Programacion Genetica

PG a Vuelo de Pajaro

- Desarrollado: EEUU en los 90s
- Pioneros: J. Koza pero…
- Generalmente aplicado a:
  - prediccion, clasificacion
- Propiedades generales:
  - compite con NN y similares
  - necesita poblaciones bastante grandes (miles?)
  - lenta
- Propiedades particulares:
  - cromosomas no lineales: arboles, grafos
  - mutacion posible pero no necesaria (disputado!)

Tableau Tecnico

Representacion	Arboles
Recombinacion	Intercambio de sub-arboles
Mutacion	Modificaciones al azar en los (sub)arboles
Seleccion de padres	Proporcional al fitness
Seleccion de sobrevivientes	Reemplazamiento generacional

Ejemplo:evaluacion de creditos

- Los bancos necesitan distinguir entre buenos aplicantes (a creditos) y malos
- Se necesita un modelo que se ajuste a datos historicos

ID	Nro de hijos	Salario	E.civil	OK?
ID-1	2	45000	Casado	0
ID-2	0	30000	Soltero	1
ID-3	1	40000	Divorciado	1
…				

- Un modelo posible:

IF (NDH = 2) AND (S > 80000) THEN bueno ELSE malo

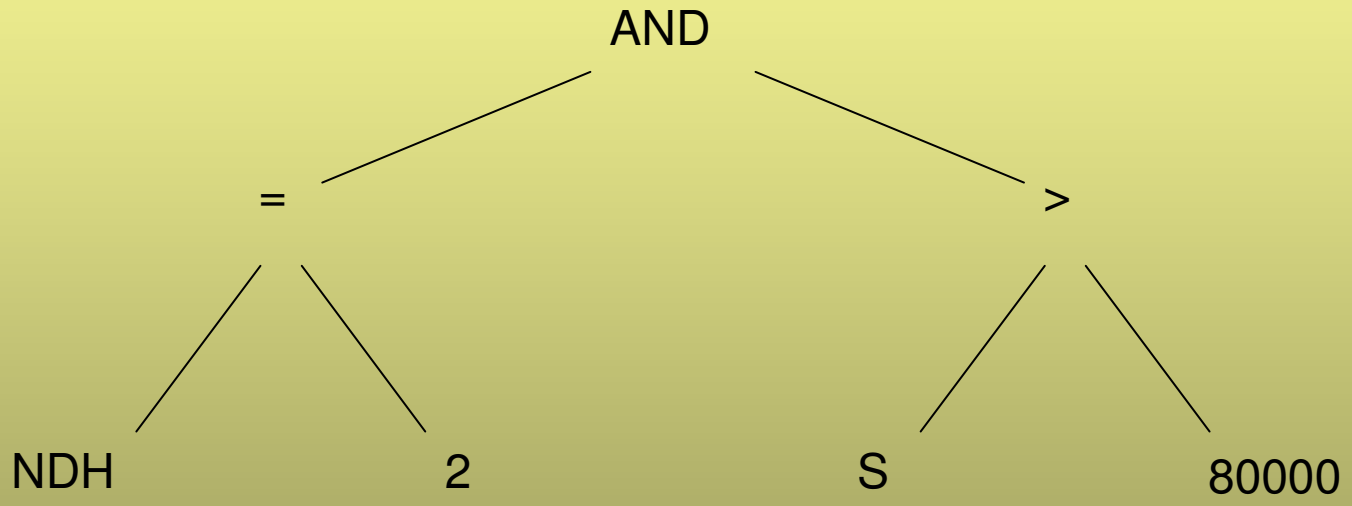
- En general:

IF formula THEN bueno ELSE malo

- No se conoce la estructura de la formula, entonces
- El espacio de busqueda es el espacion de formulas (fenotipo)
- Fitness de una formula designoide: porcentaje de casos bien clasificados
- La representacion natural de las formulas (genotipo) es un “parse tree” (arbol de clasificacion)

IF (NDH = 2) AND (S > 80000) THEN bueno ELSE malo

se representa con:



Representacion basada en Arboles

Representacion muy general, Ejemplo:

- Formula aritmetica

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

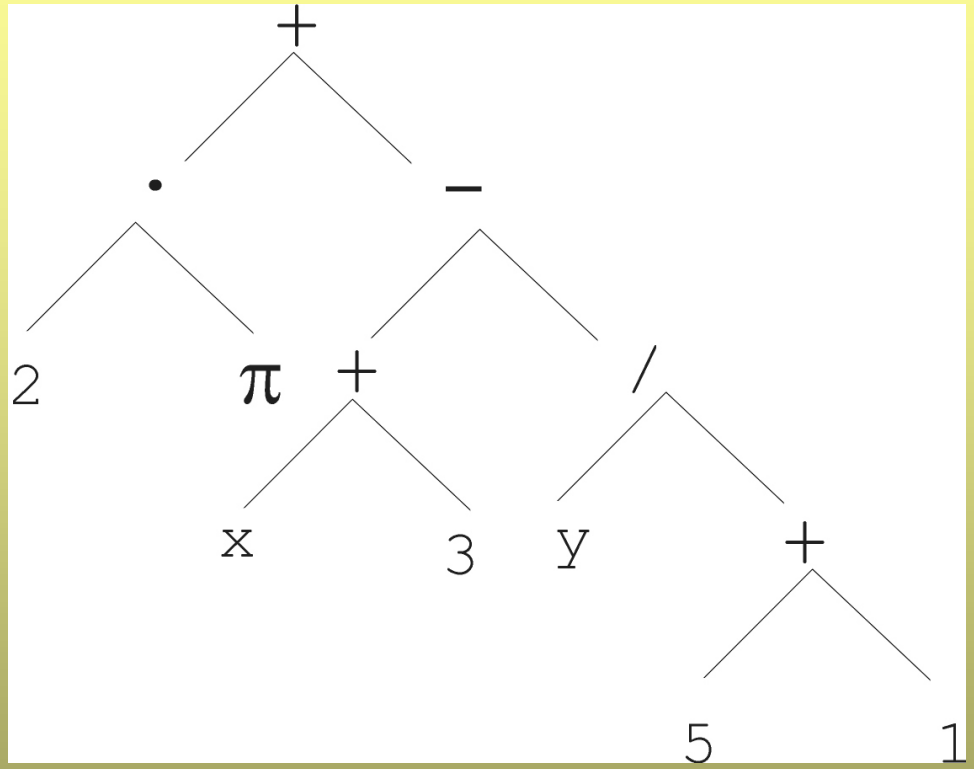
- Formula logica

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

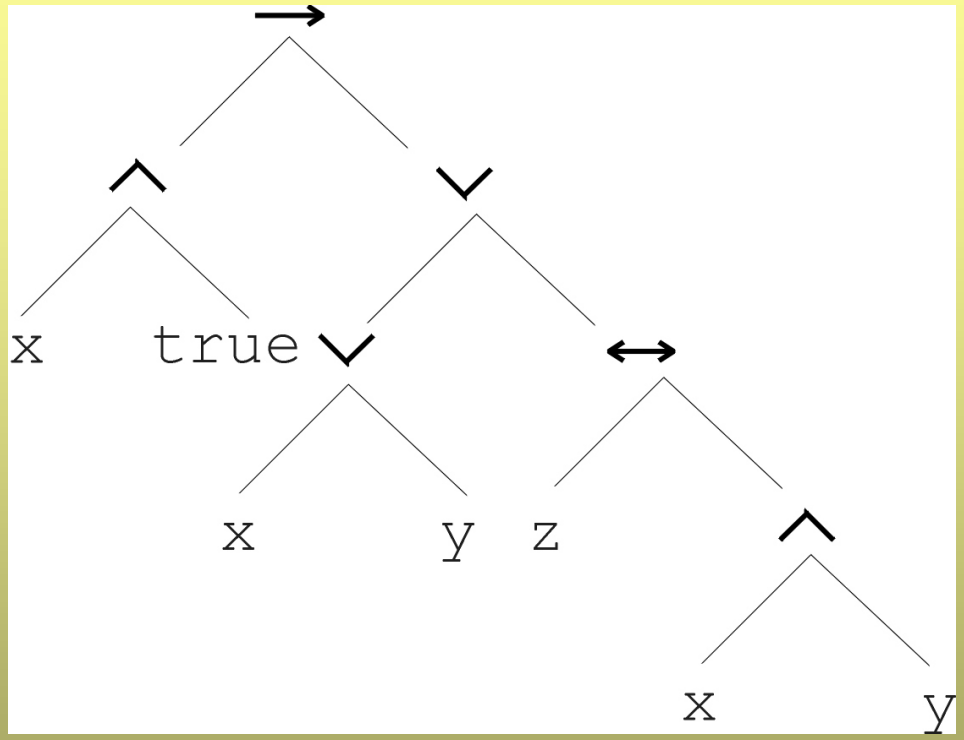
- Programa

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

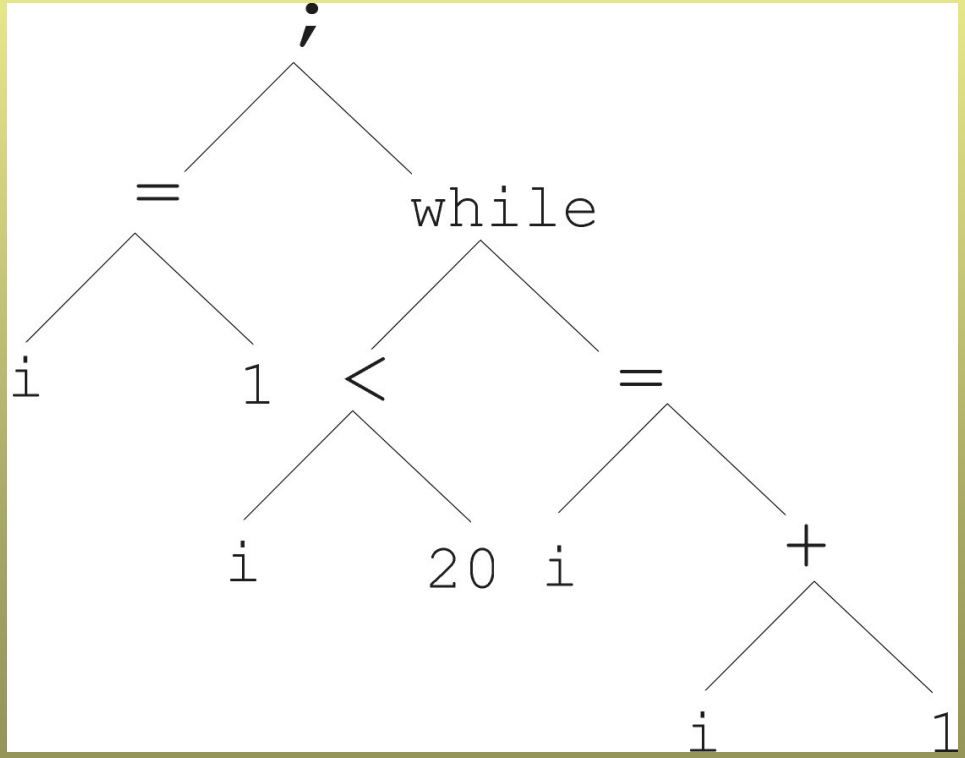
$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$



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



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



- En AGs, PE y EE la representacion se base en strings, por ejemplo, de bits, enteros, reales (vectores, permutaciones, etc)
- Estos arboles son estructuras no lineales
- En AGs, PE y EE los cromosomas (en general) son de una longitud fija
- En PG los arboles pueden variar de estructura → varian de longitud (profundidad, densidad, etc)

