Operators for Sequencing Problems**", in Gregory J.E. Rawlins (Editor), *Foundations of Genetic Algorithms*, pp. 284–300, Morgan Kaufmann Publishers, San Mateo, California, USA, 1991.

Joe K. Blanton Jr. and Roger L. Wainwright, "**Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms**", in Stephanie Forrest (Editor), *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 452–459, Morgan Kaufmann Publishers, San Mateo, California, USA, July 1993.

Inversión para Permutaciones



Con la idea de explorar localmente una solución producida por un algoritmo genético, algunos investigadores han propuesto el uso de inversión (algunos llaman "**mutación por inversión**" a este operador, para distinguirlo de la "inversión" usada por Holland, y de la cual hablaremos más adelante).

Inversión para Permutaciones

El operador es muy sencillo. Básicamente se usan 2 puntos (generados al azar) entre los cuales se invierte la sub-cadena correspondiente. Se aplica con un porcentaje dado por el usuario.

Ejemplo

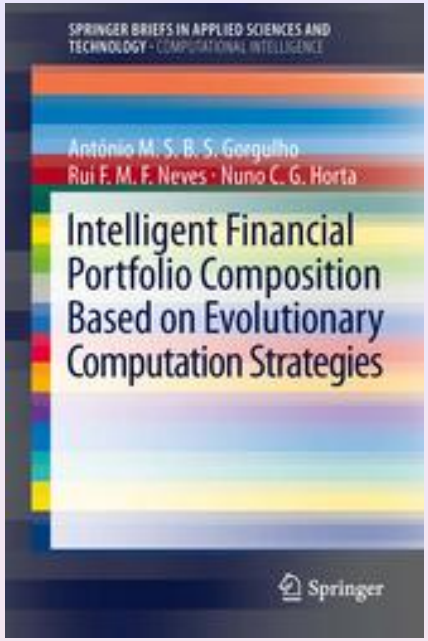
Dada la cadena:

$$P = 9\ 4\ 2\ 1\ 5\ 7\ 6\ 10\ 3\ 8$$

Y usando **v1=5** y **v2=8** como punto de inicio y punto final, respectivamente, la cadena resultante (tras aplicar inversión) sería:

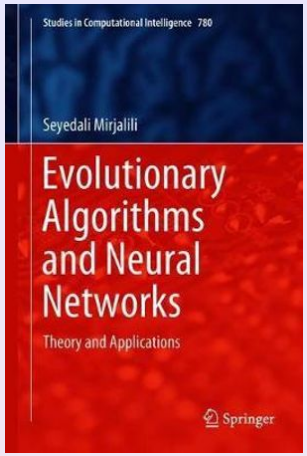
$$P' = 9\ 4\ 2\ 1\ 10\ 6\ 7\ 5\ 3\ 8$$

Mutación para Permutaciones



- Mutación por inserción
- Mutación por Desplazamiento
- Mutación por Intercambio Recíproco
- Mutación Heurística

Mutación por Inserción



Se selecciona un valor en forma aleatoria y se le inserta en una posición arbitraria.

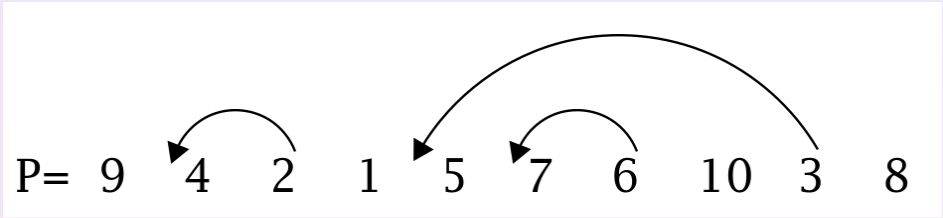
Ejemplo

Dada: $P = 9\ 4\ 2\ 1\ 5\ 7\ 6\ 10\ 3\ 8$

Si suponemos que elegimos la posición **7** y decidimos mover ese valor a la posición **2**, tendríamos: $P' = 9\ 6\ 4\ 2\ 1\ 5\ 7\ 10\ 3\ 8$

Mutación por Desplazamiento

Es una generalización de la mutación por inserción en la que en vez de mover un solo valor, se cambian de lugar varios a la vez.



La cadena después de mutación sería:

$$P' = 9 \quad 2 \quad 4 \quad 1 \quad 3 \quad 5 \quad 6 \quad 7 \quad 10 \quad 8$$

- El número **2** se movió después del número **9** y antes del número **4**.
- El número **3** se movió después del número **1** y antes del número **5**.
- El número **6** se movió después del número **5** y antes del número **7**.

Mutación por Intercambio Recíproco

En este caso, se seleccionan dos puntos al azar y se intercambian estos valores de posición.

Por ejemplo, dada:

$$P = 9 \quad 4 \quad 2 \quad 1 \quad 5 \quad 7 \quad 6 \quad 10 \quad 3 \quad 8$$

$$\wedge \qquad\qquad\qquad\qquad \wedge$$

Tendríamos:

$$P' = 9 \quad 10 \quad 2 \quad 1 \quad 5 \quad 7 \quad 6 \quad 4 \quad 3 \quad 8$$

Mutación Heurística

Fue propuesta por Cheng y Gen [1994].

