Programacion Evolutiva

PE Resumen

- Desarrollado: en EEUU en los 60s
- Pionero: D. Fogel
- Tradicionalmente aplicado a:
  - tradicional: aprendisaje computacional por automatas de estados finitos
  - moderno: optimisacion numerica
- Caracteristicas principales:
  - metodologia abierta: cualquier representacion y mutacion es OK
  - mezcla con estategias evolutivas
  - consecuentemente: es dificil decir que es “estandard” PE
- Caracteristicas especiales:
  - no Xover
  - autoadaptacion de parametros (en las PE contemporanea)

Tableau de Resumen Tecnico

Representacion	Vectores reales
Recombinacion	None
Mutacion	Perturbacion Gausiana
Seleccion de padres	Deterministica
Seleccion de sobrevivientes	Probabilistica (μ+μ)
Especialidad	auto-adaptacion de los pasos de mutacion(in meta-EP)

Perspectiva Historica

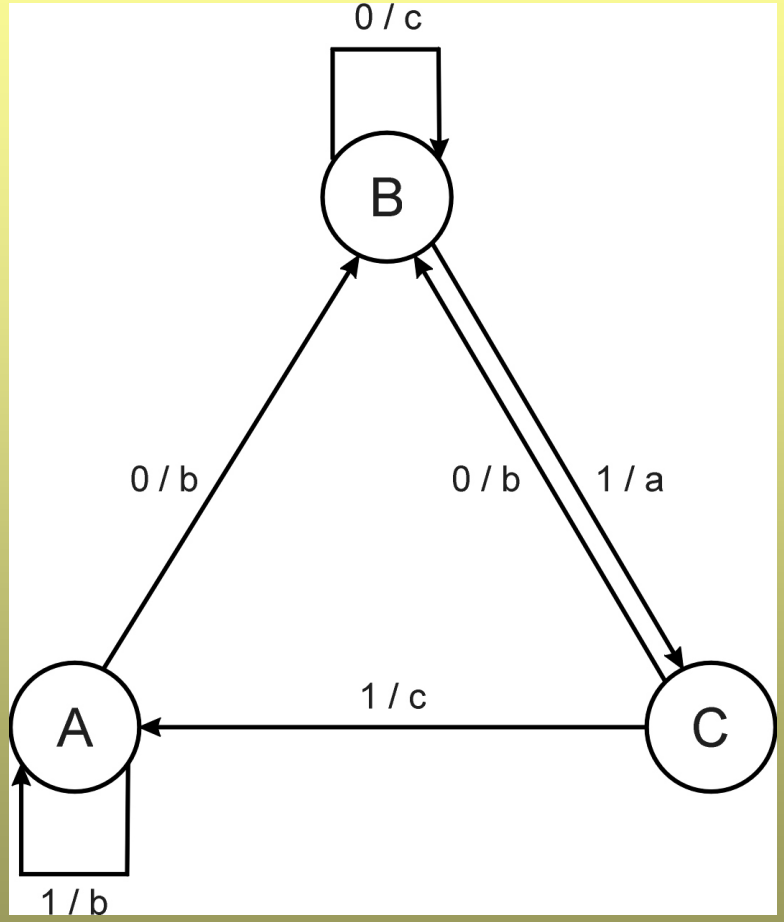
- PE estaba dirigida a producir inteligencia
- Inteligencia se veia como comportamiento adaptativo
- Prediccion del entorno se veia como requisito para comportamiento adaptativo
- En consecuencia la capacidad de predecir es clave para un comportamiento inteligente

Prediccion por Maquinas de Estados Finitos (MEF)

- MEFs:
  - Estados S
  - Inputs I
  - Outputs O
  - Funcion de transicion $\delta : S \times I \rightarrow S \times O$
  - Transforma una secuencia de inputs en una secuencia de outputs
- Puede ser usado para predecir, por ejemplo, el siguiente simbolo en una secuencia

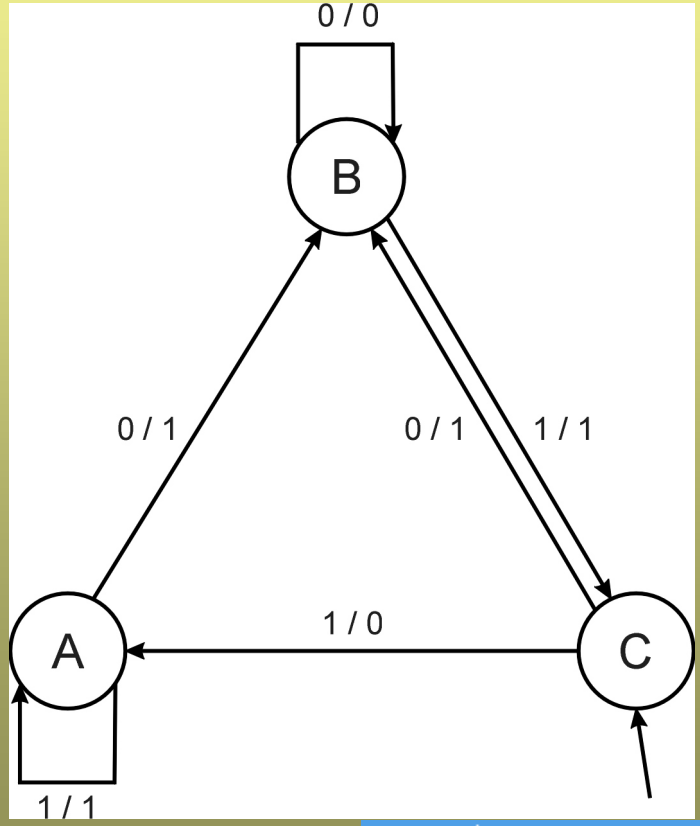
Ejemplo de MEF

- Consideren la MEF con:
  - S = {A, B, C}
  - I = {0, 1}
  - O = {a, b, c}
  - δ dada por el diagrama



