GALib

C++ Genetic Algorithm Library

Massachusetts Institute of Technology (MIT)

Matthew Wall (the Author)

Licencia

- GNU para actividades no comerciales
- Permite modificar el código fuente
- Para propósitos lucrativos, se distribuye una licencia diferente

Caracteristicas Generales

- **Multiplataforma**: Unix, Linux, MacOS, Win
- **PVM** en varias máquinas o multiples CPU´s
- **Parametrizable** por linea de comandos, archivos o en el código
- **Estadisticas** “on-line”, “off-line” en archivos estructurados

Overview

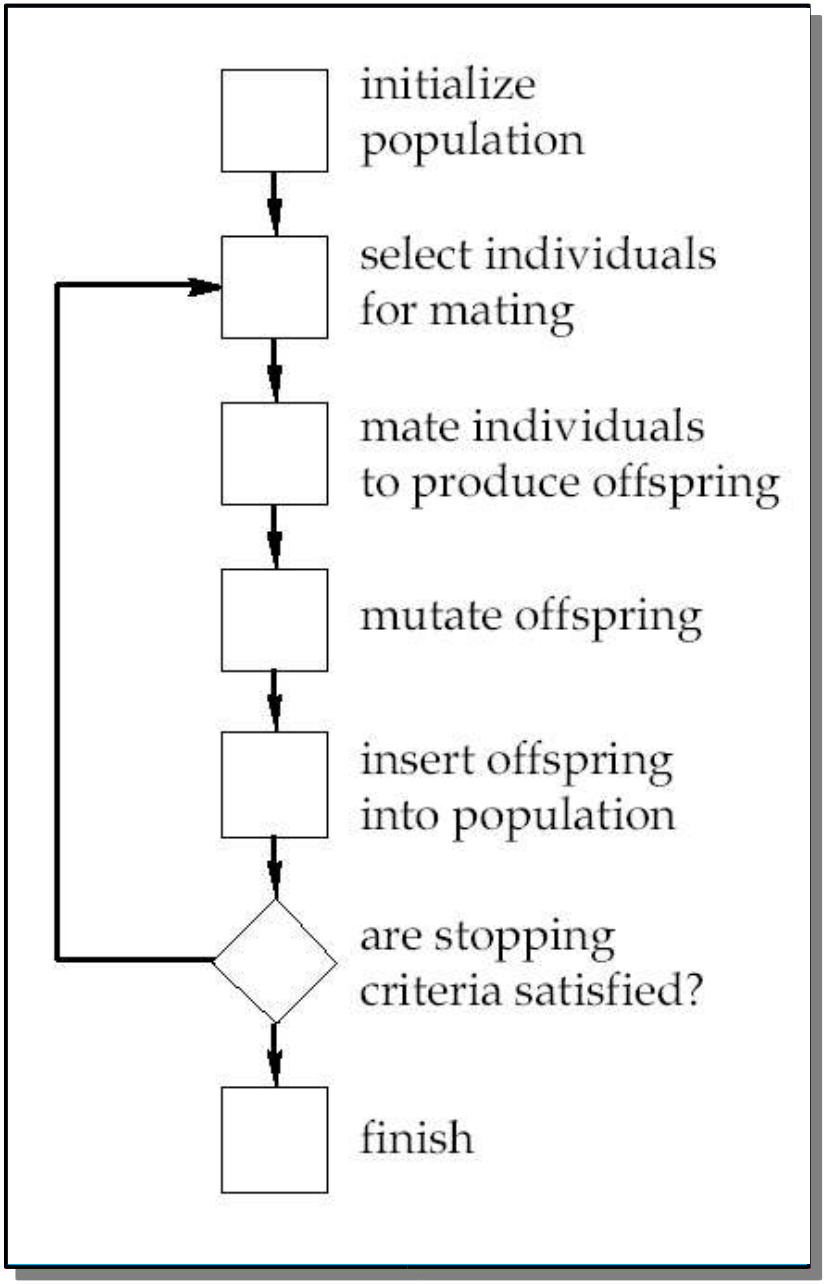
Se trabaja con 2 clases:

- El genoma (cromosoma)
- El algoritmo genetico

Se define una funcion objetivo

Se utiliza el algoritmo

Estructura del AG

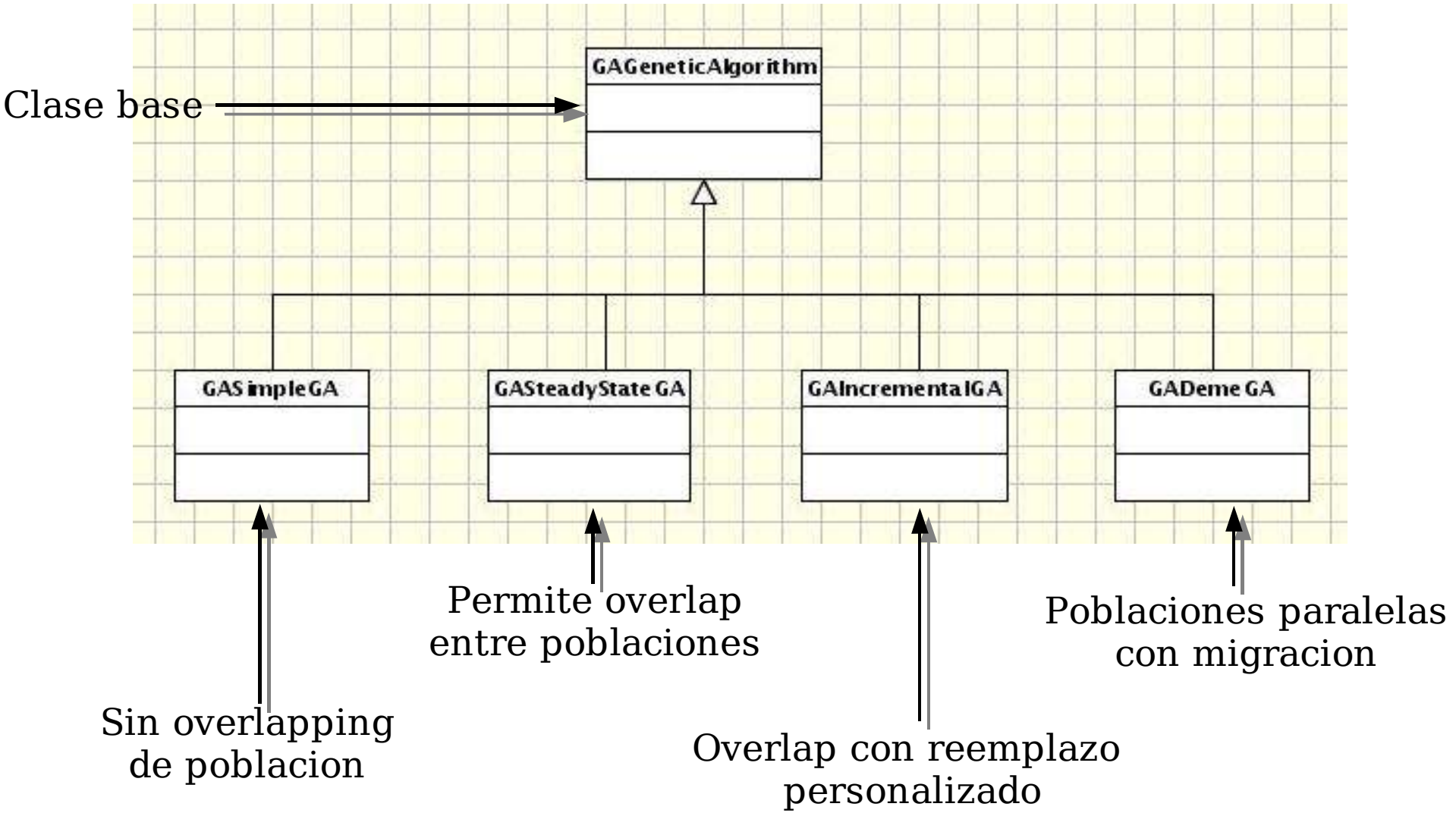


TRES PASOS A SEGUIR

1. Definir la representacion
2. Definir los operadores geneticos
3. Definir la funcion objetivo

La libreria tiene varias implementaciones para los 2 primeros puntos

Escoger el Algoritmo Genetico



El AG Contiene:

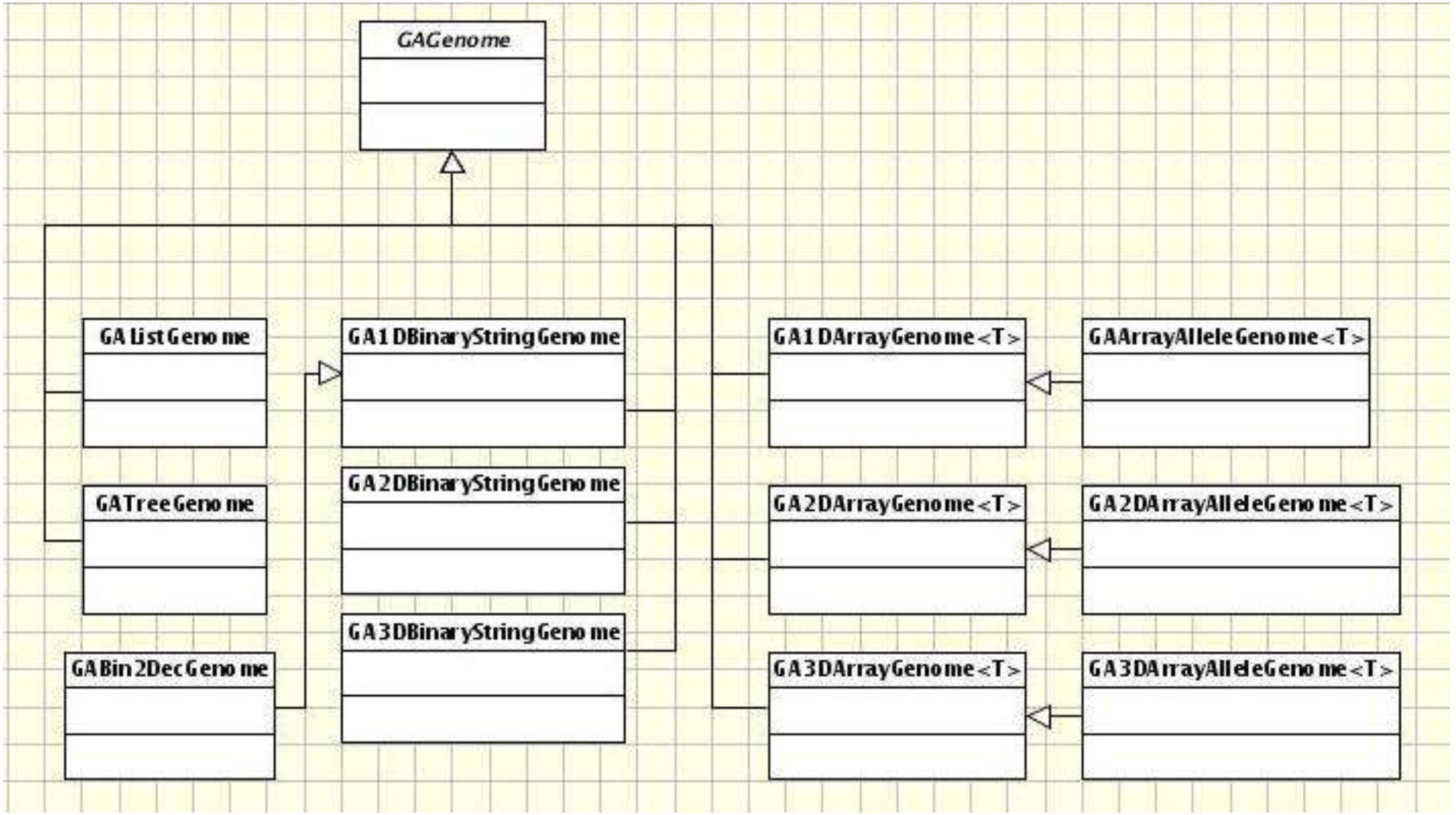
- Las estadisticas generales y de operadores
- Estrategia de reemplazo
- Parametros de ejecucion
- La poblacion y sus cromosomas
- Criterio de terminacion

