# NEORL Workshop
(NeuroEvolution Optimization with Reinforcement Learning)

Majdi I. Radaideh

Research Scientist,

Nuclear Science and Engineering,

Massachusetts Institute of Technology

*radaideh@mit.edu

*Presented at: Scientific Machine Learning for Nuclear Engineering Workshop*

American Nuclear Society
Oak Ridge/Knoxville Section

M&C 2021

Oct 3rd, 2021

Massachusetts Institute of Technology
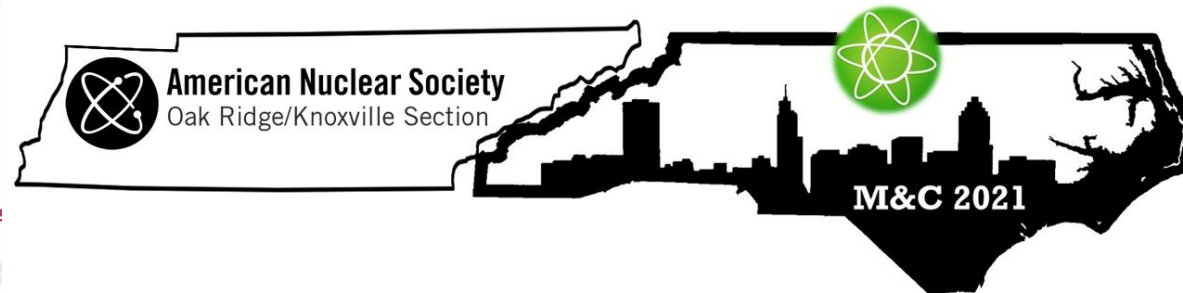
Lets Setup our Colab Early on

- Go to colab and start a new session.
  - https://colab.research.google.com/notebooks/intro.ipynb#recent=true
- Name the session the way you like (e.g. MC21_neorl.ipynb).
- Run the following commands:

```
#Remove unnecessary packages, install neorl, then test neorl quickly
!pip uninstall -y xarray arviz pymc3 tensorflow-probability pyerfa pyarrow kapre jax jaxlib datascience coveralls astropy albumentations
!pip install neorl
!neorl
```
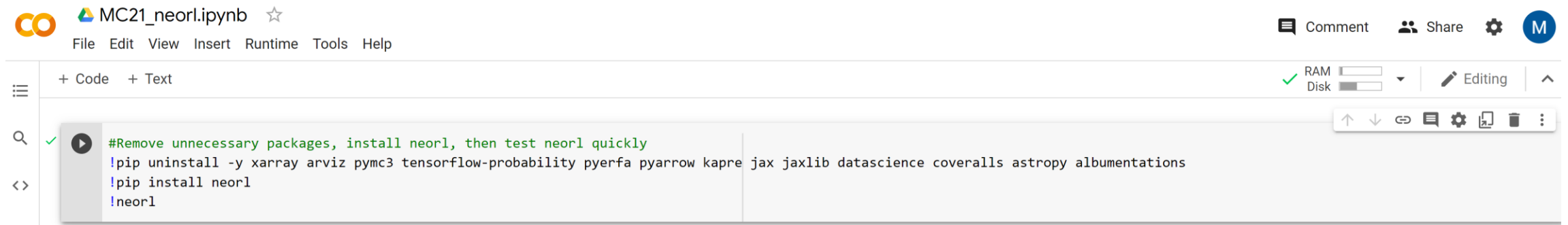
Massachusetts Institute of Technology

Content

- **Background**
- NEORL Framework
- Problem 1: Ackley Mathematical Function
- Problem 2: Pressure Vessel Design
- Problem 3: Three-bar Truss (Homework)
- Summary

Optimization

Single-objective optimization

$$\min_{\vec{x}} \ f(\vec{x}),$$

subject to,

$$g_i(\vec{x}) \geq 0, \quad i = 1, 2, ..., m,$$
$$h_j(\vec{x}) = 0, \quad j = 1, 2, ..., p,$$

Multi-objective optimization

$$\min_{\vec{x}} \ F(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x}))$$

subject to,

$$g_i(\vec{x}) \geq 0, \quad i = 1, 2, ..., m,$$
$$h_j(\vec{x}) = 0, \quad j = 1, 2, ..., p,$$
$$k \geq 2,$$

Optimization while in high school

4. An open-top box with a square bottom and rectangular sides is to have a volume of 256 cubic inches. Find the <u>dimensions</u> that require the *minimum* amount of material.

$$S = x^2 + 4xy$$
$$V = x^2 y = 256$$
$$\rightarrow S = x^2 + 4x\left(\frac{256}{x^2}\right)$$

$$S = x^2 + \frac{1024}{x}$$

$$S' = 2x - \frac{1024}{x^2}$$
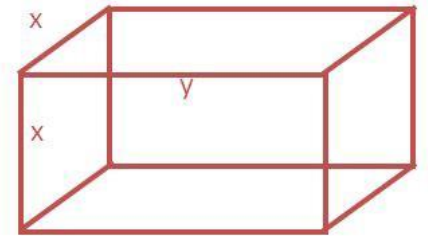
$$S'' = 2 + \frac{2048}{x^3} > 0$$

therefore a min

$$0 = 2x - \frac{1024}{x^2}$$

$$x = 8 \ \rightarrow y = 4$$
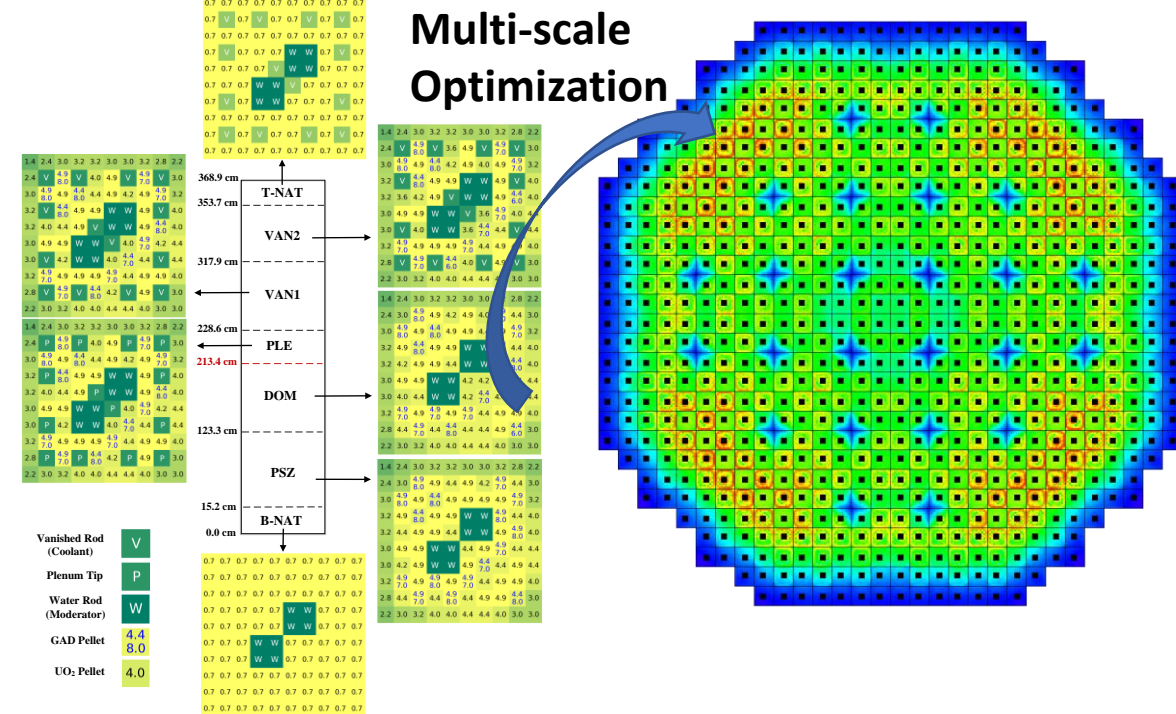
8 x 8 x 4

Large-scale optimization

Multi-scale Optimization

- We don't always have a closed-form objective function!

