Evolutionary Computation

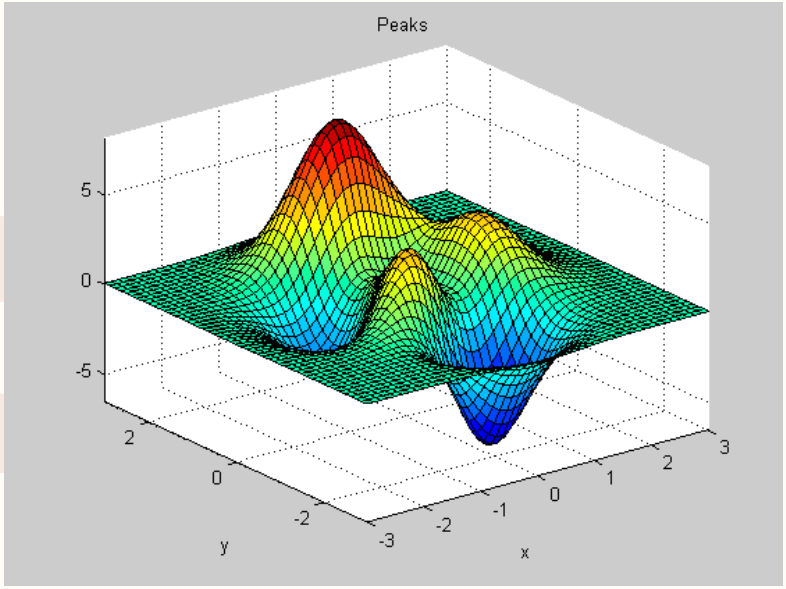
in Continuous Optimization and Machine Learning

Introduction

What? Why? How?

Introduction: **Concepts**

Optimization



Evolutionary Computation

- Problem solving using natural computational processes
- Darwinian evolution as a model for solving optimization problems
- “Organisms” as “inputs” to the model of the “environment”

Introduction: **Purpose**

Benchmark of Optimization

- Three emerging evolutionary algorithms
- On **mathematical** functions
- On **machine learning** tasks

New Algorithm

- Improve on the three algorithms in some respects in machine learning
- Add it to the benchmark

Background

Introduction to EA, Traditional EA,

Emerging EA, Machine Learning

Evolutionary Algorithms: **Introduction**

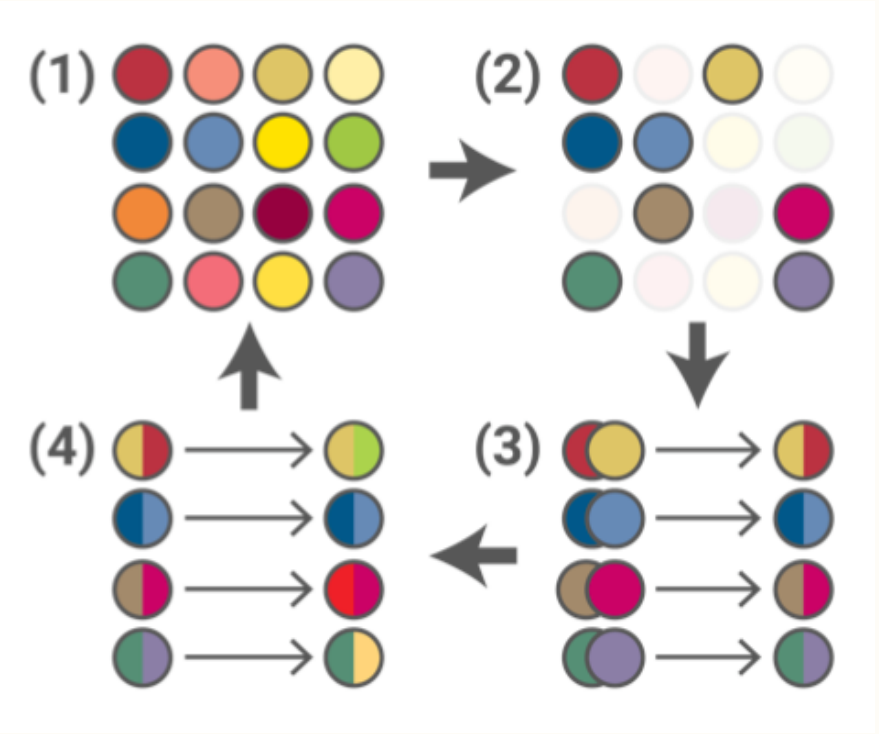
- Works with **populations**
- Which contain **individuals**
- Which are **parameter vectors**

Algorithm 1 Basic evolutionary algorithm

$t \leftarrow 0$
initialize $P(t)$
evaluate $P(t)$
while termination-condition not fulfilled **do**
  $t \leftarrow t + 1$
  select $P(t)$ from $P(t-1)$
  alter $P(t)$
  evaluate $P(t)$
end while

Evolutionary Algorithms: **Terminology**

- Population
- Representation
- Initialization
- Parent selection
- Variation
- Survivor selection
- Fitness function
- Termination condition



Evolutionary Algorithms: **Varieties**

Traditional

- Genetic Algorithms
- Evolution Strategy
- Evolutionary Programming
- Genetic Programming

Emerging

- Differential Evolution
- Particle Swarm Optimization
- Estimation of Distribution Algorithm

Differential Evolution

- Mutation based on **vector differences**
- **Few** parameters
- **Easy** to implement
- Very **good** overall

Algorithm 3 DE algorithm

Initialize population within bounds
Evaluate fitness of population
repeat
  for all Individuals x **do**
    Select three other random individuals a, b and c
    Create mutant vector $v = a + F(b - c)$
    Create trial vector u by blending x with v (taking at least one gene from u)
    Replace x with u if $fitness(u) > fitness(x)$
  end for
until End condition

Particle Swarm Optimization

- Population forms a **swarm** which acts as a **whole**
- Individuals **explore** the terrain but follow to the swarm
- **Update formula** guides the movements

Algorithm 4 PSO algorithm

Init particles with random positions $\vec{x}(0)$ and velocities $\vec{v}(0)$
repeat
  for all Particles i **do**
    Evaluate fitness $f(\vec{x_i})$
    Update $\vec{p}(t)$ and $\vec{g}(t)$
    Adapt the velocity of the particle using the above-mentioned equation
    Update the position of the particle
  end for
until $\vec{g}(t)$ is a suitable solution

$$v_{id}(t+1) = \omega v_{id}(t) + C_1\phi_1(p_{id}(t) - x_{id}(t)) + C_2\phi_2(g_{id}(t) - x_{id}(t))$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$$