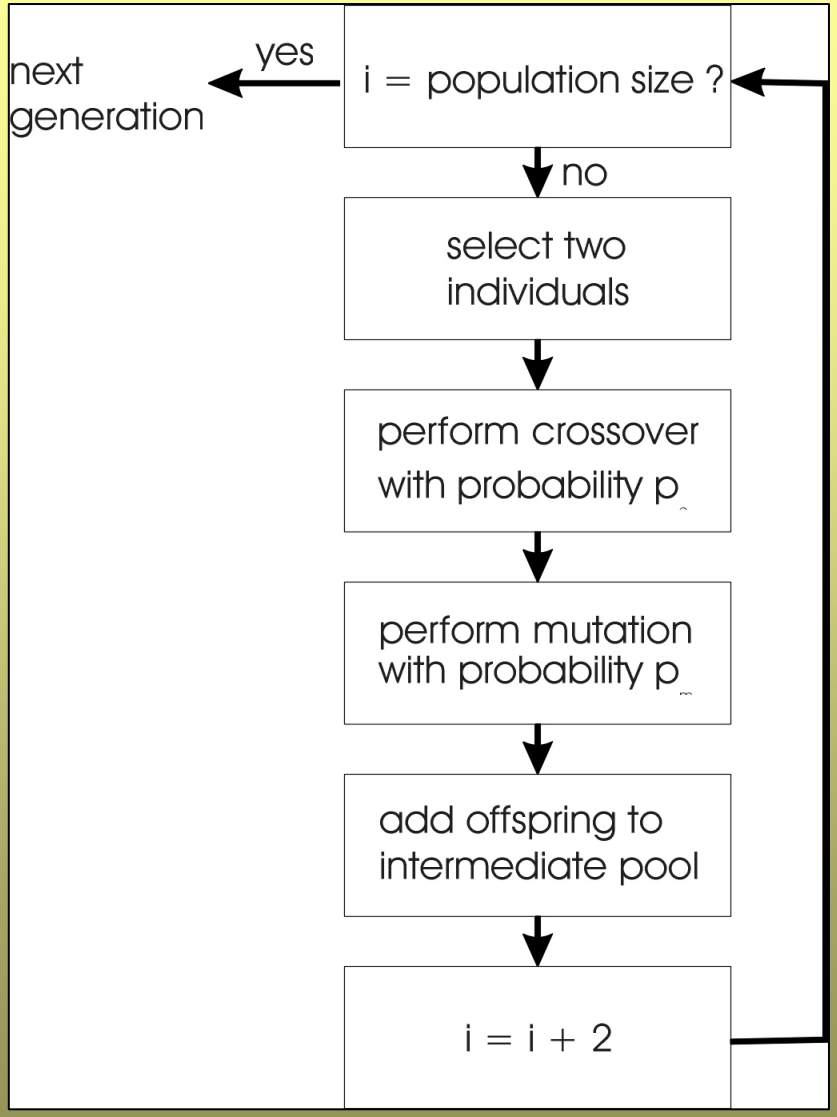
- Las expresiones representadas por arboles constan de:
  - un conjunto terminal T
  - un conjunto de funciones F (con sus aridades)
- Se adopta la siguiente definicion recursiva:
  1. Cada $t \in T$ es una expresion correcta
  2. $f(e_1, \ldots, e_n)$ es una expresion correcta if $f \in F$, arity(f)=n y $e_1, \ldots, e_n$ son expresiones correctas
  3. No hay otras formas correctas
- En general, GP no es fuertemente tipado (propiedad de clausura: cualquier $f \in F$ puede tomar $g \in F$ como argumento)

