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

COMPUTACIÓN EVOLUTIVA
PARADIGMAS DE EVOLUCIÓN

COMPUTACIÓN EVOLUTIVA

PARADIGMAS CLÁSICOS DE EVOLUCIÓN

- **ALGORITMOS GENÉTICOS**
  - **PROGRAMACIÓN GENÉTICA**
- **ESTRATEGIAS DE EVOLUCIÓN**
  - **PROGRAMACIÓN EVOLUTIVA**

OTROS MODELOS DE EVOLUCIÓN DE POBLACIONES

- **BASADOS EN MODELOS PROBABILÍSTICOS: BSC, PBIL, EDA, ...**
- **PARTICLE SWARM: ADAPTACIÓN SOCIAL**
- **BÚSQUEDA DISPERSA** (scatter search)
- **ALGORITMOS CULTURALES**
- **DIFFERENTIAL EVOLUTION**
- **ALGORITMOS MEMÉTICOS**

BIOINFORMÁTICA

TEMA 11: MODELOS DE EVOLUCIÓN BASADOS EN ESTIMACIÓN DE DISTRIBUCIONES (EDA)

1. **BÚSQUEDA MEDIANTE ADAPTACIÓN DE PROBABILIDADES**
2. **EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL**
3. **EXTENSIÓN DE LOS MODELOS PROBABILÍSTICOS BASADOS EN LA TEORÍA DE BLOQUES: EDAs**

1. BÚSQUEDA MEDIANTE ADAPTACIÓN DE PROBABILIDADES
IDEAS INICIALES

- Familia de algoritmos para la resolución de problemas binarios:

$$f: S \rightarrow \Re^{+}, \quad S = \{0,1\}^{l}$$

- Utilizan un vector de probabilidades P para generar las nuevas soluciones (puntos de S) en cada iteración:

$$P = (p_1, \ldots, p_l),\ p_i \in [0,1]$$

Se asume que p_i indica la probabilidad de generar un 1.

1. BÚSQUEDA MEDIANTE ADAPTACIÓN DE PROBABILIDADES

IDEAS INICIALES

Population #1	Population #2	Population #3
0 0 1 1	1 0 1 0	1 0 1 0
1 1 0 0	1 1 0 0	0 1 0 1
1 1 0 0	1 1 0 0	1 0 1 0
0 0 1 1	1 1 0 0	0 1 0 1
Representation	**Representation**	**Representation**
0.5, 0.5, 0.5, 0.5	1.0 0.75 0.25 0.0	0.5 0.5 0.5 0.5

1. BÚSQUEDA MEDIANTE ADAPTACIÓN DE PROBABILIDADES

IDEAS INICIALES

- En lugar de adaptar los individuos, como se hace en otros algoritmos evolutivos, en este caso se adapta el vector de probabilidades.

- Para ello, se considera la calidad de las *n* soluciones generadas en la iteración actual:

$$S = (s_1, \ldots, s_n),\ s_j \in S$$

$$s_j = (s_{j1}, \ldots, s_{jl})$$

y se "aproxima" la distribución de probabilidad a las soluciones de mayor calidad.

1. BÚSQUEDA MEDIANTE ADAPTACIÓN DE PROBABILIDADES

IDEAS INICIALES

Algoritmo evolutivo:

1. **Inicializar** P = (p_1, ..., p_l), (en general (0.5, ..., 0.5)).

2. **Generar** S = (s_1, ..., s_n) utilizando P.

3. **Evaluar** f(s_1), ..., f(s_n).

4. **Actualizar** P de acuerdo a S y f(s_1), ..., f(s_n).

5. **Ir a** 2 o **Parar**.

Diferencias entre los distintos algoritmos:
mecanismo de actualización de la distribución de probabilidad:
PBIL, cGA, EDAs, …

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

PRIMER MODELO

S. Baluja, Population-based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Tech. Rep. CMU-CS-94-163, Carnegie mellon University, 1994.

S. Baluja, R. Caruana, Removing the Genetics from the Starndard Genetic Algorithm. En: A. Prieditis, S. Russel, Int. Conf. on Machine Learning 1995 (ML-95): Morgan Kaufmann Pub., San Mateo, CA, 1995, Plenary Presentation, 38-46.

PBIL: Utiliza el concepto de población de cromosomas con distribuciones de probabilidad aplicadas a los genes.

La distribución de probabilidad se representa como un vector $P=(p_1, ..., p_l)$ con elementos en [0,1] que especifican la probabilidad de que cada posición contenga un 1.

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

El vector de probabilidades se usa para generar la siguiente población, y se actualiza usando una regla que considera la **mejor solución encontrada hasta ahora:**

$$p_i \leftarrow p_i \cdot (1.0 - LR) + \text{Mejor}[i] \cdot LR$$

LR es un parámetro que controla la velocidad de convergencia.