- We don't always have access to derivatives!

- We don't always have fast analytical computation!
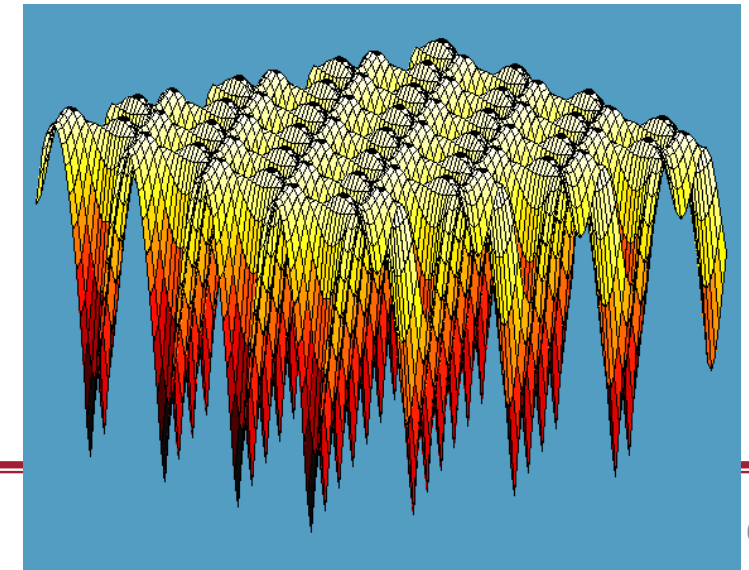
- **Thus, we need smart optimizers!**

Expensive Simulations
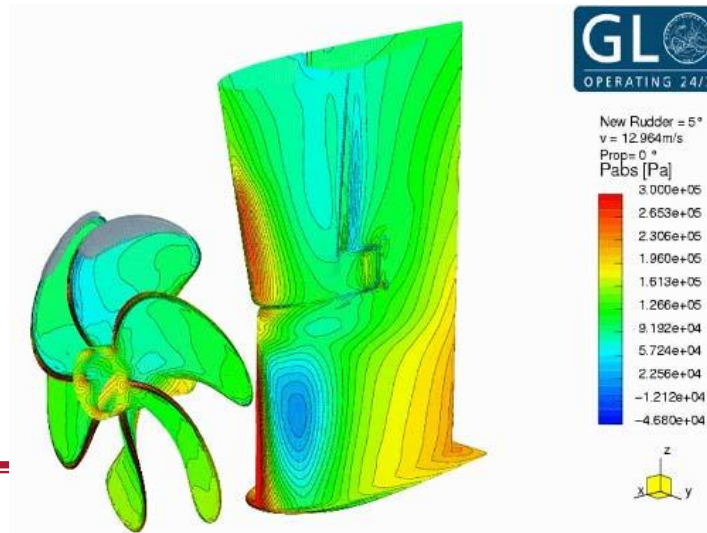
Multimodal Multi-local Optima

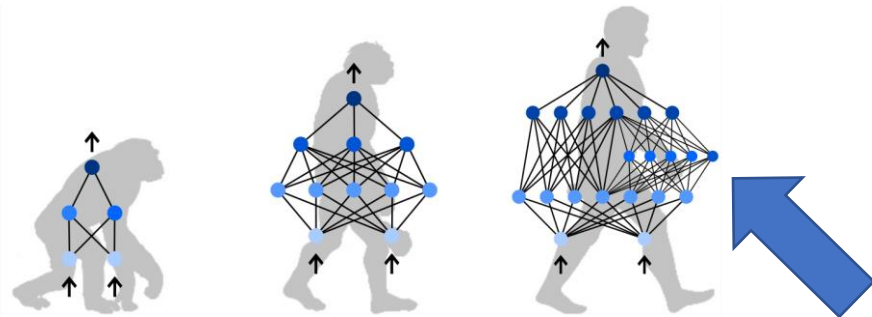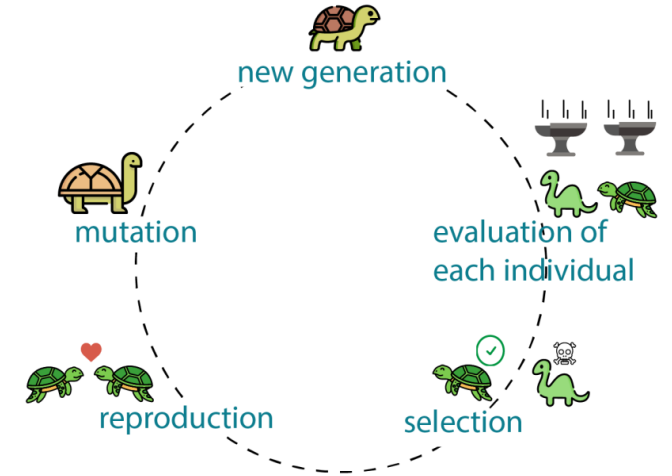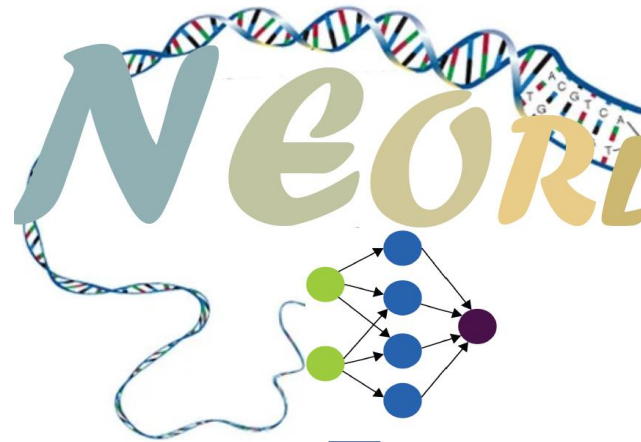**Massachusetts Institute of Technology**

Content

- Background

- **NEORL Framework**

- Problem 1: Ackley Mathematical Function

- Problem 2: Pressure Vessel Design

- Problem 3: Three-bar Truss (Homework)

- Summary

Massachusetts Institute of Technology

Optimization Categories

Hybrid Algorithms (Neuroevolution)

Evolutionary Algorithms (Metaheuristics)

new generation

mutation

evaluation of each individual

reproduction

selection

Neural Networks (Reinforcement Learning)
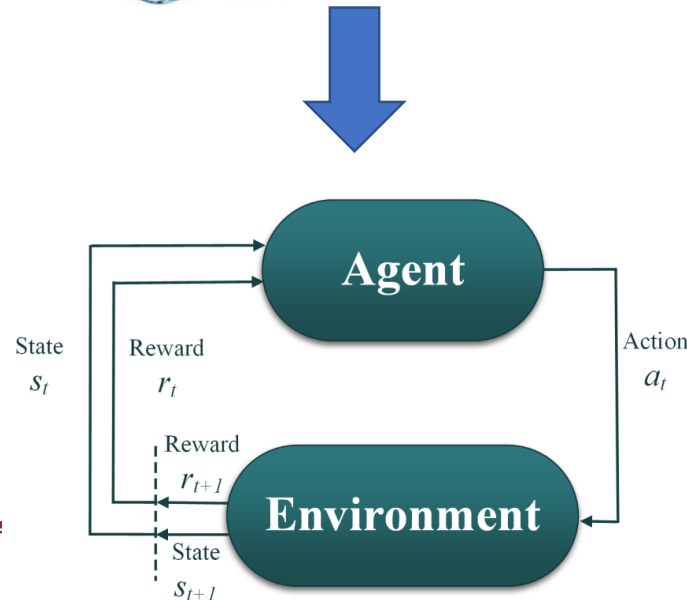
For gradient-based algorithms, see:
https://nlopt.readthedocs.io/en/latest/
https://docs.scipy.org/doc/scipy/reference/optimize.html

