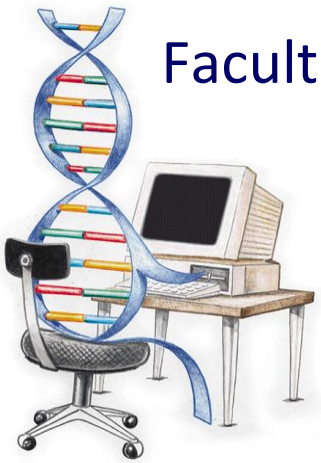
ALGORITMOS EVOLUTIVOS
Curso 2023

Tema 6: Implementación de AE

Centro de Cálculo, Instituto de Computación
Facultad de Ingeniería, Universidad de la República, Uruguay



cecal

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Implementación de algoritmos evolutivos

Contenido

Conceptos sobre implementación

- Las metaheurísticas son métodos genéricos de optimización
- Toda implementación de una metaheurística debe ser independiente del problema que se resuelve
- Por lo tanto, se debe realizar una separación en entidades:
  - *Metaheurística:* es el motor de resolución
  - *Problema:* es abstracto y se instancia para su resolución

AE y orientación a objetos

- La utilización del paradigma de orientación a objetos aporta multiples ventajas:
  - Reusabilidad
  - Legibilidad del código
  - Capacidad de abstracción e instanciación simple de problemas y codificación
  - Modularidad
- Sin embargo no contribuye a un objetivo fundamental de la implementación: la eficiencia computacional
- Necesariamente debe existir un compromiso entre ambos aspectos
  - Implementaciones no orientadas a objetos pueden ser las apropiadas para resolver problemas de tiempo real