Creacion de Hijos

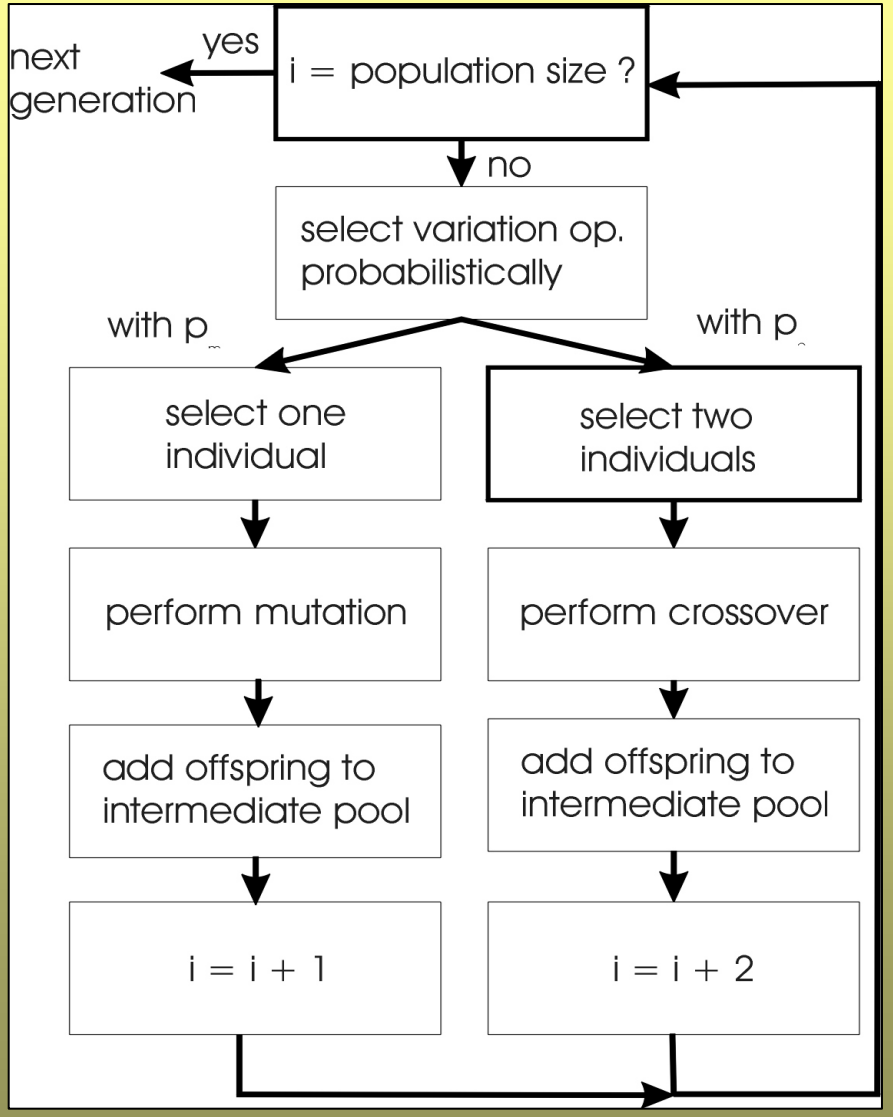
Comparen:

- Un AG usa secuencialmente mutacion y crossover (con probs)
- Un PG usa alternativamente mutacion o crossover (con probs)