Agent

State s_t Reward r_t Action a_t

Reward r_{t+1}

State s_{t+1}

Environment

NEORL in a Nutshell
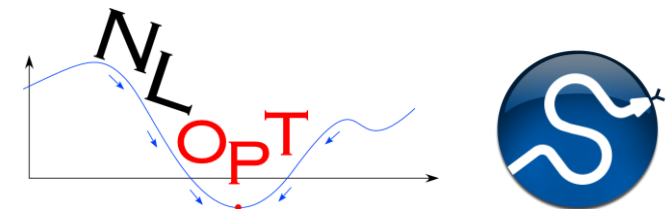
https://neorl.readthedocs.io/en/latest/index.html

Basic Features

Features	NEORL
Reinforcement Learning (standalone)	✔
Evolutionary Computation (standalone)	✔
Hybrid Neuroevolution	✔
Supervised Learning	✔
Parallel processing	✔
Combinatorial/Discrete Optimization	✔
Continuous Optimization	✔
Mixed Discrete/Continuous Optimization	✔
Hyperparameter Tuning	✔
Ipython / Notebook friendly	✔
Detailed Documentation	✔
Advanced logging	✔
Optimization Benchmarks	✔

Implemented Algorithms

NEORL offers a wide range of algorithms, where some algorithms could be used with a specific parameter space.

Algorithm	Discrete Space	Continuous Space	Mixed Space	Multiprocessing
ACER	✔	✘	✘	✔
ACKTR	✔	✔	✔	✔
A2C	✔	✔	✔	✔
PPO	✔	✔	✔	✔
DQN	✔	✘	✘	✘
ES	✔	✔	✔	✔
PSO	✔	✔	✔	✔
DE	✔	✔	✔	✔
XNES	✘	✔	✘	✔
GWO	✔	✔	✔	✔
PESA	✔	✔	✔	✔
PESA2	✔	✔	✔	✔
RNEAT	✘	✔	✘	✔
FNEAT	✘	✔	✘	✔
SA	✔	✔	✔	✔
SSA	✔	✔	✔	✔
WOA	✔	✔	✔	✔
JAYA	✔	✔	✔	✔
MFO	✔	✔	✔	✔
HHO	✔	✔	✔	✔
BAT	✔	✔	✔	✔
PPO-ES	✔	✔	✔	✔
ACKTR-DE	✔	✔	✔	✔
ACO	✘	✔	✘	✔
NGA	✘	✔	✘	✘
NHHO	✔	✔	✔	✔
CS	✔	✔	✔	✔
TS	✔	✔	✘	✘

NEORL Team

Katelin

Haijia

John

Devin

Koroush

Paul

Xubo

Majdi

Massachusetts Institute of Technology
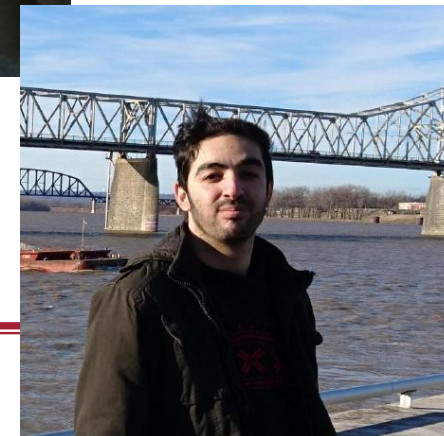
NEORL Detailed Documentation

Neuro**E**volution **O**ptimization with **R**einforcement **L**earning

---

NEORL

latest

Search docs

GENERAL GUIDE

Quick Installation

Detailed Installation

Getting Started

Reinforcement Learning

Evolutionary Algorithms

Hyperparameter Tuning

ALGORITHMS

Advantage Actor Critic (A2C)

Actor-Critic with Experience Replay (ACER)

Actor Critic using Kronecker-Factored Trust Region (ACKTR)

Deep Q Learning (DQN)

🏠 » Welcome to NEORL Website!

⌨ Edit on GitHub

Welcome to NEORL Website!

Latest News:

- September 10, 2021: First NEORL stable release 1.6 is out.

Primary contact to report bugs/issues: Majdi I. Radaideh (radaideh@mit.edu)

NEORL (**N**euro**E**volution **O**ptimisation with **R**einforcement **L**earning) is a set of implementations of hybrid algorithms combining neural networks and evolutionary computation based on a wide range of machine learning and evolutionary intelligence architectures. NEORL aims to solve large-scale optimisation problems relevant to operation & optimisation research, engineering, business, and other disciplines.

Github repository: https://github.com/mradaideh/neorl

Copyright

This repository and its content are copyright of Exelon Corporation © in collaboration with MIT Nuclear Science and Engineering 2021. All rights reserved.

