4

BIBLIOTECAS PARA ALGORITMOS GENÉTICOS

BIBLIOTECAS MAS RELEVANTES

- DEAP (Distributed Evolutionary Algorithms in Python).

- PyGAD (Python Genetic Algorithm Decimal) (Gad, 2024)

DEAP 1.4.3

DEAP 1.4.3

- DEAP (Distributed Evolutionary Algorithms in Python).
- Algoritmo genético que utiliza representación lista, matriz, conjunto, diccionario, árbol, matriz Numpy, etc.
- Programación genética utilizando árboles de prefijos tipo flexible y tipo fuerte.
- Estrategias de evolutivas (CMA-ES).
- Optimización de enjambre de partículas y Evolución diferencial.
- Optimización multiobjetivo (NSGA-II, NSGA-III, SPEA2, MO-CMA-ES).
- Funciona con mecanismos de paralelización como el multiprocesamiento.

DEAP 1.4.3

```python
from deap import base, creator, tools, algorithms
```

```python
# --- 2. Configuración de DEAP para Binario ---
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

toolbox = base.Toolbox()

# Atributo generador (gen): 0 o 1
toolbox.register("attr_bool", random.randint, 0, 1)

# Individuo: una lista de N_BITS_PER_VARIABLE * 2 bits
TOTAL_N_BITS = N_BITS_PER_VARIABLE * 2
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, n=TOTAL_N_BITS)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

# Operadores genéticos para binario
toolbox.register("evaluate", evalParaboloidBinary)
toolbox.register("mate", tools.cxTwoPoint) # Cruce de dos puntos
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05) # Probabilidad de mutar cada bit
toolbox.register("select", tools.selTournament, tournsize=3)
```

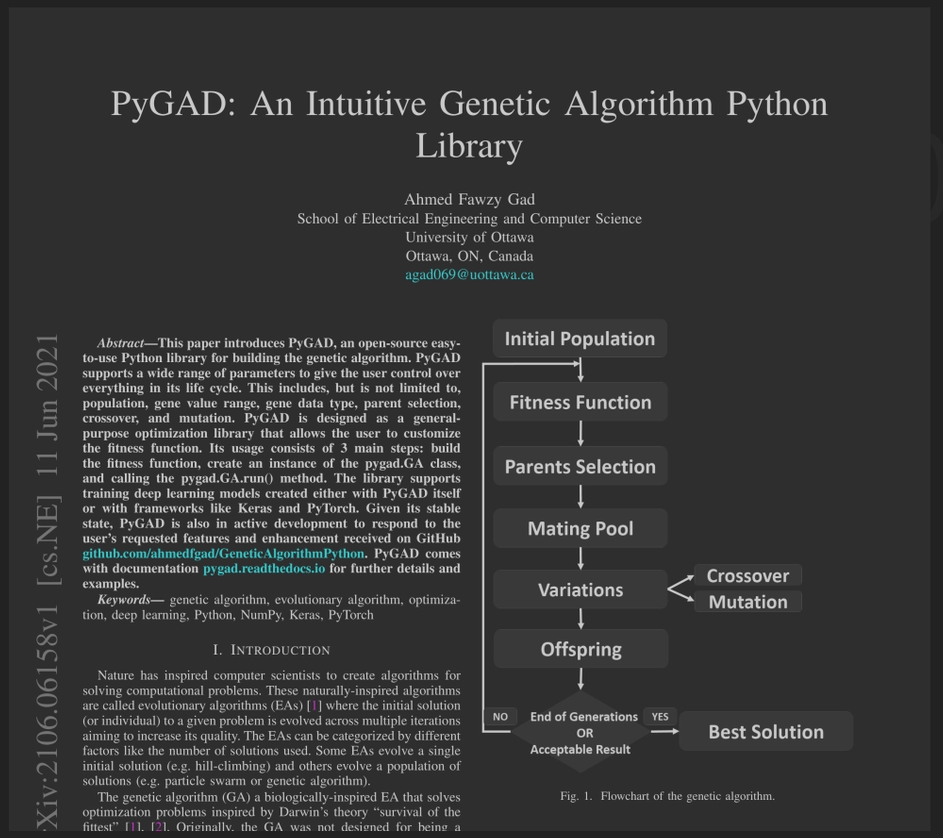
DEAP 1.4.3

```python
# --- 3. Ejecución del Algoritmo Genético ---
pop_size = 100
cruceprob = 0.7
mutacionprob = 0.2
num_generaciones = 100

pop = toolbox.population(n=pop_size)
hof = tools.HallOfFame(1)

pop, logbook = algorithms.eaSimple(pop, toolbox, cxpb=cruceprob, mutpb=mutacionprob, ngen=num_generaciones,
                                   halloffame=hof, verbose=True)
```

PYGAD

PYGAD

- PyGAD es un Biblioteca de Python de código abierto para construir el algoritmo genético.
- Problemas de optimización mono-objetivo y multiobjetivo.