Estimation of Distribution Algorithm

- Mutation accomplished by sampling probability models
- Choice of model depends on the complexity of the problem
- Simplest is normal distribution

Algorithm 5 UMDA algorithm

```
Initialize population P with size D
repeat
    Evaluate P
    Select the best t% * D individuals from P into S
    Let m be the mean of S
    Let s be the standard deviation of S
    Create a normal distribution ND from m and s
    Sample (100%-t%) * D individuals into S' from ND
    Create new generation of P by combining S' and S
until Termination condition
```

Machine Learning

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

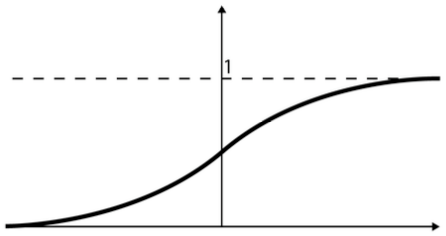


Figure 8: Sigmoid function

- Artificial Neural Networks
- Feed-Forward Neural Network
- Back-propagation

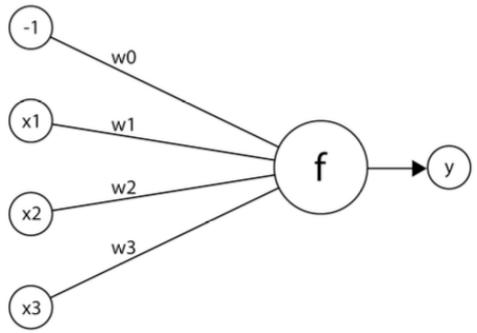


Figure 7: Artificial Neuron

$$y = f(-1 * w_0 + \sum_{i=1}^{n-1} w_i x_i)$$

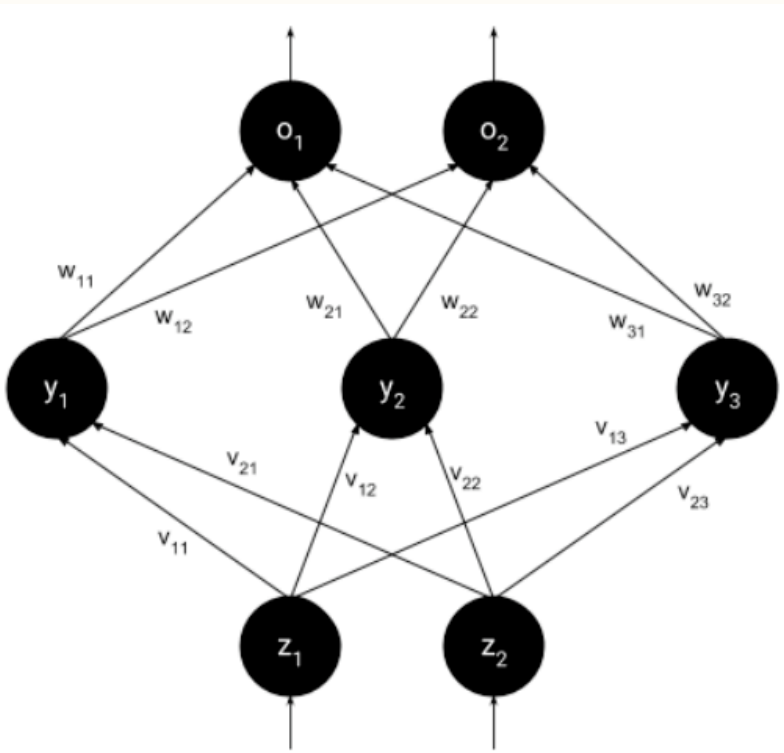


Figure 9: Feed-Forward Neural Network

Related Work

Current Research, Emerging EA, ML and the Contribution

Research

Roots

- First ideas in **1950s**
- Several concepts **emerged independently**
  - Genetic Algorithms
  - Evolution Strategy
  - Evolutionary Programming
- **No free lunch** means no single algorithm can beat them all

Current Research

- Parallelism
- Multi-population models
- Multi-objective optimization
- Dynamic environments
- Evolving executable code

Research: **Differential Evolution**

- Created 1995 by **Storn and Price**
- **One of the best** algorithms in continuous optimization
- **Hybridization** (self-adaptive, opposition-based, with local and global neighborhoods)
- Effects of parameter use **well studied**
- Comes in **10 different varieties** (DE/best/1/bin generally one of the best)

Research: **Particle Swarm Optimization**

- **Most popular** swarm optimization algorithm
- **Modified versions** (quantum-behaved, bare-bones, chaotic, fuzzy, PSOT-VAC, opposition-based)
- **Hybridized** with other evolutionary algorithms
  - Differential Evolution

Research: **Estimation of Distribution Algorithm**

- Different **probabilistic models** have been tried (normal distribution, Gaussian nets, graphical models, etc.)
- **Successful at many engineering problems** which other algorithms struggle with (antenna design, protein prediction, portfolio management, chemotherapy, etc.)
- Improvements
  - **Parallelization** of fitness function and model building
  - **Local optimization** techniques (hill climbing)

Research: **Machine Learning**

- Many attempts to combine ML and EA
- ML used to improve EC
- EC used to improve ML
  - Differential Evolution optimizing neural networks
  - Particle Swarm Optimization initializing weights in neural networks

Contributions

- Studying the application of **evolutionary computation on machine learning**
- Finding the **best optimization algorithms** for machine learning tasks
- Contributing a novel **improved algorithm**

Problem Formulation

Questions, Motivation, Outcomes and Limitations

Questions

- How do the emerging algorithms compare on **mathematical benchmarks**?
- How do the emerging algorithms compare on **machine learning benchmarks**?
- Which algorithms are **best suited for which problems**?
- How can the algorithms be **improved**?