You can read the first successful application of NEORL for nuclear fuel optimisation in this News Article.
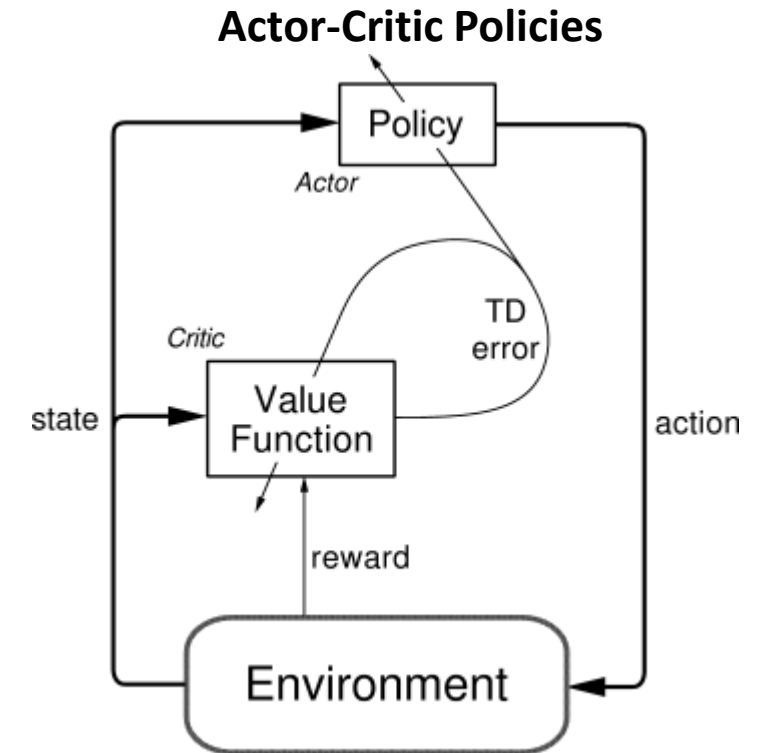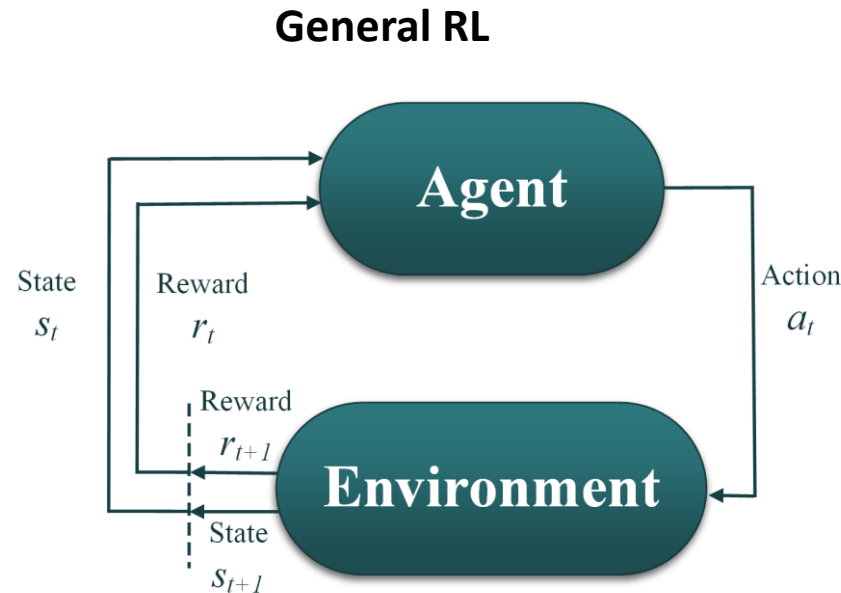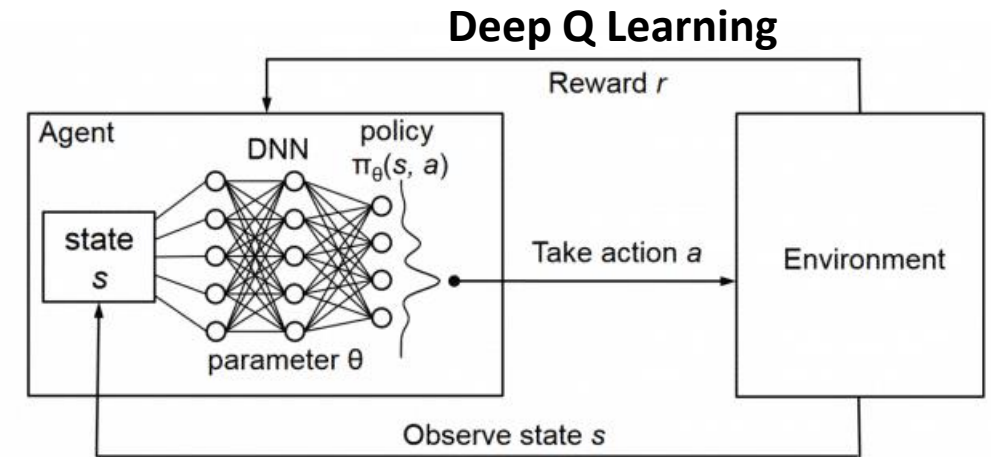
User Guide

General Guide

1

Neural-based Algorithms

- Reinforcement Learning: Combination of **state-reward-action** learning, guided by **deep neural networks**
  - PPO
  - ACKTR
  - A2C
  - DQN
  - ACER

Deep Q Learning

General RL

Actor-Critic Policies

Massachusetts Institute of Technology

Evolutionary Algorithms

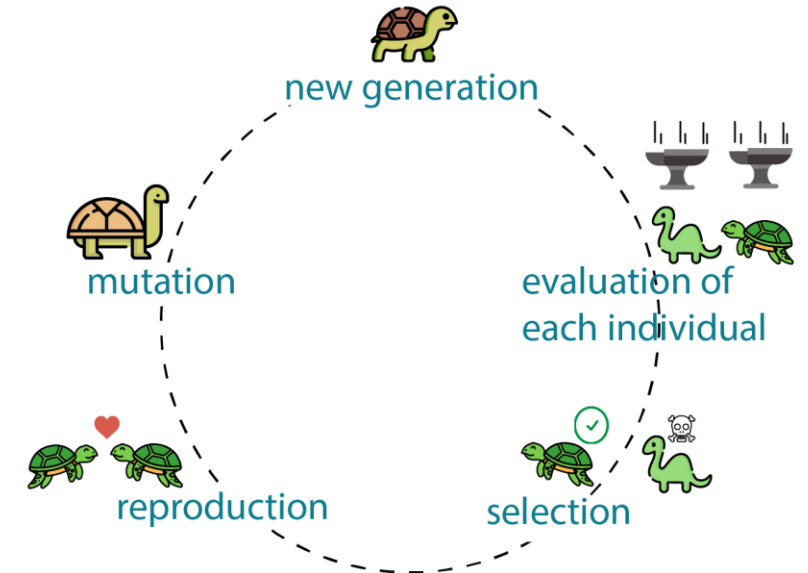- 1- Random population is generated,

- 2- Population is evaluated by a fitness function,

- 3- Best individuals are survived for next generation.

- 4- The population is updated using natural operations (crossover, mutation, hunting strategies) for next generation.

- Repeat 1-4

Particle Swarm Optimization
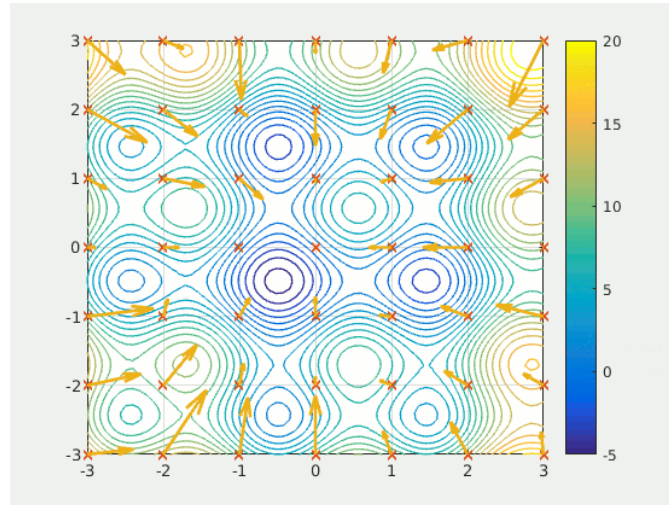
Genetic Algorithms
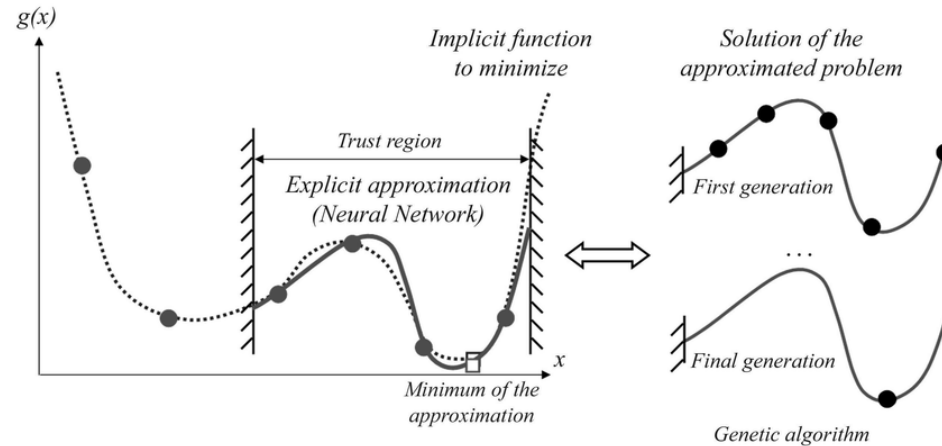