MEF como predictor

- Considere la siguiente MEF:
- Tarea: predecir el siguiente input
- Calidad: % de $in_{(i+1)} = out_i$
- Dado estado inicial C
- Secuencia input 011101
- Produce 110111
- Quality: 3 out of 5



Ejemplo introductorio: Designoid MEF para predecir primos

- P(n) = 1 si n es primo, 0 de otra manera
- I = *N* = {1,2,3,…, n, …}
- O = {0,1}
- Prediccion correcta: $out_i = P(in_{(i+1)})$
- Fitness:
  - 1 por una prediccion correcta del siguiente input
  - 0 por prediccion incorrecta
  - Penalidad por complejidad sintactica

- Seleccion de padres: cada MEF se muta una vez
- Operadores de mutacion (se selecciona uno al azar):
  - Cambiar un simbolo output
  - Cambiar una transicion
  - Agregar un estado
  - Borrar un estado
  - Cambiar el estado inicial
- Seleccion de sobrevivientes: (μ+μ)
- Resultados: "overfitting", luego de 202 inputs la mejor maquina tenia un solo estado con ambos outputs 0, esto es, siempre predice "no primo"

PE Moderna

- En general no tiene una representacion predefinida
- Consecuentemente: no hay una mutacion predefinida (en general debe corresponder de alguna manera con la representacion elegida)
- Aplica auto-adaptacion al operador de mutacion
- En lo que sigue presentamos una version de PE moderna

Representacion

- Para optimisacion de parametros continuos
- Cromosoma consiste de dos partes:
  - Variables del designoid: $x_1,\ldots,x_n$
  - Tamanio de los pasos de mutacion: $\sigma_1,\ldots,\sigma_n$
- Esto es: $\langle x_1,\ldots,x_n, \sigma_1,\ldots,\sigma_n \rangle$

Mutacion

- Cromosomas: $\langle x_1,\ldots,x_n, \sigma_1,\ldots,\sigma_n \rangle$
- $\sigma_i' = \sigma_i \cdot (1 + \alpha \cdot N(0,1))$
- $x'_i = x_i + \sigma_i' \cdot N_i(0,1)$
- $\alpha \approx 0.2$
- regla del borde: $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$
- Otras variantes:
  - Esquemas Lognormal como en Estrategias Evolutivas
  - Usar varianza en vez de las desviaciones estandares
  - Mutar σ al final
  - Otras distribuciones, e.g, Cauchy en vez de Gaussian

Recombinacion

- Ninguna
- Motivacion (ideologica): un punto en el espacio de busqueda es en realidad una especie (no un individuo) y no pueder haber entre cruzamiento entre especies.
- Devuelta el debate “mutacion vs. crossover”
- Los beneficios practicos terminan decidiendo estos debates

Seleccion de Padres

- Cada individuo crea un solo hijo por mutacion
- Entonces:
  - Deterministica
  - No depende del fitness

Seleccion de Supervivientes

- P(t): μ padres, P'(t): μ hijos
- Competicion round-robin entre pares de individuos:
  - Cada solucion x de $P(t) \cup P'(t)$ es evaluada contra otros q soluciones elegidas al azar
  - Para cada comparacion, se asigna un "win" si x tiene mejor fitness
  - Las μ con el mayor numero de "wins" constituyen la siguiente generacion
- El parametro q permite ajustar la presion selectiva
- Tipicamente $q = 10$

Ejemplo: Ackley function (Bäck et al '93)

- La funcion de Ackley (con n =30):

$$f(x) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$$

- Representacion:
  - $-30 < x_i < 30$ (solo una coincidencia con n, en general puede ser distinto)
  - 30 varianzas como el paso de mutacion
- Mutacion cambia primero las variables del designoid luego los pasos
- |poblacion|= $\mu = 200$, seleccion con q = 10
- Terminacion : luego de 200000 evaluaciones de fitnes
- Resultados: la mejor solucion promedio es $1.4 \cdot 10^{-2}$

Ejemplo: Jugadores de damas designoids (Fogel'02)

- Redes neuronales designoids se usan para evaluar las siguientes movidas
- NNs tienen estructura fija con 5046 weights, estos son evolucionados + un peso (bias) para "kings"
- Representacion:
  - vector de 5046 reales (pesos de las NN)
  - vector de 5046 reales para las σ's
- Mutacion:
  - Gaussian, esquema lognormal mutando σ primero
  - mas un mecanismo especial para los "kings"
- |poblacion|= 15

- Tamanio del torneo q = 5
- Sistema co-evolutivo, las NN juegan y evolucionan entre ellas (no hay un entrenador humano)
- Luego de 840 generaciones (6 meses) el mejor designoid se midio contra humanos en el internet
- El designoid se gano un ranking “expert class” ganandole al 99.61% de todos los jugadores rankeados
- Para los interesados leer el libro de Fogel Blondie24