Motivation

- **Insight into application** of evolutionary computation on popular machine learning tasks
- **Improved algorithms** for machine learning
- **Comparison** of algorithms

Outcomes

- Benchmarks
  - Mathematical
  - Machine Learning
    - *Function Approximation*
    - *Clustering*
    - *Classification*
    - *Game Controllers*
- Improved algorithm

Limitations

- Limited number of algorithms
- Limited number of variants of algorithms
- Small number of test cases

Method

The How

Method

Benchmark

- Statistical approach **with multiple measurements**
- Code written and developed in Matlab

Algorithm

- Experimental **iterative** development process
- Code written and developed in Matlab

The Proposed Algorithm

Differential Estimation of Distribution

The Proposed Algorithm

- Improve upon Estimation of Distribution
- Add a second variation model inspired by Differential Evolution

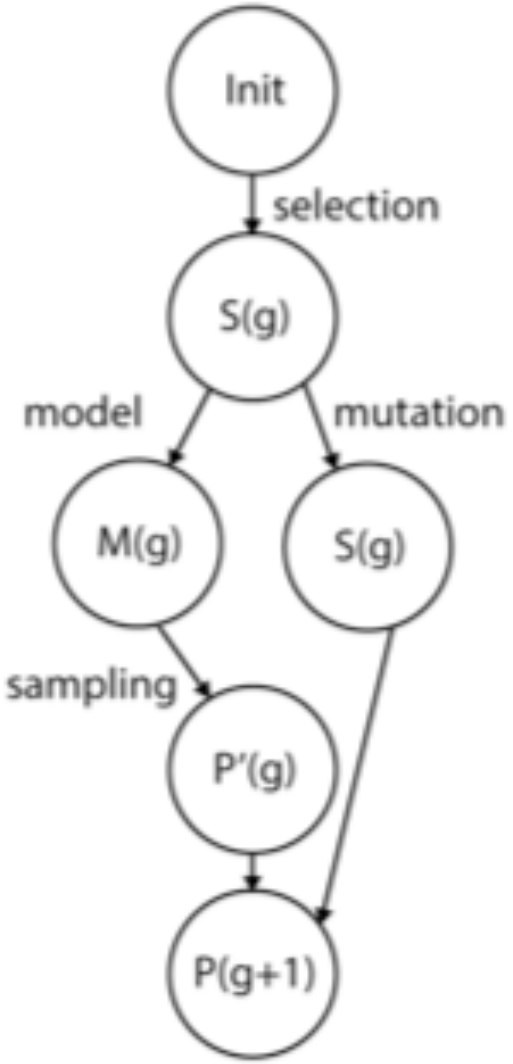


Figure 10: Flow-diagram for algorithm

Algorithm 6 Proposed algorithm

$g \leftarrow 0$
$P(g) \leftarrow$ initialize population with size N
repeat
  sort $P(g)$ by fitness
  $S(g) \leftarrow$ top t% of P(g)
  $mn \leftarrow$ calculate mean vector by dimension from $S(g)$
  $std \leftarrow$ calculate standard deviation vector by dimension from $S(g)$
  $M(g) \leftarrow$ normal distribution created from mn and std
  $P'(g) \leftarrow$ sample (100%-t%)*N individuals from M(g)
  for all Individuals x in $S(g)$ **do**
    Let a, b and c be three unique random individuals from $S(g)$ differing from x
    $v \leftarrow a + (b - c)$
    $u \leftarrow$ random blend of x with v (take at least 1 element from v)
    if $fitness(u) > fitness(x)$ **then**
      $x \leftarrow u$
    end if
  end for
  $P(g+1) \leftarrow$ merge $P'(g)$with $S(g)$
until Termination condition

Experimental Results and Evaluation

Mathematical Benchmark, ML Benchmark, Results, Discussion

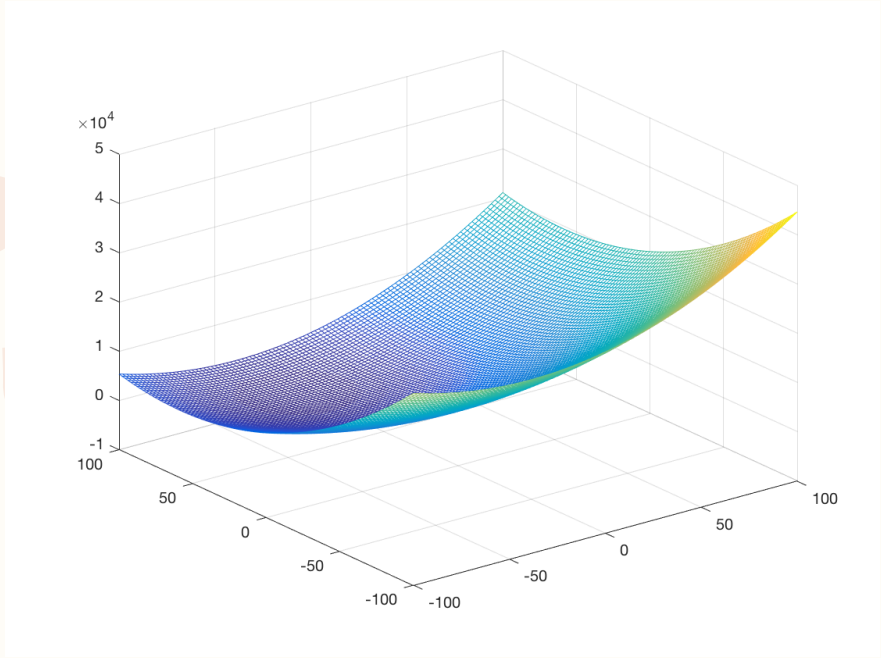
Mathematical Function Optimization

- **2005 CEC** competition test bed
- Based on popular **De Jong** test-bed
- Repeat measurements
- At **dimension 10, 30 and 50**

Parameter	Value (D=10)	Value (D=30)	Value (D=50)
Repeat measurements	30	30	30
Generations	667	1200	1429
Population Size	150	250	350

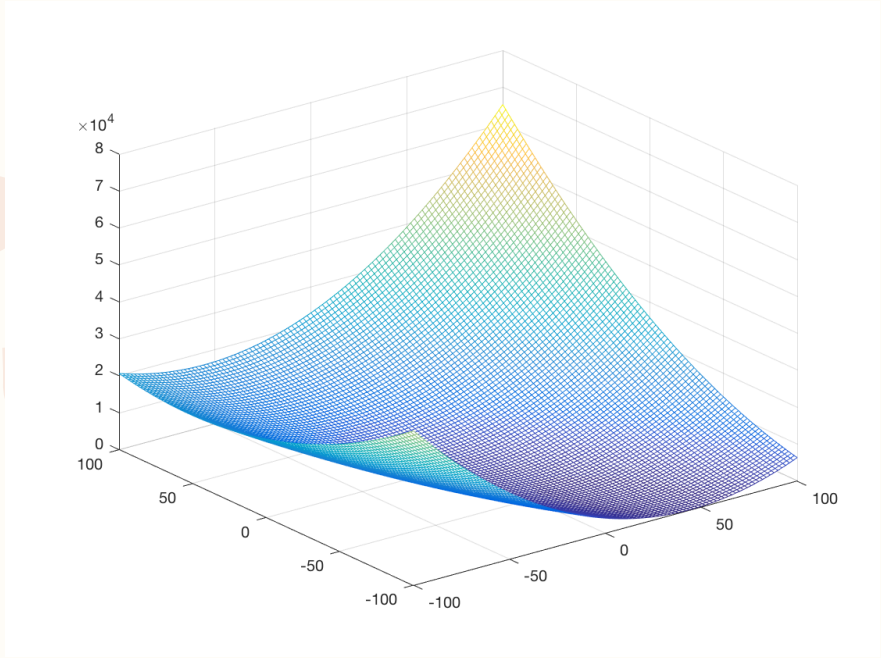
Table 1: Benchmark parameters for F_{1-10}