Grey wolf Optimization
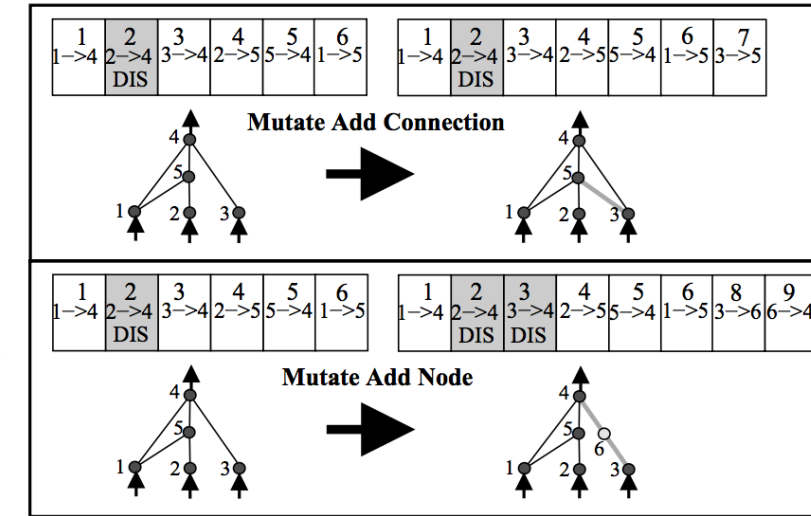

Massachusetts Institute of Technology

Hybrid/Neuroevolution Algorithms

- **Neuroevolution algorathims are advanced optimizers:**
- NEAT (RNEAT, FNEAT)
- PESA
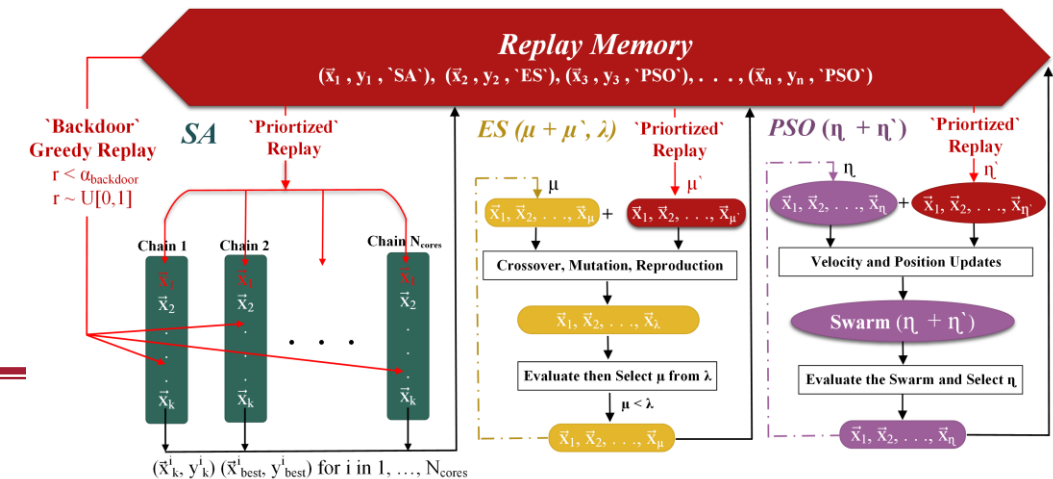- PESA2
- PPO-ES
- ACKTR-DE
- NHHO
- NGA

Surrogate-based Evolutionary Algorithms

Prioritized replay Evolutionary and Swarm Algorithm (PESA)

Massachusetts Institute of Technology

Hyperparameter Tuning and Parallel Computing

- Tune the optimizer hyperparameters **for maximum performance**
- NEORL offers **four automatic tuners**!

Parallelization in NEORL is straightforward, turn on "ncores" flag!

```
gwo=GWO(mode='min', bounds=BOUNDS,
fit=ACKLEY, nwolves=20, ncores=16, seed=1)
```


Grid Search

A module for grid search of hyperparameters of NEORL algorithms.

Original paper: Bergstra, J., & Bengio, Y. (2012). Random search for h...

Grid Search is an exhaustive search for selecting an optimal set of al... values. A multi-dimensional full grid of all hyperparameters is constr... combination of hyperparameter values is tested in serial/parallel, wh... for fine grids as well as large number of hyperparameters to tune.

What can you use?
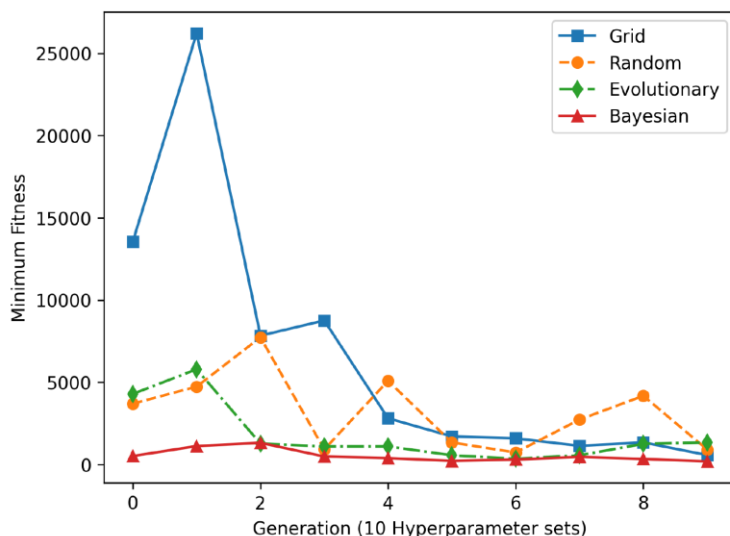
- **Automatic search** makes hyperparameter tuning a much **less burden** to **focus on the optimization** problem itself!

Toy Example (Sphere Function)

2D Sphere (n=2)


```python
#1- Import an algorithm from NEORL
from neorl import DE

#2- Define the fitness function
def FIT(individual):
    #sphere function
    y=sum(x**2 for x in individual)
    return y

#3-Setup the parameter space (n=5)
nx=5
BOUNDS={}
for i in range(1,nx+1):
    BOUNDS['x'+str(i)]=['float', -5.12, 5.12]

#4- setup and run the optimizer
de=DE(mode='min', bounds=BOUNDS, fit=FIT, npop=60, F=0.5, CR=0.7, ncores=1, seed=1)
x_best, y_best, de_hist=de.evolute(ngen=100, verbose=1)

print(x_best, y_best)
```
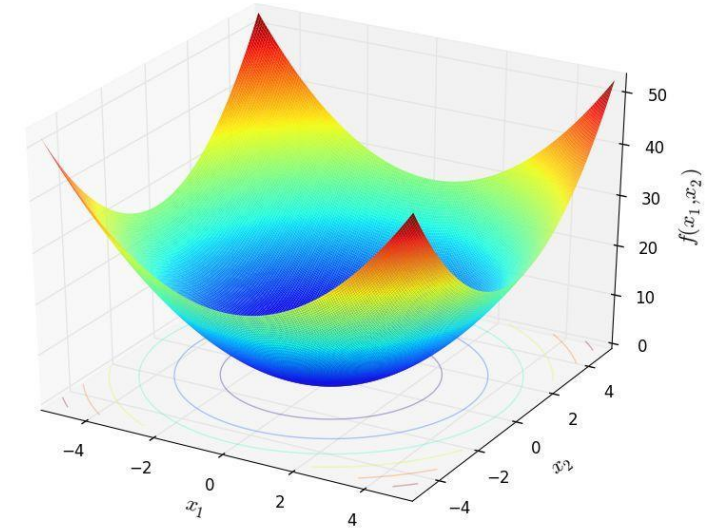
$$f(x_1 \cdots x_n) = \sum_{i=1}^{n} x_i^2$$

$$-5.12 \leq x_i \leq 5.12$$

$$\text{minimum at } f(0, \cdots, 0) = 0$$

Massachusetts Institute of Technology

Content

- Background
- NEORL Framework
- **Problem 1: Ackley Mathematical Function**
- Problem 2: Pressure Vessel Design
- Problem 3: Three-bar Truss (Homework)
- Summary

