COMPUTACIÓN BIOLÓGICA

Pedro Isasi[1]

[1]Departamento de Informática
Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Leganés (Madrid). Spain
email: isasi@ia.uc3m.es

Presentación

Temario

Conceptos

- La coevolución es una fuerza importante hacia la generación de adaptaciones muy complejas
- Relacciones depredador-presa
- Existe una gran presión para que las presas se defiendan mejor:
  - Correr más rápido
  - Generar caparazones más resistentes
  - Crear mejores camuflajes
- En respuesta los depredadores desarrollan mejores estrategias de ataque:
  - Garras y dentaduras más fuertes
  - Mejor visión
- En coevolución, el éxito de una parte es el fracaso de la opuesta que debe desarrollar respuestas para mantener sus posibilidades de supervivencia $\Rightarrow$ Reina roja

COEVOLUCIÓN COMPUTACIONAL

Existen dos tipos de coevolución:

- Cooperativa $\Rightarrow$ dos sistemas independientes cooperan para aumentar sus posibilidades de supervivencia
  - Simbiosis, "vive y deja vivir" en la segunda guerra mundial
- Competitiva $\Rightarrow$ dos sistemas independientes compiten por recursos compartidos
  - Depredador-presa
  - El fitness del primer sistema es inversamente proporcional al fitness del segundo sistema
  - Si los individuos de S_1 están poco evolucionados, los de S_2 lo están mucho
  - Al evolucionar S_1, los individuos de S_2 pierden calidad y tienen mayor presión para seguir evolucionando
  - El mecanismo se realimenta y se producen individuos en ambos sistemas cada vez mejores

Evaluación de los individuos

- En algunos ocasiones los individuos de una población tienen que ser evaluados utilizando ejemplos
- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y utilizarlos para evaluar un individuo
- Aparece el problema denominado “Ideal trainer”, entrenador ideal:
  - Existe un número enorme de posibles ejemplos, pero solo un ínfimo número de ellos se pueden utilizar para evaluar al individuo para que el proceso sea eficiente
  - El problema está en cómo determinar qué conjunto de n ejemplos (n pequeño) es el “ideal” para que sirva de evaluación
  - Si el conjunto de ejemplos es “fácil” se puede producir especialización
  - Lo que puede ser un buen conjunto de evaluación para un individuo, puede ser malo para otro y viceversa

Evaluación de los individuos

- En algunos ocasiones los individuos de una población tienen que ser evaluados utilizando ejemplos
- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y utilizarlos para evaluar un individuo
- Aparece el problema denominado “Ideal trainer”, entrenador ideal:
  - Existe un número enorme de posibles ejemplos, pero solo un ínfimo número de ellos se pueden utilizar para evaluar al individuo para que el proceso sea eficiente
  - El problema está en cómo determinar qué conjunto de n ejemplos (n pequeño) es el “ideal” para que sirva de evaluación
  - Si el conjunto de ejemplos es “fácil” se puede producir especialización
  - Lo que puede ser un buen conjunto de evaluación para un individuo, puede ser malo para otro y viceversa

Evaluación de los individuos

- En algunos ocasiones los individuos de una población tienen que ser evaluados utilizando ejemplos
- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y utilizarlos para evaluar un individuo
- Aparece el problema denominado “Ideal trainer”, entrenador ideal:
  - Existe un número enorme de posibles ejemplos, pero solo un ínfimo número de ellos se pueden utilizar para evaluar al individuo para que el proceso sea eficiente
  - El problema está en cómo determinar qué conjunto de n ejemplos (n pequeño) es el “ideal” para que sirva de evaluación
  - Si el conjunto de ejemplos es “fácil” se puede producir especialización
  - Lo que puede ser un buen conjunto de evaluación para un individuo, puede ser malo para otro y viceversa

Evaluación de los individuos

- En algunos ocasiones los individuos de una población tienen que ser evaluados utilizando ejemplos
- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y utilizarlos para evaluar un individuo
- Aparece el problema denominado “Ideal trainer”, entrenador ideal:
  - Existe un número enorme de posibles ejemplos, pero solo un ínfimo número de ellos se pueden utilizar para evaluar al individuo para que el proceso sea eficiente
  - El problema está en cómo determinar qué conjunto de *n* ejemplos (*n* pequeño) es el “ideal” para que sirva de evaluación
  - Si el conjunto de ejemplos es “fácil” se puede producir especialización
  - Lo que puede ser un buen conjunto de evaluación para un individuo, puede ser malo para otro y viceversa

Evaluación de los individuos

- En algunos ocasiones los individuos de una población tienen que ser evaluados utilizando ejemplos
- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y utilizarlos para evaluar un individuo
- Aparece el problema denominado “Ideal trainer”, entrenador ideal:
  - Existe un número enorme de posibles ejemplos, pero solo un ínfimo número de ellos se pueden utilizar para evaluar al individuo para que el proceso sea eficiente
  - El problema está en cómo determinar qué conjunto de *n* ejemplos (*n* pequeño) es el “ideal” para que sirva de evaluación
  - Si el conjunto de ejemplos es “fácil” se puede producir especialización
  - Lo que puede ser un buen conjunto de evaluación para un individuo, puede ser malo para otro y viceversa

Evaluación de los individuos

- En algunos ocasiones los individuos de una población tienen que ser evaluados utilizando ejemplos
- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y utilizarlos para evaluar un individuo
- Aparece el problema denominado “Ideal trainer”, entrenador ideal:
  - Existe un número enorme de posibles ejemplos, pero solo un ínfimo número de ellos se pueden utilizar para evaluar al individuo para que el proceso sea eficiente
  - El problema está en cómo determinar qué conjunto de *n* ejemplos (*n* pequeño) es el “ideal” para que sirva de evaluación
  - Si el conjunto de ejemplos es “fácil” se puede producir especialización
  - Lo que puede ser un buen conjunto de evaluación para un individuo, puede ser malo para otro y viceversa

Evaluación de los individuos

- En algunos ocasiones los individuos de una población tienen que ser evaluados utilizando ejemplos
- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y utilizarlos para evaluar un individuo
- Aparece el problema denominado “Ideal trainer”, entrenador ideal:
  - Existe un número enorme de posibles ejemplos, pero solo un ínfimo número de ellos se pueden utilizar para evaluar al individuo para que el proceso sea eficiente
  - El problema está en cómo determinar qué conjunto de *n* ejemplos (*n* pequeño) es el “ideal” para que sirva de evaluación
  - Si el conjunto de ejemplos es “fácil” se puede producir especialización
  - Lo que puede ser un buen conjunto de evaluación para un individuo, puede ser malo para otro y viceversa

Problema del entrenador ideal

- Una primera solución sería generar un nuevo conjunto de entrenamiento en cada generación
  - Los individuos no podrían especializarse, ya que se evalúan cada vez con ejemplos diferentes
  - Un individuo muy bueno en una generación, y que por tanto ha engendrado muchos descendientes, puede ser muy malo en la siguiente, y todos los individuos que ha engendrado ser inoperativos
  - Al cambiar brúscamente la evaluación de los individuos el "fitness landscape" se hace muy abrupto, y el método se desorienta
- Se podría cambiar de conjunto de aprendizaje cada "n" generaciones. Esto reduciría los problemas anteriores, pero no los eliminaría

Problema del entrenador ideal

- Una primera solución sería generar un nuevo conjunto de entrenamiento en cada generación
  - Los individuos no podrían especializarse, ya que se evalúan cada vez con ejemplos diferentes
  - Un individuo muy bueno en una generación, y que por tanto ha engendrado muchos descendientes, puede ser muy malo en la siguiente, y todos los individuos que ha engendrado ser inoperativos
  - Al cambiar brúscamente la evaluación de los individuos el "fitness landscape" se hace muy abrupto, y el método se desorienta
- Se podría cambiar de conjunto de aprendizaje cada "n" generaciones. Esto reduciría los problemas anteriores, pero no los eliminaría

Problema del entrenador ideal

- Se podría cambiar “ligeramente” el conjunto de aprendizaje cada “n” generaciones
  - Habría que ver cuánto y cómo se cambia el conjunto de aprendizaje, y en función de qué
  - Al utilizar el mismo conjunto de aprendizaje para todos los individuos, cambios positivos para unos podrían ser negativos para otros
- Se podría generar un conjunto de aprendizaje diferente para cada individuo, y cambiarlo “ligeramente” cada “n” generaciones
  - Se tendría el problema de cuánto y cómo cambiar el conjunto de aprendizaje
  - Si el conjunto de aprendizaje es complejo, el problema anterior no es trivial

Problema del entrenador ideal

- Se podría cambiar “ligeramente” el conjunto de aprendizaje cada “n” generaciones
  - Habría que ver cuánto y cómo se cambia el conjunto de aprendizaje, y en función de qué
  - Al utilizar el mismo conjunto de aprendizaje para todos los individuos, cambios positivos para unos podrían ser negativos para otros
- Se podría generar un conjunto de aprendizaje diferente para cada individuo, y cambiarlo “ligeramente” cada “n” generaciones
  - Se tendría el problema de cuánto y cómo cambiar el conjunto de aprendizaje
  - Si el conjunto de aprendizaje es complejo, el problema anterior no es trivial

Conjuntos de entrenamiento como sist. genéticos

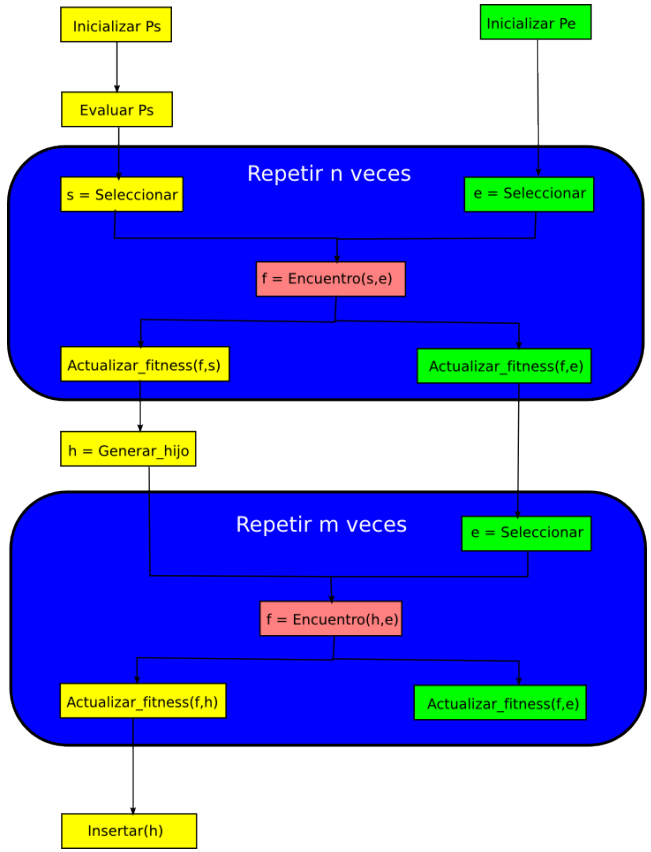
- La solución es generar una población de conjuntos de entrenamiento
- Cada individuo es un conjunto de entrenamiento, compuesto por *n* valores reales
- Cada individuo-conjunto-de-entrenamiento es evaluado con un conjunto de individuos solución
- El resultado de la evaluación se utiliza para evaluar a los individuos-ejemplo y a los individuos-solución
- una buena evaluación para un individuo-solución será mala para un individuo-ejemplo y viceversa
- Esta evaluación se utiliza dentro de un Algoritmo Genético, de forma que se evolucionan los conjuntos de entrenamiento más complicados para un individuo-solución o para un conjunto de individuos-solución

LIFE-TIME FITNESS

- Dos poblaciones, una de soluciones $P_s(t)$ de tamaño $\| P_s \|$ y otra de ejemplos P_e de tamaño $\| P_e \|$
- $P_s(t)$ evoluciona con el tiempo, P_e se mantiene fija
- Cada individuo de P_s se evalúa sobre los últimos τ ejemplos, y lo mismo cada individuo de P_E
- La evaluación de P_s sirve para seleccionar y generar nuevos individuos, mientras que la de P_e solo para seleccionar individuos
- La evaluación es la suma, u otra medida estadística, de encuentros entre soluciones y ejemplos