Para evitar una convergencia prematura, se aplica un operador de mutación sobre el vector de probabilidades.

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

```
P ← initialize probability vector. (Each position = 0.5)

loop # GENERATIONS

        # Generate Samples
        i ← loop #SAMPLES
                sample_i ← generate sample vector according to probabilities in P
                evaluation_i ← evaluate (sample_i)

        # Find Best Sample
        max. ← find vector corresponding to maximum evaluation

        # Update Probability Vector
        I ← loop #LENGTH
                P_I ← P_I * (1.0 - LR) + max_I * (LR)

        # Mutate Probability Vector
        I ← loop #LENGTH
                if (random (0,1] < MUT_PROBABILITY)
                        P_I ← P_I * (1.0 - MUT_SHIFT) + random (0.0 or 1.0) * (MUT_SHIFT)
```

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

USER DEFINED CONSTANTS:

GENERATIONS: number of iterations to allow learning.

SAMPLES: the population size, number of samples to produce per generation

LENGTH: length of encoded solution

MUT_PROBABILITY: probability of mutation occurring in each position

MUT_SHIFT: amount for mutation to affect the probability vector

LR: learning rate

VARIABLES:

P: probability vector

$sample_{i..SAMPLES}$: solution vectors

$evaluation_{i..SAMPLES}$: evaluations of the solution vectors

max: solution vector corresponding to maximum evaluation

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

Extensiones Básicas de PBIL: Aprender de los M mejores elementos de S ($\mu \leq \lambda = n$).

PBIL

Obtener un vector de probabilidades inicial $\mathbf{p}_0 = (p_1^{(0)}, p_2^{(0)}, \ldots, p_n^{(0)})$

Repetir para $t = 1, 2, \ldots$ hasta cumplir criterio parada

Utilizando $\mathbf{p}_t$ obtener λ individuos $\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}, \ldots, \mathbf{x}_\lambda^{(t)}$

Evaluar y ordenar $\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}, \ldots, \mathbf{x}_\lambda^{(t)}$

Seleccionar los $\mu \leq \lambda$ mejores individuos $\mathbf{x}_{1:\lambda}^{(t)}, \mathbf{x}_{2:\lambda}^{(t)}, \ldots, \mathbf{x}_{\mu:\lambda}^{(t)}$

$$\mathbf{p}_{t+1} = (1-\alpha)\mathbf{p}_t + \alpha \frac{1}{\mu} \sum_{k=1}^{\mu} \mathbf{x}_{k:\lambda}^{(t)}$$

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

Extensiones Básicas de PBIL: Actualizar el vector de probabilidades teniendo en cuenta los peores elementos.

- ***Moverse hacia el complemento del peor.***
- ***Dará problemas durante estados avanzados de la búsqueda, ya que el mejor y el peor se parecen.***
- ***Una buena opción es utilizar sólo los bits del peor que difieran de los del mejor.***

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

```
****** Initialize Probability Vector ******
for i :=1 to LENGTH do P[i] = 0.5;

while (NOT termination condition)
        ***** Generate Samples *****
        for i :=1 to SAMPLES do
                sample_vectors[i] := generate_sample_vector_according_to_probabilities (P);
                evaluations[i] :=Evaluate_Solution (sample[i]);

        best_vector := find_vector_with_best_evaluation (sample_vectors, evaluations);
        worst_vector := find_vector_with_worst_evaluation (sample_vectors, evaluations);

        ***** Update Probability Vector towards best solution *****
        for i :=1 to LENGTH do
                P[i] := P[i] * (1.0 - LR) + best_vector[i] * (LR);


        ***** Update Probability Away from Worst Solution *****
        for i :=1 to LENGTH do
                if (best_vector[i] ≠ worst_vector[i]) then
                        P[i] := P[i] * (1.0 - NEGATIVE_LR) + best_vector[i] * (NEGATIVE_LR);


        ***** Mutate Probability Vector *****
        for i :=1 to LENGTH do
                if (random (0,1) < MUT_PROBABILITY) then
                        if (random (0,1) > 0.5) then mutate_direction := 1
                        else mutate_direction := 0;
                        P[i] := P[i] * (1.0 - MUT_SHIFT) + mutate_direction * (MUT_SHIFT);
```

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

USER DEFINED CONSTANTS (Values Used in this Study):

SAMPLES: the number of vectors generated before update of the probability vector (100).

LR: the learning rate, how fast to exploit the search performed (0.1).

NEGATIVE_LR: the negative learning rate, how much to learn from negative examples (PBIL = 0.075, EGA = 0.0).

LENGTH: the number of bits in a generated vector (problem specific).

MUT_PROBABILITY: the probability for a mutation occurring in each position (0.02).