Massachusetts Institute of Technology

Problem Statement

- Find the optimal value of $\vec{x}$ *such that* $f(\vec{x})$ *is minimized, where d=10.*

- Solution through NEORL:
  - Set the objective function.
  - Set the parameter space.
  - Set the algorithm object.
  - Optimize!
  - Make a plot (optional)

Ackley Function

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

$x_i \in [-32, 32]$, for all $i = 1, \ldots, d.$

Sample Script

- DE group
  - Use hyperparameters **npop=80, F=0.3, CR=0.7**
  - Set **seed = 1**.
  - Use **ngen=120** in evolute function.
  - For plotting access the fitness via the key fitness in the returned dictionary.
    - **x_gwo, y_gwo, gwo_hist=gwo.evolute(ngen=120, verbose=1)**
    - Use **gwo_hist['fitness']** for plotting.

- GWO group
  - Use **nwolves=20**
  - Set **seed = 1**
  - Use **ngen=120** in evolute function
  - For plotting access the fitness directly from the variable fitness.
    - **x_de, y_de, de_hist=de.evolute(ngen=120, verbose=1)**
    - Use **de_hist** for plotting.

Required arguments in NEORL algorithms:
mode, bounds, fit

```python
#---------------------------------
# Import packages
#---------------------------------
import numpy as np
import matplotlib.pyplot as plt
from neorl import DE, GWO
from math import exp, sqrt, cos, pi
#---------------------------------
# Fitness function
#---------------------------------
def ACKLEY(individual):
    #Ackley objective function.
    d = len(individual)
    f=20 - 20 * exp(-0.2*sqrt(1.0/d * sum(x**2 for x in individual))) \
            + exp(1) - exp(1.0/d * sum(cos(2*pi*x) for x in individual))
    return f
#---------------------------------
# Parameter Space
#---------------------------------
#Setup the parameter space (d=8)
d=10
lb=-32
ub=32
BOUNDS={}
for i in range(1,d+1):
    BOUNDS['x'+str(i)]=['float', lb, ub]
#---------------------------------
# GWO
#---------------------------------
gwo=GWO(mode='min', bounds=BOUNDS, fit=ACKLEY, nwolves=20, seed=1)
x_gwo, y_gwo, gwo_hist=gwo.evolute(ngen=120, verbose=1)
#---------------------------------
# DE
#---------------------------------
de=DE(mode='min', bounds=BOUNDS, fit=ACKLEY, npop=80, F=0.3, CR=0.7, ncores=1,
seed=1)
x_de, y_de, de_hist=de.evolute(ngen=120, verbose=1)
#---------------------------------
# Plot
#---------------------------------
#Plot fitness for both methods
plt.figure()
plt.plot(gwo_hist['fitness'], label='GWO')
plt.plot(np.array(de_hist), label='DE')
plt.xlabel('Generation')
plt.ylabel('Fitness')
plt.legend()
plt.savefig('ackley_fitness.png',format='png', dpi=300, bbox_inches="tight")
plt.show()
```
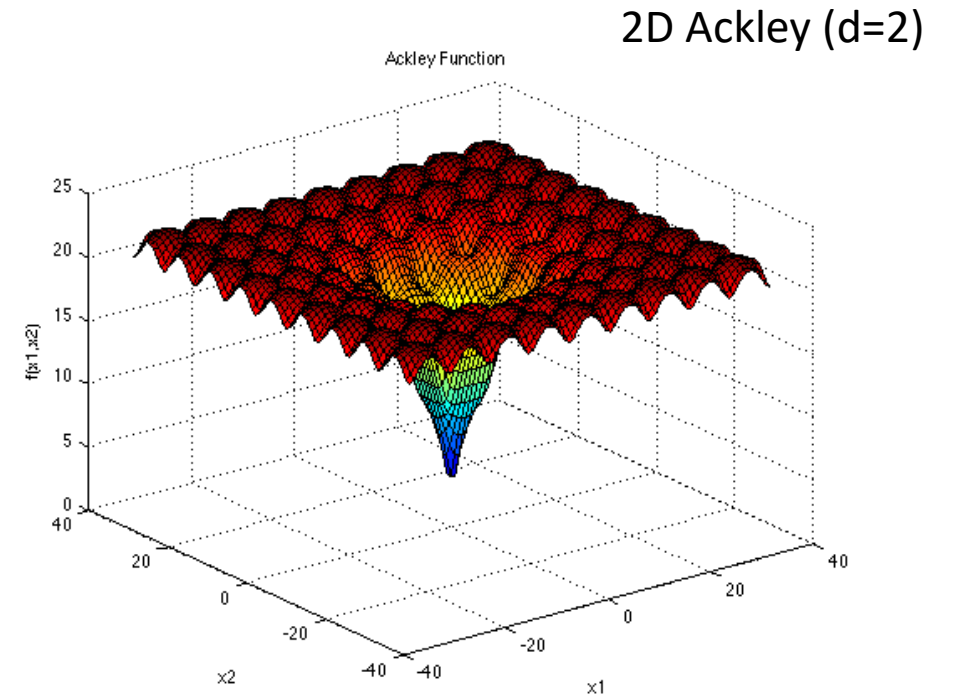
Content

- Background

- NEORL Framework

- Problem 1: Ackley Mathematical Function

- **Problem 2: Pressure Vessel Design**

- Problem 3: Three-bar Truss (Homework)

- Summary

Massachusetts Institute of Technology

Problem Statement

- Find the optimal value of $\vec{x}$
  - $T_s = x_1$
  - $T_h = x_2$
  - $R = x_3$
  - $L = x_4$
  - such that $f(\vec{x})$ is minimized (cost of pressure vessel is minimized)
- Solution through NEORL:
  - Set the objective function, including the constraints.
  - Set the parameter space.
  - Set the algorithm object.
  - Optimize!
  - Make a plot (optional).
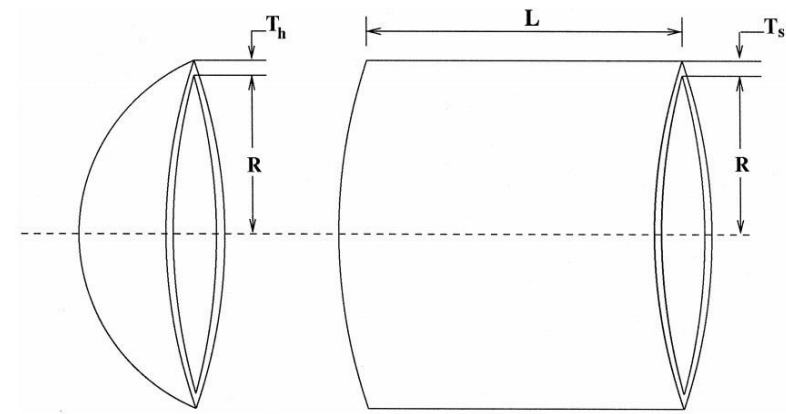
$$\min_{\vec{x}} f(\vec{x}) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3,$$