Ciclo coevolutivo

1. Realizar n veces
   1. s = seleccionar(P_s)
   2. e = seleccionar(P_e)
   3. f = encuentro(s,e)
   4. actualizar-fit(s,f)
   5. act-fitness(e,1/f)
2. s_1 = seleccionar(P_s)
3. s_2 = seleccionar(P_s)
4. s_{hijo} = cruce-mut(s_1, s_2)
5. Realizar m veces
   1. e = seleccionar(P_e)
   2. f = encuentro(s_{hijo},e)
   3. f(s_{hijo}) = actualizar-fit(s_{hijo},f)
   4. f(e) = actualizar-fit(e,1/f)
6. insertar(s_{hijo},f(s_{hijo}),P_s)



FUNCIONES

- Seleccionar(P) selecciona un elemento de la población (P), de manera proporcional a su fitness
- Actualizar fitness(i,f). Cada individuo (i) lleva una memoria del resultado de los últimos *m* encuentros, al actualizar, se elimina de la memoria el resultado del encuentro más antiguo, se añade el actual (f), y se recalcula su fitness
- *n* y *m* son valores pequeños respecto al tamaño de las poblaciones
- Encuentro(s,e). La solución *s* se evalúa con el ejemplo *e* y obtenemos un valor de fitness para este encuentro
- Insertar(s,f(s),P). Si f(s) es mejor que el fitness del peor individuo de la población, s se incluye en la población en sustitución del peor

Características

- Las soluciones solo se evalúan con un conjunto pequeño (m) de ejemplos, en vez de con todos los posibles
- Al principio todos los ejemplos tienen la misma probabilidad de ser seleccionados, pero a medida que son evaluados unos tienen más probabilidad que otros
- Los ejemplos más difíciles tendrán mejor fitness y mayor probabilidad de ser elegidos
- Al utilizar una ventana temporal de m para el cálculo del fitness, tanto soluciones como ejemplos se adaptarán unos a otros

Ejemplo de clasificación

Clasificación

Se desea ubicar un número fijo de prototipos de forma que, mediante la regla del vecino más cercano, se consiga clasificar correctamente cualquier ejemplo

SORTING NETWORKS

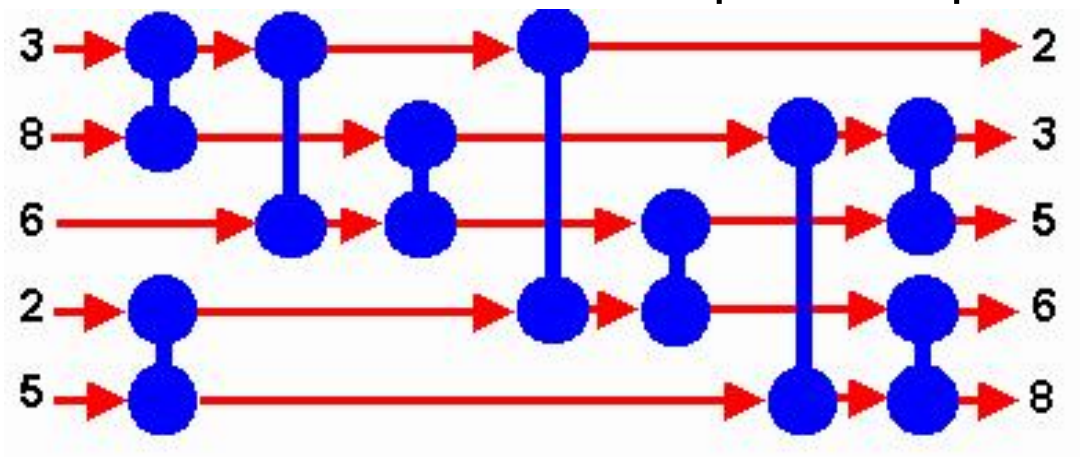
- Se desea encontrar un procedimiento que sea capaz de ordenar un vector de números de forma que:
  - Se conoce la dimensión del vector
  - El procedimiento debe ordenar cualquier combinación de números dentro del vector
  - Sea capaz de hacerlo con el menor número de comparaciones
- El procedimiento realiza un número *c* de comparaciones y funciona de la siguiente forma:
  - Se comparan dos componentes del vector, si la primera componente es menor que la segunda, se intercambian
  - En caso contrario se deja igual

SORTING NETWORKS, EJEMPLO

- Sea el siguiente vector $\vec{v_1} = \{8, 4, 9, 3, 1, 5\}$
- La siguiente red de ordenación:
  $r = \{(1, 3); (2, 3); (5, 6); (3, 5); (4, 5)\}$, es capaz de ordenarlo completamente, produciendo $r(\vec{v_1}) = \{9, 8, 5, 4, 3, 1\}$
- Sin embargo no sería capaz de ordenar el vector
  $\vec{v_2} = \{8, 9, 4, 3, 1, 5\}$, ya que $r(\vec{v_2}) = \{8, 9, 5, 3, 4, 1\}$

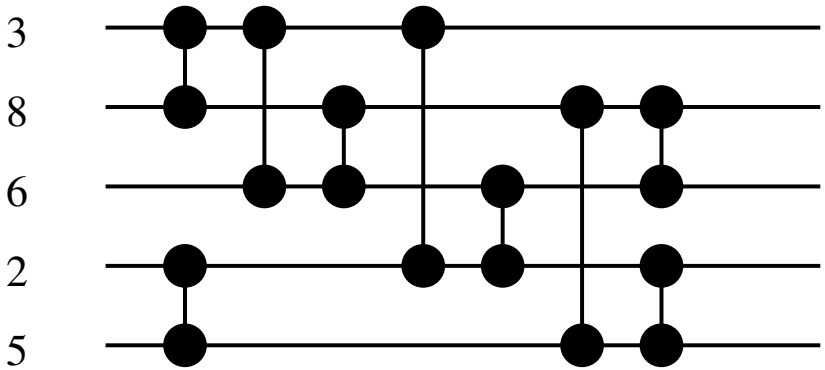
SORTING NETWORKS, REPRESENTACIÓN

- Una red de ordenación se puede representar gráficamente:



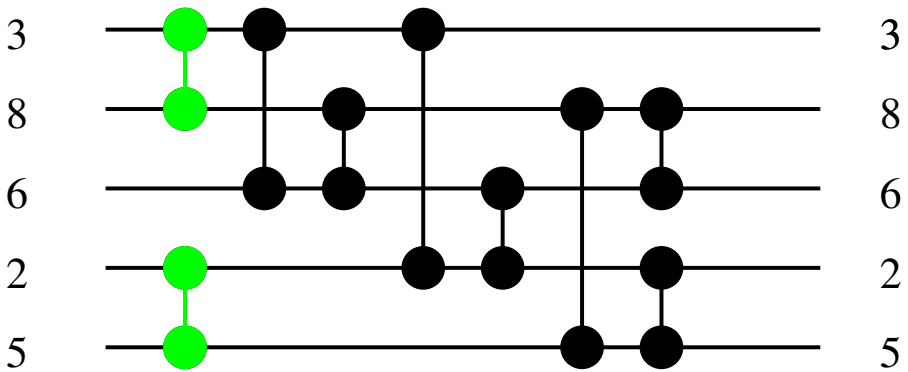
- Se inculyen tantas líneas como dimensión tengan los vectores (en este caso 5)
- Se unen las líneas que representan una comparación. La figura representa a la red
  $r = \{(1,2);(4,5);(1,3);(2,3);(1,4);(3,4);(2,5);(2,3);(4,5)\}$

SORTING NETWORKS, FUNCIONAMIENTO



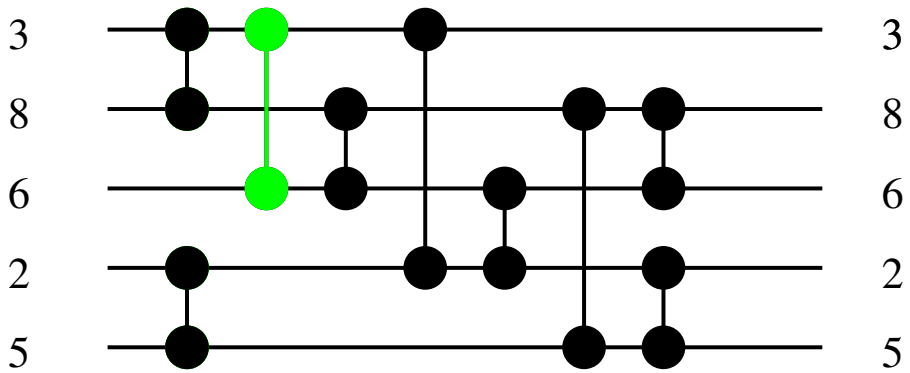
- Se hacen pasar los números por la red

SORTING NETWORKS, FUNCIONAMIENTO



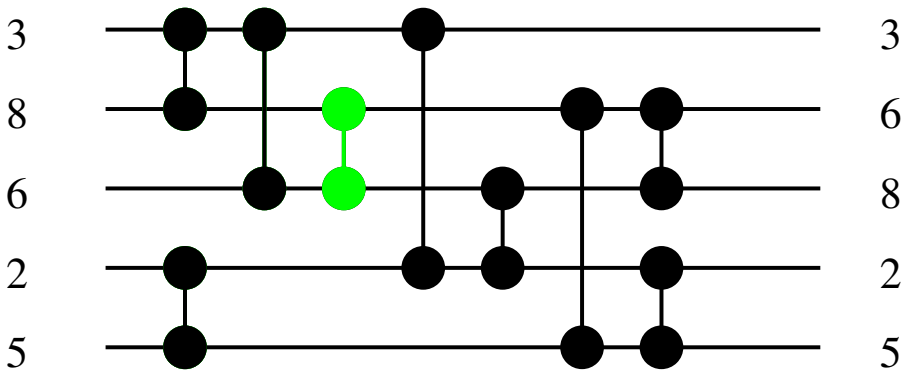
- Se hacen pasar los números por la red
- Se comprueban los valores conectados:
  - Si el primero es menor que el segundo, NO se cambian

SORTING NETWORKS, FUNCIONAMIENTO



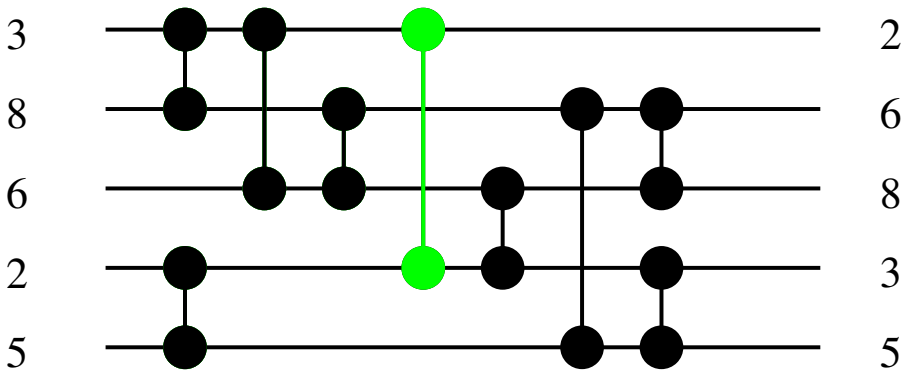
- Se hacen pasar los números por la red
- Se comprueban los valores conectados:
  - Si el primero es menor que el segundo, NO se cambian

SORTING NETWORKS, FUNCIONAMIENTO



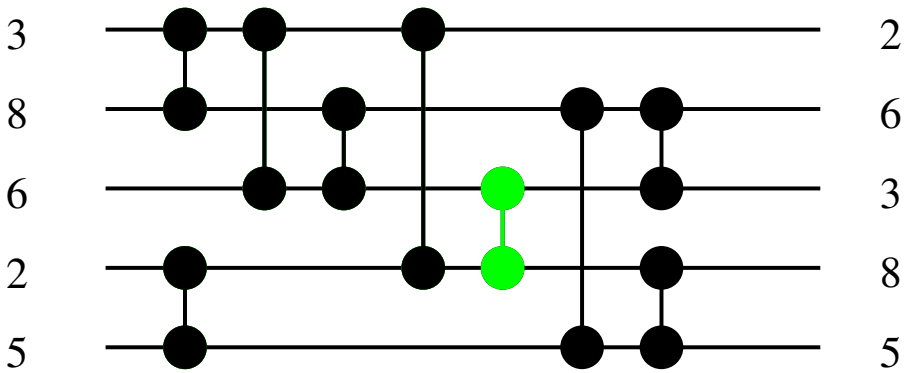
- Se hacen pasar los números por la red
- Se comprueban los valores conectados:
  - Si el primero es mayor que el segundo, se cambian