Todos son personalizables.

Escoger la Representacion

- Hacerlo apropiadamente hace parte del arte de los algoritmos geneticos
- Usar una representacion minima pero completa
- Permitir soluciones infactibles?

Representacion de Cromosomas



Un programa tipico:

```
float Objective(GAGenome&) {
  // Aqui se escribe la funcion objetivo
}
void main(){
  //Crear un genoma:
  GA2DBinaryStringGenome genome(width, height, Objective);
  GASimpleGA ga(genome);  // Crear el AG
  ga.evolve();  // Evolucionar el AG
  cout << ga.statistics() << endl;  // Ver los resultados
}
```

Parametros Configurables

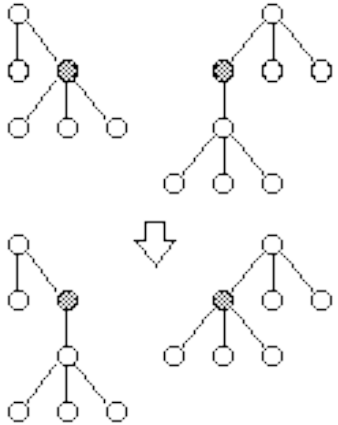
```
  ga.populationSize(popsize);
  ga.nGenerations(ngen);
  ga.pMutation(pmut);
  ga.pCrossover(pcross);

//Alternativamente:
  GASteadyStateGA ga(genome);
  ga.parameters("settings.txt");
  ga.parameters(argc, argv);
  ga.evolve();
  cout << ga.statistics() << endl;
```

Operadores Geneticos

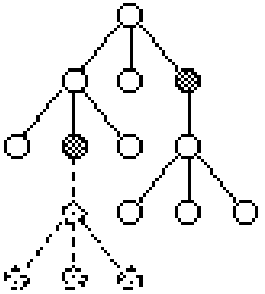
- Todo cromosoma tiene 3 operadores primarios:
  - Inicializacion
  - Mutacion
  - Cruce

```
GA1DBinaryStringGenome genome(l, Objective);
genome.crossover(GA1DBinaryStringGenome::UniformCrossover);
//...
GATournamentSelector sel;
ga.selector(sel);
```