$$g_1 = -x_1 + 0.0193 x_3 \leq 0,$$
$$g_2 = -x_2 + 0.00954 x_3 \leq 0,$$
$$g_3 = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0,$$
$$g_4 = x_4 - 240 \leq 0,$$

$$0.0625 \leq x_1 \leq 6.1875 \text{ (with step of 0.0625)}$$

$$x_2 \in \{0.0625, 0.125, 0.1875, 0.25, 0.3125, 0.375, 0.4375, 0.5, 0.5625, 0.625\}$$

$$10 \leq x_3 \leq 200,$$

$$10 \leq x_4 \leq 200$$

Problem Setup

mode, bounds, fit

- HHO group
  - Use hyperparameters **nhawks=50, int_transform='minmax'**
  - Set **seed = 1**.
  - Use **ngen=200** in evolute function.
  - For plotting access the fitness via the key fitness in the returned dictionary.
    - **x_hho, y_hho, hho_hist=hho.evolute(ngen=200, verbose=False)**
    - Use **hho_hist['global_fitness']** for plotting.

- BAT group
  - Use **nbats=50, levy = True**
  - Set **seed = 1**
  - Use **ngen=200** in evolute function
  - For plotting access the fitness directly from the variable fitness.
    - **x_bat, y_bat, bat_hist=bat.evolute(ngen=200, verbose=1)**
    - Use **bat_hist['global_fitness']** for plotting.

```python
def Vessel(individual):
    """
    Pressure vesssel design
    x1: thickness (d1)  --> discrete value multiple of 0.0625 in
    x2: thickness of the heads (d2) ---> categorical value from a pre-defined grid
    x3: inner radius (r)  ---> cont. value between [10, 200]
    x4: length (L)  ---> cont. value between [10, 200]
    """
    x=individual.copy()
    x[0] *= 0.0625   #convert d1 to "in"
    y = 0.6224*x[0]*x[2]*x[3]+1.7781*x[1]*x[2]**2+3.1661*x[0]**2*x[3]+19.84*x[0]**2*x[2];
    g1 = -x[0]+0.0193*x[2];
    g2 = -x[1]+0.00954*x[2];
    g3 = -math.pi*x[2]**2*x[3]-(4/3)*math.pi*x[2]**3 + 1296000;
    g4 = x[3]-240;
    g=[g1,g2,g3,g4]
    phi=sum(max(item,0) for item in g)
    eps=1e-5 #tolerance to escape the constraint region
    penality=1e6 #large penalty to add if constraints are violated
    if phi > eps:
        fitness=phi+penality
    else:
        fitness=y
    return fitness
```

Sample Script

```python
#######################
# Import Packages
#######################
from neorl import HHO, BAT
import math
import matplotlib.pyplot as plt
###################################
# Define Vessel Function
#Mixed discrete/continuous/grid
###############################
def Vessel(individual):
    """
    Pressure vesssel design
    x1: thickness (d1)  --> discrete value multiple of 0.0625 in
    x2: thickness of the heads (d2) ---> categorical value from a pre-defined grid
    x3: inner radius (r)  ---> cont. value between [10, 200]
    x4: length (L)  ---> cont. value between [10, 200]
    """
    x=individual.copy()
    x[0] *= 0.0625   #convert d1 to "in"
    y = 0.6224*x[0]*x[2]*x[3]+1.7781*x[1]*x[2]**2+3.1661*x[0]**2*x[3]+19.84*x[0]**2*x[2];
    g1 = -x[0]+0.0193*x[2];
    g2 = -x[1]+0.00954*x[2];
    g3 = -math.pi*x[2]**2*x[3]-(4/3)*math.pi*x[2]**3 + 1296000;
    g4 = x[3]-240;
    g=[g1,g2,g3,g4]
    phi=sum(max(item,0) for item in g)
    eps=1e-5 #tolerance to escape the constraint region
    penality=1e6 #large penality to add if constraints are violated
    if phi > eps:
        fitness=phi+penality
    else:
        fitness=y
    return fitness
#######################
# Setup the Space
#######################
bounds = {}
bounds['x1'] = ['int', 1, 99]
bounds['x2'] = ['grid', (0.0625, 0.125, 0.1875, 0.25, 0.3125, 0.375, 0.4375, 0.5, 0.5625, 0.625)]
bounds['x3'] = ['float', 10, 200]
bounds['x4'] = ['float', 10, 200]
```

```python
#######################
# Setup and evolute HHO
#######################
hho = HHO(mode='min', bounds=bounds, fit=Vessel, nhawks=50,
                int_transform='minmax', ncores=1, seed=1)
x_hho, y_hho, hho_hist=hho.evolute(ngen=200, verbose=False)
#######################
# Setup and evolute BAT
#######################
bat=BAT(mode='min', bounds=bounds, fit=Vessel, nbats=50, levy = True, seed
= 1, ncores=1)
x_bat, y_bat, bat_hist=bat.evolute(ngen=200, verbose=1)
#######################
# Plotting
#######################
plt.figure()
plt.plot(hho_hist['global_fitness'], label='HHO')
plt.plot(bat_hist['global_fitness'], label='BAT')
plt.xlabel('Generation')
plt.ylabel('Fitness')
plt.ylim([0,10000]) #zoom in
plt.legend()
plt.savefig('ex8_pv_fitness.png',format='png', dpi=300,
bbox_inches="tight")
#######################
# Comparison
#######################
print('---Best HHO Results---')
print(x_hho)
print(y_hho)
print('---Best BAT Results---')
print(x_bat)
print(y_bat)
```

Content

- Background
- NEORL Framework
- Problem 1: Ackley Mathematical Function
- Problem 2: Pressure Vessel Design
- **Problem 3: Three-bar Truss (Homework)**
- Summary

Massachusetts Institute of Technology

Three-bar Truss (Enjoy it!)

- Solve it with PSO and MFO in NEORL.
- Look for a minimum $f(\vec{x})$ of **263.9**

```python
#-------------------------------
# Fitness function
#-------------------------------
def TBT(individual):
    """Three-bar truss Design
    """
    x1 = individual[0]
    x2 = individual[1]
    y = (2*sqrt(2)*x1 + x2) * 100

    #Constraints
    if x1 <= 0:
        g = [1,1,1]
    else:
        g1 = (sqrt(2)*x1+x2)/(sqrt(2)*x1**2 + 2*x1*x2) * 2 - 2
        g2 = x2/(sqrt(2)*x1**2 + 2*x1*x2) * 2 - 2
        g3 = 1/(x1 + sqrt(2)*x2) * 2 - 2
        g = [g1,g2,g3]
    g_round=np.round(np.array(g),6)
    w1=100
    w2=100
    phi=sum(max(item,0) for item in g_round)
    viol=sum(float(num) > 0 for num in g_round)

    return y + w1*phi + w2*viol
```
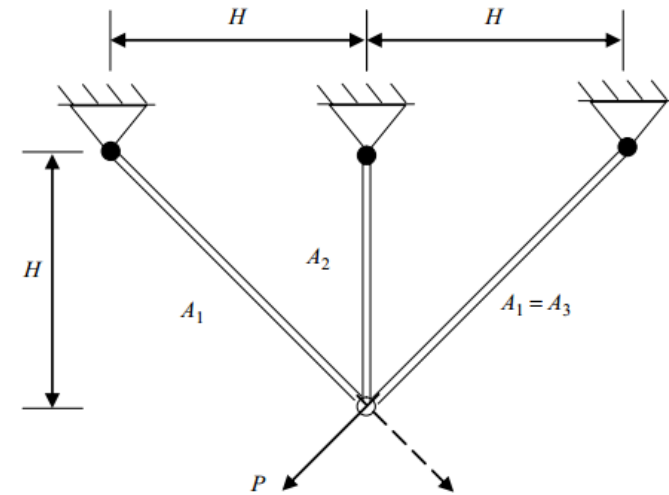

$$\min_{\vec{x}} f(\vec{x}) = (2\sqrt{2}x_1 + x_2) \times H,$$

$$g_1 = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0,$$

$$g_2 = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0,$$

$$g_3 = \frac{1}{x_1 + \sqrt{2}x_2}P - \sigma \leq 0,$$

where $0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1, H = 100cm, P = 2KN/cm^2$, and $\sigma = 2KN/cm^2$