Algoritmo

1. Seleccionar λ genes al azar.
2. Generar vecinos de acuerdo a todas las permutaciones posibles de los genes seleccionados.
3. Evaluar todos los vecinos y seleccionar el mejor.

Mitsuo Gen and Runwei Cheng, "**Genetic Algorithms & Engineering Optimization**", John Wiley & Sons, Wiley Series in Engineering Design and Automation, New York, USA, 2000.

Mutación Heurística

Ejemplo

$$P = 9\ 4\ 2\ 1\ 5\ 7\ 6\ 10\ 3\ 8$$

$$\wedge \quad \wedge \quad \wedge$$

Generar todas las permutaciones de:

$$4 \quad 5 \quad 10$$

1) 4 10 5 2) 5 4 10 3) 5 10 4 4) 10 5 4 5) 10 4 5

Mutación Heurística

Individuos generados:

$P1 = 9\ \ 4\ \ 2\ \ 1\ \ 10\ \ 7\ \ 6\ \ \ 5\ \ 3\ \ 8$

$P2 = 9\ \ 5\ \ 2\ \ 1\ \ \ 4\ \ 7\ \ 6\ \ 10\ \ 3\ \ 8$

$P3 = 9\ \ 5\ \ 2\ \ 1\ \ 10\ \ 7\ \ 6\ \ \ 4\ \ 3\ \ 8$

$P4 = 9\ \ 10\ \ 2\ \ 1\ \ \ 5\ \ 7\ \ 6\ \ \ 4\ \ 3\ \ 8$

$P5 = 9\ \ 10\ \ 2\ \ 1\ \ \ 4\ \ 7\ \ 6\ \ \ 5\ \ 3\ \ 8$

De entre ellas, se selecciona a la mejor. En este caso, supondremos que es **P4**:

$P' = 9\ \ 10\ \ 2\ \ 1\ \ 5\ \ 7\ \ 6\ \ 4\ \ 3\ \ 8$

Cruzas para Representación Real

- Un punto
- Dos puntos
- Uniforme
- Intermedia
- Aritmética simple
- Aritmética total
- Simulated Binary Crossover
- Cruza basada en dirección

Cruzas para Representación Real

Cruza Simple

Se trata realmente de la cruza de un punto aplicada a vectores de números reales.

Ejemplo

Dados:

P1 = $< 3.2,\ 1.9,\ |\ -0.5 >$

P2 = $< 2.9,\ 0.8,\ |\ 1.4 >$

Los hijos serían:

H1 = $< 3.2, 1.9, 1.4 >$

H2 = $< 2.9, 0.8, -0.5 >$

Cruzas para Representación Real

Cruza de Dos Puntos

Es igual que para el caso binario.

Ejemplo

Dados:

P1 $= < 1.6,\ -2.1,\ |\ 3.5,\ 0.4,\ |\ 5.6 >$

P2 $= < 0.2,\ 4.5,\ |\ -2.3,\ 8.6,\ |\ -1.4 >$

Los hijos serían:

H1 $= < 1.6,\ -2.1,\ -2.3,\ 8.6,\ 5.6 >$

H2 $= < 0.2,\ 4.5,\ 3.5,\ 0.4,\ -1.4 >$

Cruzas para Representación Real

Cruza Uniforme

Es igual que para el caso binario.

Ejemplo

Dados:

P1 = $< 1.6,\ -2.1,\ 3.5,\ 0.4,\ 5.6,\ 7.8 >$

P2 = $< 0.2,\ 4.5,\ -2.3,\ 8.6,\ -1.4,\ 0.5 >$

Si **Pc=0.5**, entonces los hijos podrían ser:

H1 = $< 1.6,\ 4.5,\ -2.3,\ 0.4,\ 5.6,\ 0.5 >$

H2 = $< 0.2,\ -2.1,\ 3.5,\ 8.6,\ -1.4,\ 7.8 >$

Cruza Intermedia

En este caso, si suponemos que tenemos dos padres:
P1 = $< V_1, \ldots, V_m >$ y **P2** = $< W_1, \ldots, W_m >$, supongamos que éstos se cruzan en la posición k.

Entonces los hijos producidos son:
H1 =
$V_1, \ldots, V_k, W_{k+1} * a + V_{k+1} * (1 - a), \ldots, W_m * a + V_m * (1 - a)$
H2 = $W_1, \ldots, W_k, V_{k+1} * a + W_{k+1} * (1 - a), \ldots, V_m * a +$
$W_m * (1 - a)$
donde: a $\in [0 \ldots 1]$

Cruza Intermedia (Ejemplo)

Supongamos que: $k = 3$, a =0.3 y:
P1 = $< 1.6, -2.1, 3.5, 0.4, 5.6, 7.8 >$
P2 = $< 0.2, 4.5, -2.3, 8.6, -1.4, 0.5 >$

Los hijos serían:
H1 = $< 1.6, -2.1, 3.5, 8.6 * 0.3 + 0.4 * (1 - 0.3), -1.4 * 0.3 + 5.6 * (1 - 0.3), 0.5 * 0.3 + 7.8 * (1 - 0.3) >$
H2 = $< 0.2, 4.5, -2.3, 0.4 * 0.3 + 8.6 * (1 - 0.3), 5.6 * 0.3 - 1.4 * (1 - 0.3), 7.8 * 0.3 + 0.5 * (1 - 0.3) >$