SORTING NETWORKS, FUNCIONAMIENTO



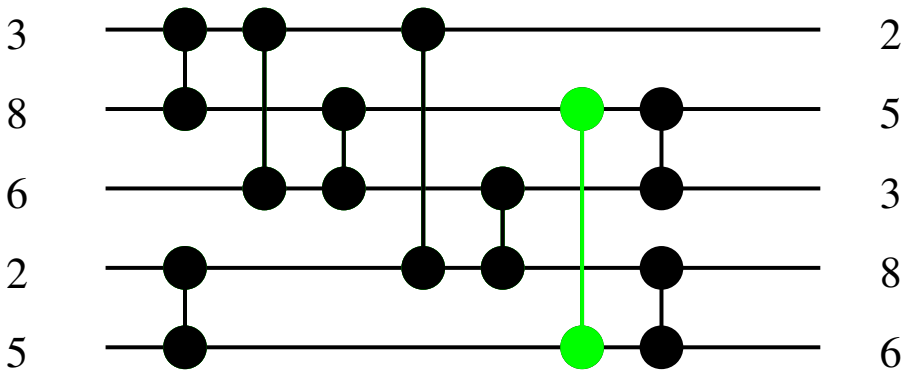
- Se hacen pasar los números por la red
- Se comprueban los valores conectados:
  - Si el primero es mayor que el segundo, se cambian

SORTING NETWORKS, FUNCIONAMIENTO



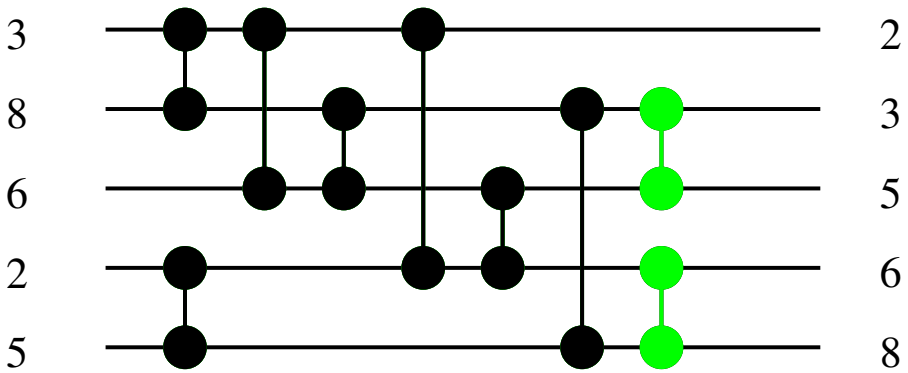
- Se hacen pasar los números por la red
- Se comprueban los valores conectados:
  - Si el primero es mayor que el segundo, se cambian

SORTING NETWORKS, FUNCIONAMIENTO



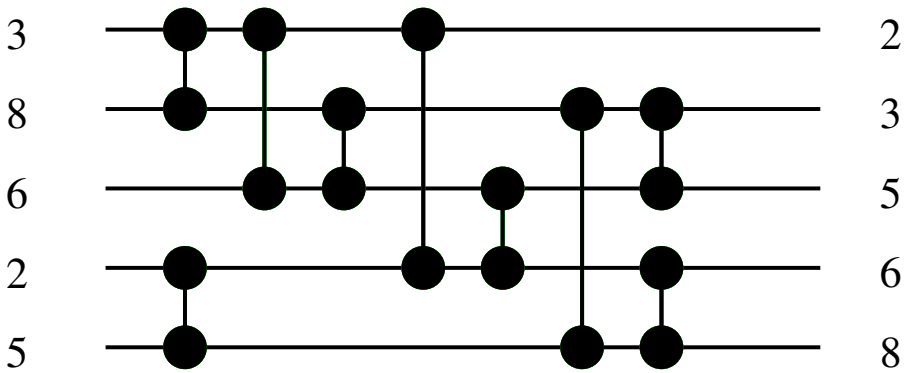
- Se hacen pasar los números por la red
- Se comprueban los valores conectados:
  - Si el primero es mayor que el segundo, se cambian

SORTING NETWORKS, FUNCIONAMIENTO



- Se hacen pasar los números por la red
- Se comprueban los valores conectados:
  - Si el primero es mayor que el segundo, se cambian

SORTING NETWORKS, FUNCIONAMIENTO

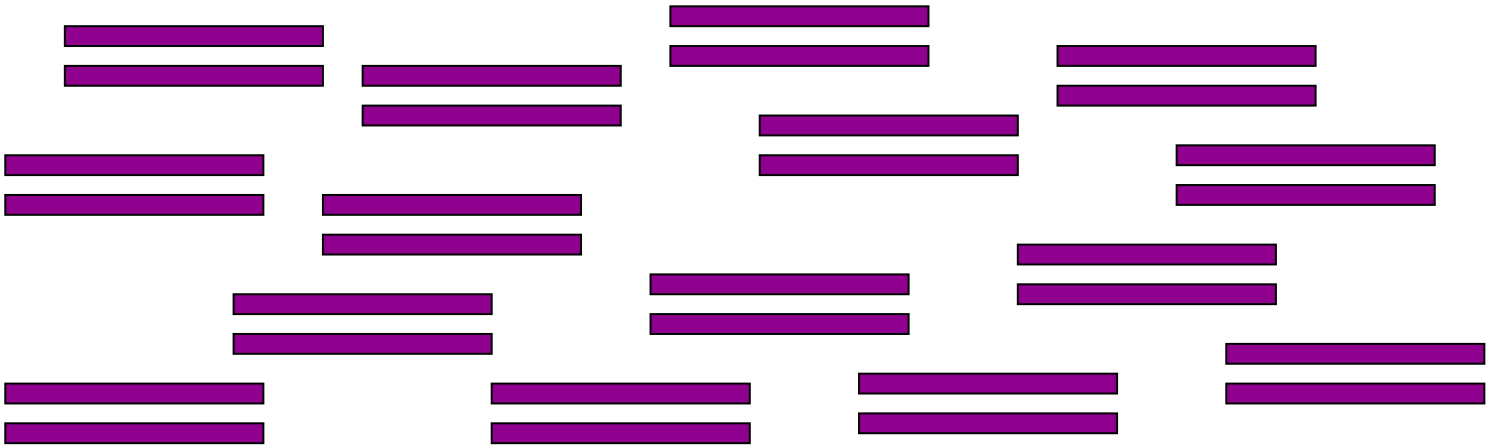


- Se hacen pasar los números por la red
- Se comprueban los valores conectados:
  - Si el primero es mayor que el segundo, se cambian

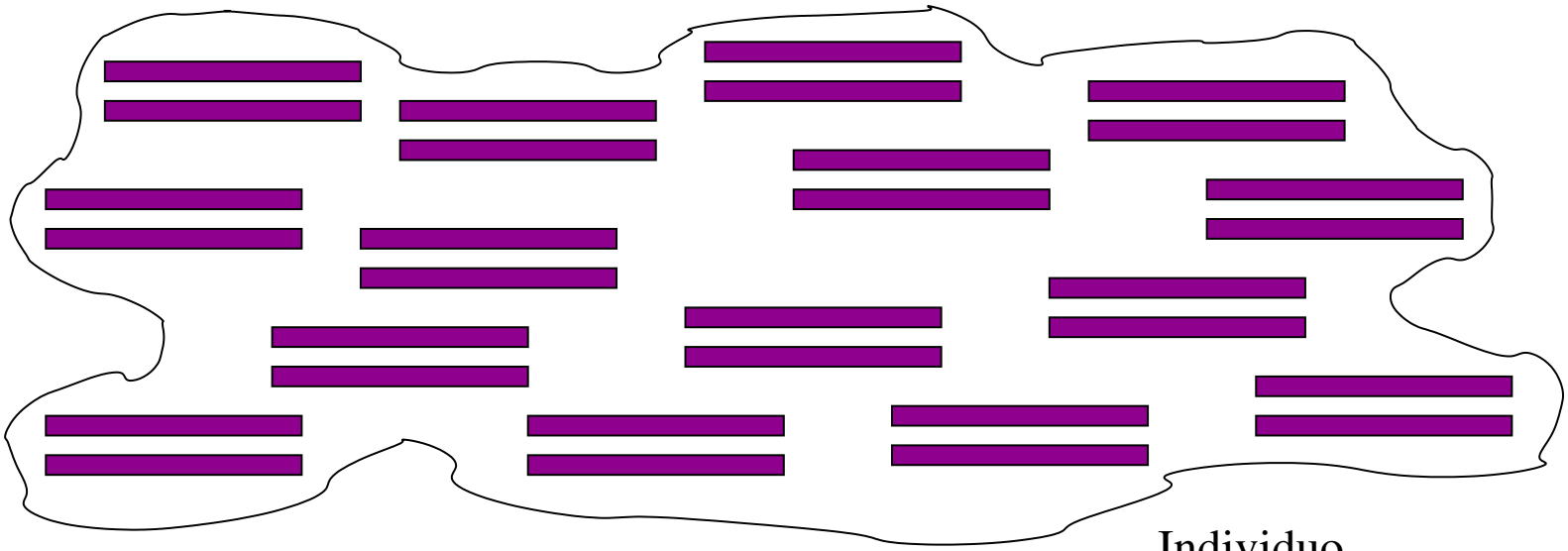
TRABAJOS PREVIOS

- El tema ha sido objeto de estudio sobre todo en los años 60. Existe una recopilación sobre el tema en el libro “Art of computer programming” de Knuth de 1973
- Se han propuesto numerosas soluciones para el problema con dimensión 16:
  - Bose y Nelson en 1962 dan una solución con 65 comparaciones
  - Batcher y Floyd y Knuth en 1964 dan dos soluciones con 63 comparaciones
  - Shapiro en 1969 da una solución con 62 comparaciones
  - Green en 1969 da una solución con 60 comparaciones, que es la mejor hasta la fecha

SORTING NETWORKS, CODIFICACIÓN

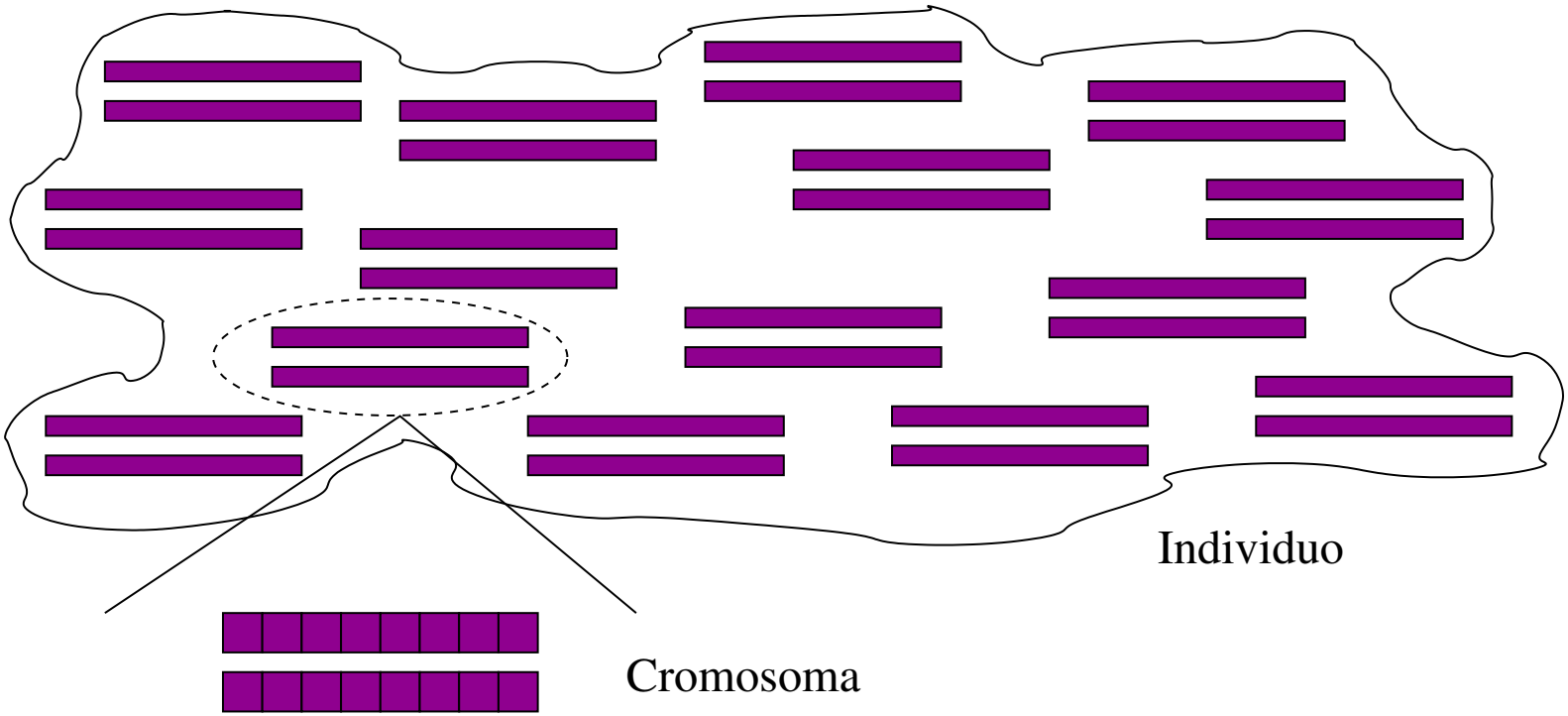


SORTING NETWORKS, CODIFICACIÓN

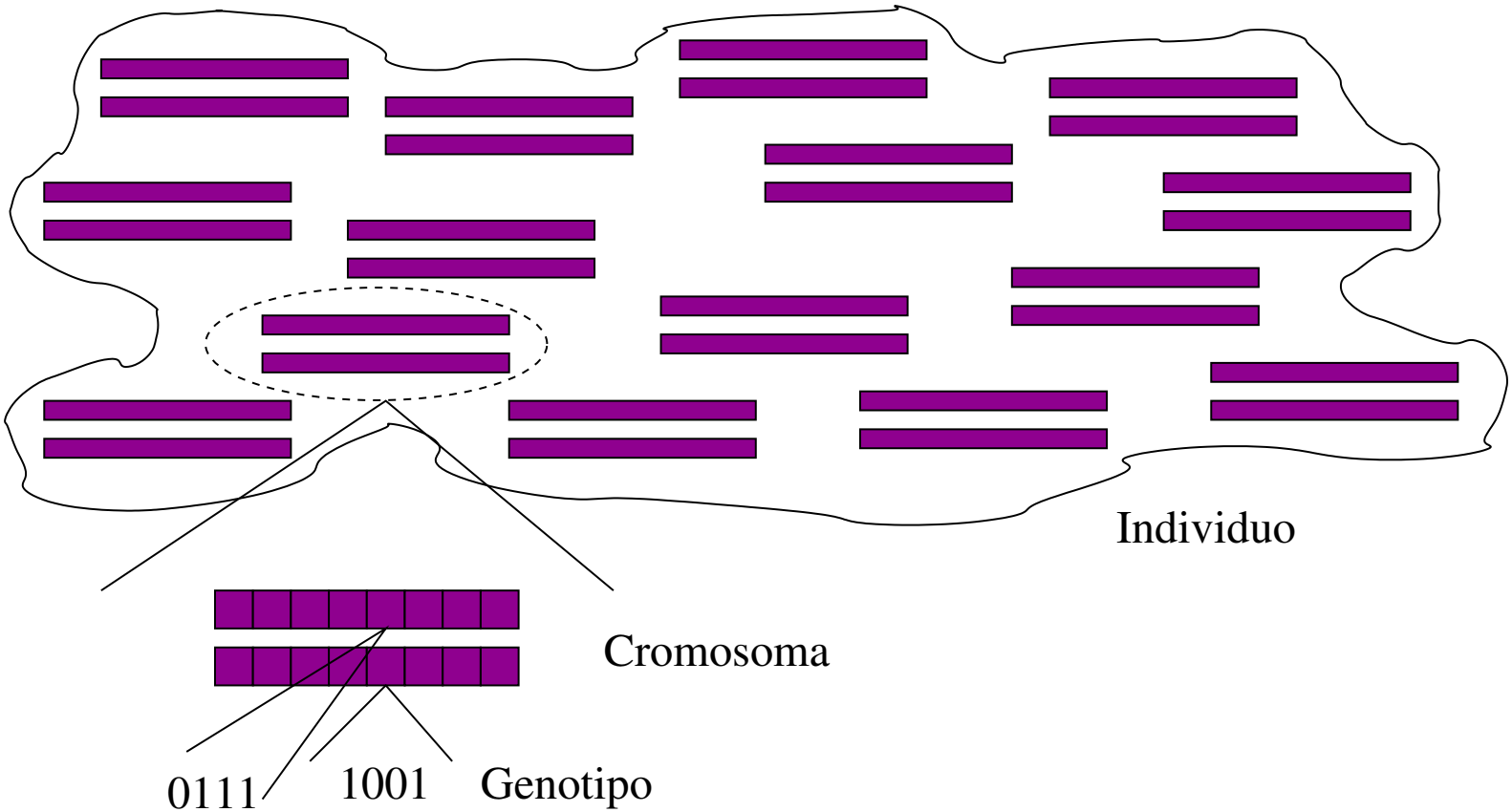


Individuo

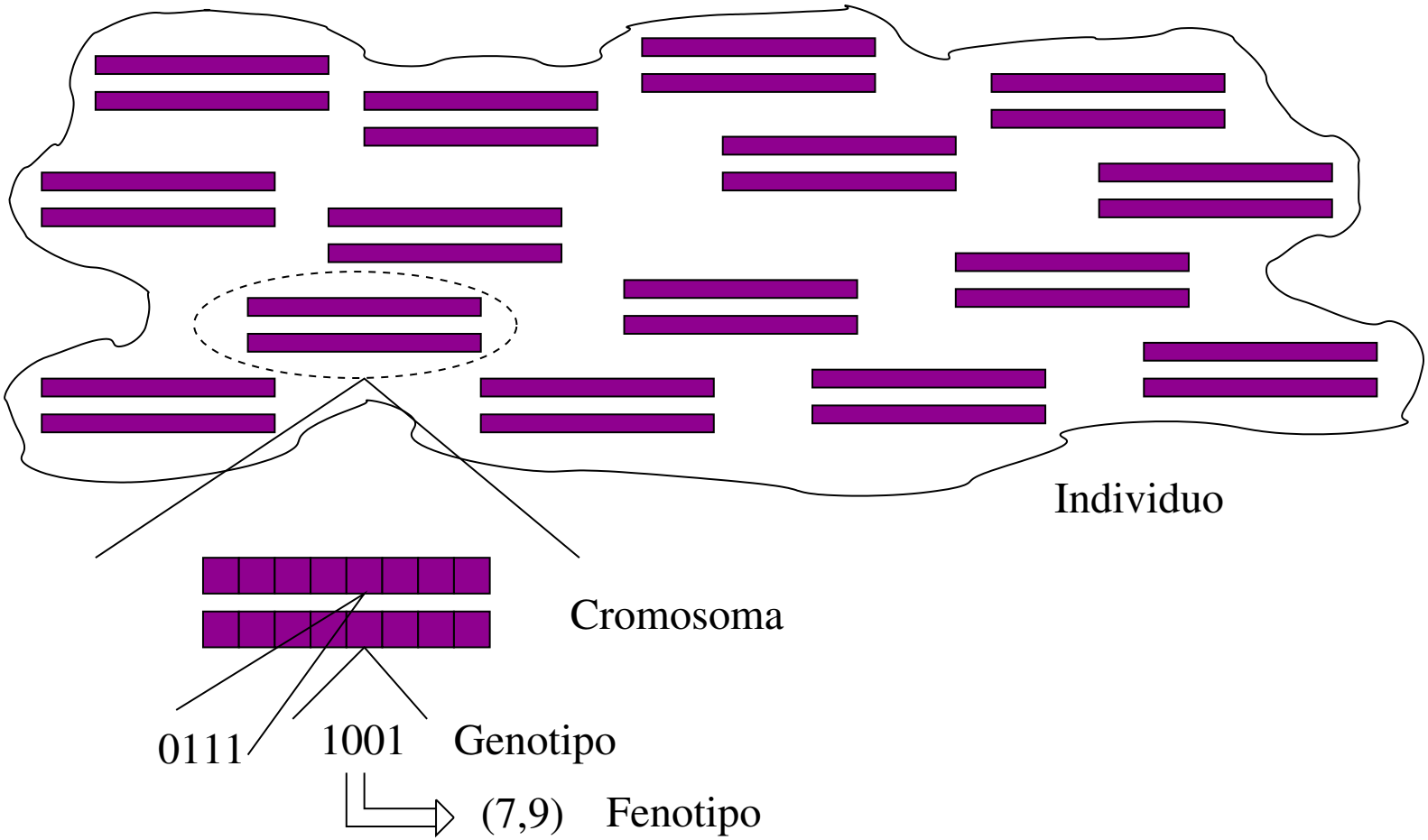
SORTING NETWORKS, CODIFICACIÓN



SORTING NETWORKS, CODIFICACIÓN



SORTING NETWORKS, CODIFICACIÓN



CODIFICACIÓN

Cada individuo:

- 15 pares de cromosomas de 32 bits. Codificación diploide
- Cada 4 bits es un nodo. Hay 16 nodos, 8 en cada cromosoma homólogo
- Cada 8 bits es una comparación, ya que codifican 2 líneas. Cada nodo de cada cromosoma homólogo es un elemento de la comparación
- Cada cromosoma tiene ocho nodos, cada par de cromosomas 8 comparaciones
- cada individuo tiene $8 * 15 = 120$ comparaciones. Cada individuo es una red con 120 comparaciones

CODIFICACIÓN

Cada individuo:

- 15 pares de cromosomas de 32 bits. Codificación diploide
- Cada 4 bits es un nodo. Hay 16 nodos, 8 en cada cromosoma homólogo
- Cada 8 bits es una comparación, ya que codifican 2 líneas. Cada nodo de cada cromosoma homólogo es un elemento de la comparación
- Cada cromosoma tiene ocho nodos, cada par de cromosomas 8 comparaciones
- cada individuo tiene $8 * 15 = 120$ comparaciones. Cada individuo es una red con 120 comparaciones

CODIFICACIÓN

Cada individuo:

- 15 pares de cromosomas de 32 bits. Codificación diploide
- Cada 4 bits es un nodo. Hay 16 nodos, 8 en cada cromosoma homólogo
- Cada 8 bits es una comparación, ya que codifican 2 líneas. Cada nodo de cada cromosoma homólogo es un elemento de la comparación
- Cada cromosoma tiene ocho nodos, cada par de cromosomas 8 comparaciones
- cada individuo tiene $8 * 15 = 120$ comparaciones. Cada individuo es una red con 120 comparaciones

CODIFICACIÓN

Cada individuo:

- 15 pares de cromosomas de 32 bits. Codificación diploide
- Cada 4 bits es un nodo. Hay 16 nodos, 8 en cada cromosoma homólogo
- Cada 8 bits es una comparación, ya que codifican 2 líneas. Cada nodo de cada cromosoma homólogo es un elemento de la comparación
- Cada cromosoma tiene ocho nodos, cada par de cromosomas 8 comparaciones
- cada individuo tiene $8 * 15 = 120$ comparaciones. Cada individuo es una red con 120 comparaciones

CODIFICACIÓN

Cada individuo:

- 15 pares de cromosomas de 32 bits. Codificación diploide
- Cada 4 bits es un nodo. Hay 16 nodos, 8 en cada cromosoma homólogo
- Cada 8 bits es una comparación, ya que codifican 2 líneas. Cada nodo de cada cromosoma homólogo es un elemento de la comparación
- Cada cromosoma tiene ocho nodos, cada par de cromosomas 8 comparaciones
- cada individuo tiene $8 * 15 = 120$ comparaciones. Cada individuo es una red con 120 comparaciones

Reducción de comparaciones

- Problema
  - La solución anterior no es buena, ya que el número de comparaciones es fija
  - Se necesita que sea variable, para así obtener la red con menor número de comparaciones
- Solución
  - Como son individuos diploides, si son homocigóticos en un gen se reduce el número de comparaciones
  - Si un gen es homocigótico expresa un solo nodo, si es heterocigótico expresa dos nodos

Reducción de comparaciones

- Problema
  - La solución anterior no es buena, ya que el número de comparaciones es fija
  - Se necesita que sea variable, para así obtener la red con menor número de comparaciones
- Solución
  - Como son individuos diploides, si son homocigóticos en un gen se reduce el número de comparaciones
  - Si un gen es homocigótico expresa un solo nodo, si es heterocigótico expresa dos nodos

Reducción de comparaciones

- Si todos son heterocigóticos
  - Como hay 8 genes por cromosomas, y 15 pares de cromosomas, el número total de nodos es $15 \cdot 2 \cdot 8 = 240$
  - Como cada dos nodos es una comparación habrá 120 comparaciones
- Si todos son homocigóticos
  - El número total de nodos será $15 \cdot 8 = 120$,
  - Por lo tanto 60 comparaciones
- De esta forma es posible codificar cualquier número y combinación de comparaciones entre 60 y 120

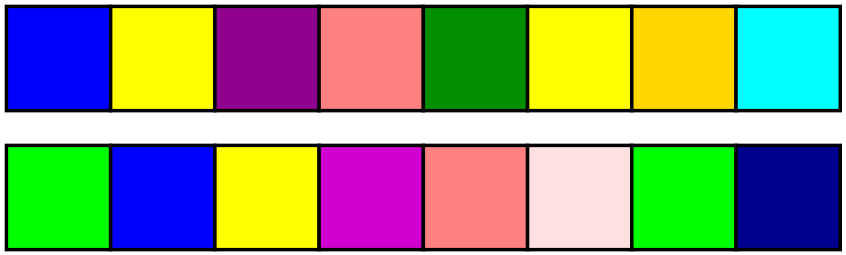
Reducción de comparaciones

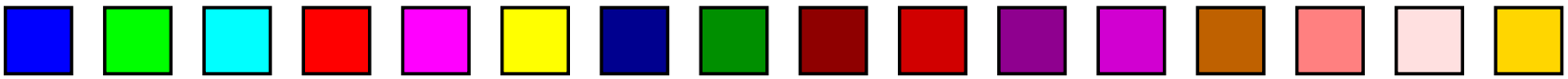
- Si todos son heterocigóticos
  - Como hay 8 genes por cromosomas, y 15 pares de cromosomas, el número total de nodos es $15 \cdot 2 \cdot 8 = 240$
  - Como cada dos nodos es una comparación habrá 120 comparaciones
- Si todos son homocigóticos
  - El número total de nodos será $15 \cdot 8 = 120$,
  - Por lo tanto 60 comparaciones
- De esta forma es posible codificar cualquier número y combinación de comparaciones entre 60 y 120

Reducción de comparaciones

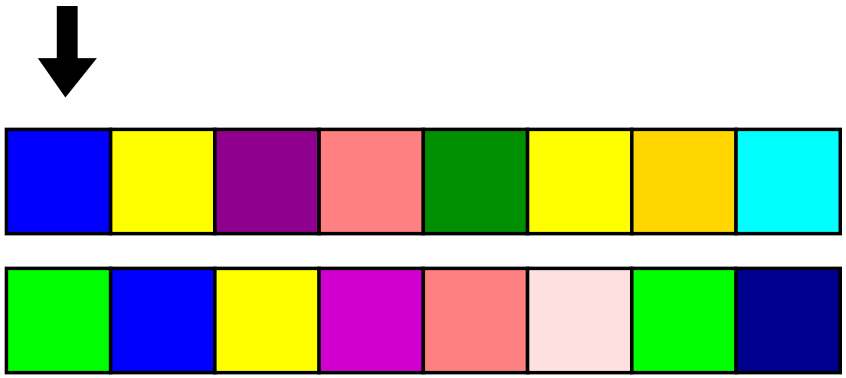
- Si todos son heterocigóticos
  - Como hay 8 genes por cromosomas, y 15 pares de cromosomas, el número total de nodos es $15 \cdot 2 \cdot 8 = 240$
  - Como cada dos nodos es una comparación habrá 120 comparaciones
- Si todos son homocigóticos
  - El número total de nodos será $15 \cdot 8 = 120$,
  - Por lo tanto 60 comparaciones
- De esta forma es posible codificar cualquier número y combinación de comparaciones entre 60 y 120

CODIFICACIÓN DIPLOIDE

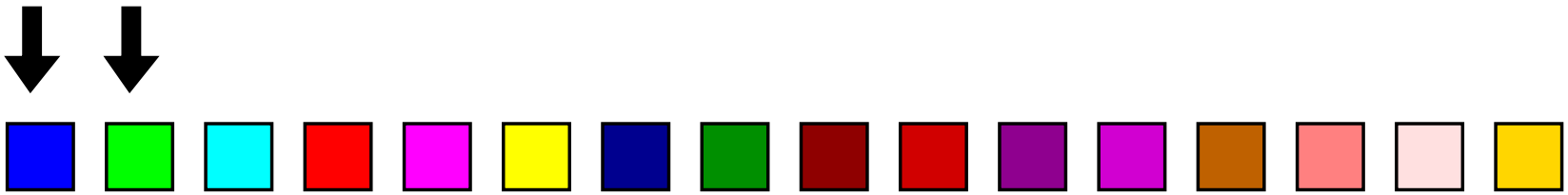




CODIFICACIÓN DIPLOIDE

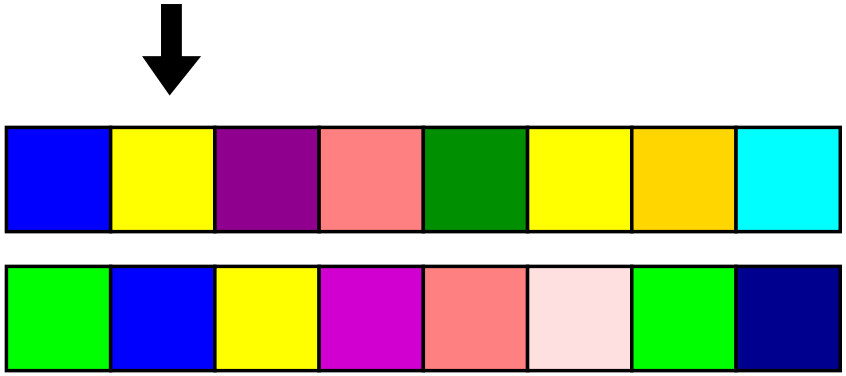


(1,2)

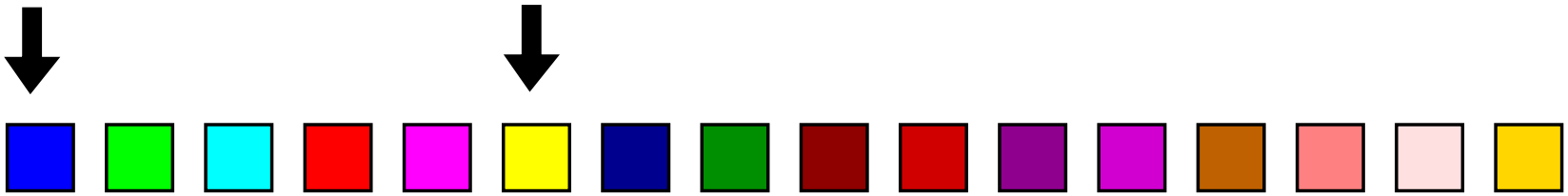


1 2

CODIFICACIÓN DIPLOIDE

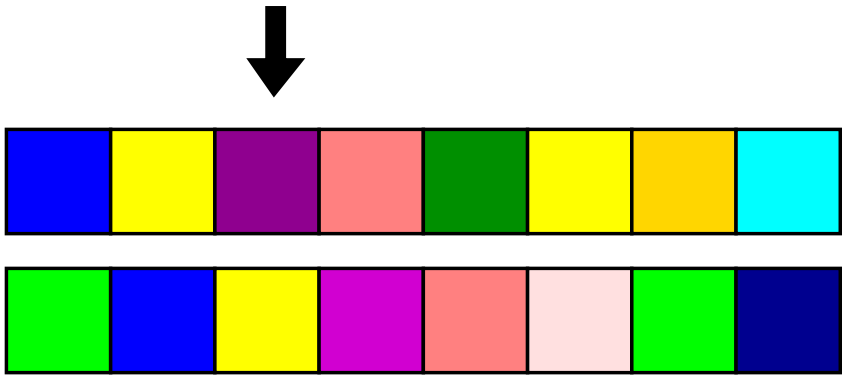


(1,2) (6,1)

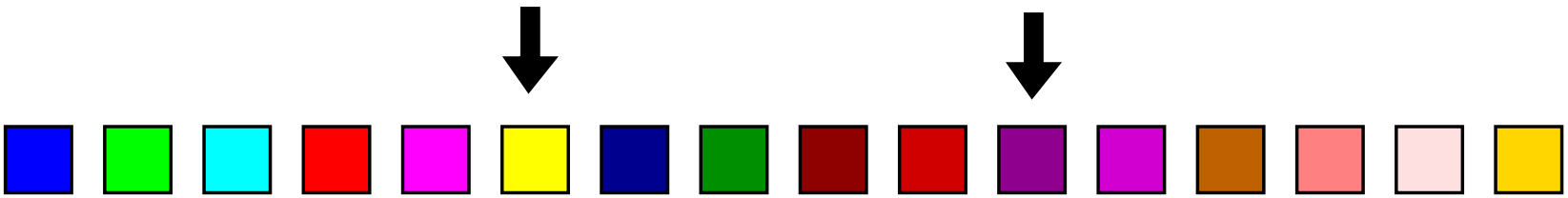


1 6

CODIFICACIÓN DIPLOIDE

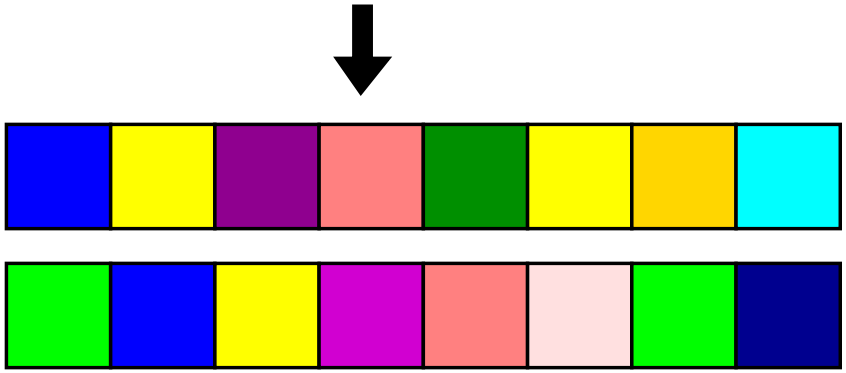


(1,2) (6,1) (11,6)

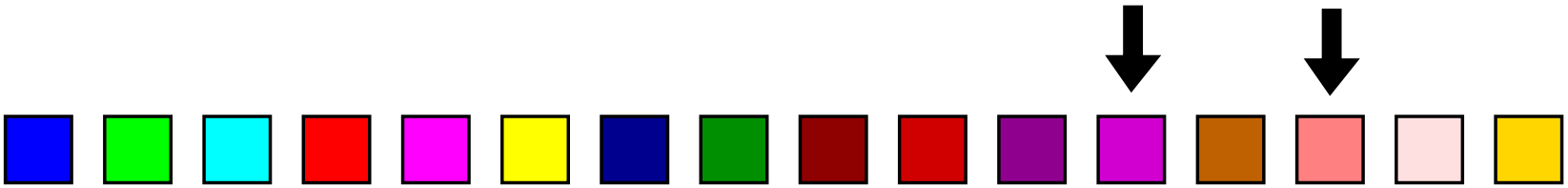


6 11

CODIFICACIÓN DIPLOIDE

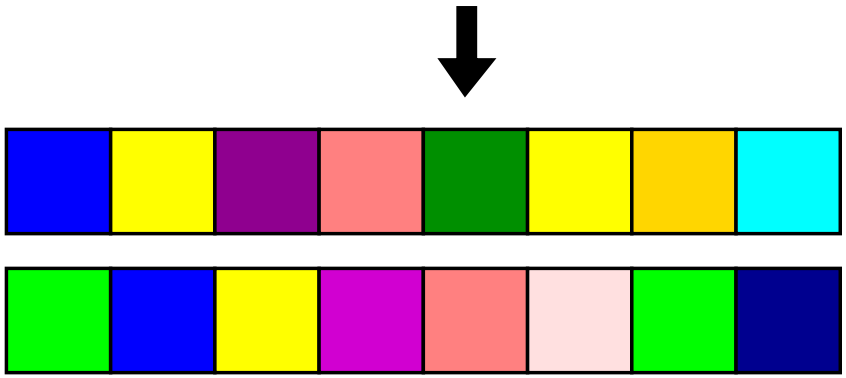


(1,2) (6,1) (11,6) (14,12)

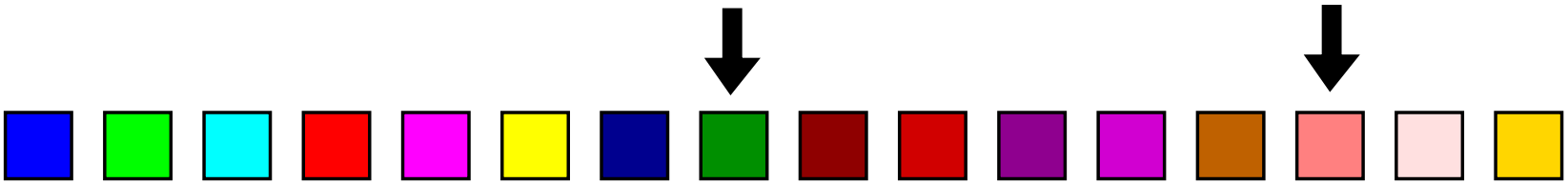


12 14

CODIFICACIÓN DIPLOIDE

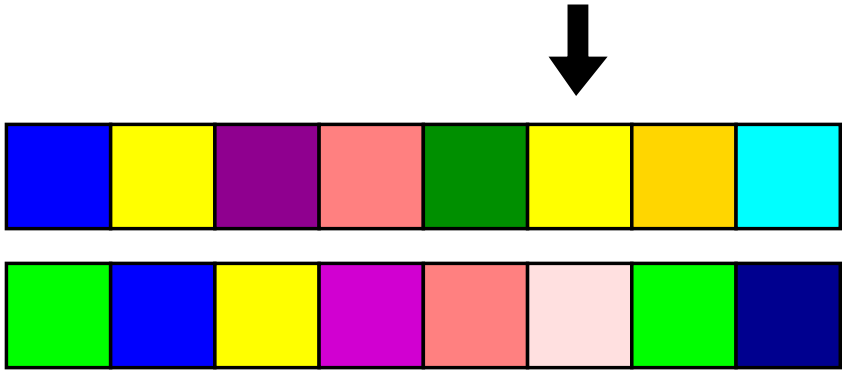


(1,2) (6,1) (11,6) (14,12) (8,14)

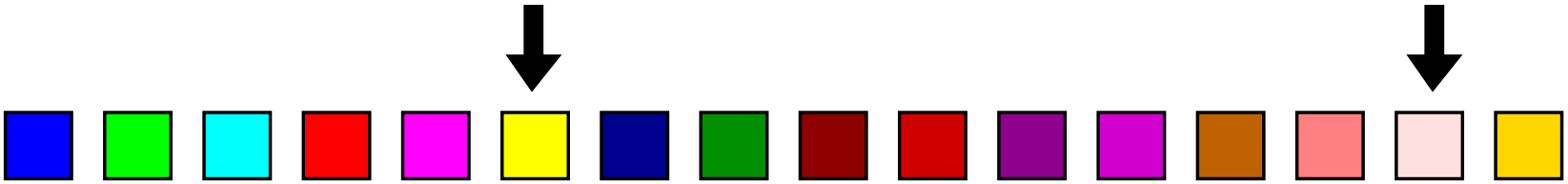


8 14

CODIFICACIÓN DIPLOIDE

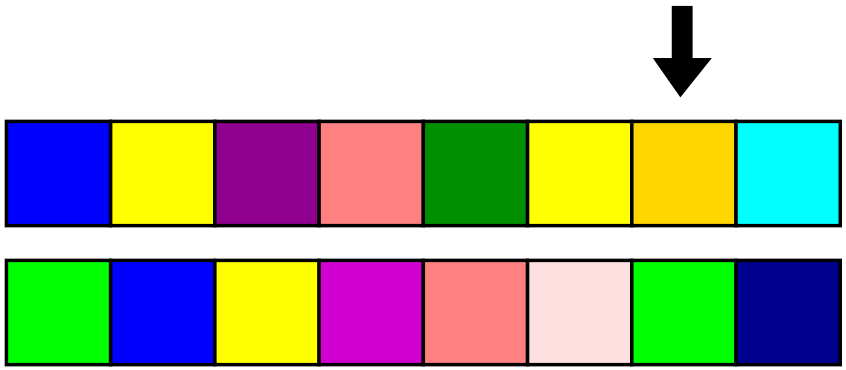


(1,2) (6,1) (11,6) (14,12) (8,14) (6,15)

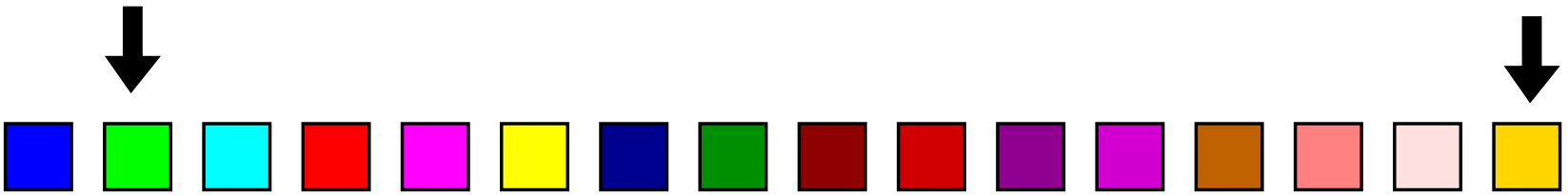


6 15

CODIFICACIÓN DIPLOIDE

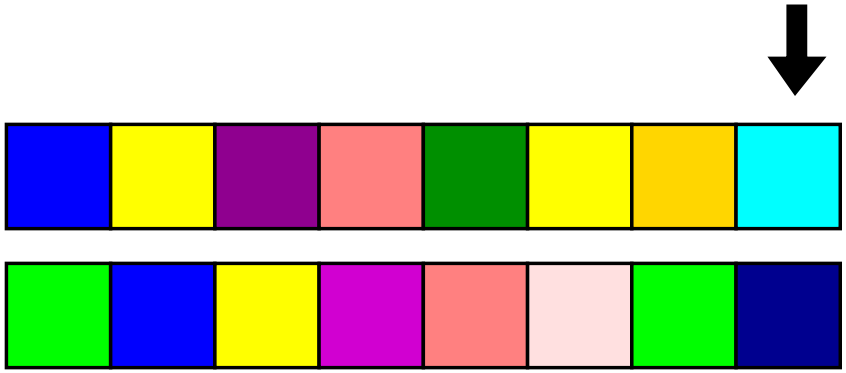


(1,2) (6,1) (11,6) (14,12) (8,14) (6,15) (16,2)



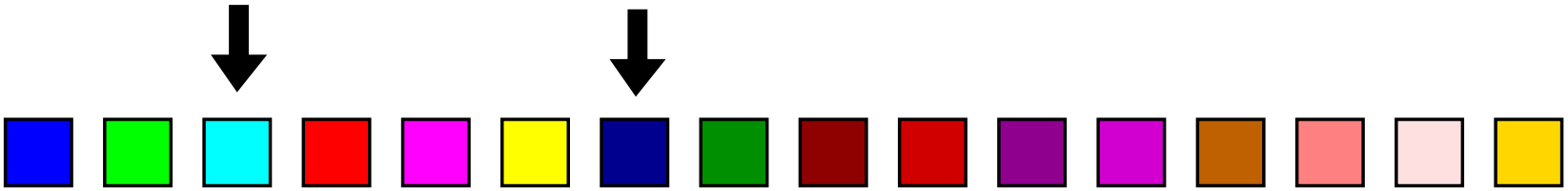
2 16

CODIFICACIÓN DIPLOIDE



(1,2) (6,1) (11,6) (14,12) (8,14) (6,15) (16,2) (3,7)

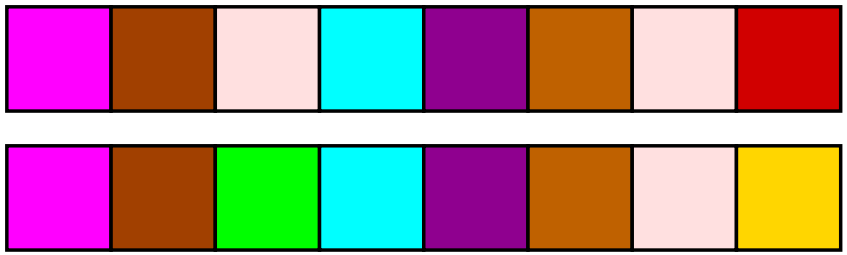
8 Comparaciones

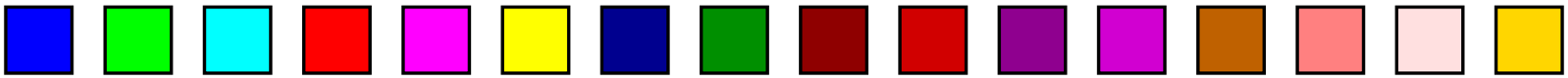


3 7

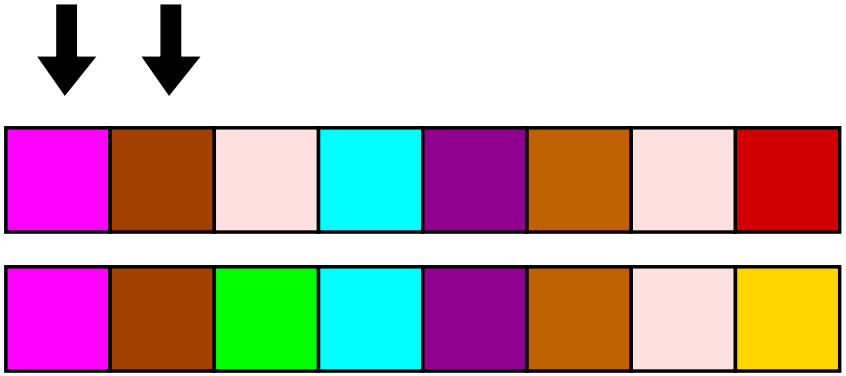
Ningun homocigotico

CODIFICACIÓN DIPLOIDE

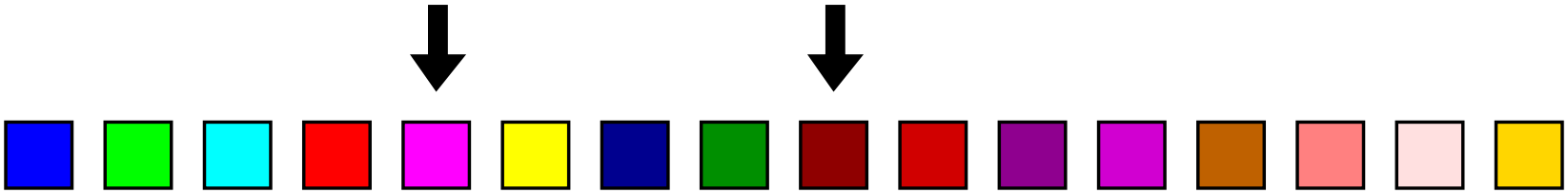




CODIFICACIÓN DIPLOIDE

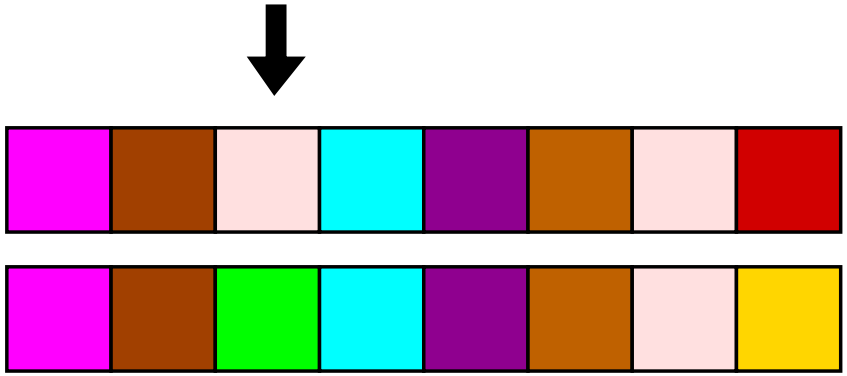


(5,9)

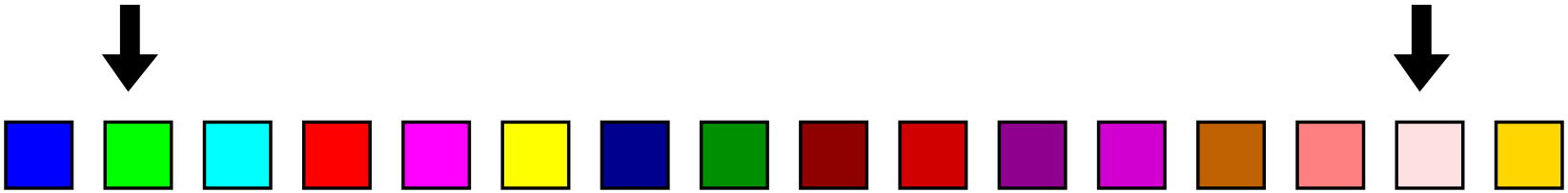


5 9

CODIFICACIÓN DIPLOIDE

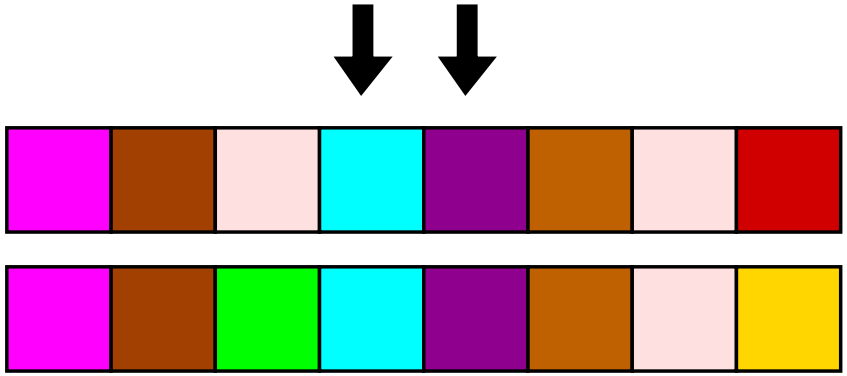


(5,9) (15,2)

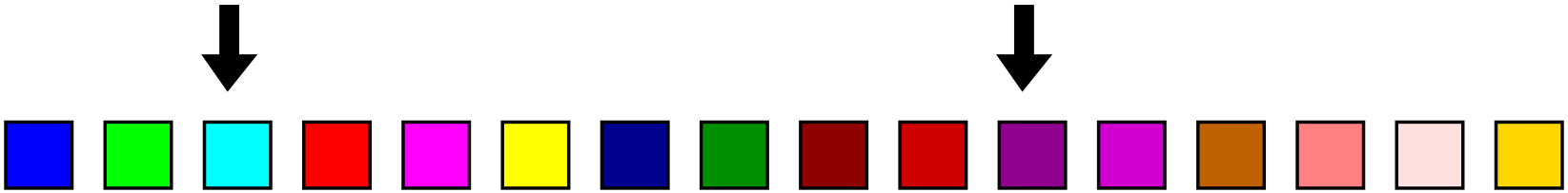


2 15

CODIFICACIÓN DIPLOIDE

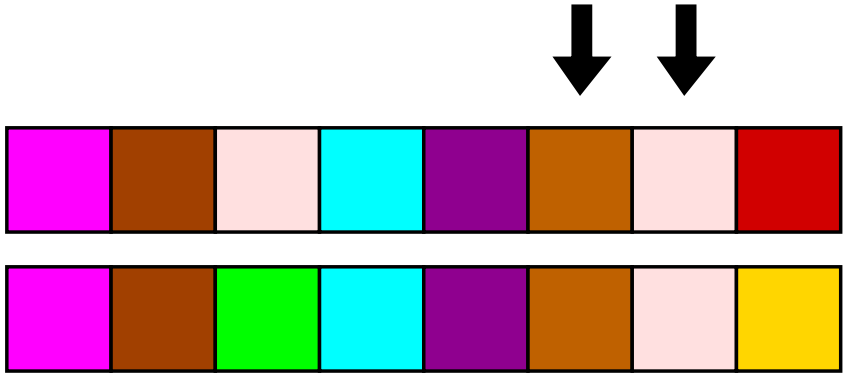


(5,9) (15,2) (3,11)

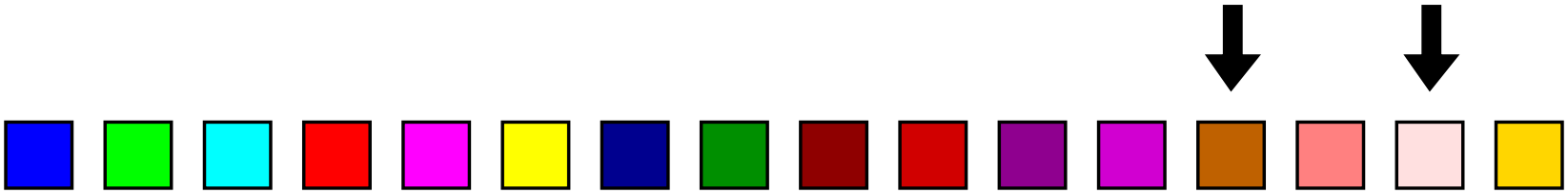


3 11

CODIFICACIÓN DIPLOIDE

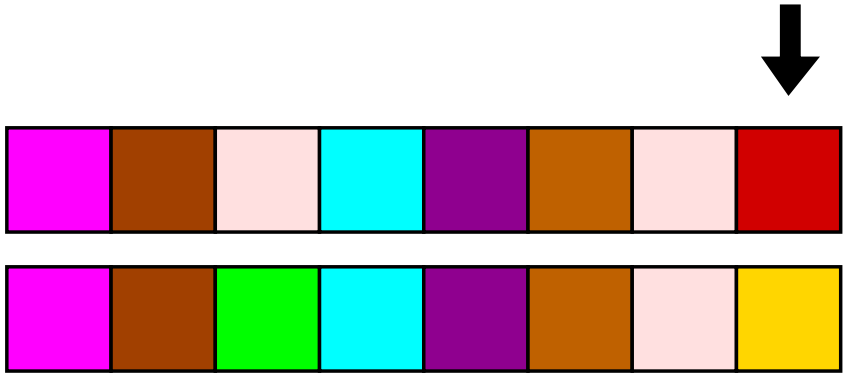


(5,9) (15,2) (3,11) (13,15)

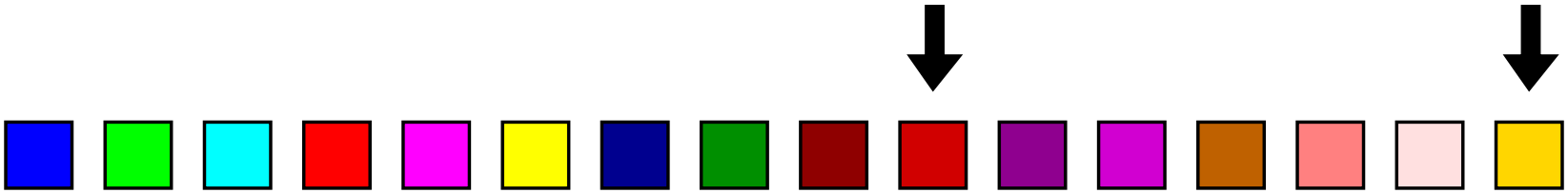


13 15

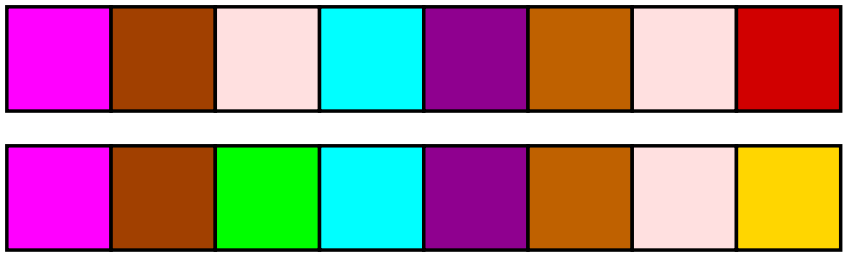
CODIFICACIÓN DIPLOIDE



(5,9) (15,2) (3,11) (13,15) (10,16)



CODIFICACIÓN DIPLOIDE



(5,9) (15,2) (3,11) (13,15) (10,16)

5 Comparaciones

6 homocigotico

EVALUACIÓN DE LOS INDIVIDUOS

- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y comprobar cuántos de ellos son ordenados por la red
- Si se generan 100 ejemplos de vectores desordenados, la función de evaluación será qué porcentaje de ellos, después de utilizar la red codificada por el individuo a evaluar, aparecen perfectamente ordenados
- Si una red ordena 17 de 100 vectores diferentes, su evaluación será 1,7. Si ordena 98 será 9,8
- También se puede tener en cuenta cuantos elementos están desordenados, en los vectores que no han sido totalmente ordenados

Evaluación de los individuos

- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y comprobar cuántos de ellos son ordenados por la red
- Si se generan 100 ejemplos de vectores desordenados, la función de evaluación será qué porcentaje de ellos, después de utilizar la red codificada por el individuo a evaluar, aparecen perfectamente ordenados
- Si una red ordena 17 de 100 vectores diferentes, su evaluación será 1,7. Si ordena 98 será 9,8
- También se puede tener en cuenta cuantos elementos están desordenados, en los vectores que no han sido totalmente ordenados

Evaluación de los individuos

- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y comprobar cuántos de ellos son ordenados por la red
- Si se generan 100 ejemplos de vectores desordenados, la función de evaluación será qué porcentaje de ellos, después de utilizar la red codificada por el individuo a evaluar, aparecen perfectamente ordenados
- Si una red ordena 17 de 100 vectores diferentes, su evaluación será 1,7. Si ordena 98 será 9,8
- También se puede tener en cuenta cuantos elementos están desordenados, en los vectores que no han sido totalmente ordenados

Evaluación de los individuos

- Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y comprobar cuántos de ellos son ordenados por la red
- Si se generan 100 ejemplos de vectores desordenados, la función de evaluación será qué porcentaje de ellos, después de utilizar la red codificada por el individuo a evaluar, aparecen perfectamente ordenados
- Si una red ordena 17 de 100 vectores diferentes, su evaluación será 1,7. Si ordena 98 será 9,8
- También se puede tener en cuenta cuantos elementos están desordenados, en los vectores que no han sido totalmente ordenados

Conjuntos de entrenamiento como sist. genéticos

- La solución es generar una población de conjuntos de entrenamiento
- Cada individuo es un conjunto de entrenamiento, compuesto por *n* valores reales
- Cada individuo-conjunto-de-entrenamiento es evaluado con un conjunto de redes de ordenación
- Cada vez que el conjunto de entrenamiento no es ordenado por una red de ordenación se suma 1 a su valor de evaluación
- De esta forma la función de evaluación del conjunto de entrenamiento será el número de redes de ordenación que NO han sido capaces de ordenar a dicho conjunto
- Esta evaluación se utiliza dentro de un Algoritmo Genético, de forma que se evolucionan los conjuntos de entrenamiento más complicados para un conjunto de redes de ordenación

COEVOLUCIÓN COMPETITIVA

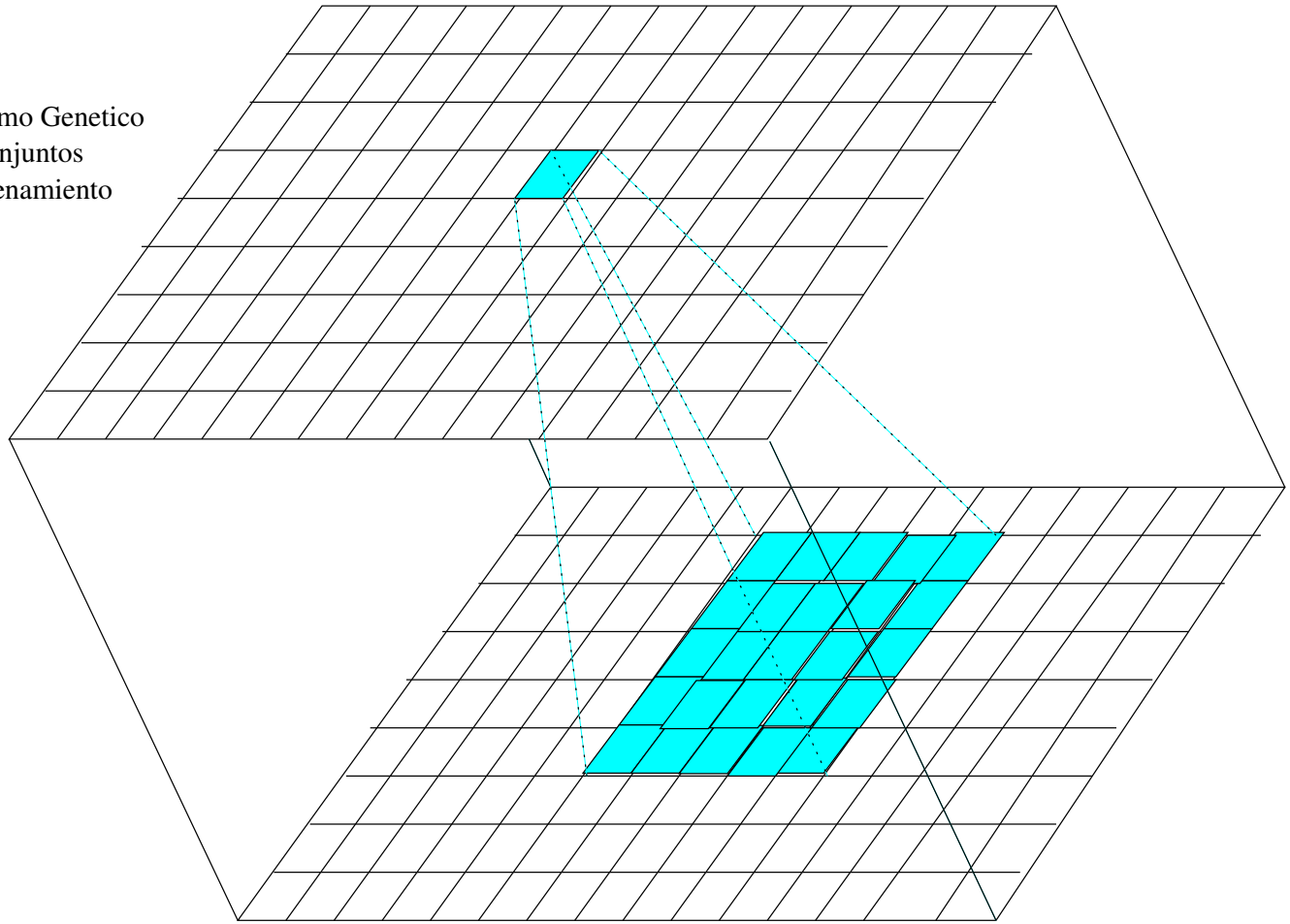
- Se dispone de dos algoritmos genéticos que coevolucionan, cada uno hace evolucionar al otro más eficazmente, de forma competitiva
- Por un lado el AG de las redes de ordenación intenta generar individuos que ordenen el mayor número de los vectores con los que son evaluados
- Por otra parte cada vector trata de que le ordene el menor número de redes de ordenación
- La evaluación de uno es inversa a la evaluación del otro $\Rightarrow$ coevolución competitiva

Distribución espacial de las dos poblaciones

- Cada AG se distribuye en una rejilla bidimensional, y se disponen una encima de la otra, de forma que:
  - Cada individuo vector es evaluado con las redes que están justo debajo en su entorno
  - Cada individuo red es evaluado con los vectores que están justo encima en su entorno
- En cada Población, los individuos próximos en la rejilla se corresponden con codificaciones (genotipos) similares
- Esto se consigue haciendo que el sobrecruzamiento se produzca entre individuos que estén próximos, y que los descendientes reemplacen a individuos próximos que sean peores

CODIFICACIÓN DIPLOIDE

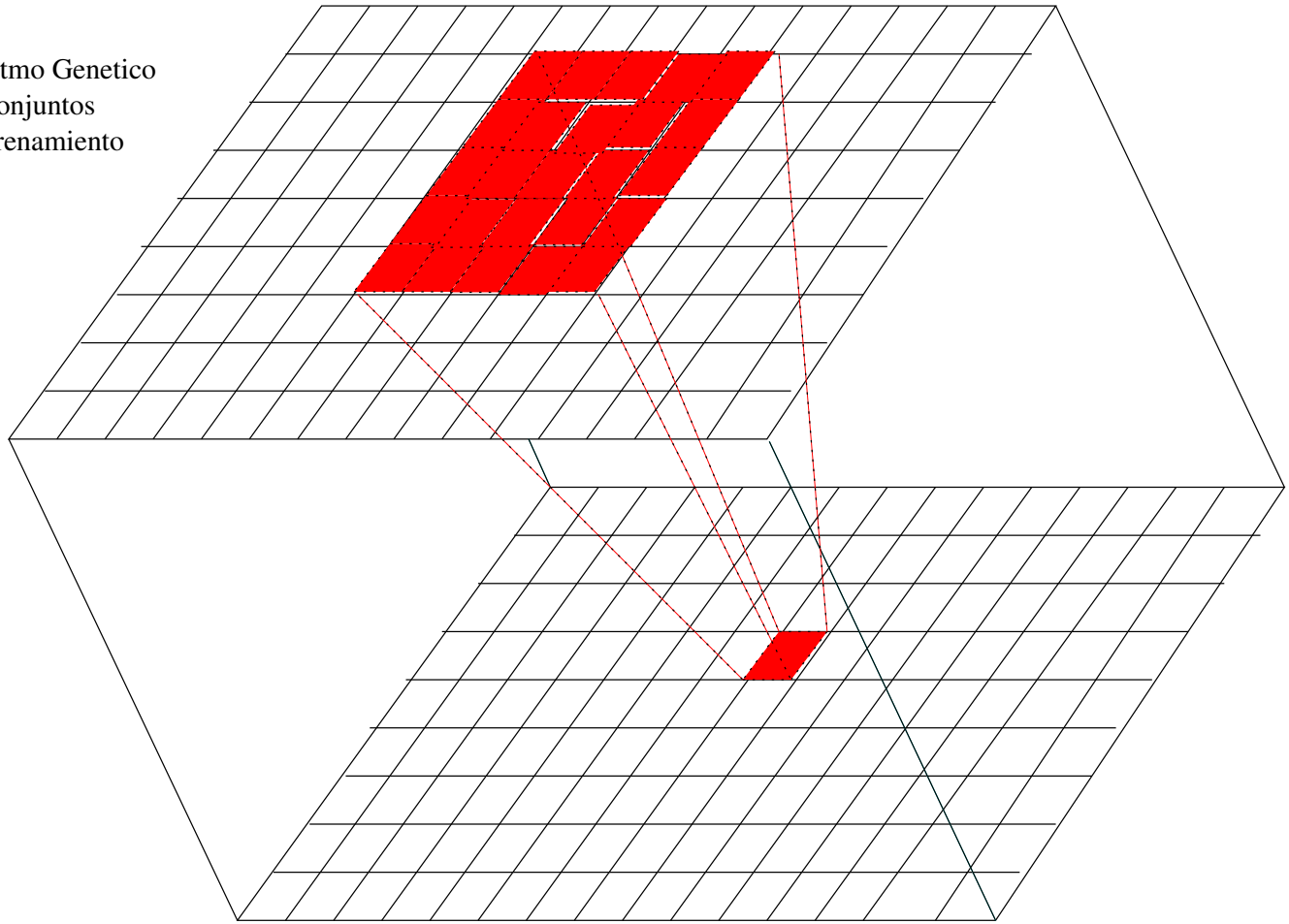
Algoritmo Genetico
de conjuntos
de entrenamiento



Algoritmo Genetico
de redes
de ordenacion

CODIFICACIÓN DIPLOIDE

Algoritmo Genetico
de conjuntos
de entrenamiento



Algoritmo Genetico
de redes
de ordenacion

Solución al problema del entrenador ideal

- Solución $\Rightarrow$ Coevolución competitiva
- Evolución de un conjunto de ejemplos para cada red de ordenación
- Solapamiento por proximidad de los conjuntos de ejemplos
- Individuos parecidos tienen conjuntos de ejemplos similares y viceversa
- Para cada red se evoluciona el peor conjunto de vectores con el que podría ser evaluado, gracias a que sus individuos-vectores les “parasitan”
- Permite disponer de conjunto de vectores pequeños y representativos

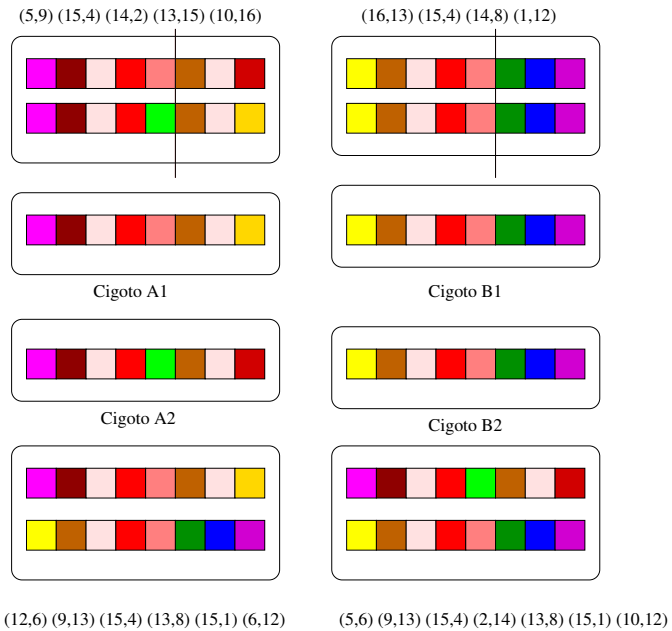
¿Cómo reducir el tamaño de las redes?

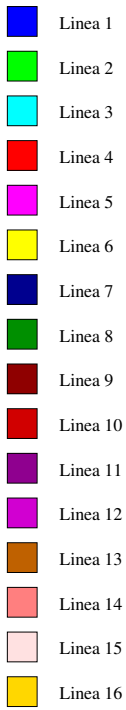
- El objetivo final es obtener redes que clasifiquen, pero con el menor número posible de comparaciones
- Habría que introducir un sesgo, presión selectiva, en el procedimiento para que se prefirieran redes más pequeñas
- Lo habitual es incluir dicho sesgo en la función de evaluación de la red
- Hillis utiliza una propiedad biológica, que se cumple gracias a la codificación elegida:

> *En la naturaleza se produce una presión selectiva hacia la homocigosis, ya que esta protege a los genes más importantes. Si un gen importante es homocigótico, tiene menos posibilidades de ser eliminado por meiosis, y no propagado a los descendientes del individuo*

¿Cómo reducir el tamaño de las redes?

En este caso hay una presión selectiva indirecta hacia la minimalidad (redes con número mínimo de comparaciones), debido a que, como en la naturaleza, la homocigosis protege a las comparaciones cruciales





Temario

Temario