F1: Shifted Sphere Function



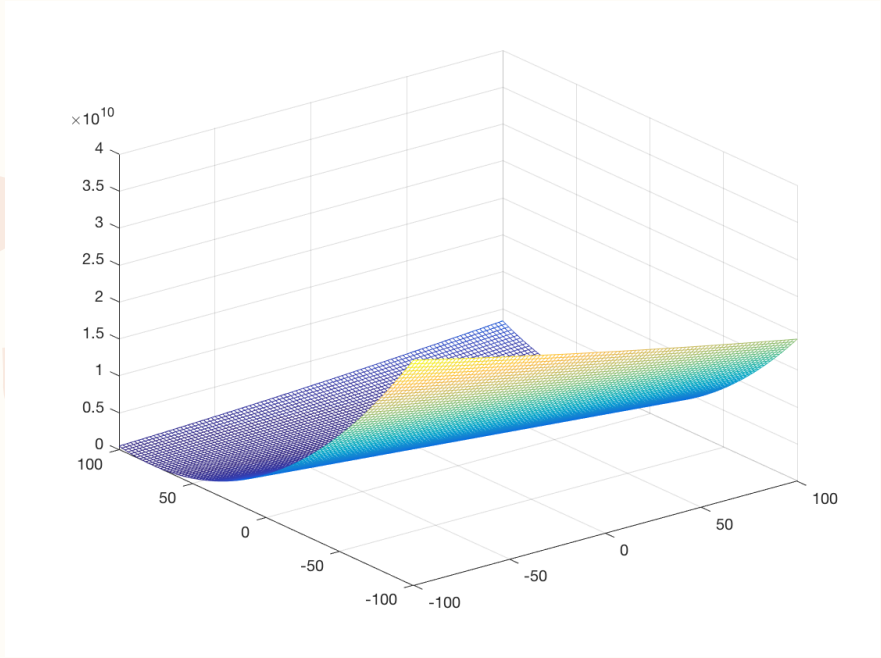
- Unimodal
- Separable

F2: Shifted Schwefel's Problem



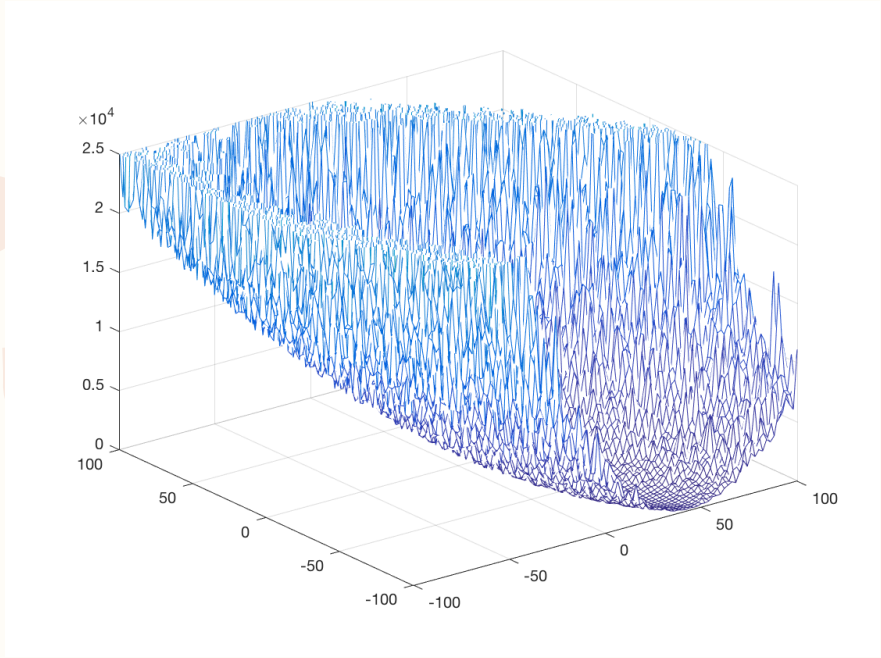
- Unimodal
- Non-Separable

F3: Shifted Rotated High Conditioned Elliptic Function



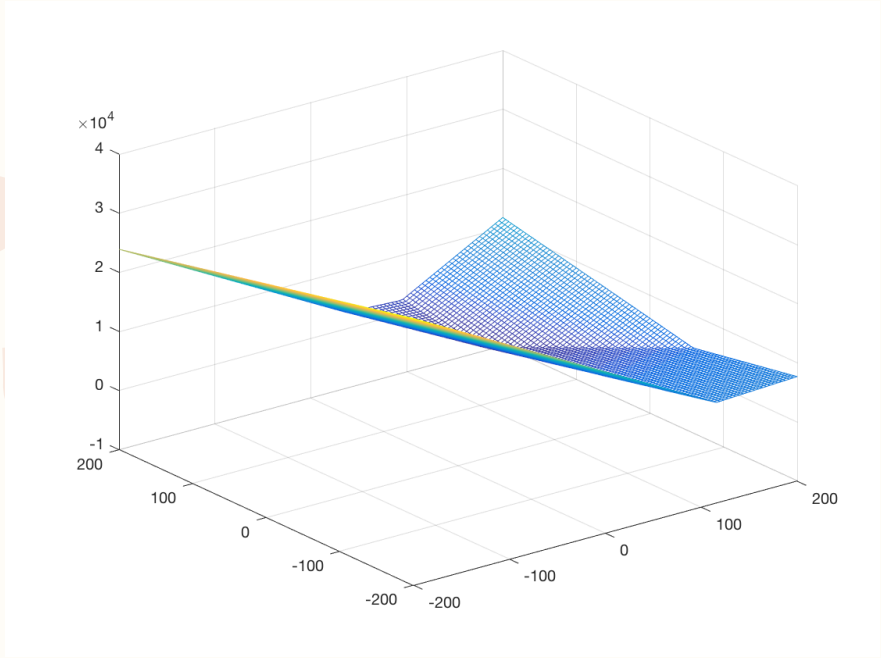
- Unimodal
- Non-Separable

F4: Shifted Schwefel's Problem with Noise in Fitness



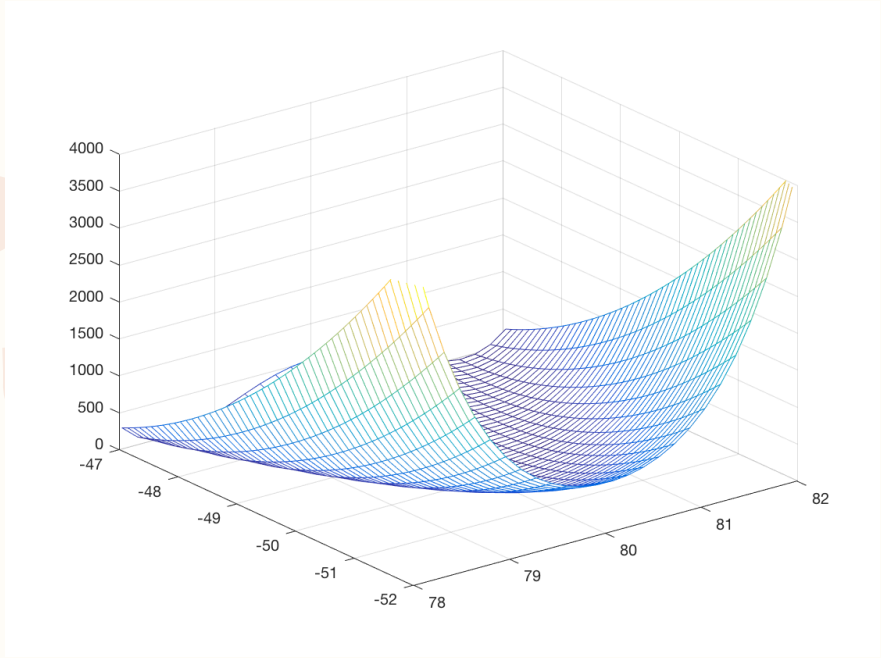
- Unimodal
- Non-Separable

F5: Schwefel's Problem with Global Optimum on Bounds



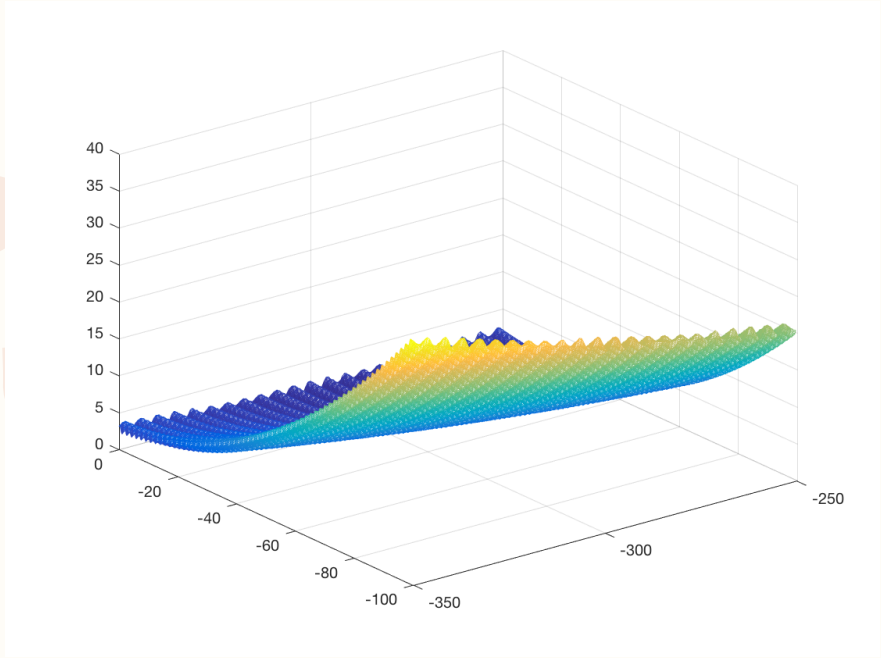
- Unimodal
- Non-Separable

F6: Shifted Rosenbrock's Function



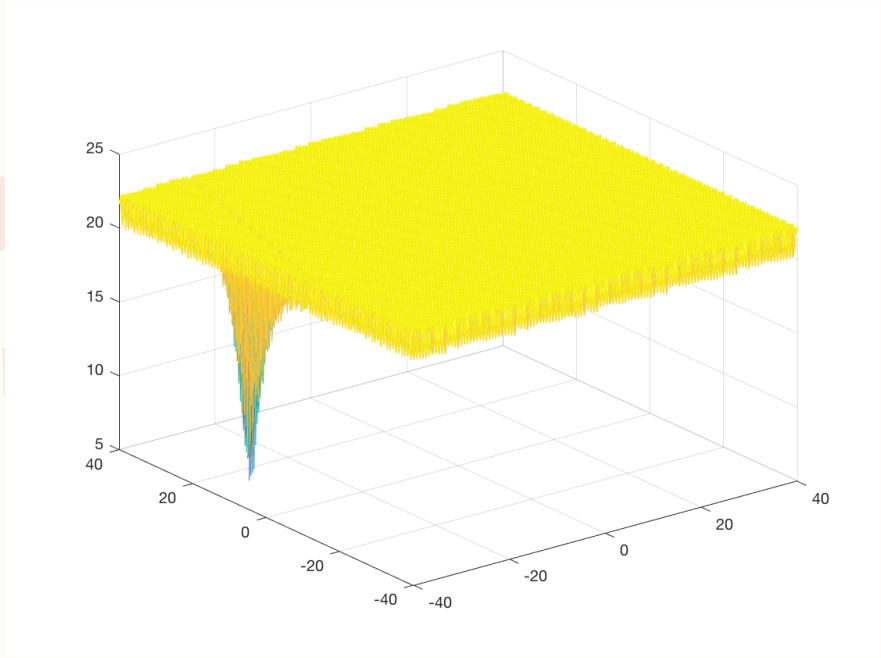
- Multimodal
- Non-Separable

F7: Shifted Rotated Griewank's Function without Bounds



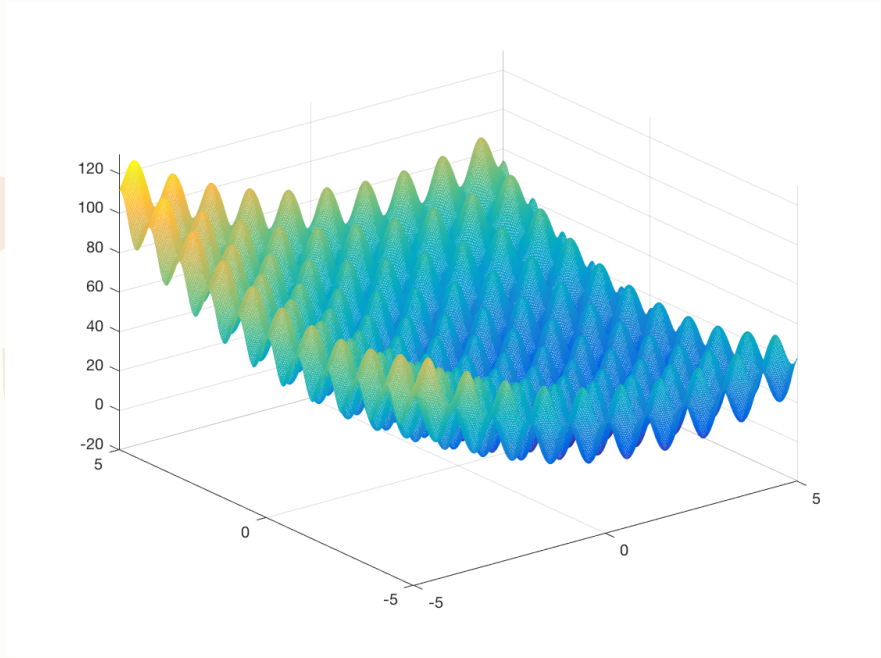
- Multimodal
- Non-Separable

F8: Shifted Rotated Ackley's Function with Global Optimum on Bounds



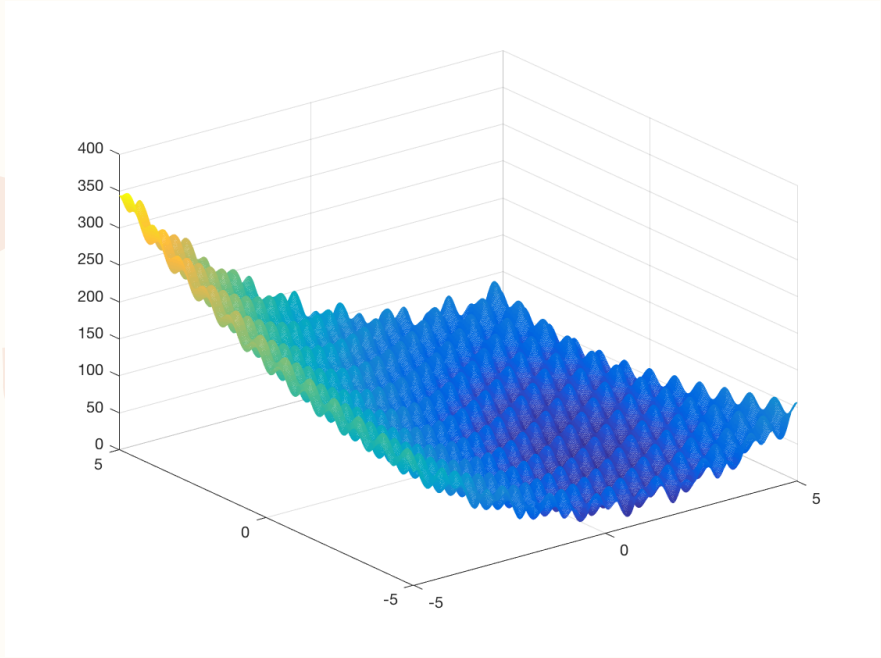
- Multimodal
- Non-Separable

F9: Shifted Rastrigin's Function



- Multimodal
- Separable

F10: Shifted Rotated Rastrigin's Function