Implementación de AE

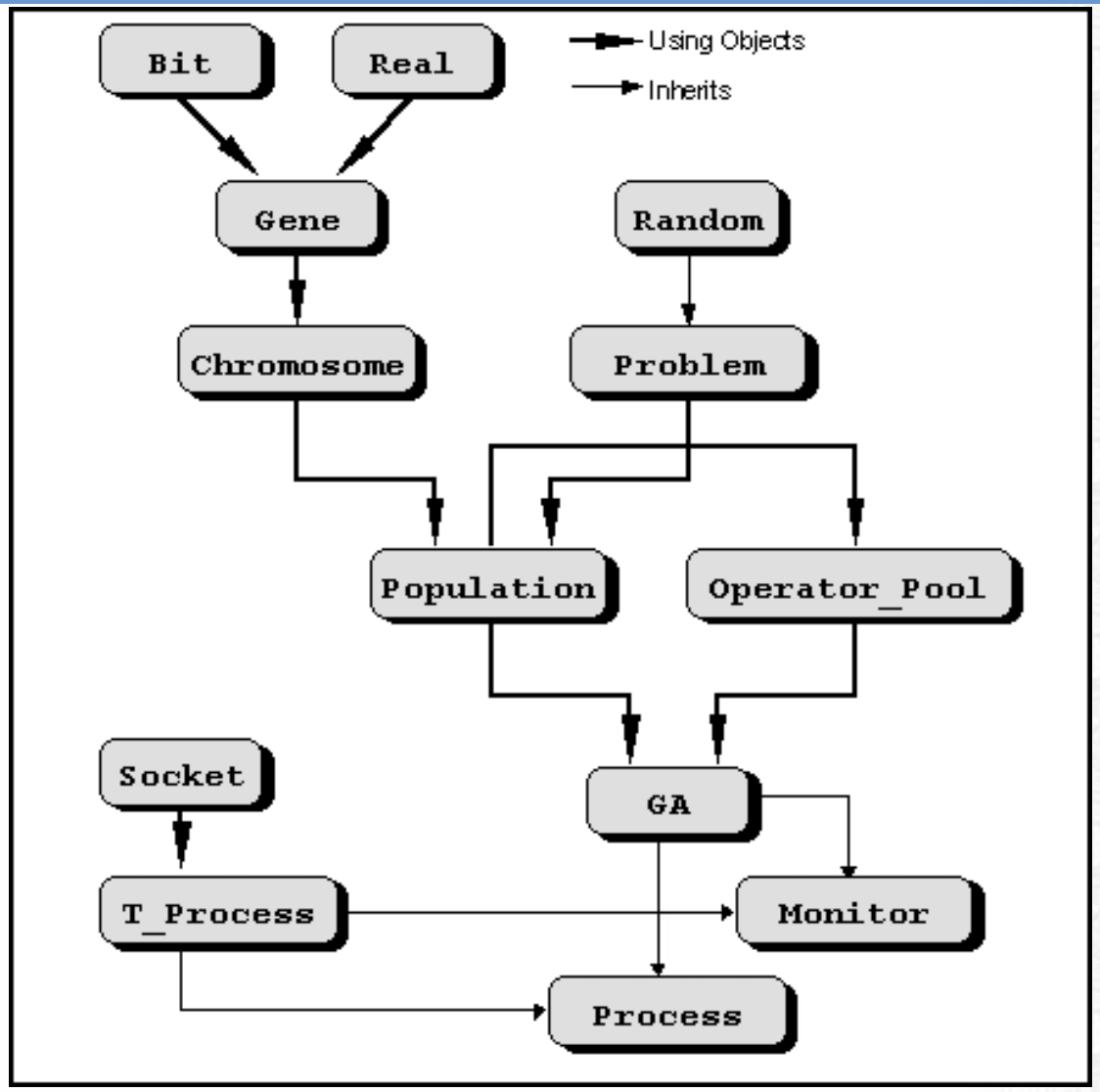
AE y orientación a objetos

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Las entidades de la implementación de un AE se mapearán a clases en un lenguaje orientado a objetos.
- Para permitir la incorporación de nuevos operadores es necesario separar completamente la clase de individuos de la de operadores.
- Los individuos también deben estar separados del genotipo para permitir su manipulación en forma independiente.
- También es necesario considerar una clase particular para encapsular el problema que se quiere resolver.
- A partir de estos postulados, se presenta a continuación la propuesta de Alba y Troya (1997) para la implementación de AE (disponible en http://neo.lcc.uma.es/TutorialEA/semEC/main.html)

AE orientado a objetos

Propuesta de jerarquía de clases (Alba y Troya, 1997)



Implementación de AE

Aspectos importantes

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- No utilizar un tamaño fijo para la población ni para el pool de operadores.
  - Dar flexibilidad al uso de operadores no convencionales.
- Evitar evaluaciones múltiples de la misma solución, fundamentalmente para poblaciones numerosas y problemas complejos.
- Dejar la mayor parte posible de “código abierto” para instanciar por el usuario
- Para lograr una implementación eficiente, no utilizar lenguajes interpretados, ni de programación lógica o funcional

Implementación de AE

Una (breve) lista

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Implementación	lenguaje	año	referencia
aealib	Java	2001	Gorges-Schleuter
AForge.NET	C#		P. Adamidis
ANNEvolve	C, Phyton		
ASPARAGOS	C	1991	Baeck
copdeb	C	1996	(Michalewicz,
DEAP	Phyton		Cantú-Paz
DEAP			Jankovic
DGENESIS	C/C++	1994	Adcock
DGPF	Java		Perone
DREAM	Java		Mc Nab
EAlib	C++		Meffert
EasyLocal++	C++		Luke
ECF	C++		Merelo
ECJ	Java		Merelo
Egglets	Java		Faulpel
EO/ ParadisEO/ MOEO/ PEO	C++		Dolan
EO/EOlib	C++	2002	Wall
Evo	C#		Levine
Evocosm	C++		Hunter
EvoJ	JAva		Mathworks
FOM	Java		
FORTRAN GA driver	FORTRAN	1999	Gagné
Framework for distributed EAs in computational grids		2010	Poli
FREVO	Java	2012	
GA Playground	Java		Rybarski
GAEDALib	C++	2007	Streichert

