BIOINFORMÁTICA
2013 - 2014

PARTE I. INTRODUCCIÓN

- Tema 1. Computación Basada en Modelos Naturales

PARTE II. MODELOS BASADOS EN ADAPTACIÓN SOCIAL (Swarm Intelligence)

- Tema 2. Introducción a los Modelos Basados en Adaptación Social
- Tema 3. Optimización Basada en Colonias de Hormigas
- Tema 4. Optimización Basada en Nubes de Partículas (Particle Swarm)

PARTE III. COMPUTACÍON EVOLUTIVA

- Tema 5. Introducción a la Computación Evolutiva
- Tema 6. Algoritmos Genéticos I. Conceptos Básicos
- Tema 7. Algoritmos Genéticos II. Diversidad y Convergencia
- Tema 8. Algoritmos Genéticos III. Múltiples Soluciones en Problemas Multimodales
- Tema 9. Estrategias de Evolución y Programación Evolutiva
- Tema 10. Algoritmos Basados en Evolución Diferencial (Diferential Evolution – DE)
- Tema 11. Modelos de Evolución Basados en Estimación de Distribuciones (EDA)
- Tema 12. Algoritmos Evolutivos para Problemas Multiobjetivo
- Tema 13. Programación Genética
- Tema 14. Modelos Evolutivos de Aprendizaje

PARTE IV. OTROS MODELOS DE COMPUTACIÓN BIOINSPIRADOS

- Tema 15. Sistemas Inmunológicos Artificiales
- Tema 16. Otros Modelos de Computación Natural/Bioinspirados

BIOINFORMÁTICA

TEMA 13: PROGRAMACIÓN GENÉTICA

1. INTRODUCCIÓN
2. FUNDAMENTOS DE LA PROGRAMACIÓN GENÉTICA
3. EJEMPLO DE APLICACIÓN: REGRESIÓN SIMBÓLICA
4. ALGORITMOS GA-P
5. APLICACIONES
6. DISCUSIÓN FINAL

Bibliografía

J.R. Koza, Genetic Programming. MIT Press, 1992.

W. Banzhaf, P. Nordin, R. Keller, F.D. Francone, Genetic Programming. An Introduction. Morgan Kaufmann, 1998.

W. B. Langdon, R. Poli. Foundations of Genetic Programming. Springer-Verlag, 2002.

1. INTRODUCCIÓN: Diferencias entre Programación Genética y Algoritmos Genéticos

Podemos definir la Programación Genética (PG) como un caso particular de Algoritmos Genéticos (AGs) usados para inducir programas de ordenador de un modo automático.

La disciplina data de 1985, aunque el gran difusor ha sido John Koza a principios de la década de los 90.

La diferencia principal entre AGs y PG es la representación de las soluciones. Los cromosomas codifican, de algún modo, programas de ordenador.

Naturalmente, los operadores genéticos han de adaptarse a la representación empleada.

1. INTRODUCCIÓN: Rápido Resumen

- Desarrollo: USA, en los años 90
- Primeros autores: J. Koza
- Aplicación típica:
  - Regresión simbólica
- Características atribuidas:
  - Cromosomas no lineales: árboles, grafos
  - Cruce muy destructivo, que genera mucha diversidad
  - Mutación posible pero no necesaria (a discutir en las aplicaciones)

Genetic Programming

Representation	Tree structures
Recombination	Exchange of subtrees
Mutation	Random change in trees
Parent selection	Fitness proportional
Survivor selection	Generational replacement

Genetic Programming

Introductory example: credit scoring

- **Bank wants to distinguish good from bad loan applicants**
- **Model needed that matches historical data**

ID	No of children	Salary	Marital status	OK?
ID-1	2	45000	Married	0
ID-2	0	30000	Single	1
ID-3	1	40000	Divorced	1
…				

Genetic Programming

Introductory example: credit scoring

- **A possible model:**

  IF (NOC <= 2) AND (S > 40000) THEN good ELSE bad

- **In general:**

  IF formula THEN good ELSE bad

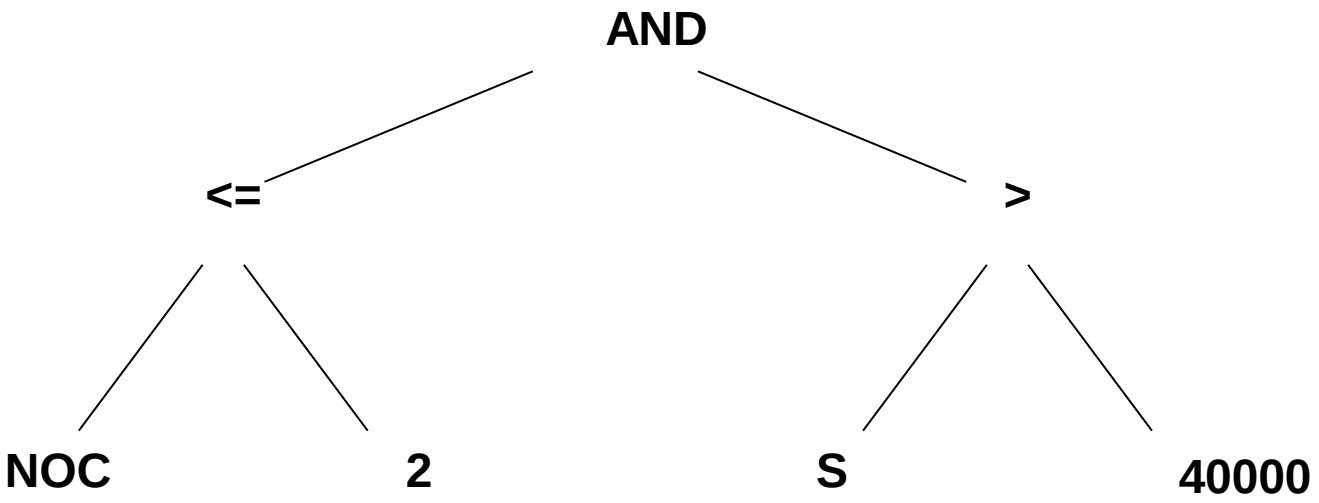
- **Only unknown is the right formula, hence**
- **Our search space (phenotypes) is the set of formulas**
- **Natural fitness of a formula: percentage of well classified cases of the model it stands for**
- **Natural representation of formulas (genotypes) is: parse trees**

Genetic Programming

Introductory example: credit scoring

IF (NOC = 2) AND (S > 40000) THEN good ELSE bad

can be represented by the following tree



2. FUNDAMENTOS DE LA PROGRAMACIÓN GENÉTICA

- REPRESENTACIÓN DE PROGRAMAS
- LA REPRESENTACIÓN DE ÁRBOL Y EL MANEJO DE GRAMÁTICAS. 3 EJEMPLOS
- LA FUNCIÓN DE ADAPTACIÓN
- GENERACIÓN DE LA POBLACIÓN INICIAL
- OPERADORES EN PROGRAMACIÓN GENÉTICA

Representación de Programas

Existen diversas formas de representar programas:

- **Árboles de expresiones.**
- **Secuencias de instrucciones de máquinas de estados finitos.**
- **Secuencias de instrucciones en código máquina.**

La representación más usada en PG es la de árbol de expresiones.

Habitualmente se trabaja con lenguajes muy sencillos, que no emplean bucles.

La Representación de Árbol y el Manejo de Gramáticas

- La representación de programas en forma de árboles de expresión se basa en la existencia de una **gramática libre de contexto** que define las sentencias válidas del lenguaje.

- Sin embargo, al trabajar directamente con los árboles y las reglas necesarias para evaluarlos, no es necesario hacer referencia al lenguaje libre de contexto.

- **El cromosoma como tal codifica la expresión del árbol utilizando cualquier recorrido sobre él, habitualmente el preorden.**

La Representación de Árbol y el Manejo de Gramáticas: Principales características

- AG: Representación lineal de estructuras (cadenas de bits, enteros, valores reales o permutacions)
- Los cromosomas en forma de árboles son estructuras no lineales.
- Los AG tienen cromosomas de tamaño fijo
- Los árboles en PG pueden variar en profundidad y anchura

La Representación de Árbol y el Manejo de Gramáticas (2)

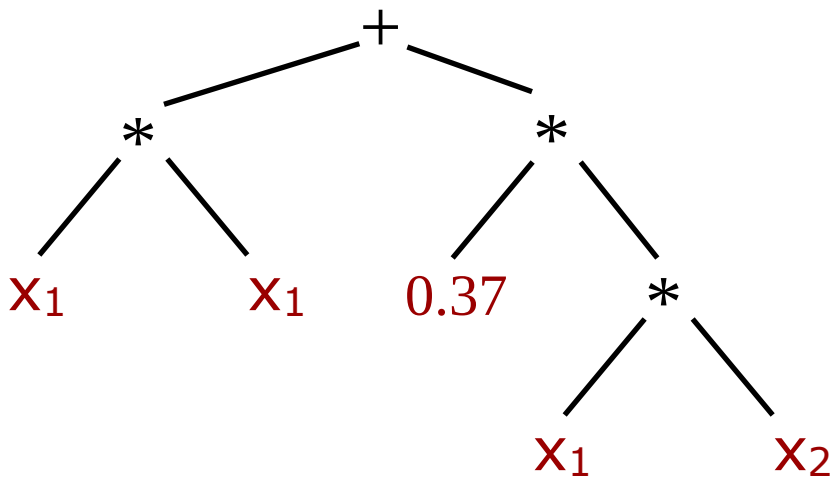
EJEMPLO

GRAMÁTICA:

$S \rightarrow x_1 \mid x_2 \mid N \mid + S\ S \mid * S\ S$

$N \rightarrow \Re$

ÁRBOL VÁLIDO:



Cromosoma: + * x_1 x_1 * 0.37 * x_1 x_2

Expresión: $x_1^2 + 0.37 * x_1 * x_2$

La Representación de Árbol y el Manejo de Gramáticas (3)

Para emplear esta representación, es necesario definir el conjunto de símbolos terminales de la gramática:

- **Variables y constantes del problema: {x_1, x_2, $\Re$, ...}**
- **Funciones/Instrucciones del programa: {+, -, *, /, seno, cos, and, if-then-else, ...}**

- El conjunto de símbolos y funciones debe ser suficiente para resolver el problema.
- Cada función debe aceptar como argumento cualquier terminal y cualquier valor del tipo de datos devuelto por otra función:

$$x / y = \begin{cases} x / y, & y \neq 0 \\ 0, & \text{en otro caso} \end{cases} \qquad \sqrt{x} = \sqrt{|x|}$$

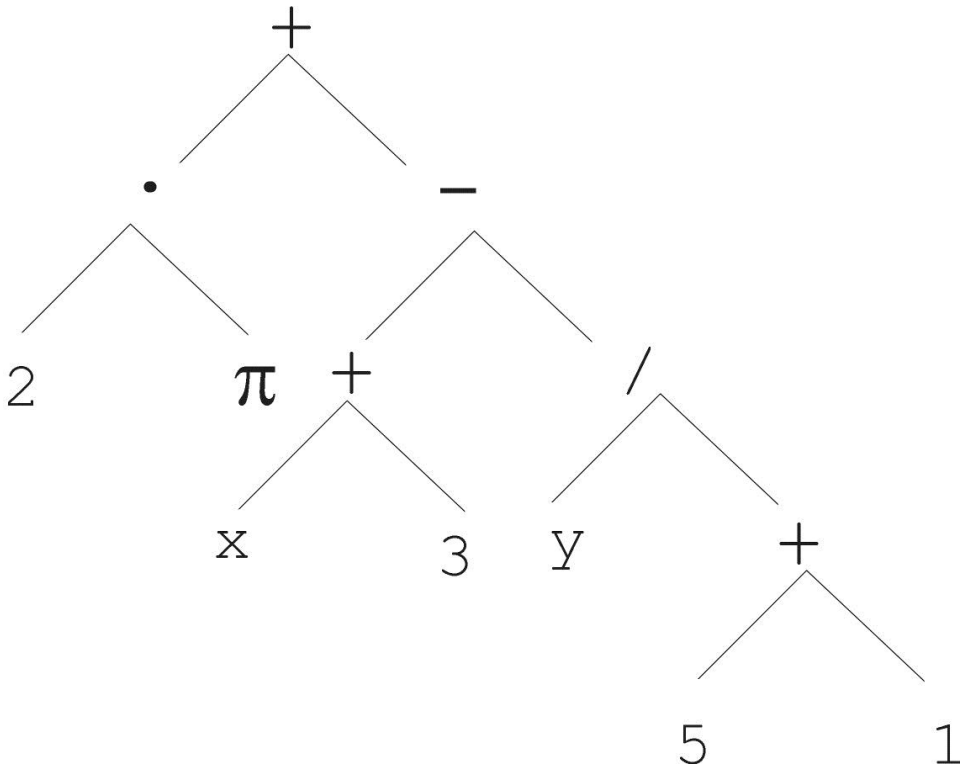
La Representación de Árbol: 3 ejemplos

Los árboles de expresión son una forma universal de representación que nos permite codificar distintos objetos como:

- Formulas aritméticas: $2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$
- Formulas lógicas: $(\mathrm{x} \wedge \mathrm{true}) \rightarrow ((\ \mathrm{x} \vee \mathrm{y}\ ) \vee (\mathrm{z} \leftrightarrow (\mathrm{x} \wedge \mathrm{y})))$
- Programas:

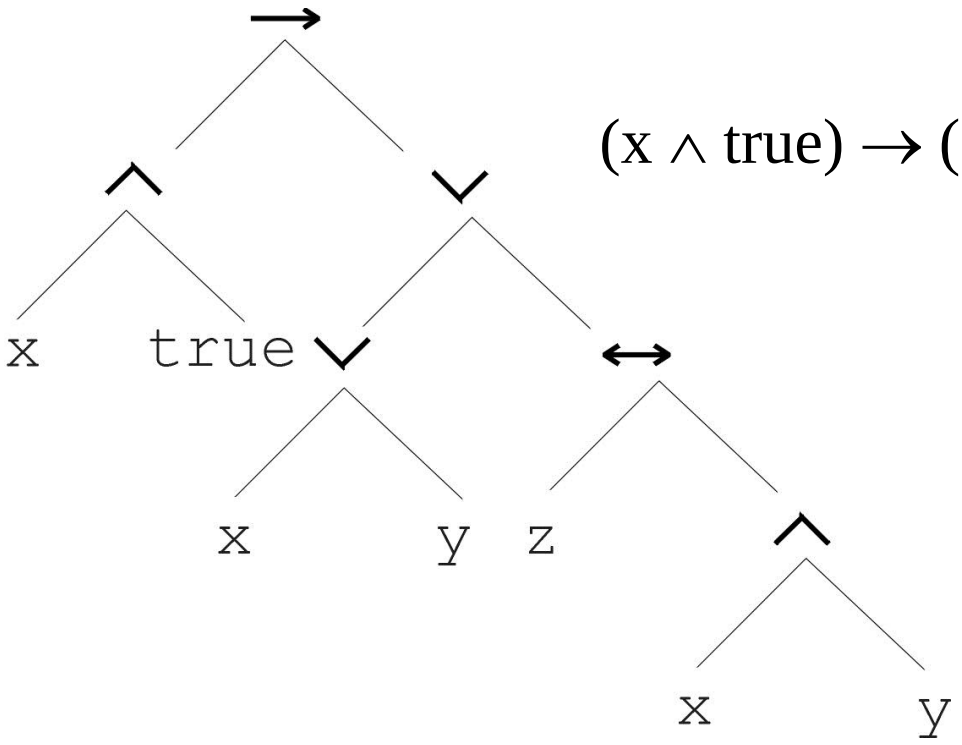
```
i =1;
while (i < 20)
{
        i = i +1
}
```

La Representación de Árbol: 3 ejemplos (2)



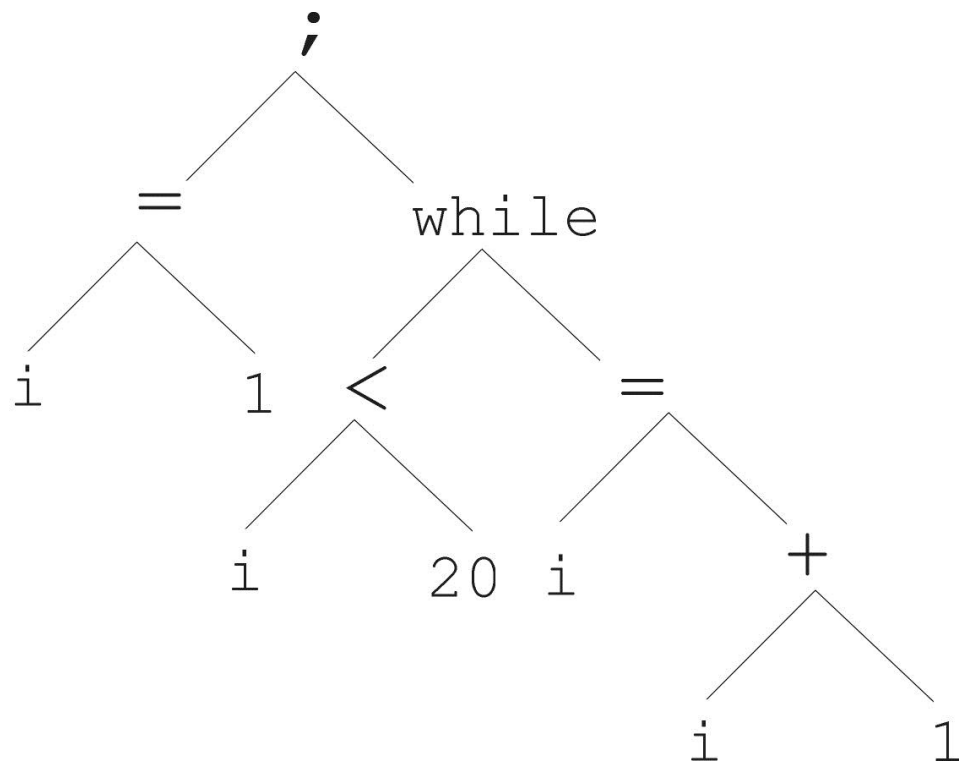
$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

La Representación de Árbol: 3 ejemplos (3)



$$(x \wedge \text{true}) \rightarrow ((\ x \vee y\ ) \vee (z \leftrightarrow (x \wedge y)))$$

La Representación de Árbol: 3 ejemplos (4)




```
i =1;
while (i < 20)
{
        i = i +1
}
```

La Función de Adaptación

En PG, la función de adaptación debe especificar una medida de la calidad del programa codificado en la resolución del problema planteado.

En principio, para poder evaluar los árboles de expresiones, es necesario una evaluación recursiva, consistente en dar valor a los atributos de las hojas e ir subiendo en el árbol hasta devolver el valor asociado a la raíz.

Sin embargo, se puede implementar de forma iterativa haciendo uso de una o más pilas para ir operando con la representación en preorden del árbol.

Generación de la Población Inicial

Se basa en una generación aleatoria de distintos programas (árboles de expresiones) que puede ser dirigida o no dirigida:

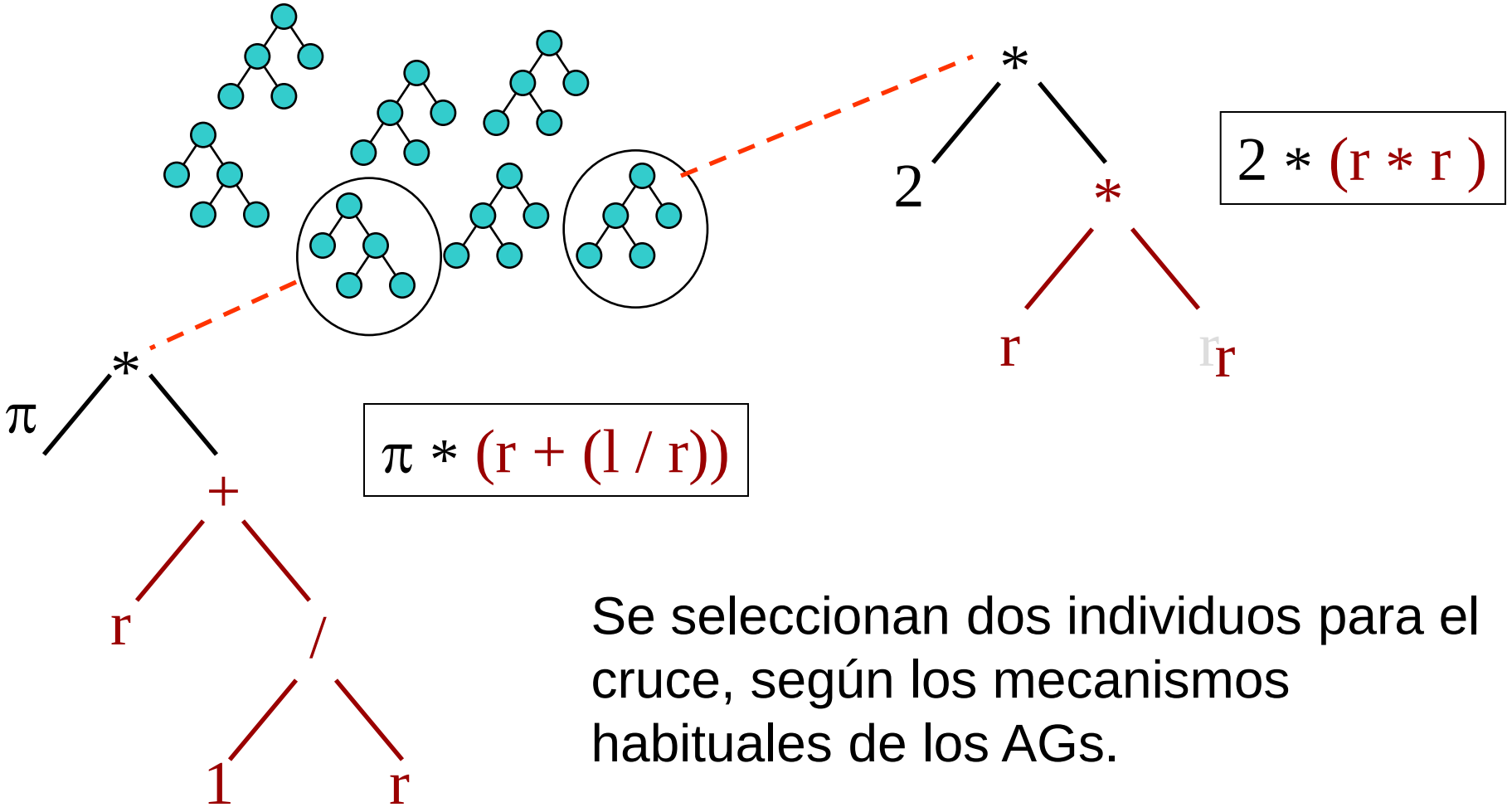
- El tamaño de los árboles ha de ser menor que un tope máximo permitido.
- Para introducir diversidad, se puede:
  - Generar un porcentaje de árboles de cada tamaño válido.
  - Generar programas puramente aleatorios o que hagan uso de todas las entradas.
- Para generar un árbol aleatorio se aplican progresiva y aleatoriamente las reglas de producción la gramática, de un modo directo o implícito.

Operadores en Programación Genética

Existen dos bloques de operadores:

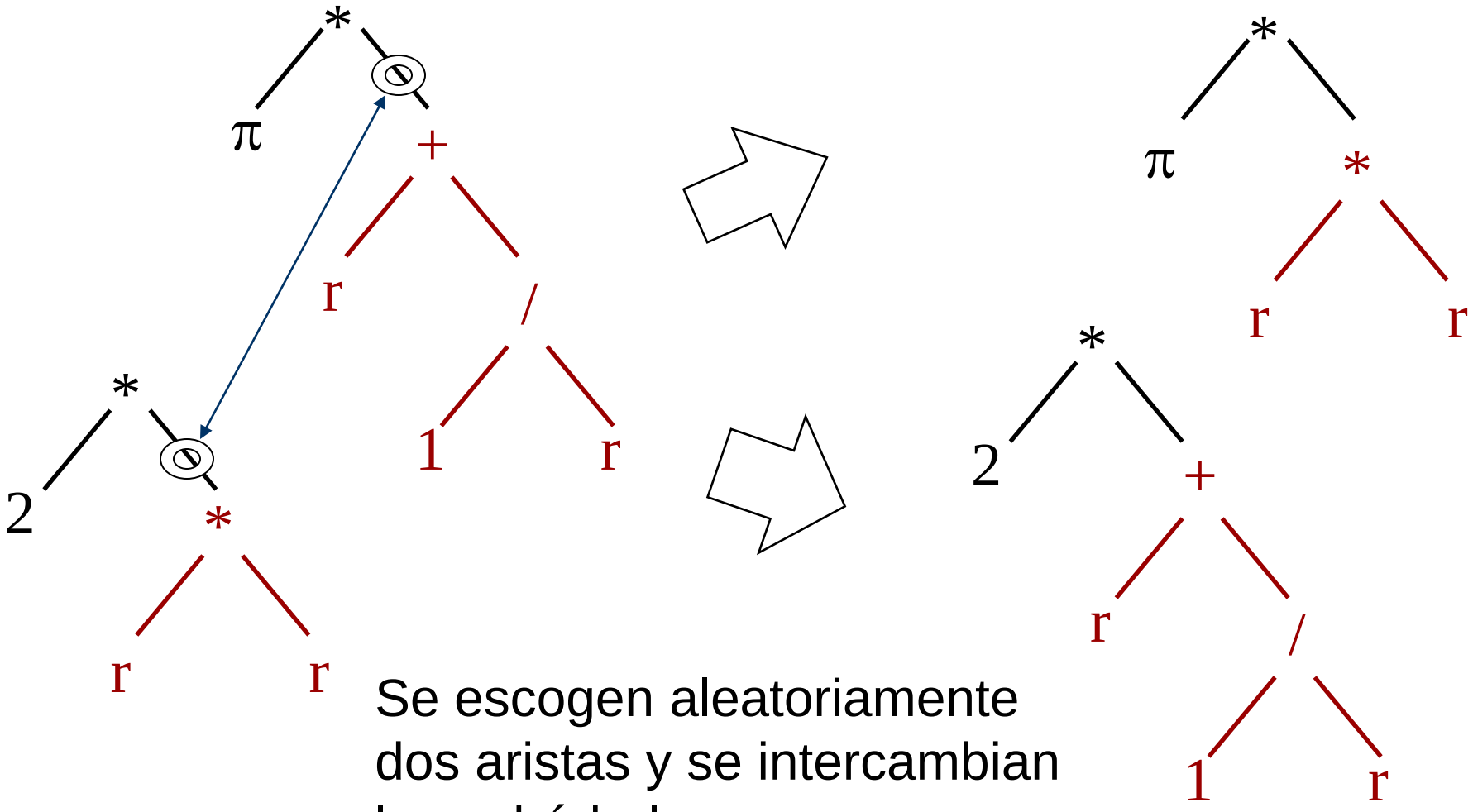
- Operadores primarios:
  - Reproducción
  - Cruce
  - Mutación
- **Operadores secundarios:**
  - **Permutación**
  - **Edición**
  - **Encapsulación**
  - **Decimación**

Reproducción



Se seleccionan dos individuos para el cruce, según los mecanismos habituales de los AGs.

Cruce de Árboles de Expresiones



Se escogen aleatoriamente dos aristas y se intercambian los subárboles.

Consideraciones Acerca del Cruce PG:
Selección de los Puntos de Corte

- El algoritmo es más eficiente si se sesga la probabilidad de selección de los arcos conectados a nodos terminales para hacerla menor que la de los conectados a nodos internos.
- Un valor adecuado es que la probabilidad de selección de los arcos de nodos terminales sea menor que 1/nº_arcos_árbol.
- Así, se consiguen descendientes más diferentes de sus padres.
- Una vez escogido aleatoriamente el punto de corte en el primer árbol, se hace una elección aleatoria dirigida en en el segundo para verificar restricciones sintácticas como:
  - Cruzar con un subárbol que evalúe a un tipo de dato compatible.
  - Cruzar con un subárbol de tamaño adecuado.

Consideraciones Acerca del Cruce PG (2):
Características del Operador

- El cruce de PG es un operador muy disruptivo: los árboles tienden a crecer indefinidamente si no se limita el tamaño en la operación de cruce.
- Lo habitual es no efectuar el cruce si el subarbol escogido en el segundo árbol provoca que el descendiente supere el tamaño máximo o el tamaño deseado.
- Otra posibilidad consiste en incorporar criterios que penalicen una excesiva complejidad de los programas a la función de adaptación.

Consideraciones Acerca del Cruce PG (3):
Aspectos de Implementación

- La dificultad de implementación del operador radica en determinar correctamente los subárboles a intercambiar sobre las expresiones en preorden de los árboles.

- Esta operación se puede implementar de dos formas:

  - Aplicando las producciones de la gramática para obtener el árbol codificado, deteniéndose aleatoriamente en una y considerando el subárbol que resta por generar para el intercambio.

  - Escogiendo una posición aleatoriamente y determinando los puntos de inicio y final del subárbol que cuelga de ella mediante un recorrido directo sobre la expresión en preorden.

Consideraciones Acerca del Cruce PG (4):
Diferencias entre el Cruce AG y el Cruce PG

- El cruce de PG de dos individuos iguales no genera dos descendientes iguales como el de AG.
- Esto provoca que el cruce de dos árboles muy parecidos con buen fitness (cuando el algoritmo está próximo a converger) no necesariamente produzca descendientes adecuados.
- Así, el algoritmo explora bien el espacio pero no lo explota adecuadamente.
- Como consecuencia, en PG es necesario considerar tamaños de población mucho mayores que en los AGs.

Mutación de Árboles de Expresiones

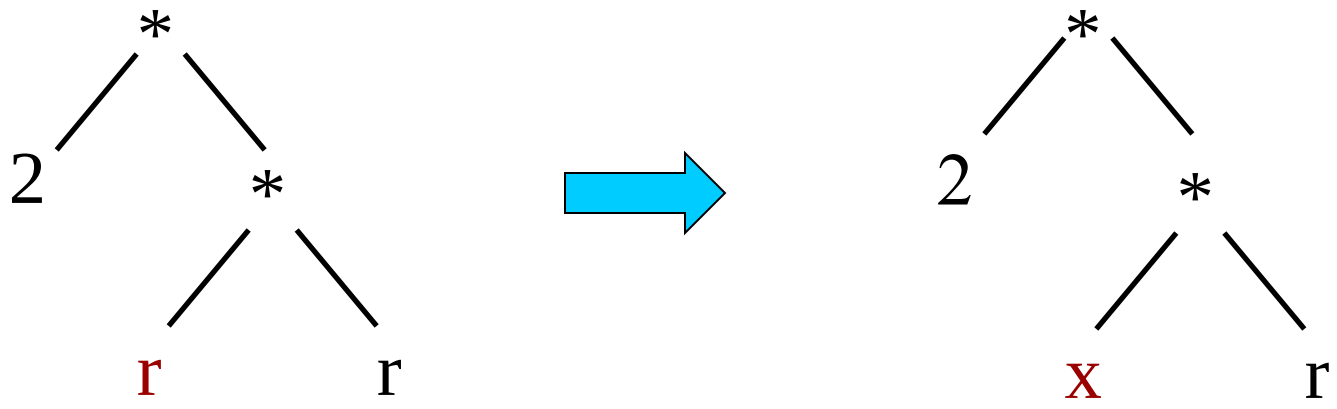
Existen distintas posibilidades, que producen una mayor o menor alteración en el descendiente:

- Mutación en un punto: Se escoge un nodo y se cambia su valor por otro del mismo tipo.
- Mutación por subárbol aleatorio: Se escoge una arista y se substituye el subárbol conectado a ella por otro generado aleatoriamente.

Tiene menor importancia que en AGs puesto que el operador de cruce PG se basta para mantener la diversidad.

Mutación en un Punto

Se define una lista de terminales que pueden intercambiarse entre sí y se reemplaza un nodo escogido al azar por otro de la lista: variable por variable, función por función, constante por constante, etc.

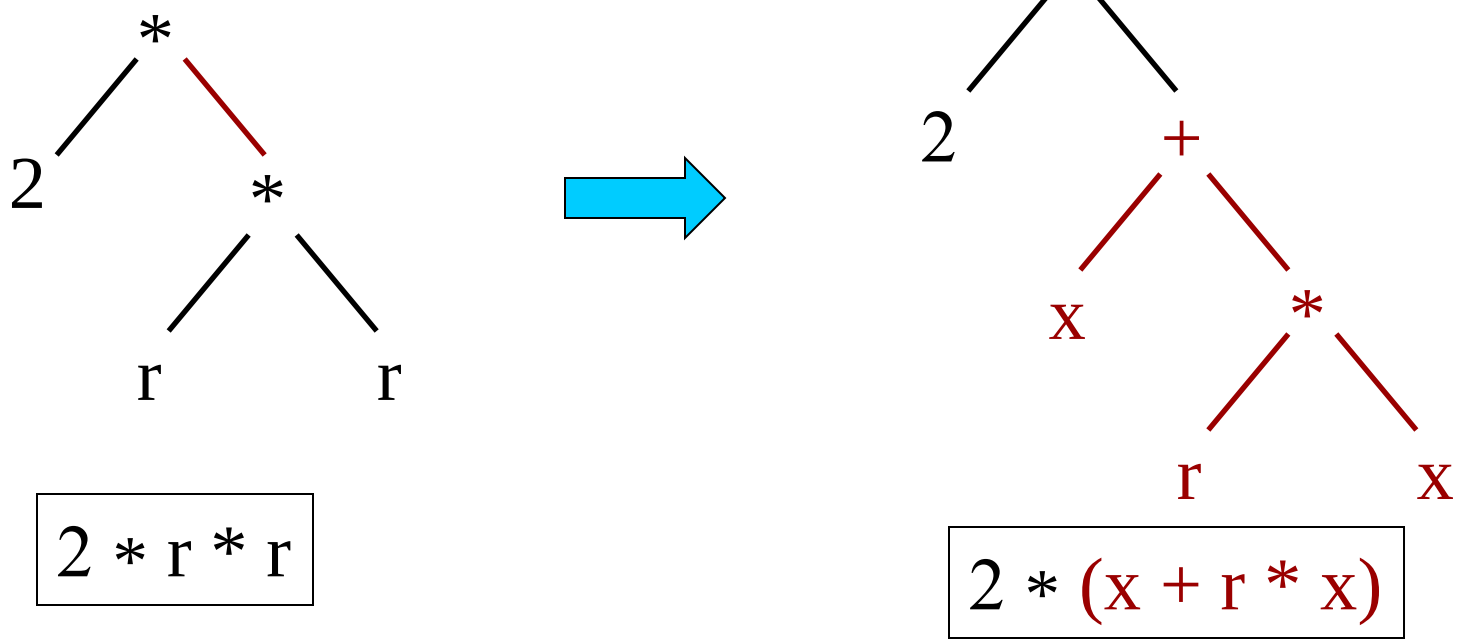


2 * r * r

2 * x * r

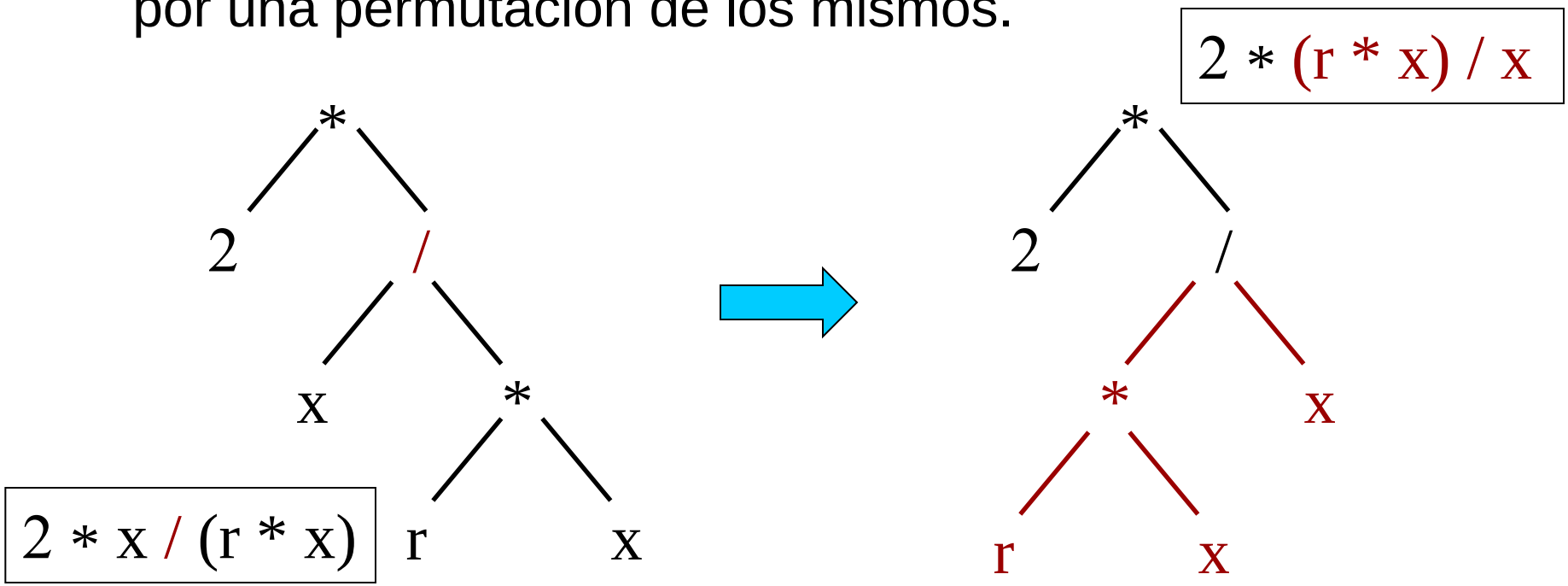
Mutación por Subárbol Aleatorio

Se escoge una arista aleatoria y se reemplaza el subárbol que cuelga de ella por otro generado aleatoriamente, sin superar el tamaño máximo.



Permutación

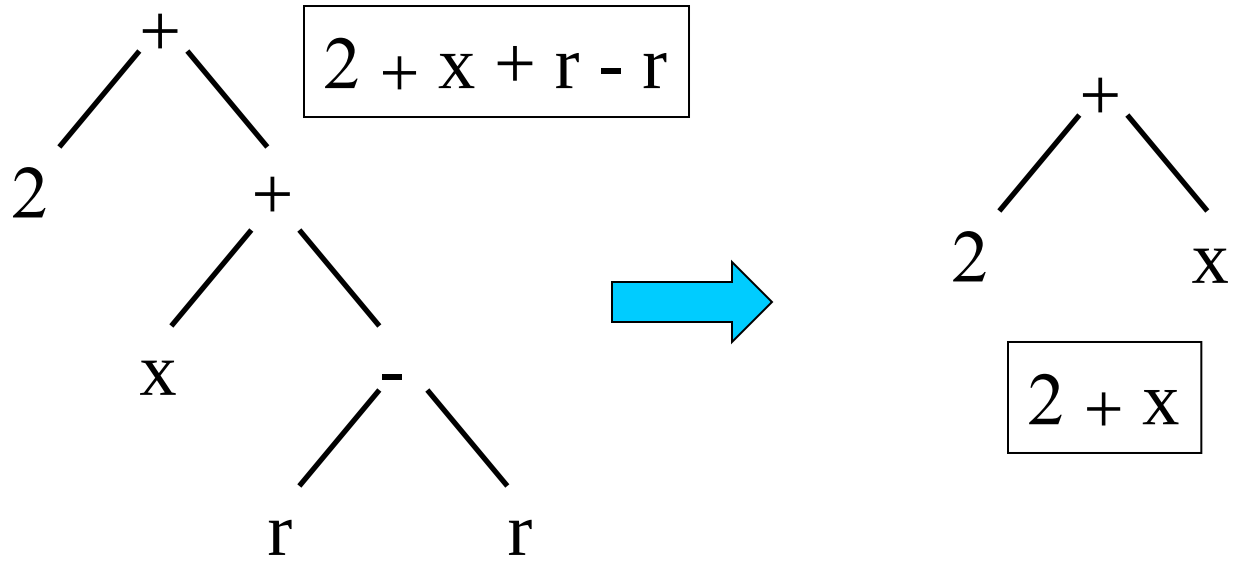
Se reemplaza una lista de argumentos a una función por una permutación de los mismos.



Equivale al operador de inversión de los AGs.

Edición

Simplifica una expresión, reemplazándola por otra más sencilla.

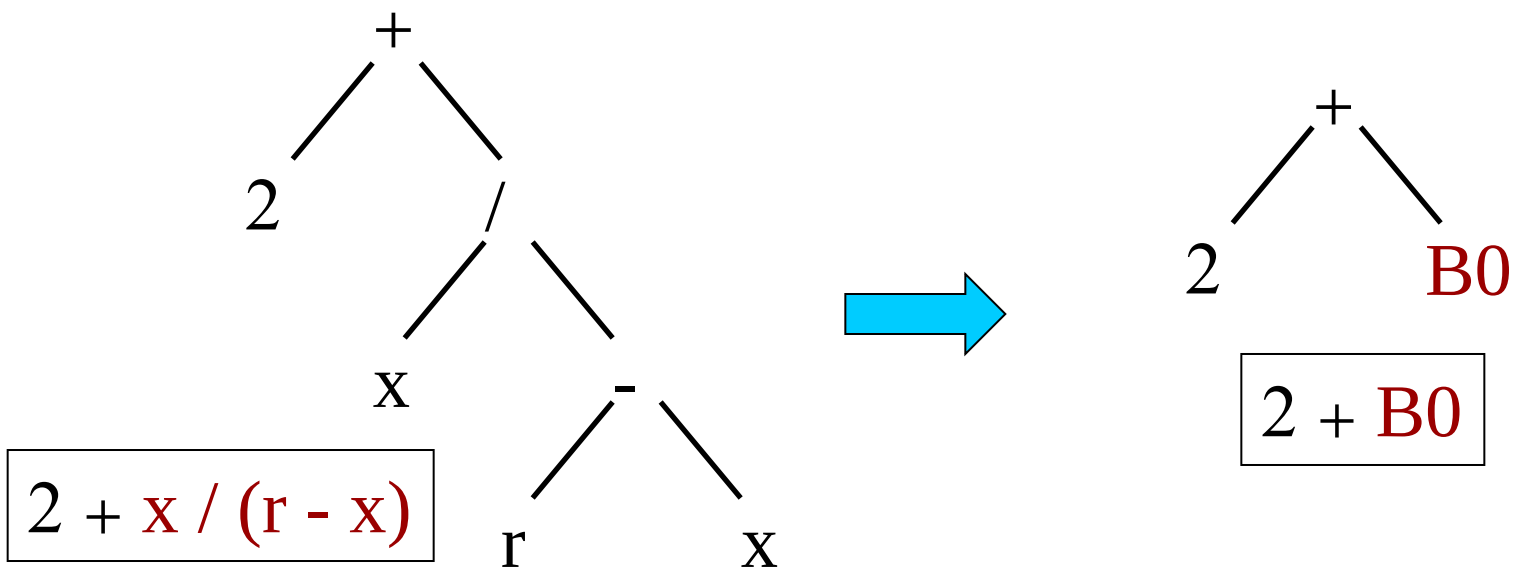


Si se aplica sobre todos los individuos generados, es muy costosa y no se sabe si mejora la convergencia (puede reducir la diversidad).

Sólo se aplica por motivos estéticos de aspecto de las expresiones.

Encapsulación

Consiste en dar un nombre a un subárbol, convirtiéndolo en una nueva función que puede intervenir en otros árboles:



Es ventajosa para estructurar los resultados. Equivale a una restricción en los posibles puntos de cruce del árbol.

Decimación

Eliminación de un porcentaje de individuos de la población en función de su fitness, para evitar tener que evaluar soluciones que se desecharán en generaciones posteriores.

En cierto modo, es una aproximación a las poblaciones de tamaño variable de los AGs.

3. EJEMPLO DE APLICACIÓN: REGRESIÓN SIMBÓLICA

La principal aplicación de la PG es la Regresión Simbólica.

Las técnicas de regresión permiten obtener expresiones matemáticas *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 esté fijada a priori (regresión lineal, cuadrática, exponencial, logarítmica, etc.).

Únicamente determina los coeficientes numéricos de dicha expresión usando una técnica de optimización numérica.

Regresión Simbólica (2)

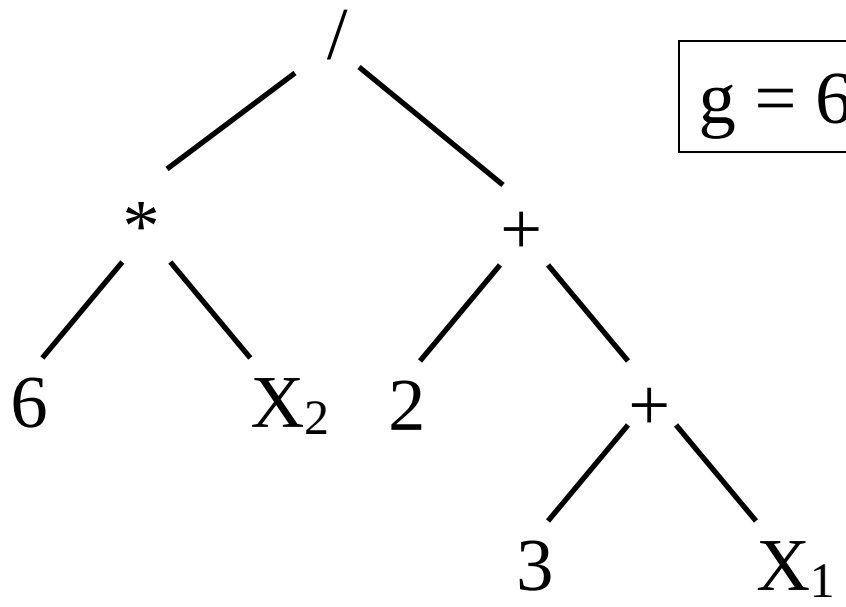
En cambio, la regresión simbólica permite obtener tanto la estructura de la expresión como los valores de los coeficientes de un modo automático.

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

Pueden aplicarse técnicas de inducción de programas para determinar la expresión algebraica g.

Basta con codificar las expresiones de un modo adecuado (por ejemplo, como árboles) y aplicar un algoritmo de PG.

Regresión Simbólica mediante PG



$$g = 6*X_2 / (2 + 3*X_1)$$

- Símbolos terminales: {X_i, constantes}
- Funciones: { +, -, *, /, $\sqrt{\ }$, ...}

Regresión Simbólica mediante PG (2)

Como hemos visto, las expresiones de la función *g* son más sencillas si se permite el uso de constantes reales.

El valor de estas constantes se genera aleatoriamente al generar la población inicial de programas.

El problema es que su valor sólo puede alterse vía mutación. Así, es difícil que tomen el valor deseado y el algoritmo de PG opera con ellas hasta aproximar el resultado:

$$g = (0.5/0.4)*X_1 + X_2*X_2/0.4 - X_2*X_2*0.4*0.4$$

$$g = 1.25*X_1 + 2.34*X_2^2$$

Ejemplo de Funcionamiento

PROBLEMA: Regresión Simbólica sobre 10 puntos de la función:

$$y = \frac{x^2}{2}$$

Parámetros	Valores
Objetivo:	Hacer evolucionar una función que aproxime 10 puntos de una parábola.
Representación:	Arbol Sintáctico.
Gramática del lenguaje:	S → número S → x S → + S S S → * S S
Valores de los terminales "número":	Reales entre -1 y +1
Tamaño de la población:	100
Probabilidad de cruce:	95 por ciento
Probabilidad de mutación:	5 por ciento
Selección:	Elitista. Torneo, tamaño 4
Criterio de terminación:	Ninguno
Máximo número de generaciones:	100
Altura máxima de un árbol:	5
Método de inicialización:	Muestreo uniforme

La función de adaptación considerada es el Error Cuadrático Medio entre el valor devuelto por la expresión codificada para cada punto y el valor real.

Ejemplo de Funcionamiento (2)

El mejor individuo en la primera generación es:

$$y = (1.85195\text{-}1.64984)*X^2 = 0.20211*X^2$$

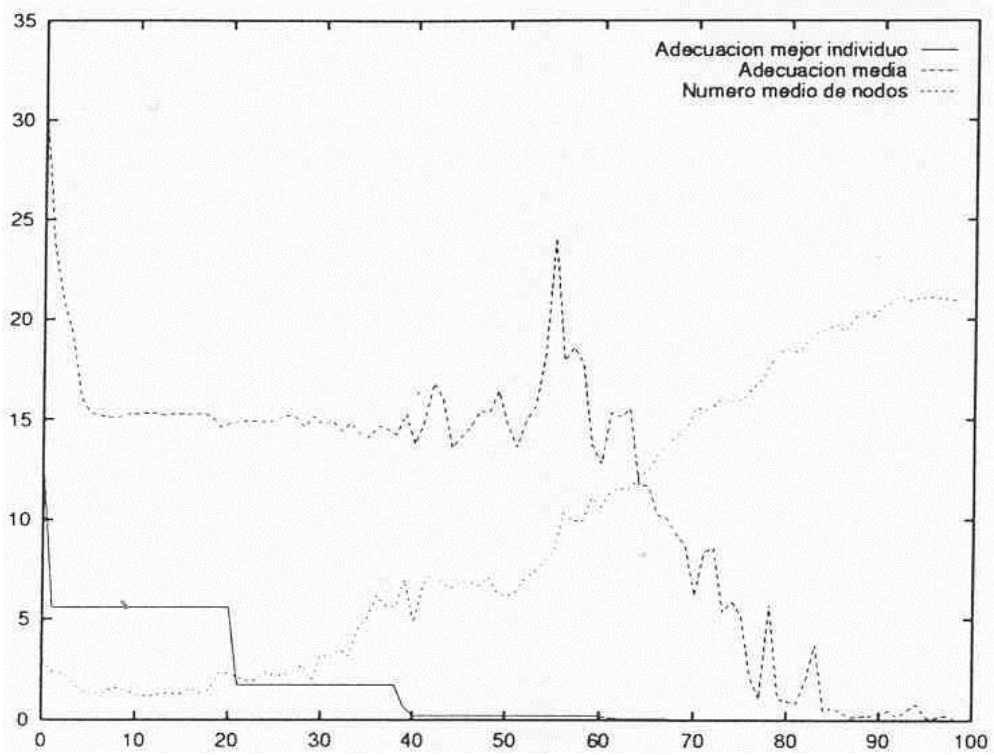
con un error cuadrático de 13.1329.

El mejor de la última generación (100) es:

$$y = (-0.733508)*(-0.733508)*(-0.262223)*(-0.262223)*X^2 + (-0.733508)*(-0.733508)*X^2 = 0.56515*X^2$$

con un error cuadrático de 0.017576.

Ejemplo de Funcionamiento (3)



La adecuación media de la población mejora progresivamente, mientras que la longitud media de las soluciones aumenta.

La razón es la necesidad de combinar varias constantes para obtener expresiones complejas.

Ejemplo de Funcionamiento (4)

Example 2: Symbolic regression using 10 points of the function

Independent variable *X*	Dependent variable *Y*
-1.00	1.00
-0.80	0.84
-0.60	0.76
-0.40	0.76
-0.20	0.84
0.00	1.00
0.20	1.24
0.40	1.56
0.60	1.96
0.80	2.44
1.00	3.00

Ejemplo de Funcionamiento (5)

Example 2: Symbolic regression using 10 points of the function

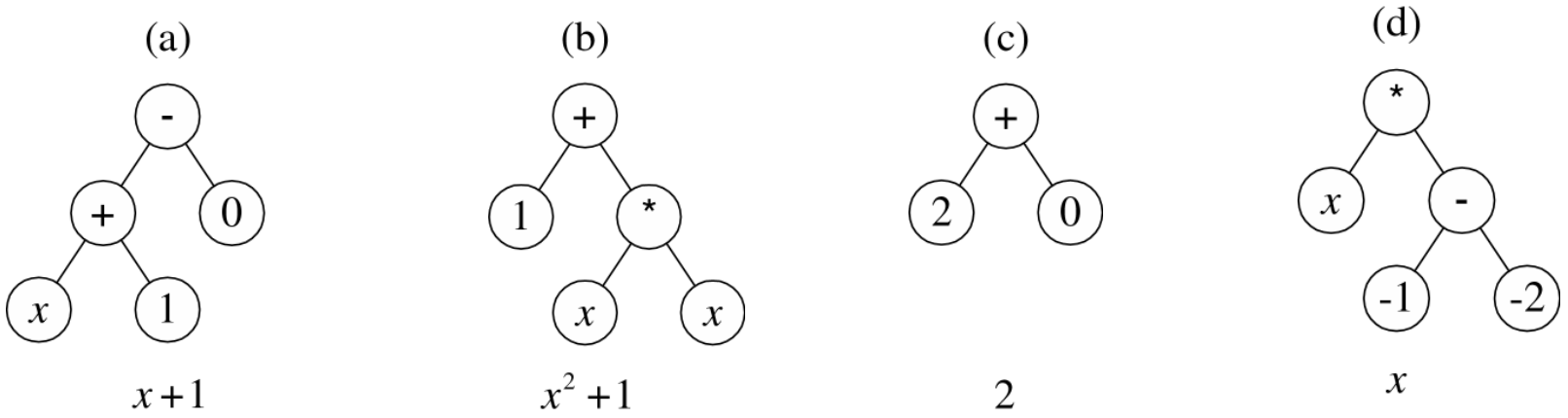
$y = x^2 + x + 1$

Objective:	Find program whose output matches $x^2 + x + 1$ over the range $-1 \leq x \leq +1$.
Function set:	$+$, $-$, % (protected division), and $\times$; all operating on floats
Terminal set:	x, and constants chosen randomly between -5 and $+5$
Fitness:	sum of absolute errors for $x \in \{-1.0, -0.9, \ldots 0.9, 1.0\}$
Selection:	fitness proportionate (roulette wheel) non elitist
Initial pop:	ramped half-and-half (depth 1 to 2. 50% of terminals are constants)
Parameters:	population size 4, 50% subtree crossover, 25% reproduction, 25% subtree mutation, no tree size limits
Termination:	Individual with fitness better than 0.1 found

Ejemplo de Funcionamiento (6)

Example 2: Symbolic regression using 10 points of the function

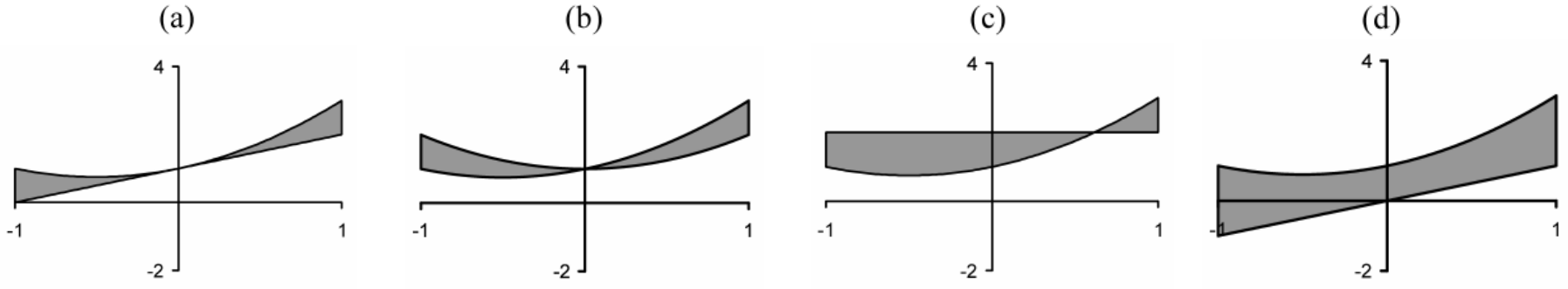
POPULATION OF 4 RANDOMLY CREATED INDIVIDUALS FOR GENERATION 0



Ejemplo de Funcionamiento (7)

Example 2: Symbolic regression using 10 points of the function

FITNESS OF THE 4 INDIVIDUALS IN GEN 0



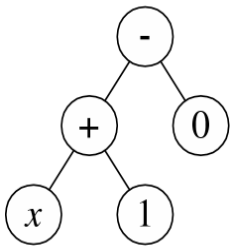
$x + 1$	$x^2 + 1$	2	x
0.67	**1.00**	**1.70**	**2.67**

Ejemplo de Funcionamiento (8)

Example 2: Symbolic regression using 10 points of the function

GENERATION 1

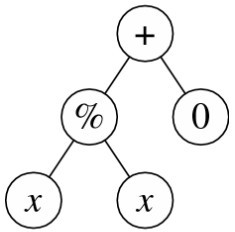
(a)



$x+1$

Copy of (a)

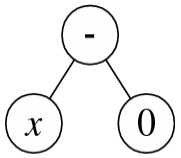
(b)



1

Mutant of (c) picking "2" as mutation point

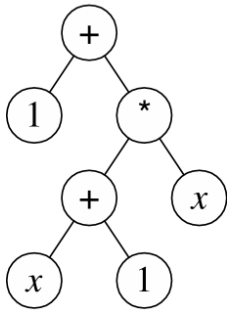
(c)



x

First offspring of crossover of (a) and (b) picking "+" of parent (a) and left-most "x" of parent (b) as crossover points

(d)



x^2+x+1

Second offspring of crossover of (a) and (b) picking "+" of parent (a) and left-most "x" of parent (b) as crossover points

4. ALGORITMOS GA-P

- JUSTIFICACIÓN
- REPRESENTACIÓN DE PROGRAMAS EN GA-P
- OPERADORES GENÉTICOS EN GA-P
- VENTAJAS DE LOS ALGORITMOS GA-P
- CONSIDERACIONES SOBRE EL COMPOR-TAMIENTO DE LOS ALGORITMOS GA-P
- EJEMPLO DE FUNCIONAMIENTO

L. M. Howard, D.J. D'Angelo, The GA-P: A Genetic Algorithm and Genetic Programming Hybrid. IEEE Expert, June 1995, pp. 11-15.

Justificación

El tratamiento de las constantes en PG puede mejorarse de las siguientes formas:

- Modificando el operador de mutación PG para que las constantes reales muten como en un AG con codificación real.
- Incorporando un método de optimización numérica que ajuste el valor de las constantes en las expresiones antes de evaluarlas.
- Evolucionando simultáneamente las expresiones (PG) y las constantes (AG).

La última opción se lleva a cabo mediante algoritmos GA-P.

L. M. Howard, D.J. D'Angelo, The GA-P: A Genetic Algorithm and Genetic Programming Hybrid. IEEE Expert, June 1995, pp. 11-15.

Representación de Programas en GA-P

Los algoritmos GA-P son híbridos entre AGs y PG:

- Cada individuo consta de una expresión y una cadena de coeficientes.
- Ambas partes evolucionan simultáneamente.

GRAMÁTICA:

$S \rightarrow C \mid P$

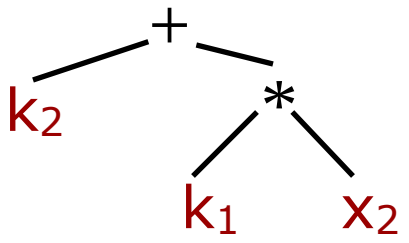
$C \rightarrow$ VALOR-K1 | VALOR-K2

VALOR-K1 $\rightarrow \Re$

VALOR-K2 $\rightarrow \Re$

$P \rightarrow x_1 \mid x_2 \mid k_1 \mid k_2 \mid$
$+ P P \mid * P P$

INDIVIDUO VÁLIDO:



Parte GP

3.27425	1.12997
k_1	k_2

Parte GA

$1.12997 + 3.27425 * x_2$

Operadores Genéticos en GA-P

Los operadores de cruce y mutación se aplican de forma independiente en las componentes GA y GP.

Así, existen una probabilidad de cruce y mutación independientes para cada parte: P_c^{GA}, P_m^{GA}, P_c^{GP}, P_m^{GP}.

Una de las decisiones a tomar es qué operadores GA usar, así como los valores de las probabilidades.

Ventajas de los Algoritmos GA-P

- Pueden manejar información numérica y simbólica simultáneamente.
- La precisión de los resultados finales es superior.
- La población puede ser de menor tamaño que en PG, con lo que el algoritmo converge antes.
- La representación es más compacta, se gana visión acerca de la estructura real del problema con respecto a PG.
- Permite extender la PG a problemas ya resueltos con AGs, ponderando la importancia del aprendizaje de la estructura (PG = sólo estructura) y los datos numéricos (AG = sólo parámetros).

Consideraciones Sobre el Comportamiento de los Algoritmos GA-P

- La población GA-P evoluciona hasta organizarse en varias subpoblaciones de individuos con la misma parte expresional (GP) y distintos valores para las constantes.

- Estos individuos compiten entre sí y con los individuos de otras subpoblaciones simultáneamente.

- El tamaño de la población puede ser dinámico como en AGs, de forma que al principio coexisten varias expresiones distintas y al final sólo sobrevive la subpoblación con la mejor estructura.

  De este modo, al comienzo del algoritmo se tiene un algoritmo de PG y al final un AG.

Ejemplo de Funcionamiento

PROBLEMA: Regresión Simbólica sobre 10 puntos de la función:

$$y = \frac{x^2}{2}$$

Parámetros	Valores
Objetivo:	Hacer evolucionar una función que aproxime 10 puntos de una parábola
Gramática:	POLNOM → MONOMIO POLNOM → MONOMIO + POLNOM MONOMIO → MONOMIO * x MONOMIO → k_1 | k_2 | k_3 | k_4
Representación	GA-P
Tamaño de la población	100
Probabilidad de cruce	95 por ciento
Probabilidad de mutación	5 por ciento
Selección:	Elitista. Torneo, tamaño 4
Criterio de terminación	Ninguno
Máximo número de generaciones	100
Altura máxima de un árbol	10
Método de inicialización	Muestreo uniforme

Función de adaptación: Error Cuadrático Medio.

Ejemplo de Funcionamiento (2)

El mejor individuo en la primera generación es:

$$y = 7.26724 + (-2.63539)$$

con un error cuadrático de 15.2234.

El mejor de la generación 2 es:

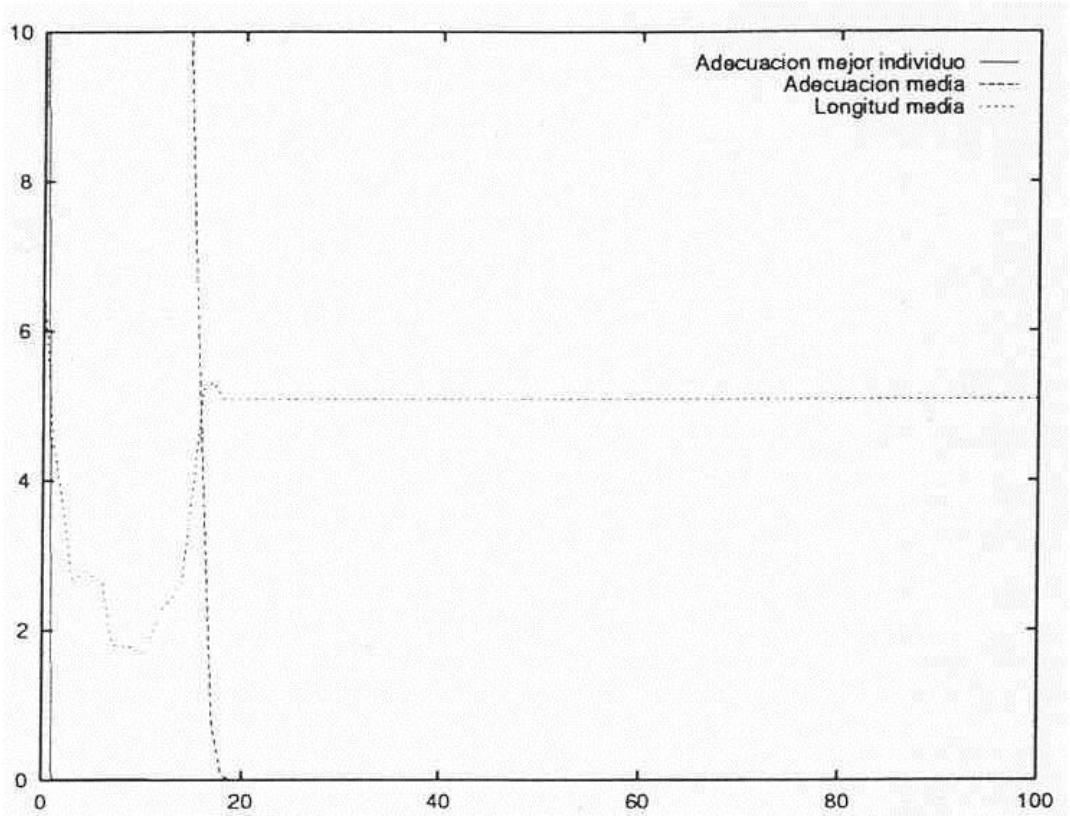
$$y = 0.513389 * X^2$$

con un error cuadrático de 0.0265324.

y el mejor de la generación 26 es la solución óptima:

$$y = 0.5 * X^2$$

Ejemplo de Funcionamiento (3)



Se puede observar como la convergencia es más rápida y la longitud media más pequeña que en el algoritmo de PG.

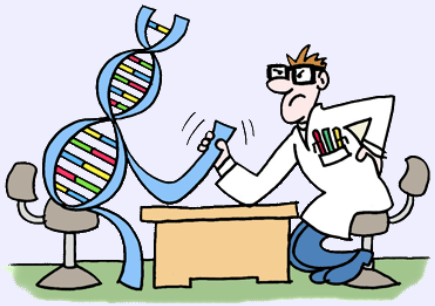
5. APLICACIONES

Tipo de problema	Ejemplos
Control óptimo y modelos inversos	Centrado del péndulo invertido Problemas cinemáticos
Planificación de trayectorias	Hormiga artificial Robot que sigue una trayectoria
Aprendizaje de conceptos booleanos	Problema del multiplexador Funciones booleanas Sumador de dos bits
Regresión simbólica y descubrimiento empírico	Leyes de Kepler Modelos econométricos Series caóticas
Resolución de ecuaciones	Solución aproximada de ecuaciones diferenciales y ecuaciones funcionales Integración simbólica
Programación automática	Generador de números aleatorios Diseño de redes neuronales
Estrategias de juegos	Coevolución Evolución de estrategias de mercado
Clasificación e inducción de árboles de decisión	Inducción de clasificadores Extracción de características
Programación de autómatas celulares	Autómatas de una y dos dimensiones

5. APLICACIONES

http://www.genetic-programming.org/hc2005/main.html

ANNUAL "**HUMIES**" AWARDS
FOR HUMAN-COMPETITIVE RESULTS
PRODUCED BY GENETIC AND EVOLUTIONARY COMPUTATION
HELD AT THE
ANNUAL GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE



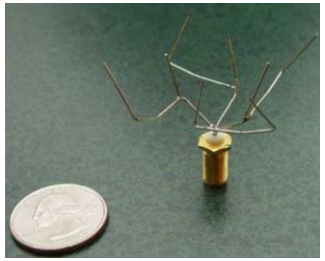
The Annual "Humies" Awards — 2004-2012

Call For Entries for 2012 Human-Competitive Awards

A result produced by an automated method must earn the rating of "human competitive" regardless of the fact that it was automatically generated

A $10,000 prize is awarded to projects that have produced automatically-created results which equal or better those produced by humans

GP applications win some of these prizes.
For example, one of the two 2004 gold medals was given for the GP-based design of an antenna for deployment on NASA's Space Technology 5 Mission (Lohn, Hornby, and Linden)



Genetic Programming

Application: GP has Done Well?

GP has been especially productive in areas having some or all of the following properties:

- **The interrelationships among the relevant variables are unknown or poorly understood**
- **Finding the size and shape of the ultimate solution is a major part of the problem**
- **Significant amounts of test data are available in computer-readable form**

Genetic Programming

Application: GP has Done Well?

- There are good simulators to test the performance of tentative solutions to a problem, but poor methods to directly obtain good solutions
- Conventional mathematical analysis does not, or cannot, provide analytic solutions
- An approximate solution is acceptable (or is the only result that is ever likely to be obtained)
- Small improvements in performance are routinely measured (or easily measurable) and highly prized

Genetic Programming and predictions about data

Application:

Distilling Free-Form Natural Laws from Experimental Data
Michael Schmidt, *et al.*
Science **324**, 81 (2009);
DOI: 10.1126/science.1165893

"Without any prior knowledge about physics, kinematics, or geometry, the algorithm discovered Hamiltonians, Lagrangians, and other laws of geometric and momentum conservation. The discovery rate accelerated as laws found for simpler systems were used to bootstrap explanations for more complex systems, gradually uncovering the "alphabet" used to describe those systems."

Genetic Programming and predictions

Science
AAAS

Distilling Free-Form Natural Laws from Experimental Data
Michael Schmidt, *et al.*
Science **324**, 81 (2009);
DOI: 10.1126/science.1165893

"Symbolic regression is an established method based on evolutionary computation for searching the space of mathematical expressions while minimizing various error metrics"

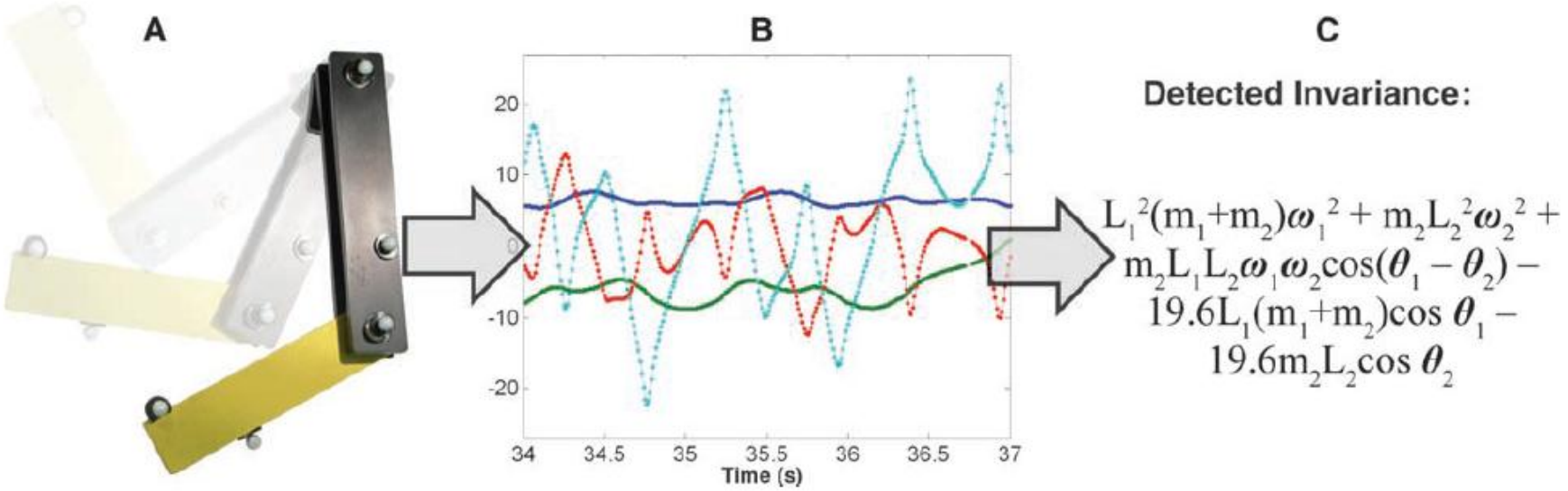


Fig. 1. Mining physical systems. We captured the angles and angular velocities of a chaotic double-pendulum (**A**) over time using motion tracking (**B**), then we automatically searched for equations that describe a single natural law relating these variables. Without any prior knowledge about physics or geometry, the algorithm found the conservation law (**C**), which turns out to be the double pendulum's Hamiltonian. Actual pendulum, data, and results are shown.

Genetic Programming and predictions

Science
AAAS

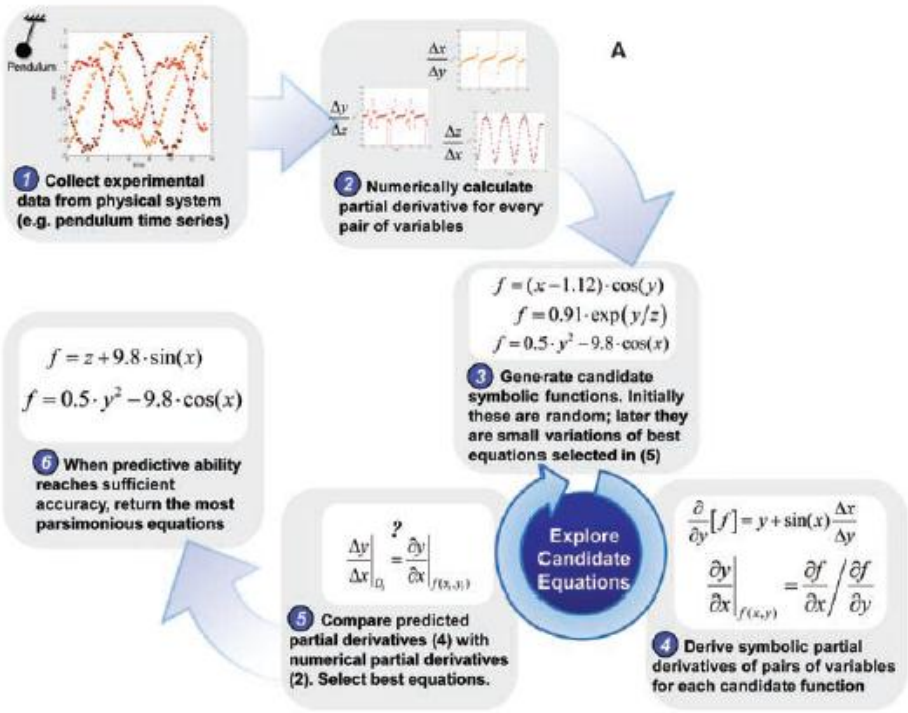
Distilling Free-Form Natural Laws from Experimental Data
Michael Schmidt, *et al.*
Science **324**, 81 (2009);
DOI: 10.1126/science.1165893

Fig. 2. Computational approach for detecting conservation laws from experimentally collected data.

(A) First, calculate partial derivatives between variables from the data, then search for equations that may describe a physical invariance. To measure how well an equation describes an invariance, derive the same partial derivatives symbolically to compare with the data.

(B) The representation of a symbolic equation in computer memory is a list of successive mathematical operations (see SOM section S6).

(C) This list representation corresponds to a graph, where nodes represent mathematical building blocks and leaves represent parameters and system variables. Both (B) and (C) correspond to the same equation. The algorithm varies these structures to search the space of equations.

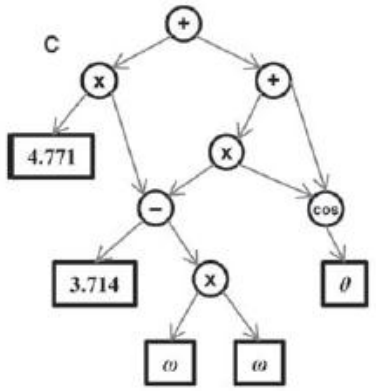


B

$$f(\theta,\omega) = 4.771\cdot(3.714-\omega^2)+\cos(\theta) + (3.714-\omega^2)\cdot\cos(\theta)$$

```
(0)  <- load [3.714]
(1)  <- load [ω]
(2)  <- mul  (1), (1)
(3)  <- sub  (0), (2)
(4)  <- load [θ]
(5)  <- cos  (4)
(6)  <- mul  (3), (5)
(7)  <- load [4.771]
(8)  <- mul  (7), (3)
(9)  <- add  (8), (5)
(10) <- add  (9), (6)
```

C



Genetic Programming and predictions

Science
AAAS

Distilling Free-Form Natural Laws from Experimental Data
Michael Schmidt, *et al.*
Science **324**, 81 (2009);
DOI: 10.1126/science.1165893

- **Given position and velocity data over time, the algorithm converged on the energy laws of each system (Hamiltonian and Lagrangian equations).**
- **Given acceleration data also, it produced the differential equation of motion corresponding to Newton's second law for the harmonic oscillator and pendulum systems.**
- **Given only position data for the pendulum, the algorithm converged on the equation of a circle, indicating that the pendulum is confined to a circle.**
- **An interesting approximate law for the double pendulum that emerged was conservation of angular momentum.**

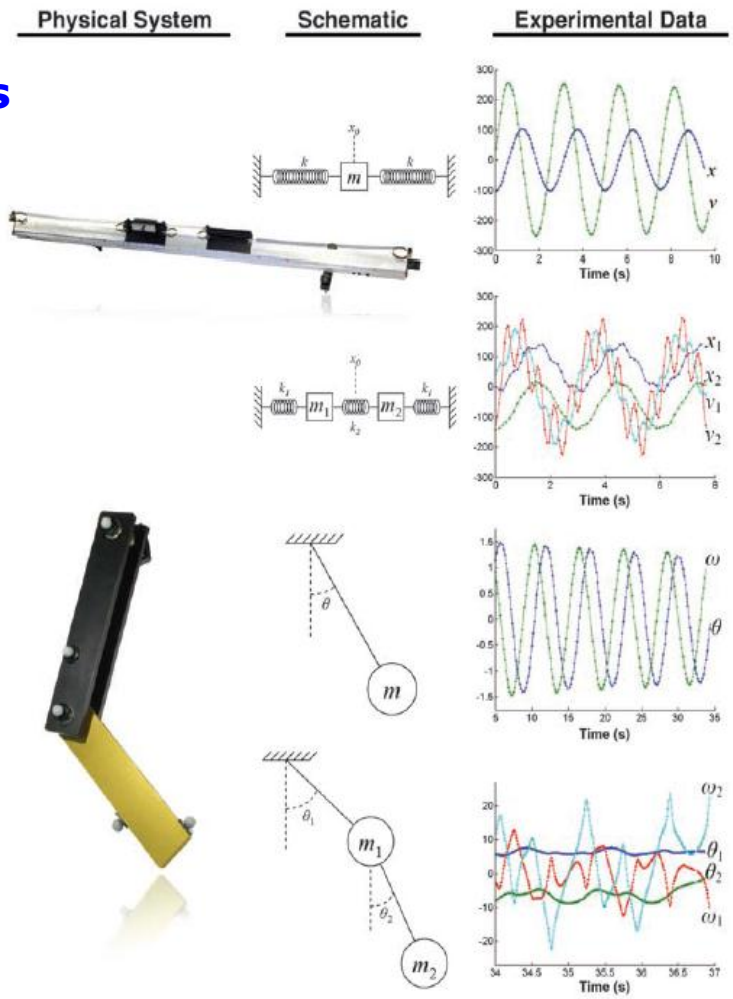


Fig. 3. Summary of laws inferred from experimental data collected from physical systems. Depending on the types of variables provided to the algorithm, it detects different types of laws. Given solely position information, the algorithm detects position manifolds; given velocities, the algorithm detects energy laws; given accelerations, it detects equations of motion and sum of forces laws (θ, angle; ω, angular velocity; α, angular acceleration).

Genetic Programming and predictions

Science
AAAS

Distilling Free-Form Natural Laws from Experimental Data
Michael Schmidt, *et al.*
Science **324**, 81 (2009);
DOI: 10.1126/science.1165893

Interesting reflections by autors:

"We have demonstrated the discovery of physical laws, from scratch, directly from experimentally captured data with the use of a computational search. We used the presented approach to detect nonlinear energy conservation laws, Newtonian force laws, geometric invariants, and system manifolds in various synthetic and physically implemented systems without prior knowledge about physics, kinematics, or geometry. The concise analytical expressions that we found are amenable to human interpretation and help to reveal the physics underlying the observed phenomenon. Many applications exist for this approach, in fields ranging from systems biology to cosmology, where theoretical gaps exist despite abundance in data."

"Might this process diminish the role of future scientists? Quite the contrary: Scientists may use processes such as this to help focus on interesting phenomena more rapidly and to interpret their meaning."

6. DISCUSIÓN FINAL

¿Qué es la PG?

- El arte de evolucionar programas de ordenador
- Un medio para automatizar la programación de ordenadores
- Un AG con otra representación (que permite el manejo de soluciones de longitud variable).

...

Es un área de la Computación Evolutiva que genera grandes pasiones y también críticas (a las pasiones) por aquellos que la consideran una forma más de AGs.

BIOINFORMÁTICA
2013 - 2014

PARTE I. INTRODUCCIÓN

- Tema 1. Computación Basada en Modelos Naturales

PARTE II. MODELOS BASADOS EN ADAPTACIÓN SOCIAL (Swarm Intelligence)

- Tema 2. Introducción a los Modelos Basados en Adaptación Social
- Tema 3. Optimización Basada en Colonias de Hormigas
- Tema 4. Optimización Basada en Nubes de Partículas (Particle Swarm)

PARTE III. COMPUTACÍON EVOLUTIVA

- Tema 5. Introducción a la Computación Evolutiva
- Tema 6. Algoritmos Genéticos I. Conceptos Básicos
- Tema 7. Algoritmos Genéticos II. Diversidad y Convergencia
- Tema 8. Algoritmos Genéticos III. Múltiples Soluciones en Problemas Multimodales
- Tema 9. Estrategias de Evolución y Programación Evolutiva
- Tema 10. Algoritmos Basados en Evolución Diferencial (Diferential Evolution – DE)
- Tema 11. Modelos de Evolución Basados en Estimación de Distribuciones (EDA)
- Tema 12. Algoritmos Evolutivos para Problemas Multiobjetivo
- Tema 13. Programación Genética
- Tema 14. Modelos Evolutivos de Aprendizaje

PARTE IV. OTROS MODELOS DE COMPUTACIÓN BIOINSPIRADOS

- Tema 15. Sistemas Inmunológicos Artificiales
- Tema 16. Otros Modelos de Computación Natural/Bioinspirados