- Multimodal
- Non-Separable

Function Approximation

- Training Feed-Forward Neural Networks to perform function optimization
- Matlab sample data-sets were used
- Repeat measurements
- Sum of squared errors (SSE) used as fitness function

$$sse(x,t) = \frac{1}{2n}\sum_{i=1}^{n}(y(x)-t)^2$$

Parameter	Value
Repeat measurements	5
Generations	250
Population Size	50

Table 2: Benchmark parameters for FA_{1-6}

Data-set	Description	Input-Dimension	Output-Dimension	Weight-Dimension
simplefit	Simple fitting	1	1	4
bodyfat	Body fat percentage	13	1	196
chemical	Chemical sensor	8	1	81
cho	Cholesterol	21	3	528
engine	Engine behavior	2	2	12
house	House value	13	1	196

Table 3: Data-sets for FA_{1-6}

Classification

- Training Feed-Forward Neural Networks to perform classification
- Matlab sample data-sets were used
- Repeat measurements
- Sum of squared errors used as fitness function

Parameter	Value
Repeat measurements	5
Generations	250
Population Size	50

Table 4: Benchmark parameters for CLS_{1-7}

Data-set	Description	Input-Dimension	Output-Dimension	Weight-Dimension
simpleclass	imple pattern recognition	2	4	18
cancer	Breast cancer	9	2	110
crab	Crab gender	6	2	56
glass	Glass chemical	9	2	110
iris	Iris flower	4	3	35
thyroid	Thyroid function	21	3	528
wine	Italian wines	13	3	224

Table 5: Data-sets for CLS_{1-7}

Clustering

- Training Feed-Forward Neural Networks to perform clustering
- Matlab sample data-sets were used
- Repeat measurements
- Sum of squared errors used as fitness function

Parameter	Value
Repeat measurements	5
Generations	250
Population Size	50

Table 6: Benchmark parameters for CLU_1

Data-set	Description	Input-Dimension	Output-Dimension	Weight-Dimension
simplecluster	Simple clustering	2	4	18

Table 7: Data-sets for CLU_1

Intelligent Game Controller

- Training Feed-Forward Neural Networks to control classic **game of "Snake"**
- **12-dimensional** vector represents the snakes senses
- **4-dimensional** vector represents the decision matrix

Parameter	Value
Individuals	140
Generations	1000
Grid dimension	10*10
Weight dimension	140
Repeat measurements	10
Repeat fitness measurements	5