Implementación de AE

Una (breve) lista

Implementación	lenguaje	año	referencia
GAGS	C++	1994	Jakobovic
GAJIT	Java		Krupowicz
GAlib	C++		
GALOPPS	C	2002	
GAUL	C++	2003	Ventura
GECO	Lisp	2011	Arenas
Genepool	Ocaml		Ladd
Genesis	C	1984	Smith
GENEsYs	C	1992	Tsvetkov
Genetic Algorithm Library	C++	2008	De Rainville
Genetic and Evolutionary Algorithm Toolbox	Matlab		Rummler
GENOCOP3	C	1996	Hu
HeuristicLab	C#		Jiang
HFC		2005	De la Fuente
Java Grid-enabled Genetic Algorithm	Java	2008	
JavaEvA	Java		Filipic
JavaEvA/EvA/EvA2	Java		Ineichen
JCLEC	Java		Leon
jDeal	Java		Emmerich
JEAF	Java	2010	Alba
jGal	Java		Talbi
jGAP	Java		
jMetal	Java		
Mallba	C++	2002	
Metco	C++	2009	Martin Lukasiewycz and Helwig
Modular MO framework	C++	2012	Geihs

Implementación de AE

Una (breve) lista

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Implementación	lenguaje	año	referencia
Modular MO framework	C++	2012	Geihs
MOEA	Java		Garrett
MOS	C++	2010	Ramírez
OpenBeagle	C++	2002	Mueller
OPT4J	Java		Limmer
OptiGen	C++	2005	
Optimization Algorithm Toolkit	Java		
pALS	Java		Goodman
ParadisEO	C++		Carroll
Parallel EC framework for SO and MO parallel		2009	Sobe
pCMALib	FORTRAN	2009	Williams
PGAPack	C		Rubinsteyn
PIPE	C	1997	Salustowicz
PMF	C++	2010	Li
Pyevlove	Phyton	2009	Di Gaspero and Schaerf
pygene	Phyton	2005	Kronfeld et al.
Sugal	C		Parejo et al.
TEA	C++	2001	Brownlee
TinyGP	C		
Watchmaker	Java		Cahon et al.
C	10		
C++	20		
Java	24		

La bibilioteca Mallba

- El proyecto Mallba fue realizado por las universidades de MÁlaga, La Laguna y Barcelona.
- Mallba es una biblioteca de esqueletos algorítmicos para optimización combinatoria que incluye métodos exactos, heurísticos e híbridos.
- Facilita el pasaje de implementaciones secuenciales a paralelas, ya que el usuario provee una implementación secuencial y la biblioteca le provee (automáticamente) las implementaciones paralelas.
- La arquitectura es flexible y extensible, por lo cual pueden agregarse nuevos esqueletos o implementar hibridaciones de modo relativamente simple.

La bibilioteca Mallba

Algoritmos de optimización

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Los métodos de resolución genéricos implementados en Mallba incluyen:
  - Genetic Algorithm (GA)
  - Simulated Annealing (SA)
  - Algoritmo CHC
  - Evolution Strategy (ES)
  - Ant Colony Optimization (ACO)
  - Híbridos GA+SA y GASA
  - Cooperative Local Search (CLS)
  - Particle Swarm Optimization (PSO)
- Las características particulares del problema deben ser provistas por el usuario.

La bibilioteca Mallba

Diseño y clases

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Las *clases provistas (.pro.cc)* implementan aspectos internos de cada esqueleto, de forma independiente a las particularidades de los problemas. Salvo que se deseen introducir cambios estructurales específicos en los algoritmos, las clases provistas no deben ser modificadas por el usuario
- Las *clases requeridas (.req.cc)* especifican la información vinculada directamente al problema que se intenta resolver, así como la interacción con el método de resolución. Deben ser instanciadas por el usuario para resolver cada problema específico
- Esta separación conceptual permite que la interfaz de las clases requeridas esté fija, pero no se provea de ninguna implementación para que las clases provistas puedan ser utilizadas en forma genérica

La bibilioteca Mallba

Algoritmos genéticos

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- El esqueleto del algoritmo genético requiere las clases Problem, Solution, Crossover y Mutation
- La clase Problem corresponde a la definición de la instancia del problema (el usuario debe implementar completamente esta clase para cada problema a resolver)
- La clase Solution corresponde a la definición de las soluciones factibles o no del problema. (el usuario debe implementar completamente esta clase, determinando las funciones de codificación y decodificación y todas las operaciones requeridas sobre las soluciones)
- Las clases Crossover y Mutation corresponden a las definiciones de los operadores de cruzamiento y mutación

La bibilioteca Mallba

Algoritmos genéticos

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Deben configurarse una serie de parámetros para permitir la ejecución del AG:
  - Número de ejecuciones independientes
  - Número de generaciones
  - Tamaño de la población
  - Número de hijos creados en cada generación
  - Estrategia de reemplazo (si se reemplazan padres por hijos o si solamente los nuevos hijos pueden ser padres)
  - Parámetros de la selección (parámetros de la selección de los padres y los hijos)
  - Parámetros de las operaciones: cruzamiento y mutación
- Se configuran en el archivo ".cfg", en el caso de AE, "newGA.cfg"

La bibilioteca Mallba

Algoritmos genéticos

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- MALLBA implementa varios métodos de selección:
  - Aleatorio, torneo, rueda de ruleta, ranking, best, worst.
- MALLBA incluye un benchmark de problemas resueltos como ejemplo:
  - OneMax, MaxSat, Sphere, VRP, DNAfa, entre otros.
- En los diversos ejemplos de MALLBA están implementados:
  - Operadores de cruzamiento tradicionales:
    - SPX, 2PX, UPX, cruzamientos para permutaciones (PMX, UX), otros.
  - Operadores de mutación:
    - Inversión de bits, mutaciones para permutaciones, otros.

La bibilioteca Mallba

Enlaces

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- La biblioteca MALLBA está disponible públicamente en

  neo.lcc.uma.es/mallba/easy-mallba/index.html

- Un spin-off actualizado, desarrollado y mantenido por nuestro grupo es Malva
- "The Malva Project: A framework for computational intelligence in C++", disponible públicamente en

  https://github.com/themalvaproject

Implementación de AE

Coloreo de mapas

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Teorema de los cuatro colores:
  - Problema a resolver: *colorear el mapa de barrios de Montevideo de acuerdo al teorema de los cuatro colores.*
- El mapa se encuentra disponible en el catálogo de datos abiertos:
  - https://catalogodatos.gub.uy/dataset/limites-barrios
  - 63 barrios
- Implementaciones en Malva, ECJ y WatchMaker disponibles en:
  - https://github.com/hpc-cecal-uy/cursoAE/

Implementación de AE

Coloreo de mapas

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Representación: tuplas de enteros de largo #regiones, donde cada gen tiene un valor en {0,1,2,3} representando al color asignado a esa región.
- Función de fitness:

$$F = -\sum_{i=1}^{\#B}\sum_{j=0}^{\#L_i} x_{i,j}$$

$$x_{i,j} = \begin{cases} 1 & \text{si limitan}(i,j) \text{ e igual\_color}(i,j) \\ 0 & \text{en caso contrario} \end{cases}$$

$$B = \{b_1, b_2 \dots b_n\} \text{ Conjunto de Barrios}$$

$$L_i = \{l_1, l_2 \dots l_k\} \text{ Barrios límitrofes al barrio } i$$

- Operadores:
  - Selección: torneo de tamaño 2
  - Cruzamiento: de dos puntos (2PX)
  - Mutación: uniforme en {0,1,2,3}
- Condición de parada: *fitness* = 0 o 10.000 generaciones

ECJ Framework

Introducción

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- ECJ es una biblioteca de computación evolutiva escrita en Java desarrollada en la Universidad George Mason.
- Ampliamente utilizada para programación genética, también soporta AE y otras metaheurísticas.
- Disponible en: https://cs.gmu.edu/~eclab/projects/ecj/
  - Última versión (publicada en julio 2018): **ECJ 26.**
  - La versión 25 no es compatible hacia atrás con las versiones previas.
- Documentación y soporte:
  - Manual completo (más de 280 páginas).
  - Tutoriales
  - Problemas de ejemplo
  - Lista de correos activa para consultas

ECJ Framework

Características y funcionalidades

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Múltiples representaciones para individuos ya implementadas.
- Diversos operadores de selección, cruzamiento y mutación ya implementados.
- Soporta algoritmos multiobjetivo (NSGA-II, SPEA2, etc.).
- Manejo de subpoblaciones y migraciones.
- Ofrece paralelismo:
  - Multithreading para la evaluación de las soluciones.
  - Distribuído (TCP/IP) para el manejo de islas.
- Posibilidad de hacer Checkpointing (pausar/reiniciar una ejecución).

Pasaje de parámetros

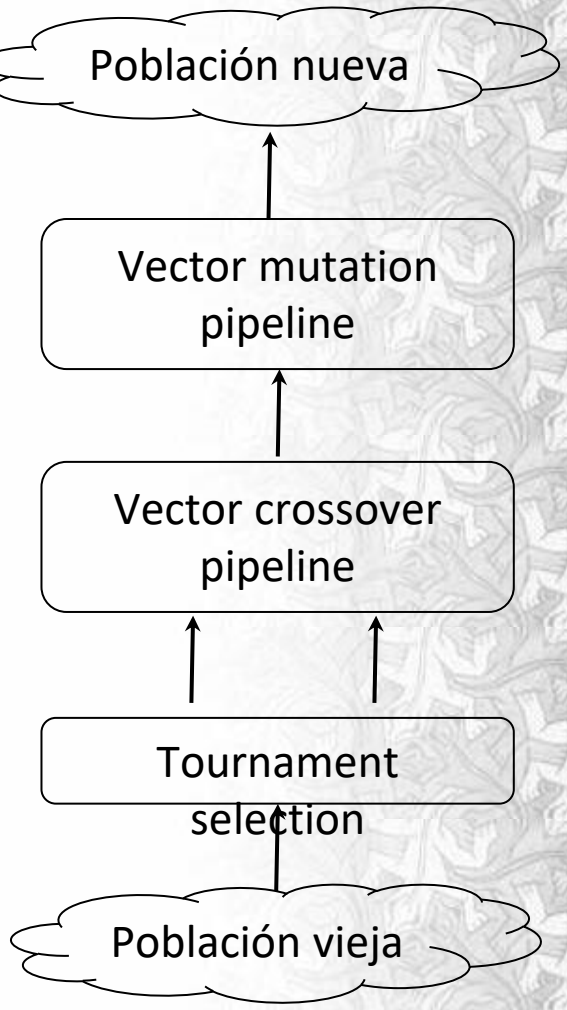
- ECJ utiliza un gran número de parámetros definidos por el usuario para tomar decisiones en tiempo de ejecución.
- Los parámetros se almacenan en un archivo especificado al momento de la ejecución.
- Estos archivos pueden derivar de otros archivos de parámetros, sobrescribiendo los parámetros de acuerdo a la jerarquía establecida.
- Los parámetros se especifican mediante pares clave-valor:
  - Parameter name = parameter value
- Se pueden definir parámetros al ejecutar ECJ de forma que sobreescriban aquellos definidos en los archivos.

Breeding pipelines

- Un BreedingPipeline es el mecanismo básico para generar los individuos de la nueva población a partir de la población anterior.
- Toman Individuals de una o más BreedingSources (que pueden ser SelectionMethods u otros BreedingPipelines) y producen un nuevo individuo.
- Los BreedingPipelines están diseñados para ser fácilmente encadenados y para establecer jerarquías entre ellos, lo que permite una mayor flexibilidad a la hora de diseñar los operadores evolutivos.