PyGAD: An Intuitive Genetic Algorithm Python Library

Ahmed Fawzy Gad
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, ON, Canada
agad069@uottawa.ca

***Abstract*—This paper introduces PyGAD, an open-source easy-to-use Python library for building the genetic algorithm. PyGAD supports a wide range of parameters to give the user control over everything in its life cycle. This includes, but is not limited to, population, gene value range, gene data type, parent selection, crossover, and mutation. PyGAD is designed as a general-purpose optimization library that allows the user to customize the fitness function. Its usage consists of 3 main steps: build the fitness function, create an instance of the pygad.GA class, and calling the pygad.GA.run() method. The library supports training deep learning models created either with PyGAD itself or with frameworks like Keras and PyTorch. Given its stable state, PyGAD is also in active development to respond to the user's requested features and enhancement received on GitHub github.com/ahmedfgad/GeneticAlgorithmPython. PyGAD comes with documentation pygad.readthedocs.io for further details and examples.**

***Keywords*—** genetic algorithm, evolutionary algorithm, optimization, deep learning, Python, NumPy, Keras, PyTorch

I. Introduction

Nature has inspired computer scientists to create algorithms for solving computational problems. These naturally-inspired algorithms are called evolutionary algorithms (EAs) [1] where the initial solution (or individual) to a given problem is evolved across multiple iterations aiming to increase its quality. The EAs can be categorized by different factors like the number of solutions used. Some EAs evolve a single initial solution (e.g. hill-climbing) and others evolve a population of solutions (e.g. particle swarm or genetic algorithm).

The genetic algorithm (GA) a biologically-inspired EA that solves optimization problems inspired by Darwin's theory "survival of the fittest" [11, 12]. Originally, the GA was not designed for being a

Fig. 1. Flowchart of the genetic algorithm.

PYGAD

```
!pip install pygad
import pygad
```

```
# --- 2. Configuración del Algoritmo Genético con PyGAD (más parecido a DEAP) ---

# Parámetros del AG
num_generations = 100
num_parents_mating = 10
sol_per_pop = 50
num_genes = 2

# Límites para cada gen (x e y)
gene_space = [{'low': BOUND_LOW, 'high': BOUND_UP}, {'low': BOUND_LOW, 'high': BOUND_UP}]

# Parámetro K para la selección por torneo (tamaño del torneo)
K_tournament = 3
```


PyGAD

PYGAD

```python
# Inicializ. la instancia de GA
ga_instance = pygad.GA(num_generations=num_generations,
                       num_parents_mating=num_parents_mating,
                       fitness_func=fitness_func,
                       sol_per_pop=sol_per_pop,
                       num_genes=num_genes,
                       gene_type=float, # Genes flotantes
                       gene_space=gene_space,
                       parent_selection_type="tournament", # Selección por Torneo
                       K_tournament=K_tournament,          # Tamaño del torneo
                       keep_elitism=1, # Mantener al mejor individuo (similar a HallOfFame(1))
                                       # keep_parents=-1 también puede tener un efecto similar si
                                       # la selección de padres es fuerte.
                       crossover_type="two_points", # Cruce de dos puntos
                       mutation_type="random",      # Mutación: reemplaza gen con valor aleatorio en su rango
                       mutation_percent_genes=10)   # Probabilidad de mutar un gen
```

PYGAD

- Cruza multipunto flotante:

Padre 1 = [1.0, 2.5, 3.1, 4.8, 5.2]

Padre 2 = [0.5, 1.1, 6.3, 0.9, 2.0]

Padre 1: [1.0 | 2.5, 3.1 | 4.8, 5.2]

Padre 2: [0.5 | 1.1, 6.3 | 0.9, 2.0]

Hijo 1 = [1.0, 1.1, 6.3, 4.8, 5.2]

Hijo 2 = [0.5, 2.5, 3.1, 0.9, 2.0]

REFERENCIAS BIBLIOGRÁFICAS Y WEB (I)

- https://pypi.org/project/deap/
- https://pygad.readthedocs.io/en/latest/
- Gad, A. F. (2024). Pygad: An intuitive genetic algorithm python library. Multimedia tools and applications, 83(20), 58029-58042.