MUT_SHIFT: the amount a mutation alters the value in the bit position (0.05).

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

OTRAS Extensiones:

- ***PBIL Distribuido***

M. Schmid, K. Kristensen, T.R. Jensen, Adding Genetics to Standard PBIL Algorithm. Proc. CEC99, Washington, 1999, 1527-1534.

- ***PBIL Continuo con Distribuciones Gaussianas (PBILc)***

M. Sebag, A. Ducoulombier, Extended Popula-tion-Based Incremental Learning to Continuous Search Spaces. 5th. Int. Conf. Parallel Problem Solving from nature – PPSN V, 1998, A.E. Eiben, T.Bäck, M. Schoenauer, H-P. Schwefel (Eds.), Springer-Verlag, LCNC-418-427.

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

Extensión del Modelo PBIL con actualizaciones basadas en la comparación de cromosomas.

Compact GA: Ref. G. R. Harik, F. G. Lobo, D.E. Goldberg, The Compact Genetic Algorithm. IEEE Transactions on Evolutionary Computation Vol. 3, No. 2, 1999, 287-297.

Compact-GA: Basado en la comparación de dos cromosomas

Compact-GA extendido: Basados en la comparación de poblaciones

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

```
1) initialize probability vector
   for i := 1 to ℓ do p[i] := 0.5;

2) generate two individuals from the vector
   a := generate(p);
   b := generate(p);

3) let them compete
   winner, loser := compete(a, b);

4) update the probability vector towards the better one
   for i := 1 to ℓ do
      if winner[i] ≠ loser[i] then
         if winner[i] = 1 then p[i] := p[i] + 1/n
                          else p[i] := p[i] − 1/n;

5) check if the vector has converged
   for i := 1 to ℓ do
      if p[i] > 0 and p[i] < 1 then
         return to step 2;

6) p represents the final solution


compact GA parameters:
   n: population size.
   ℓ: chromosome length.
```

Fig. 2. Pseudocode of the compact GA.

Compact-GA: Basado en la comparación de dos cromosomas

2. EVOLUCIÓN DE POBLACIONES MEDIANTE APRENDIZAJE INCREMENTAL: PBIL

```
1) generate s individuals from the vector and store them in S
   for i := 1 to s do
       S[i] := generate(p);

2) rearrange S so that S[1] is the individual with higher fitness

3) let S[1] compete with the other individuals
   for j := 2 to s do
   begin
       winner, loser := compete(S[1], S[j]);
       update probability vector (step 4 of cGA code)
   end
```

Compact-GA extendido: Basados en la comparación de poblaciones

Fig. 5. Modification of the compact GA that implements tournament selection of size s. This would replace steps 2–4 of the cGA code.

3. EXTENSIÓN DE LOS MODELOS PROBABILÍSTICOS BASADOS EN LA TEORÍA DE BLOQUES: EDAs

Los modelos probabilísticos se extienden mediante modelos explícitos de construcción de soluciones prometedoras basadas en la teoría de bloques.

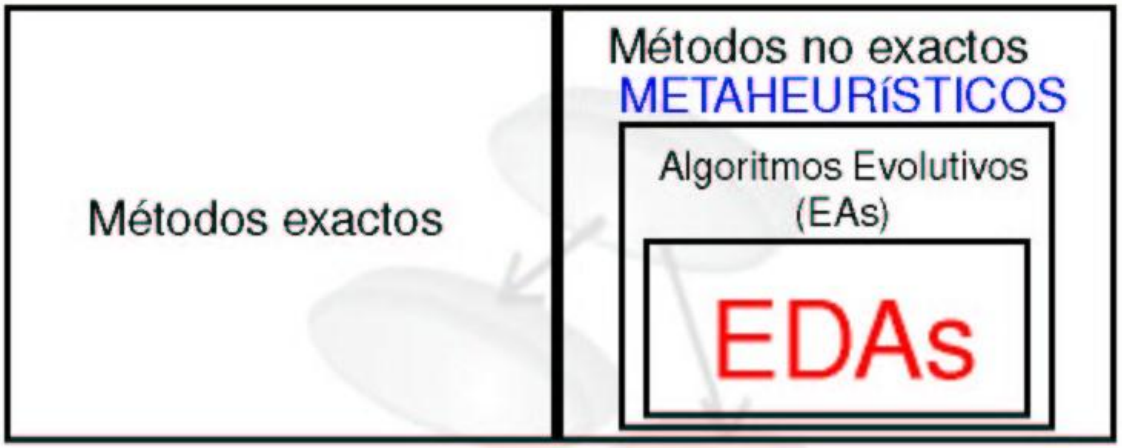
Las Redes Bayesianas se utilizan como herramienta para modelar soluciones prometedoras.

Estimation of Distribution Algorithms (EDAs).