Table 8: Benchmark parameters for FA_{1-6}

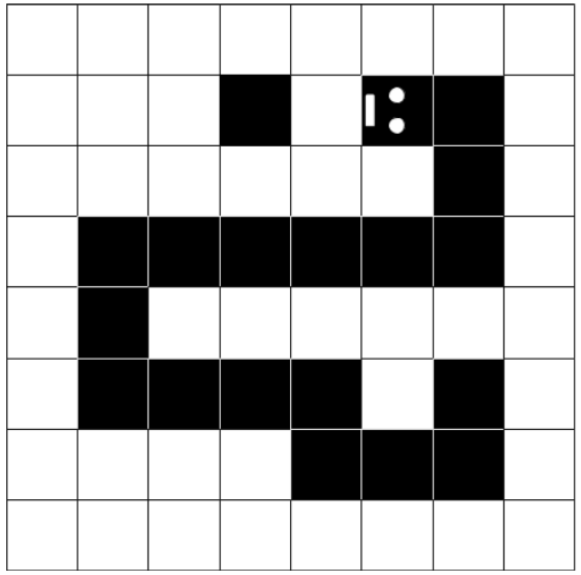


Figure 21: Snake $8 * 8$ game-field with snake hunting food

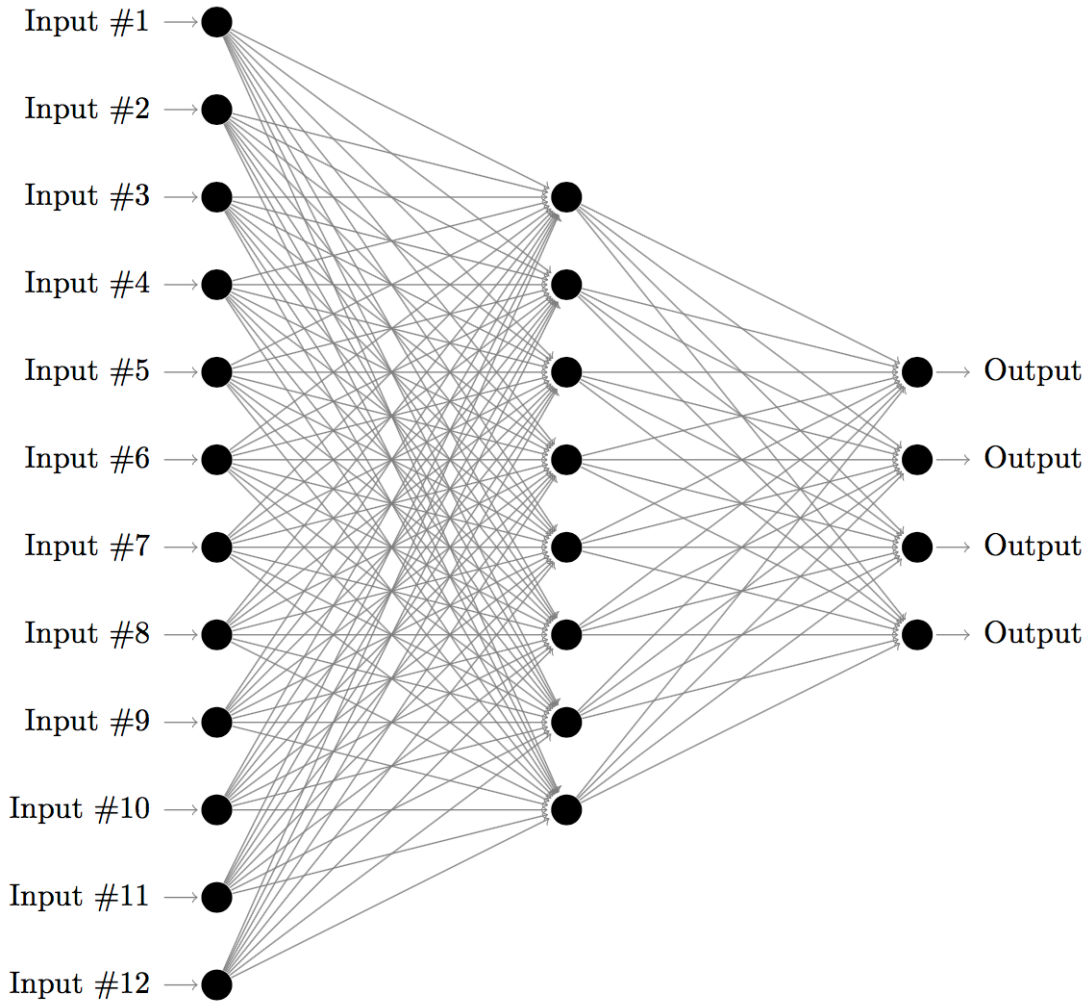


Figure 22: FFNN snake controller

$$fitness = \text{foodEaten} + \frac{1}{1 - e^{-\text{movesMade}}}$$

Function Optimization at D=10

F	DE		PSO		EDA		**DEDA**	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
1	3,88E-17	3,42E-17	1,65E-16	3,64E-16	**1,67E-27**	4,07E-28	0,00E+00	0,00E+00
2	1,83E-08	1,24E-08	2,07E-06	3,36E-06	2,51E+02	1,42E+02	**4,23E-27**	3,10E-27
3	5,19E-01	2,29E-01	4,88E+05	7,21E+05	1,09E+05	8,32E+04	**4,96E-02**	3,01E-02
4	3,36E-07	2,59E-07	6,31E-05	9,10E-05	3,21E+02	2,16E+02	**1,73E-26**	2,16E-26
5	**4,24E-13**	7,82E-13	4,94E+02	4,49E+02	1,67E+02	1,37E+02	6,57E+00	1,26E+01
6	**5,43E-05**	1,24E-04	2,11E+02	7,15E+02	1,34E+04	4,30E+04	6,71E+00	5,83E-01
7	4,87E-01	7,61E-02	2,74E-01	1,79E-01	**1,87E-01**	2,23E-01	3,41E-01	1,24E-01
8	2,04E+01	7,20E-02	**2,03E+01**	8,82E-02	2,04E+01	8,15E-02	2,04E+01	6,88E-02
9	2,50E+01	4,44E+00	**1,76E+00**	1,49E+00	2,35E+01	3,08E+00	1,80E+01	3,52E+00
10	3,26E+01	4,25E+00	**1,73E+01**	7,50E+00	2,11E+01	3,43E+00	2,05E+01	3,62E+00

Table 10: Benchmark results for F_{1-10} $D = 10$

Function Optimization at D=30

F	DE		PSO		EDA		**DEDA**	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
1	1,03E+00	1,53E-01	4,54E-10	7,66E-10	1,81E-25	7,66E-10	**8,58E-28**	1,40E-28
2	3,07E+03	6,91E+02	1,42E+03	7,01E+02	1,71E+03	7,01E+02	**4,84E+02**	1,80E+02
3	3,28E+07	5,84E+06	1,92E+07	1,12E+07	**4,93E+06**	1,12E+07	1,88E+07	4,86E+06
4	6,06E+03	1,23E+03	1,50E+04	6,87E+03	3,63E+03	6,87E+03	**6,74E+02**	1,96E+02
5	1,38E+03	3,54E+02	6,92E+03	2,55E+03	2,70E+03	2,55E+03	**1,31E+02**	9,21E+01
6	2,99E+03	1,24E+03	**5,30E+02**	8,60E+02	8,57E+04	8,60E+02	7,32E+03	2,21E+04
7	1,22E+00	5,36E-02	**5,77E-02**	5,18E-02	5,86E+00	5,18E-02	2,54E-01	4,25E-01
8	2,09E+01	4,24E-02	**2,09E+01**	5,74E-02	2,10E+01	5,74E-02	2,10E+01	5,41E-02
9	1,91E+02	9,58E+00	**2,53E+01**	5,97E+00	1,53E+02	5,97E+00	1,60E+02	8,27E+00
10	2,19E+02	1,09E+01	**1,14E+02**	3,78E+01	1,59E+02	3,78E+01	1,60E+02	8,53E+00

Table 11: Benchmark results for F_{1-10} $D = 30$

Function Optimization at D=50

F	DE		PSO		EDA		**DEDA**	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
1	5,64E+02	8,54E+01	1,20E-04	1,49E-04	1,04E-24	9,53E-26	**2,14E-26**	4,42E-27
2	5,00E+04	4,88E+03	3,69E+04	1,02E+04	**3,55E+03**	6,45E+02	3,96E+04	5,00E+03
3	2,52E+08	3,33E+07	8,68E+07	4,64E+07	**1,31E+07**	2,94E+06	1,96E+08	3,16E+07
4	7,02E+04	9,89E+03	1,12E+05	3,09E+04	**8,17E+03**	1,71E+03	5,08E+04	7,70E+03
5	1,33E+04	1,03E+03	1,48E+04	3,64E+03	4,31E+03	2,95E+02	**2,71E+03**	3,84E+02
6	7,19E+06	2,18E+06	1,62E+04	3,11E+04	**1,18E+04**	2,68E+04	2,23E+04	1,09E+05
7	3,32E+01	6,43E+00	1,05E+00	1,24E-01	1,25E+01	3,91E+00	**6,04E-01**	6,19E-01
8	2,12E+01	2,28E-02	**2,11E+01**	5,09E-02	2,11E+01	4,97E-02	2,11E+01	3,28E-02
9	3,94E+02	1,24E+01	**7,93E+01**	1,46E+01	3,14E+02	1,41E+01	3,27E+02	1,13E+01
10	4,48E+02	1,87E+01	**2,79E+02**	6,36E+01	3,20E+02	1,17E+01	3,27E+02	9,26E+00

Table 12: Benchmark results for F_{1-10} $D = 50$

Function Approximation

F	DE		PSO		EDA		**DEDA**	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
simplefit	**6,91E-01**	1,67E-16	6,91E-01	3,90E-04	6,64E+00	7,12E-01	1,09E+00	1,13E-01
bodyfat	1,92E+01	2,59E+00	1,90E+01	2,14E+00	1,96E+01	2,11E+00	**1,69E+01**	1,26E+00
chemical	2,58E+01	5,53E-04	2,46E+01	1,97E+00	1,15E+05	2,38E+03	**2,43E+01**	2,11E+00
cho	2,07E+03	2,16E+02	**1,86E+03**	4,84E+01	5,32E+03	6,24E+02	1,97E+03	1,10E+02
engine	1,29E+05	7,09E+04	7,86E+04	6,86E+03	9,91E+05	2,59E+03	**5,61E+04**	1,27E+04
house	3,19E+01	4,81E-01	**2,00E+01**	1,93E+00	2,45E+01	2,68E+00	2,03E+01	5,90E+00

Table 13: Benchmark results for F_{1-10}

Classification

F	DE		PSO		EDA		**DEDA**	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
simpleclass	1,76E-01	1,29E-02	1,83E-01	7,07E-03	1,97E-01	2,56E-02	**1,43E-01**	7,76E-03
cancer	7,74E-02	9,95E-03	6,48E-02	1,40E-02	**3,23E-02**	9,32E-04	5,26E-02	6,65E-03
crab	1,83E-01	3,25E-02	1,27E-01	2,54E-02	**1,20E-01**	3,70E-02	6,35E-02	5,26E-02
glass	1,25E-01	3,54E-02	**7,45E-02**	1,95E-02	9,71E-02	2,11E-02	9,91E-02	4,33E-02
iris	1,63E-01	7,31E-03	1,26E-01	3,22E-02	1,31E-01	3,18E-02	**6,67E-02**	5,23E-02
thyroid	2,00E-01	3,66E-02	2,25E-01	4,84E-02	**1,17E-01**	1,24E-02	1,80E-01	3,57E-02
wine	3,45E-01	2,33E-02	3,06E-01	2,25E-02	2,79E-01	2,52E-02	**2,59E-01**	7,80E-02

Table 14: Benchmark results for CLS_{1-6}

Clustering

F	DE		PSO		EDA		**DEDA**	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
simplecluster	1,65E-01	1,39E-02	1,74E-01	1,57E-02	2,17E-01	2,05E-02	**1,51E-01**	8,12E-03

Table 15: Benchmark results for CLU_1

Intelligent Game Controllers

F	DE		PSO		EDA		**DEDA**	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
snake	2,32E+01	2,19E+00	**2,92E+01**	4,16E+00	2,78E+01	1,34E+01	2,69E+01	3,59E+00

Table 16: Benchmark results for IGC_1

Discussion

- All algorithms perform comparatively well
- **PSO and DEDA perform better** then the rest on average
- PSO is best at multimodal functions and game controllers
- **DEDA is best at unimodal functions and optimizing neural networks**
- EDA performs somewhat worse than DEDA
- EDA performs well at higher dimensional mathematical optimization

Conclusions

The End

Two-Fold Conclusion

Benchmark

- DE, PSO, EDA and DEDA were compared on mathematical and ML benchmarks
- **PSO** and **DEDA** were the best performers overall
- **DEDA showed good results** in the benchmarks

New Algorithm

- Inspired by **EDA**
- **Extended** mutation and recombination operator inspired by DE
- **Performed well at ML** tasks (approximation, clustering, classification, etc.)

Future Work

What's next?

Future Work

- There exist many **variations** of the tested algorithms
- It would be interesting to **compare the different variations** of the best performing algorithms in the benchmarks to try to improve performance
- It would be interesting to try **different approaches at hybridizing** differential mutation and recombination with the principles of model building and distribution sampling to find the most efficient approach

The End

Questions?