GA flowchart

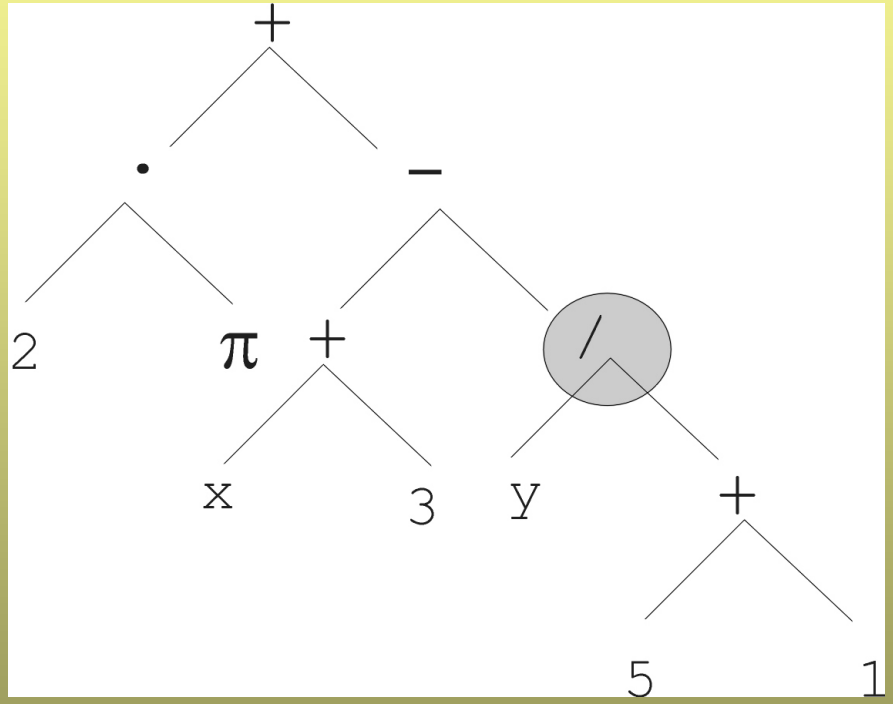


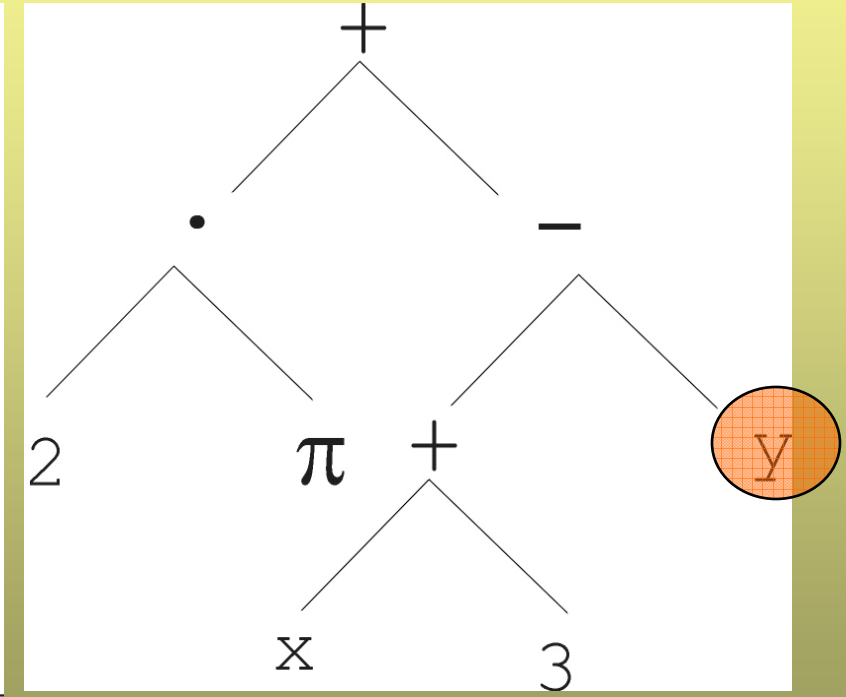
GP flowchart



Mutacion

- La mutacion mas comun: reemplazar al azar un subtree con uno nuevo creado al azar





- La mutacion tiene dos parametros:
  - Probabilidad p_m de seleccionar mutacion en vez de crossover
  - Probabilidad de seleccionar un nodo interno como la raiz del subarbol a mutar