Content

- Background

- NEORL Framework

- Problem 1: Ackley Mathematical Function

- Problem 2: Pressure Vessel Design

- Problem 3: Three-bar Truss (Homework)

- **Summary**

Massachusetts Institute of Technology

Summary

- Bring your **objective function** and let NEORL **takes care of the rest**.

- NEORL has:
  - more than 25 algorithms,
  - neural, evolutionary, and neuroevolution categories,
  - Discrete, continuous, and mixed optimization spaces,
  - parallel computing,
  - hyperparameter tuning.
  - friendly interface,
  - detailed documentation,

RL-informed Differential Evolution (ACKTR-DE)
Ant Colony Optimization (ACO)
Neural Genetic Algorithms (NGA)
Neural Harris Hawks Optimization (NHHO)
Cuckoo Search (CS)
Tabu Search (TS)

HYPERPARAMETER TUNING
Grid Search
Random Search
Bayesian Search
Evolutionary Search

EXAMPLES
Example 1: Traveling Salesman Problem
Example 2: Ackley with EA
Example 3: Welded-beam design
Example 4: Benchmarks
Example 5: CEC'2017 Test Suite
Example 6: Three-bar Truss Design

Projects

This is a list of projects using NEORL. Please contact us if you want your project to appear on this page.

Physics-informed Reinforcement Learning Optimisation with NEORL

Optimization of nuclear fuel assemblies if performed effectively, will lead to fuel efficiency improvement, cost reduction, and safety assurance. However, assembly optimization involves solving high-dimensional and computationally expensive combinatorial problems. As such, fuel designers' expert judgement has commonly prevailed over the use of stochastic optimization (SO) algorithms such as genetic algorithms and simulated annealing. To improve the state-of-art, we explore a class of artificial intelligence (AI) algorithms, namely, reinforcement learning (RL) in this work. We propose a physics-informed AI optimization methodology by establishing a connection through reward shaping between RL and the tactics fuel designers follow in practice by moving fuel rods in the assembly to meet specific constraints and objectives. The methodology utilizes RL algorithms, deep Q learning and proximal policy optimization, and compares their performance to SO algorithms. The methodology is applied on two boiling water reactor assemblies of low-dimensional (combinations) and high-dimensional (combinations) natures. The results demonstrate that RL is more effective than SO in solving high dimensional problems, i.e., 10 × 10 assembly, through embedding expert knowledge in form of game rules and effectively exploring the search space. For a given computational resources and timeframe relevant to fuel designers, RL algorithms outperformed SO through finding more feasible patterns, 4–5 times more than SO, and through increasing search speed, as indicated by the RL outstanding computational efficiency. The results of this work clearly demonstrate RL effectiveness as another decision support tool for nuclear fuel assembly optimization.
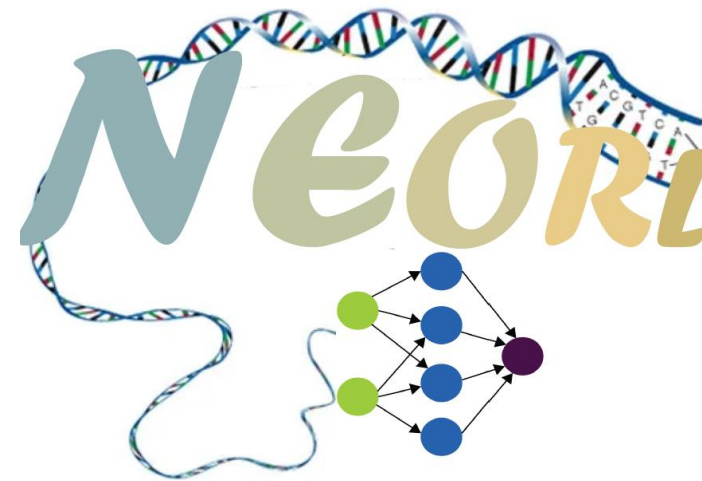
Authors: Majdi I. Radaideh et al., 2021.
Reference: https://doi.org/10.1016/j.nucengdes.2020.110966

Questions?

- For detailed examples and info, check our website and Github:

- https://neorl.readthedocs.io/en/latest/index.html

- https://github.com/mradaideh/neorl

Thanks to our collaborators and sponsor:
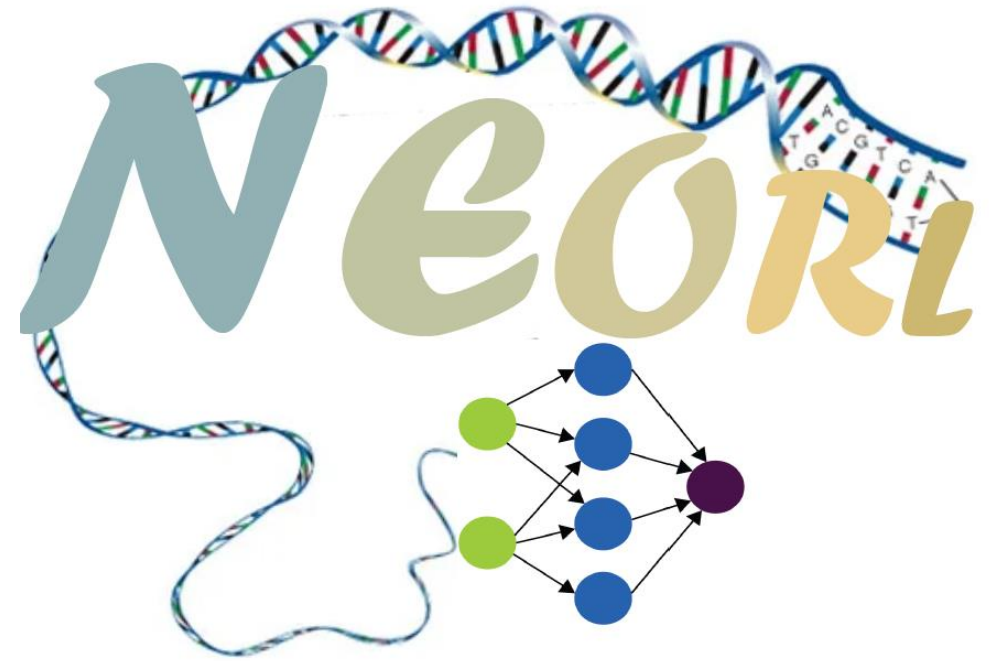
You can download all workshop materials/scripts from this link: