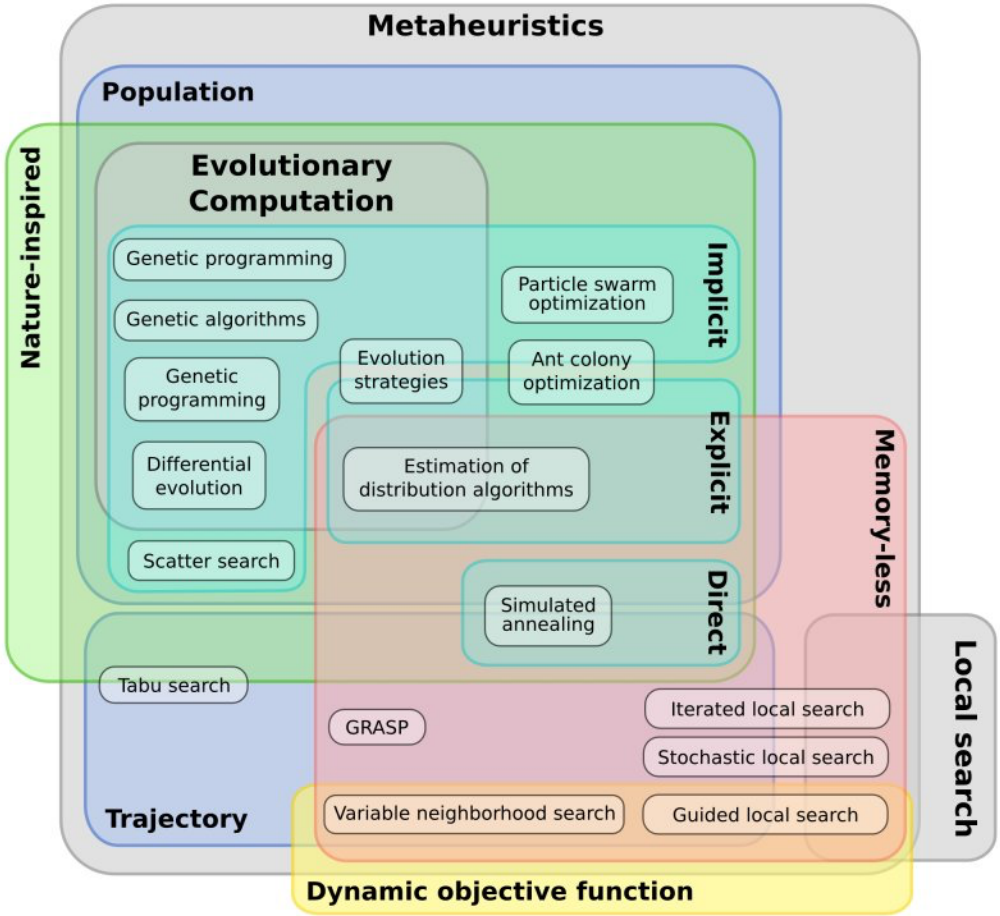
Tema 4. Metaheurísticas basadas en poblaciones (II) Programación genética

Profesor: Sebastián Ventura

KDIS Lab

UNIVERSIDAD D CORDOBA


Metaheuristics
Population
Nature-inspired
Evolutionary Computation
Genetic programming
Genetic algorithms
Genetic programming
Differential evolution
Scatter search
Implicit
Particle swarm optimization
Evolution strategies
Ant colony optimization
Explicit
Estimation of distribution algorithms
Direct
Simulated annealing
Memory-less
Local search
Tabu search
GRASP
Iterated local search
Stochastic local search
Trajectory
Variable neighborhood search
Guided local search
Dynamic objective function

Objetivos

- Conocer las características de la programación genética y sus diferencias con los algoritmos genéticos
- Entender el funcionamiento y estructura general de los algoritmos de programación genética
- Conocer los diferentes paradigmas dentro de la programación genética, sus ventajas e inconvenientes.

Índice

- **Introducción a la Programación Genética**
- **Variantes de la Programación Genética**
  - *Canonical Genetic Programming*
  - *Strongly Typed Genetic Programming*
  - *Gene Expression Programming*
  - *Grammar Guided Genetic Programming*
  - *Grammatical Evolution*
- **Representaciones híbridas en GP: Modelo GA-P**

Introducción a la Programación Genética

Introducción a la Programación Genética

- La Programación Genética (*Genetic Programming*, GP) es una metaheurística poblacional cuya estructura es básicamente igual a la de un AG.
- La característica diferencial de GP es que los individuos representan "entidades ejecutables", que solucionan un determinado problema.
- Estas entidades ejecutables pueden ser:
  - Código fuente para interpretar o compilar
  - Estructuras directamente ejecutables

Introducción a la Programación Genética
Un poco de historia

- La primera referencia sobre GP es del año 1985

  N.L. Cramer. A Representation for the Adaptive Generation of Simple Sequential Programs. *Proceedings of an International Conference on Genetic Algorithms and Their Applications*. Carnegie-Mellon University. Pittsburgh, PA. 1985

- Sin embargo, el autor que más renombre ha tenido en esta área de investigación es John Koza, de manera que la formulación de GP que él propuso se denomina *canonical genetic programming:*

  J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

Introducción a la Programación Genética
Características básicas

- Los algoritmos de GP tienen la estructura básica de otros EAs
- El esquema algorítmico más popular es el de **estado estacionario**, u otros en los que se aplique un elitismo fuerte, dado que los operadores son muy disruptivos

Algoritmo GP
Inicio
  Crear población inicial
  Mientras (no criterio_parada()) **hacer**
    Seleccionar padres
    Crear hijos aplicando operadores
    Actualizar población
  Fin Mientras
Fin Algoritmo GP

Variantes de la Programación Genética

Canonical Genetic Programming

- Esta es la estructura descrita por John Koza en el paper:

- y también en el libro *Genetic Programming*, publicado en 1992.

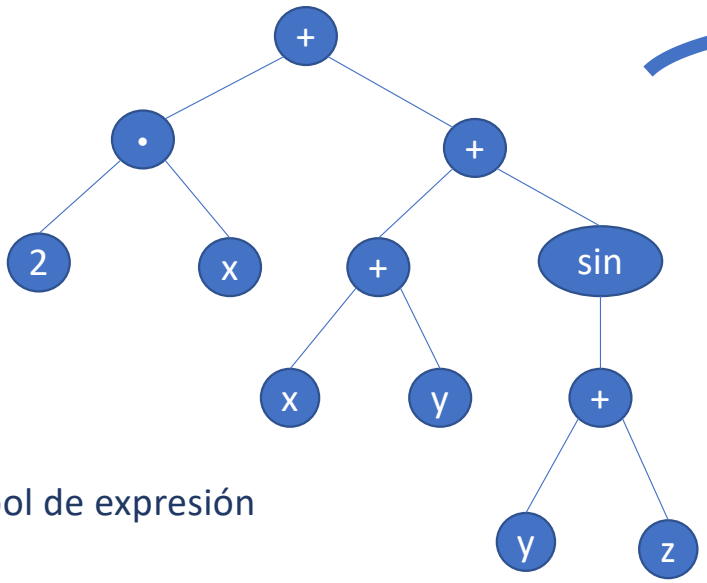
Canonical Genetic Programming

Fundamentos

- **Individuo:**
  - Una o más expresiones expresadas como árboles
- **Nodos del árbol** de expresión:
  - Terminales o nodos hoja (variables, constantes, funciones sin argumentos)
  - Funciones o nodos internos (operadores – principalmente unarios o binarios)
- El árbol hace la doble función de genotipo y fenotipo:
  - Genotipo: experimenta transformaciones genéticas
  - Fenotipo: El recorrido del árbol genera la expresión asociada a éste

Canonical Genetic Programming

Árboles de expresión



Árbol de expresión

(+(• 2 x) (+ (+ x y) (sin (+ y z)))

Expresión en preorden

2x + (x + y) + sin (y+z) = 3x + y + sin (x+z)

Expresión convencional

Canonical Genetic Programming

Evaluación de árboles de expresión

- La evaluación de los árboles consiste en ejecutar el programa representado por el árbol.
- La ejecución se hace de forma distinta según la plataforma de desarrollo:
  - La formulación original de Koza utilizaba expresiones S en el lenguaje LISP.
  - Existen implementaciones en C, C++, Java…
    - Implementaciones recursivas
    - Implementaciones basadas en uso de una pila
    - Traducción y posterior ejecución
    - …
  - La implementación influye en el coste y rendimiento del algoritmo

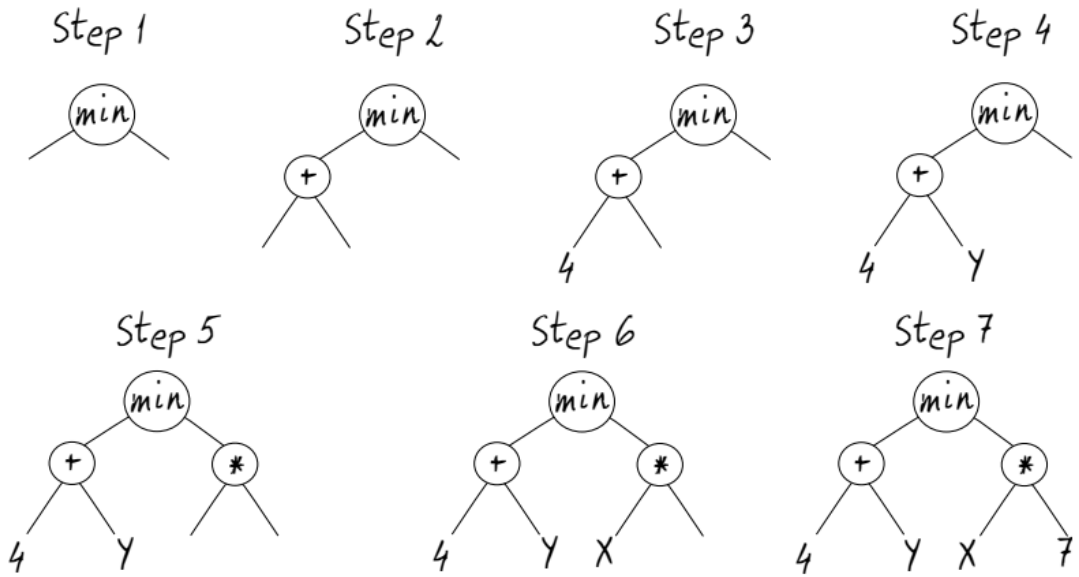
Canonical Genetic Programming

Inicialización de la población

- La creación de la población inicial es un proceso aleatorio
- La profundidad de un árbol se calcula como el número de nodos existentes desde la raíz hasta la hoja más lejana
- Las formas más conocidas de generación son:
  - *Full Method*: genera árboles cuyas hojas están a la misma profundidad. Que los árboles tenga la misma profundidad no quiere decir que tengan la misma forma (sólo ocurre si las funciones tienen la misma aridad).
  - *Grow Method*: genera árboles de cualquier forma con la única restricción de parar al alcanzar la profundidad máxima.
  - El método *Half and Half* construye un 50% de árboles con el método *Full* y el otro 50% con el método *Grow*.
- La estrategia Ramped establece múltiples niveles de profundidad máxima

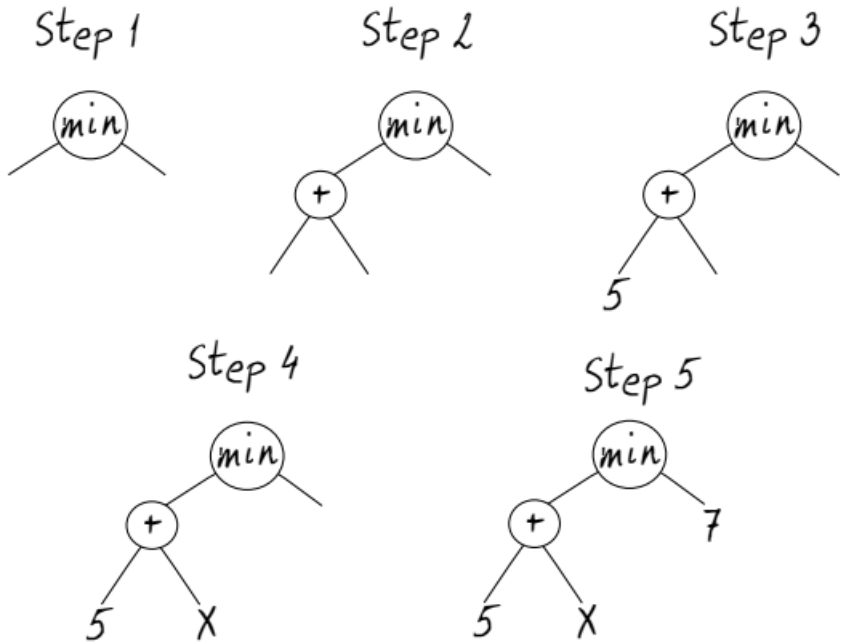
Canonical Genetic Programming

Inicialización de la población: Método Full



Canonical Genetic Programming

Inicialización de la población: Método Grow



Canonical Genetic Programming

Operadores genéticos: Cruce

- **Operador de cruce**: Se selecciona un nodo de un padre de manera aleatoria. Una vez seleccionado el punto de corte en el primer padre, se hace una elección aleatoria dirigida en el segundo padre para verificar restricciones como:
  - Cruzar con un subárbol que evalúe a un tipo de dato compatible
  - Cruzar con un subárbol de tamaño adecuado
- En programación genética, el **cruce es un operador muy disruptivo**
- Los **árboles tienden a crecer indefinidamente** si no se limita el tamaño de la operación de cruce (descendiente que no supere tamaño máximo)

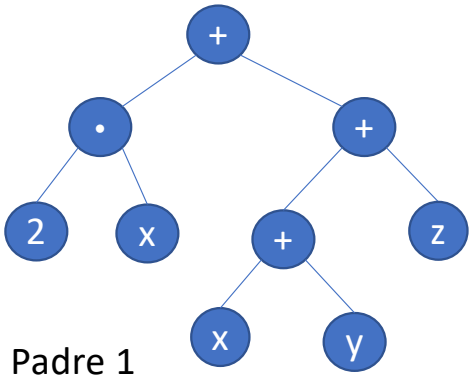
Canonical Genetic Programming

Operadores genéticos: Cruce (II)

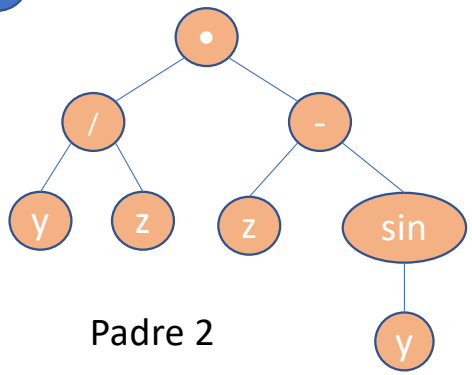
- El **operador de cruce** en programación genética de dos individuos iguales no genera dos descendientes iguales como en los algoritmos genéticos. Esto provoca:
  - Dos árboles muy parecidos que están próximos a converger al óptimo no necesariamente producen descendientes adecuados
  - El algoritmo explora bien el espacio pero no lo explota bien

Canonical Genetic Programming

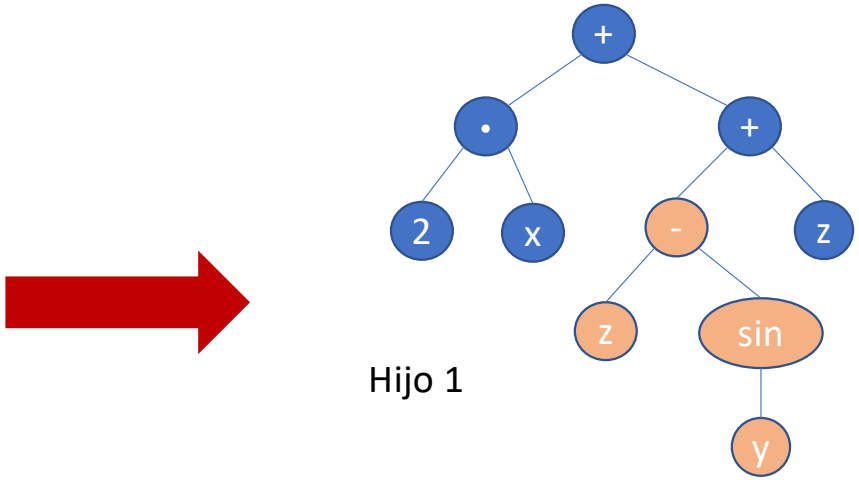
Operadores genéticos: Ejemplo de Cruce



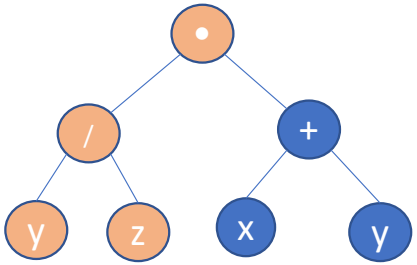
Padre 1



Padre 2



Hijo 1



Hijo 2

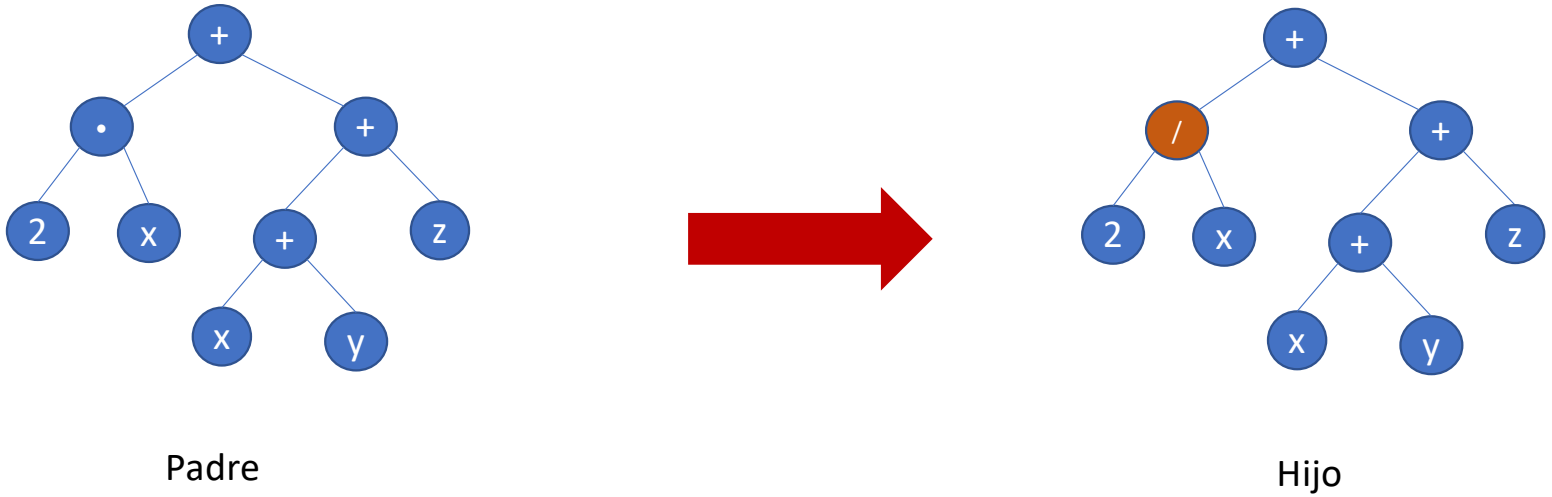
Canonical Genetic Programming

Operadores genéticos: Mutación

- Existen dos grandes tipos de mutación que producen una mayor o menor alteración:
  - **Un punto**: Se escoge un nodo y se cambia su valor por otro del mismo tipo
  - **Subárbol aleatorio**: Se escoge una arista y se sustituye el subárbol conectado a ella por otro generado aleatoriamente
- En programación genética, el **operador de mutación tiene menos importancia que en los algoritmos genéticos**, puesto que el operador de **cruce se basta para mantener la diversidad** (cuando la población es suficientemente grande y tiene suficiente variedad genética)

Canonical Genetic Programming

Operadores genéticos: Ejemplo de Mutación en un Punto

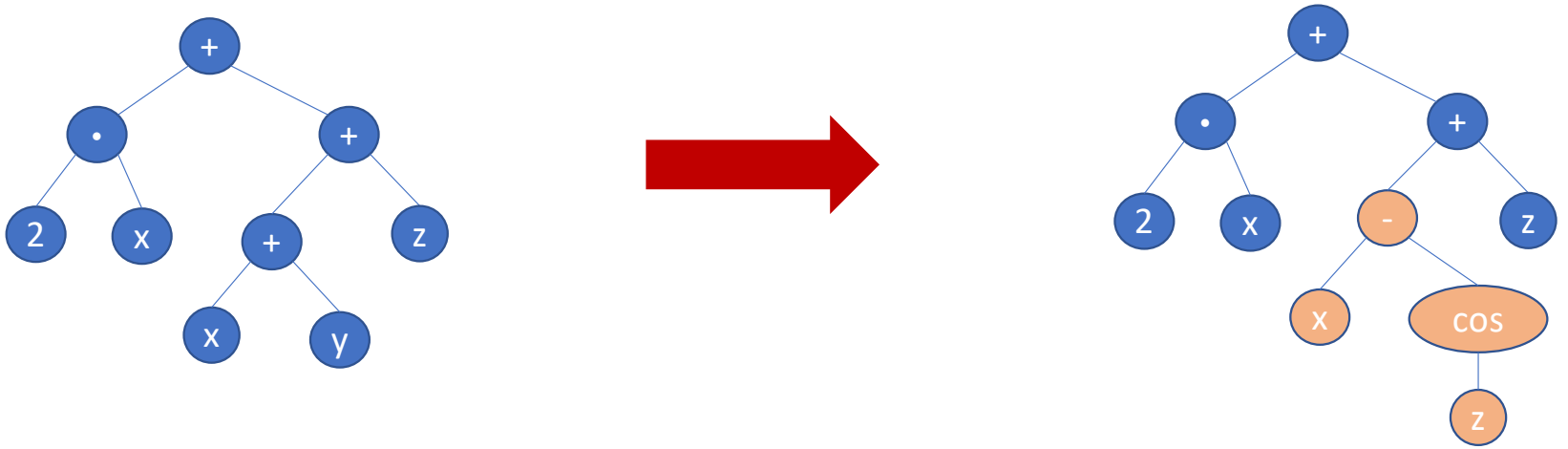


Padre

Hijo

Canonical Genetic Programming

Operadores genéticos: Ejemplo de Mutación de Subárbol



Padre

Hijo

Canonical Genetic Programming

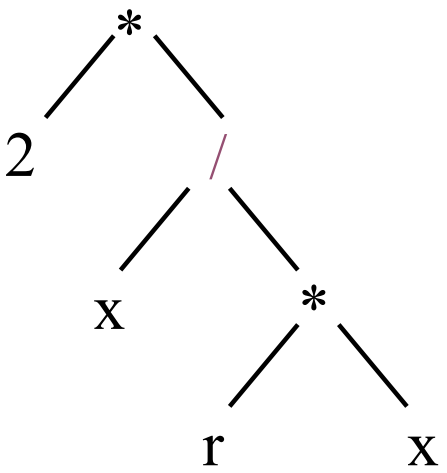
Otros operadores de mutación

- **Permutación**: Se reemplaza un subárbol (lista de argumentos a un nodo función) por una permutación de los mismos
- **Edición**: Simplifica una expresión reemplazándola por otra más sencilla. Si se aplica sobre todos los individuos generados, es muy costosa y no se sabe si mejora la convergencia (puede reducir la diversidad). Sólo se aplica por motivos estéticos de aspecto de las expresiones
- **Encapsulación**: permite dar un nombre a un subárbol, convirtiéndolo en una nueva función que puede intervenir en otros árboles. Es equivalente a restringir los posibles puntos de cruce del árbol

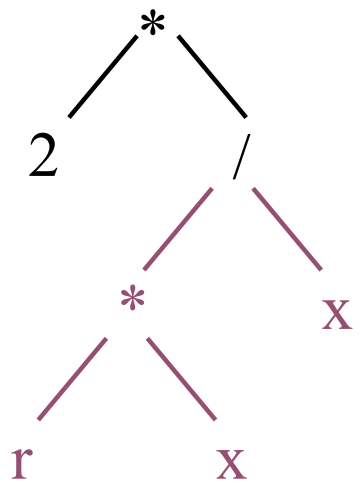
Canonical Genetic Programming

Operador de Permutación

Padre



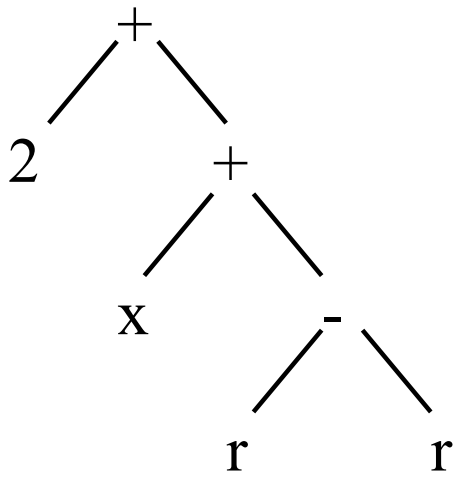
Hijo



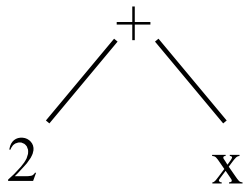
Canonical Genetic Programming

Operador de edición

Padre



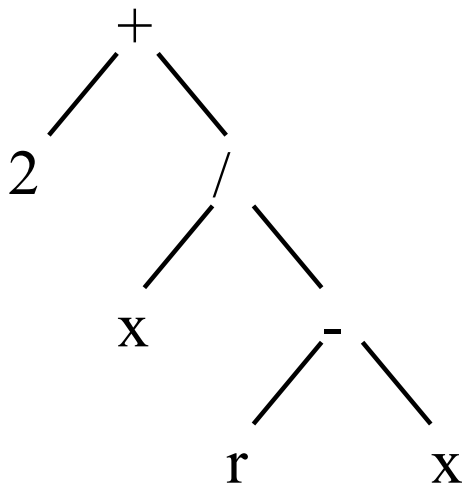
Hijo



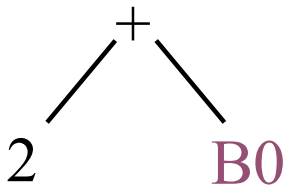
Canonical Genetic Programming

Operador de Encapsulación

Padre



Hijo



Strongly-Typed Genetic Programming

- El problema que tenía la GP canónica es que no permite la combinación de tipos de datos.
- Se pueden establecer simplificaciones (por ejemplo, 1 y 0 para los valores lógicos) pero, en general, se complica la resolución de algunos problemas.
- Tampoco se permite la sobrecarga de operadores
- David Montana (1994) propone STGP para solucionar los problemas anteriores.

Strongly-Typed Genetic Programming

Fundamentos

- Los individuos en STGP son similares a los individuos en CGP, es decir, son árboles de instrucciones
- Los nodos definidos en STGP están **tipados**:
  - Los argumentos que reciben (hijos) han de ser de un determinado tipo.
  - El valor que devuelven es de un tipo determinado (no tiene por qué ser el mismo que el de los argumentos de entrada
    - La función > tiene el prototipo (numérico, numérico):boolean
  - Los nodos función admiten sobrecarga, es decir, un mismo nodo puede devolver un resultado distinto según los nodos sean de un tipo o de otro:
    - Ejemplo, la función + podría tener los prototipos
      - (entero, entero): entero
      - (real, real): real
      - (vector de reales, vector de reales) : vector de reales

Strongly-Typed Genetic Programming

Fundamentos

- La clave de STGP está en la tabla de compatibilidades que se establece al definir los nodos función y los nodos terminal.
- Esta tabla contiene, para cada nodo:
  - Número de argumentos
  - Tipo de dato para cada uno de los argumentos
  - Tipo devuelto
- Si un nodo está sobrecargado, tendrá varias entradas como esta
- A partir de esa tabla, se puede construir una segunda en la que se incluyen los nodos que pueden actuar como hijos de cada uno de los nodos existentes. Esto es lo que permite manipular los árboles para que siempre resulten correctos.

Strongly-Typed Genetic Programming

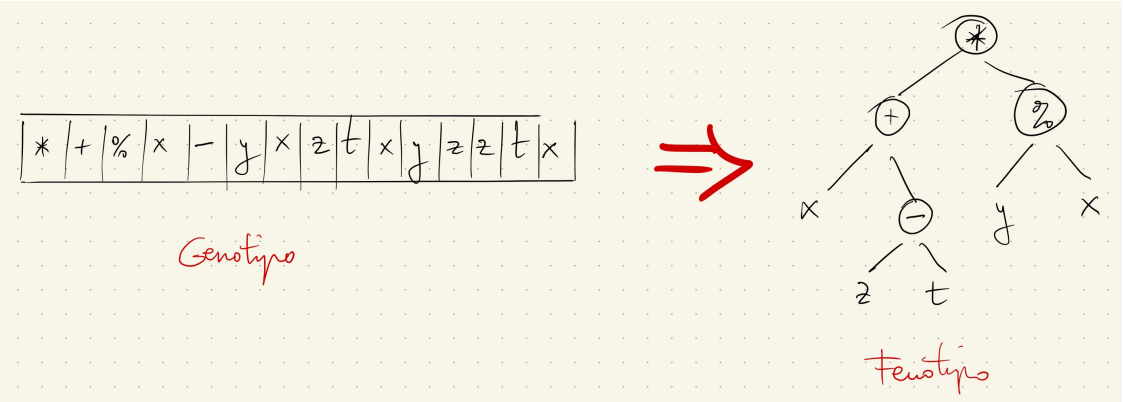
Creación de árboles y operadores genéticos

- El proceso de creación de árboles en STGP es muy parecido al de CGP. La única diferencia es que, los nodos se eligen al azar de entre los que son considerados como compatibles de un nodo dado.
- En el caso de operadores genéticos (cruce y mutación) estamos en la misma situación:
  - Para el cruce, se eligen ramas cuyos tipos devueltos sean iguales o compatibles con el lugar de destino.
  - Para la mutación, se cambian unas funciones por otras con el mismo prototipo y, cuando hay que generar un subárbol se hace usando la tabla de compatibilidad de funciones.

Gene Expression Programming (GEP)

Introducción

- Paradigma propuesto por Candida Ferreira (2002)
- Presenta una doble codificación:
  - Genotipo lineal (parecido a los individuos de AG con codificación entera)
  - Fenotipo árbol de instrucciones (ejecutable y evaluable)



Gene Expression Programming (GEP)

Codificación

- Los cromosomas en GEP presentan dos zonas
  - Cabeza (*head*) – Desde el extremo izquierdo hasta la posición h
  - Cola (*tail*) – Desde la posición h+1 hata la posición h+t
- Los elementos de la cabeza pueden ser funciones o terminales, pero los de la cola sólo pueden ser terminales.
- Los valores *h* y *t* se denominan *tamaño de cabeza* y *tamaño de cola*, y se definen para garantizar que la traducción del genotipo al fenotipo producirá siempre un árbol de instrucciones válido.

$$t = h\,(n_{max} - 1) + 1$$

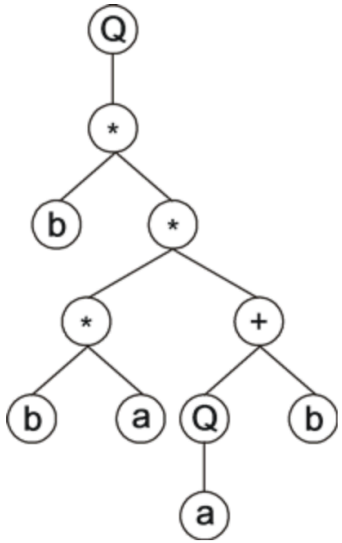
Gene Expression Programming (GEP)

Decodificación

Para pasar del genotipo al fenotipo se va recorriendo el cromosoma lineal, y con su contenido se van rellenando los nodos del árbol por capas: primero la raíz, segundo la capa bajo la raíz y así hasta completar el árbol

Q*b**+baQbaabbbaabb

cabeza cola



Gene Expression Programming (GEP)

Operadores genéticos

- Existen múltiples operadores genéticos en GEP:
  - Replicación
  - Recombinación
  - Mutación
  - Transposición
  - Inversión
- Todos los operadores se aplican con una determinada probabilidad durante la fase de reproducción.

Grammar-Guided Genetic Programming

- Los primeros modelos de GP (canónica o fuertemente tipada) presentan un algunos problemas de difícil solución:
  - Fenómeno del hinchamiento (bloat). En general, los individuos en GP tienen a aumentar de tamaño progresivamente, introduciendo lo que se denominan “intrones” (trozos de código que no realizan ninguna acción productiva).
  - El cruce es muy disruptivo

Grammatical Evolution (GE)

Representaciones híbridas en GP: Modelo GA-P