Breeding pipelines - Ejemplo

- Definición en el archivo de parámetros:
  - pop.subpop.0.species.pipe = ec.vector.VectorMutationPipeline
  - pop.subpop.0.species.pipe.source.0 = ec.vector.VectorCrossoverPipeline
  - pop.subpop.0.species.pipe.source.0.source.0 = ec.select.TournamentSelection
  - pop.subpop.0.species.pipe.source.0.source.1 = same
  - pop.subpop.0.species.pipe.source.0.source.0.size = 2



Breeding pipelines - Ejemplo

- El operador específico a ejecutar y sus parámetros pertenecen a la clase Species y no al pipeline.
- Los operadores a aplicar dependen de la especie de los individuos. Continuando el ejemplo se definen los siguientes parámetros:
  - pop.subpop.0.species = ec.vector.FloatVectorSpecies
  - pop.subpop.0.species.crossover-type = one
  - pop.subpop.0.species.mutation-prob = 0.1
  - pop.subpop.0.species.mutation-type = gauss
  - pop.subpop.0.species.mutation-stdev = 0.01

ECJ Framework

Resumen

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Como mínimo para resolver un problema usando ECJ hay que definir:
  - La clase Java que define al problema, con su función de fitness.
  - El archivo de parámetros:
    - Apuntando a la clase Java del problema a resolver.
    - Definiendo la representación de los individuos, los pipelines a utilizar, los operadores específicos, sus probabilidades y parámetros, entre otros.
- Si se desea utilizar una representación distinta a las que ya vienen incorporadas en ECJ, es necesario crear subclases de Species e Individual o modificar alguna existente.
- Puede ser deseable también crear otros operadores distintos a los que ya incorpora ECJ o estadísticas adaptadas al problema a resolver.

Otras bibliotecas de AE

jMetal

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- jMetal es una biblioteca para metaheurísticas (en especial algoritmos evolutivos) multiobjetivo
- Desarrollada en Java
- Implementa múltiples MOEAs, incluyendo:
  - NSGA-II, SPEA-2, MOCell, MOCHC, otros.
- Disponible para descarga en jmetal.sourceforge.net
- Versión en C++: jMetalcpp
- Disponible para descarga en jmetalcpp.sourceforge.net

Otras bibliotecas de AE

ParadisEO y WatchMaker

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- ParadisEO: framework en C++ para el uso de metaheuristicas
  - Orientado a objetos, portable en Windows, Unix y MacOS.
  - Incluye AEs, PSO, TS, ILS, algoritmos híbridos, y otros.
  - Disponible para descarga en paradiseo.gforge.inria.fr
- WatchMaker: biblioteca de computación evolutiva en Java.
  - JAR disponible en: http://watchmaker.uncommons.org/
  - Código fuente en: https://github.com/dwdyer/watchmaker
  - Soporta multi-threading para la evaluación y modelo de islas.
  - Operadores evolutivos implementados para tipos de datos simples (string de bits, string de caracteres, arrays y listas).

Otras bibliotecas de AE

DEAP y HeuristicLab

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- DEAP: framework para AE en Python.
  - Soporta múltiples representacioness: List, Array, Set, Dictionary, Tree, Numpy Array, etc.
  - Implementa la evaluación paralela de soluciones según un modelo maestro-esclavo.
  - Soporta algoritmos multiobjetivo (NSGA-II, SPEA2, MO-CMA-ES).

Disponible en: https://github.com/DEAP/deap

- HeuristicLab: framework para metaheurísticas en C# usando .NET.
  - Implementa AE, MOEAs, PSO, SA, TS, y muchos otros.
  - Diversos problemas de ejemplo de distintas áreas.
  - Interfaz gráfica para las ejecuciones y para visualizar los resultados.

Disponible en: dev.heuristiclab.com