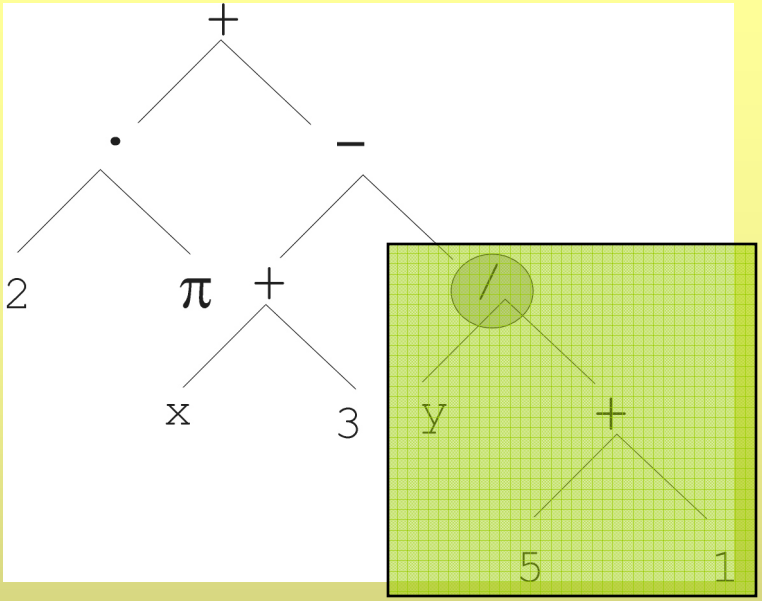
- Notablemente Koza recomienda $p_m = 0$ (Koza'92) o muy chica, como por ejemplo 0.05 (Banzhaf et al. '98)

- Notar que el tamanio del hijo puede exceder el tamanio del padre

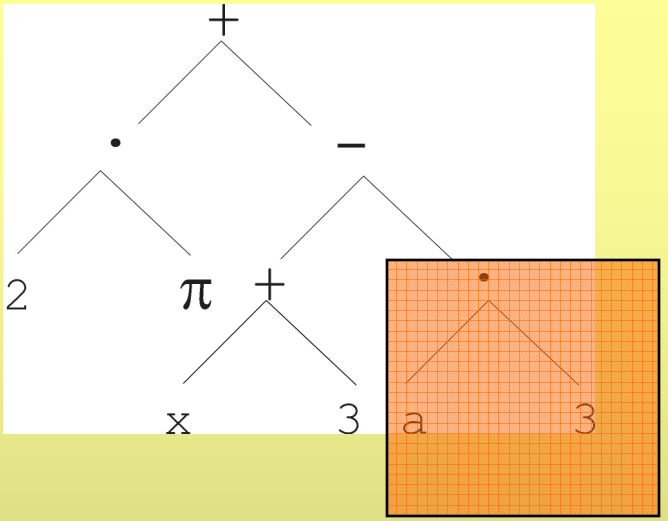
Recombinacion

- La recombinacion mas comun: intercambiar dos subarboles entre los padres
- Recombinacion usa dos parametros:
  - Probabilidad p_c de elegir recombinacion y no mutacion
  - La probabilidad de elegir un nodo interno para recombinar
- Aqui tambien el resultado puede ser un hijo mas grande que los padres

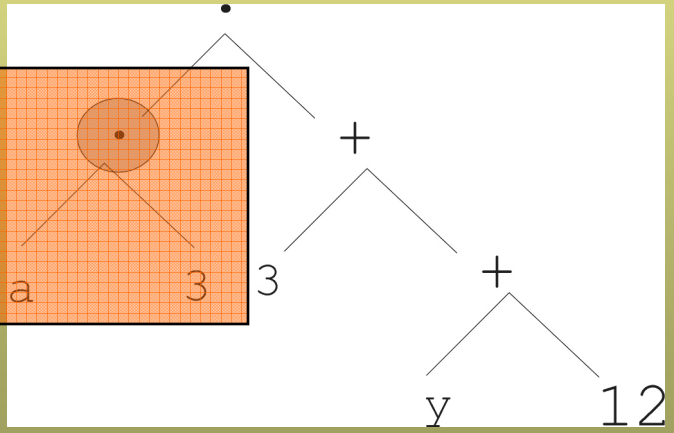
P1



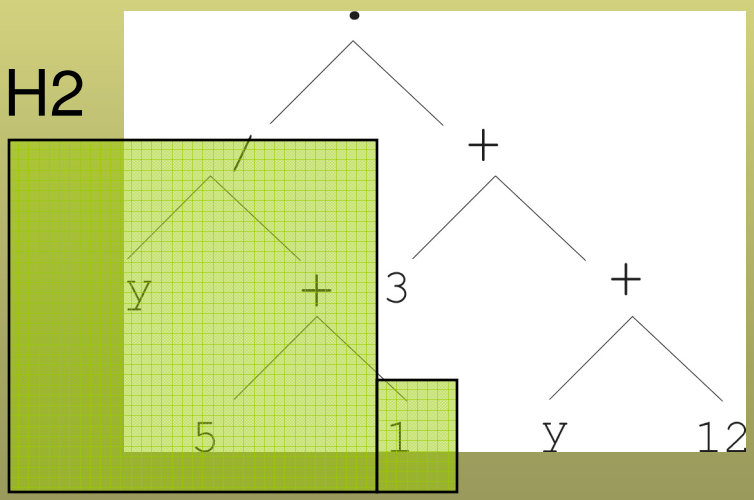
H1



P2



H2



Seleccion

- Para la seleccion de padres se usa generalmente proporcional al fitness
- Sobre seleccion en poblaciones grandes:
  - se rankea y divide la poblacion en dos grupos:
  - grupo 1: los mejores x% de la poblacion, grupo 2 el resto
  - El 80% de los padres viene del grupo 1 (para la reproduccion) y solo un 20% del segundo
  - para |poblacion| = 1000, 2000, 4000, 8000 x = 32%, 16%, 8%, 4%
  - motivacion: incrementar la eficiencia, el % es un “rule of thumb”
- Seleccion de sobrevivientes
  - Tipicamente: generacional, esto es hijos reemplazan padres
  - Sin embargo recientemente la gente utilizo steady state por el elitismo

Initialisacion

- Se define una profundidad maxima D_{max} para los arboles
- Metodo completo (cada rama realiza D_{max}):
  - nodos con profundidad $d < D_{max}$ elegidos al azar del conjunto F
  - nodes con profundidad $d = D_{max}$ elegidos al azar del conjunto T
- Metodo constructivo (cada rama con prof $\leq D_{max}$):
  - nodos con profundidad $d < D_{max}$ elegidos al azar de la union de F y T
  - nodes con profundidad $d = D_{max}$ elegidos al azar del conjunto T
- Inicialisaciones: “ramped half-and-half”, donde cada uno de los anteriores produce la mitad de la poblacion

“Bloat”

- Bloat = “Seleccion del mas gordo”, i.e., los arboles en la poblacion tienden a crecer
- Hay un debate sobre las razones de bloat
- Tomar contra-medidas, e.g.
  - Operadores que crean chicos mas bien “deportivos” y no tan “gordos”
  - Parsimonia: penalizar los designoides grandes sin cambiar los operadores
  - Hibrido de ambos metodos

PG para controladores

- Arboles para “data fiting” vs. arboles (programas) ejecutables
- La ejecucion de un programa podria cambiar el entorno donde el designoide esta embebido → cambio de la funcion de fitness (si es que la evolucion acontece on-line)
- Ejemplo: robot controller
- Los designoides (controladores) se evaluan off-line por simulaciones que suelen ser o caras o MUY caras
- Sin embargo tienden a ser bastante buenos

Problema mas tipico: regresion simbolica

- Dados puntos ejemplos en $\mathbf{R}^2$, $(x_1, y_1), \ldots, (x_n, y_n)$
- Encontrar una funcion f(x) s.t. $\forall i = 1, \ldots, n : f(x_i) = y_i$
- PG al rescate:
  - Representacion con F = {+, -, /, sin, cos}, T = $\mathbf{R} \cup \{x\}$
  - El inverso del error es el fitnes
  - Operadores estandard $err(f) = \sum_{i=1}^{n} (f(x_i) - y_i)^2$
  - |poblacion|= 1000, inicializacion ramped half-half
  - Terminacion: n "hits" o 50000 evaluaciones de fitness (donde "hit" es $|f(x_i) - y_i| < 0.0001$, presicion)

Discusion

Que es PG:

El arte de evolucionar programas computacionales ?

El medio para programacion automatica?

Un AG con otra representacion?