Operadores

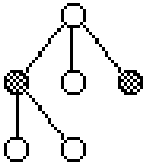


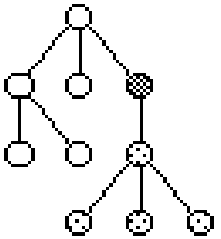
Cruce de arbol
punto simple



Mutacion por
Intercambio
de un subarbol

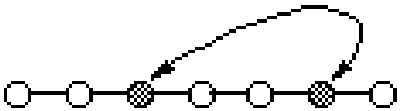
Mutacion por
intercambio
de nodos



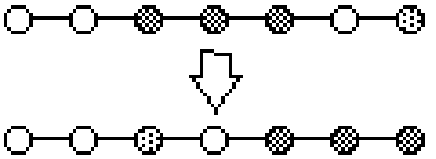


Mutacion por
destruccion de
un subarbol

Mutacion por
destruccion en
una lista

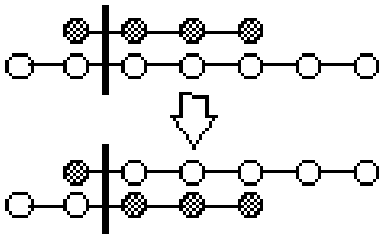


Mutacion por
intercambio en
la lista

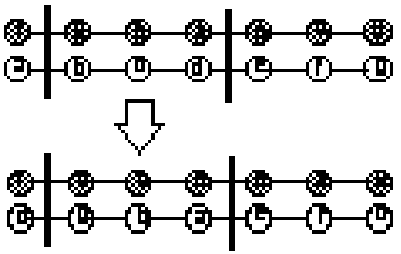


Mutacion por
Intercambio de
una sublista

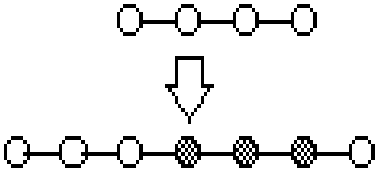
Operadores



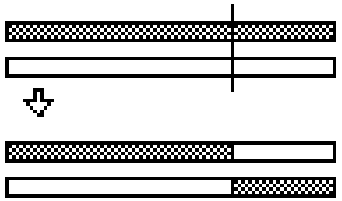
Cruce de lista de punto unico



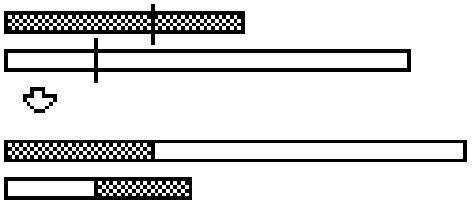
Cruce de lista basado en ordenamietno



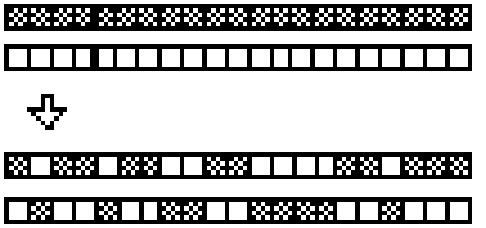
Mutacion de lista por generacion



Cruce de unico punto en arreglos con longitud fija



Cruce de unico punto en arreglos de longitud variable



Cruce de arreglos uniforme

La Poblacion

- Es el contenedor de los cromosomas
- Tiene un inicializador y un evaluador
- Mantiene informacion estadistica
- Tiene definido el operador de seleccion

Funcion Objetivo y Fitness

- Es totalmente definida por el programador
- Se puede realizar evaluacion para individuos o evaluacion para poblaciones
- Una funcion objetivo poblacional puede estar compuesta por funciones objetivo individuales

Funcion Objetivo y Fitness

```
float Objective(GAGenome& g){
  GA1DBinaryStringGenome & genome = (GA1DBinaryStringGenome &)g;
  float pesos[]={2.0,3.0,4.0,2.5,3.8,1.9,2.6,0.7,4.2,1.40};
  float valores[]={10.0,21.0,1.0,14.0,3.0,1.0,23.0,12.0,15.0,19.0};
  float pesoMaximo = 8.00, valorTotal = 0.0, pesoTotal = 0.0;
  for(int i = 0; i < genome.length(); i++)
  {
    valorTotal += genome.gene(i)*valores[i];
    pesoTotal += genome.gene(i)*pesos[i];
  }
  if(pesoTotal > pesoMaximo)  return 0;
  else return valorTotal;
}
```

Estadisticas

0100001101

250 # current generation

1 # current convergence

6500 # number of selections since initialization

5626 # number of crossovers since initialization

625 # number of mutations since initialization

6250 # number of replacements since initialization

5739 # number of genome evaluations since initialization

251 # number of population evaluations since initialization

75 # maximum score since initialization

0 # minimum score since initialization

73.8134 # average of all scores ('on-line' performance)

75 # average of maximum scores ('off-line' performance)

73.256 # average of minimum scores ('off-line' performance)