P. Larrañaga, J.A. Lozano (Eds.), Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, 2001. ISBN 0-7923-7466-5.

3. EXTENSIÓN DE LOS MODELOS PROBABILÍSTICOS BASADOS EN LA TEORÍA DE BLOQUES: EDAs

EDAs: ¿Qué son?



¿Cómo generan nuevos individuos?

3. EXTENSIÓN DE LOS MODELOS PROBABILÍSTICOS BASADOS EN LA TEORÍA DE BLOQUES: EDAs

Pseudocódigo para un EDA general

EDA

$D_0 \leftarrow$ Generar aleatoriamente M individuos (población inicial)

Repetir para $t = 1, 2, \ldots$ hasta cumplir criterio parada

$D_{t-1}^{Se} \leftarrow$ Seleccionar $N \leq M$ individuos de D_{t-1} de acuerdo al método de selección

$p_t(\mathbf{x}) = p(\mathbf{x}|D_{t-1}^{Se}) \leftarrow$ Estimar la distribución de probabilidad conjunta a partir de los individuos seleccionados

$D_t \leftarrow$ Muestrear M individuos (la nueva población) de $p_t(\mathbf{x})$

3. EXTENSIÓN DE LOS MODELOS PROBABILÍSTICOS BASADOS EN LA TEORÍA DE BLOQUES: EDAs

El algoritmo UMDA$_c$ con selección por torneo

UMDA$_c$

$\mathbf{X}^0 = (X_1^0, \ldots, X_n^0) \sim \prod_{i=1}^{n} f_{\mathcal{N}(\mu_i^0, \sigma_i^0)}$ → Obtener aleatoriamente μ_i^0 y σ_i^0 para cada una de las variables

mientras no convergencia **hacer**

Repetir para $1 \leq j \leq N$

Muestreando $\mathbf{X}^t$ obtener 2 individuos:

$$\mathbf{x}_{1,j}^t = (x_{1,j}^{1,t}, \ldots, x_{1,j}^{n,t}) \, , \, \mathbf{x}_{2,j}^t = (x_{2,j}^{1,t}, \ldots, x_{2,j}^{n,t})$$

Evaluar $\mathbf{x}_{1,j}^t$, $\mathbf{x}_{2,j}^t$

Seleccionar el mejor de ellos:

$$\mathbf{x}_{(1:2),j}^t = argmin_{\mathbf{x}}\{G(\mathbf{x}_{1,j}^t), G(\mathbf{x}_{2,j}^t)\}$$

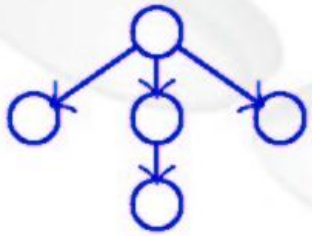
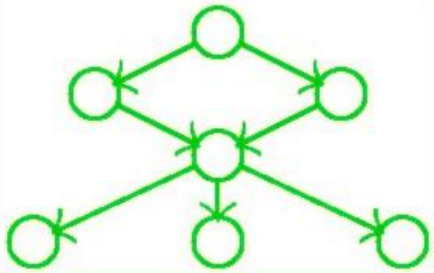
Repetir para $1 \leq i \leq n$

Estimar los parámetros de las nuevas funciones de densidad:

$$\mu_i^{t+1} = \frac{\sum_{j=1}^{N} x_{(1:2),j}^{i,t}}{N}, \quad \sigma_i^{t+1} = \sqrt{\frac{\sum_{j=1}^{N} (x_{(1:2),j}^{i,t} - \mu_i^{t+1})^2}{N}}$$

3. EXTENSIÓN DE LOS MODELOS PROBABILÍSTICOS BASADOS EN LA TEORÍA DE BLOQUES: EDAs

Clasificación de los EDAs

Modelo	Algoritmo	Representación gráfica	Factorización de $p_t(\mathbf{x})$
Independencia	UMDA		$p_t(\mathbf{x})=\prod_{i=1}^{n} p_t(x_i)$
Dependencias a pares	MIMIC		$p_t(\mathbf{x})=\prod_{i=1}^{n} p_t(x_i \mid x_{j(i)})$
	TREE		
Dependencias múltiples	EBNA		$p_t(\mathbf{x})=\prod_{i=1}^{n} p_t(x_i \mid \mathbf{pa}_i)$

COMENTARIOS FINALES

En este tema se presentan nuevas propuestas de Evolución de Poblaciones, los algoritmos EDA.

Dos temas abiertos y de estudio en EDA son:

- El uso de "codificación real/variables reales" no está resuelto satisfactoriamente.

- La escalabilidad cuando los cromosomas crecen de tamaño: la red bayesiana que hay que aprender para cada población es muy grande, esto puede ocasionar problemas de convergencia/eficiencia.

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