# How would Darwin solve an optimization problem?

Machine Learning Meeting

15 Apr 2019

# Why should we still care about evolutionary algorithms?

Machine Learning Meeting

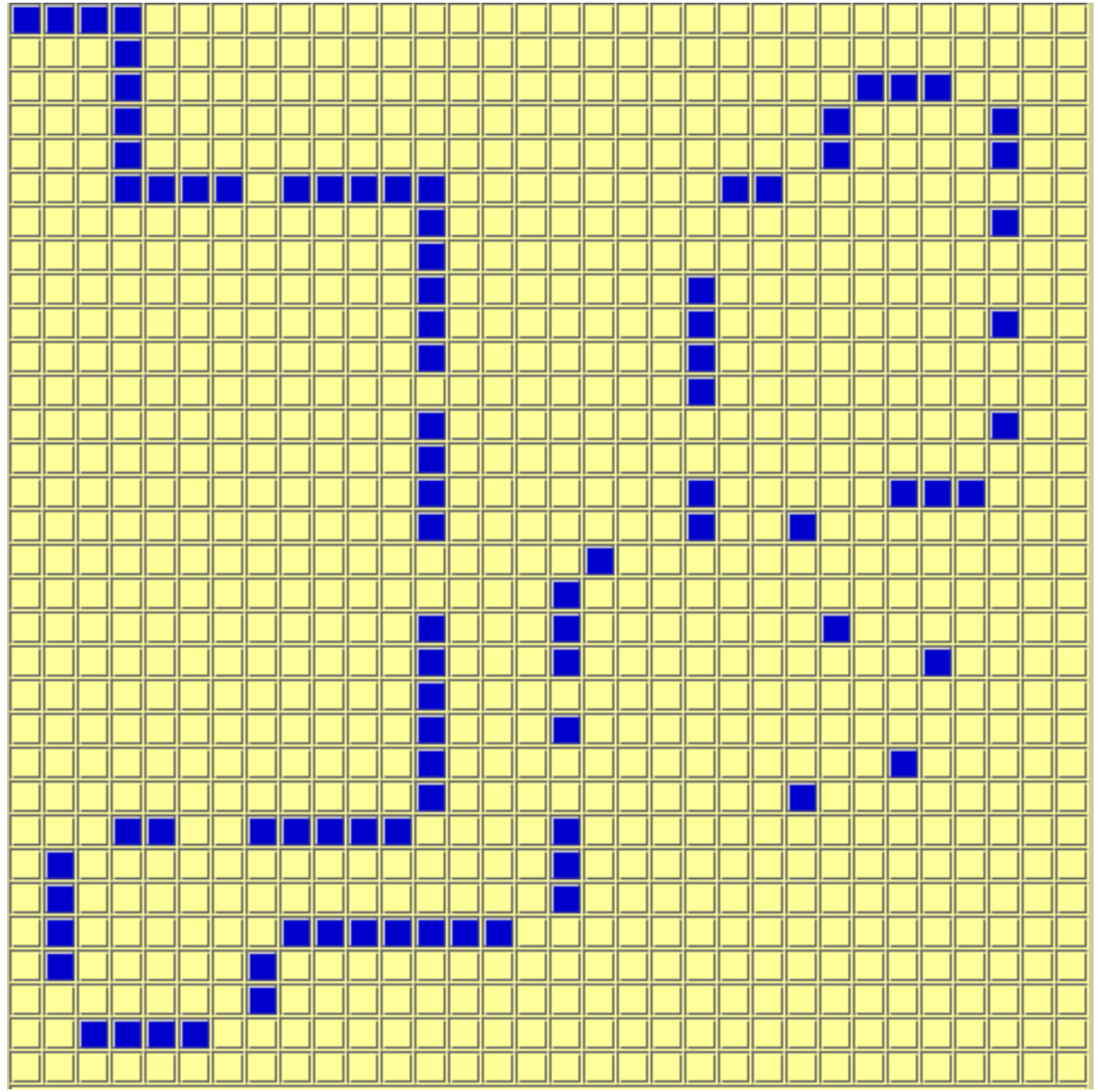
15 Apr 2019

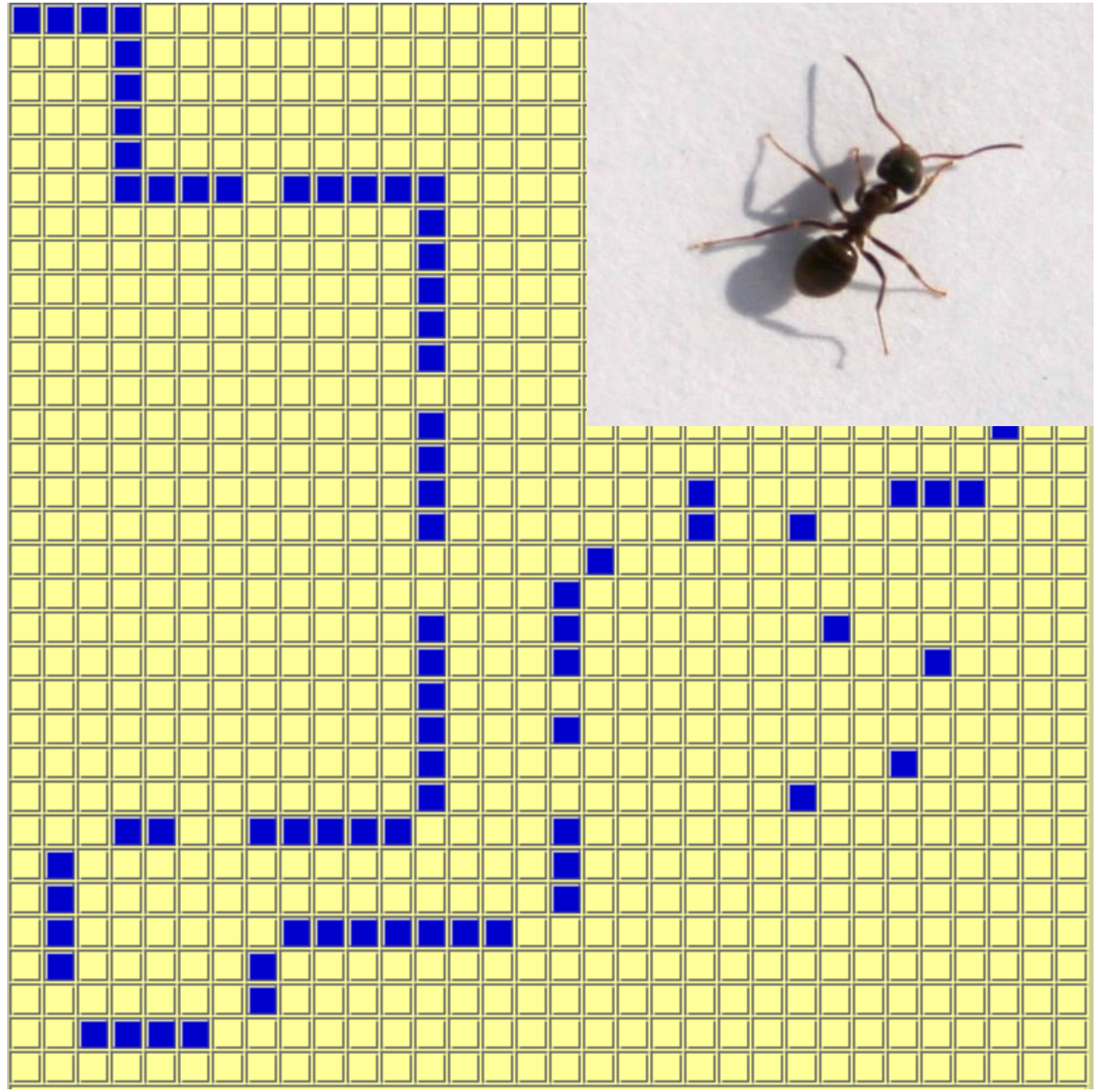

## IEEE CEC 2020

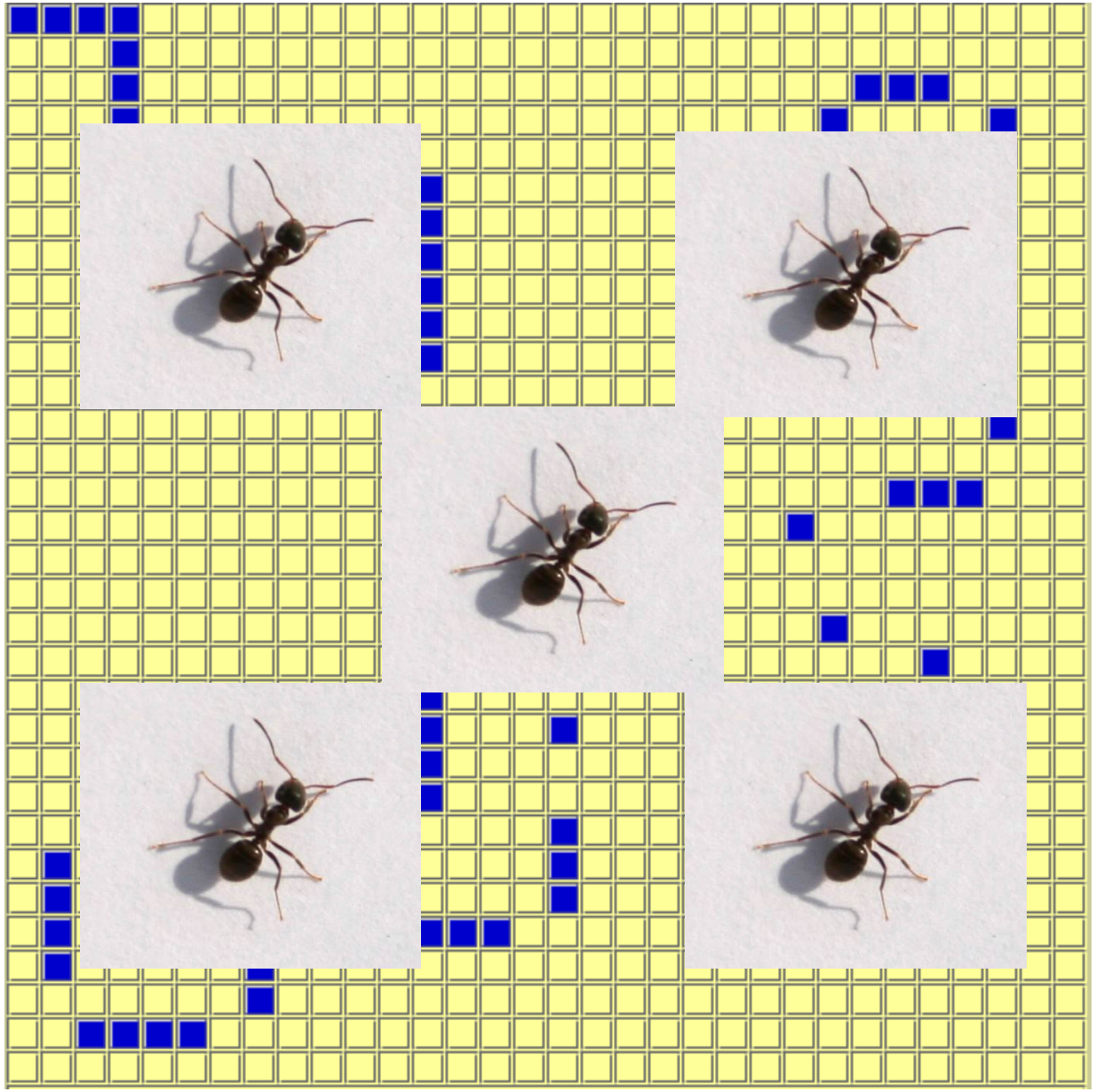
The IEEE Congress on Evolutionary Computation (IEEE CEC) covers all topics in evolutionary computation from theory to real-world applications.

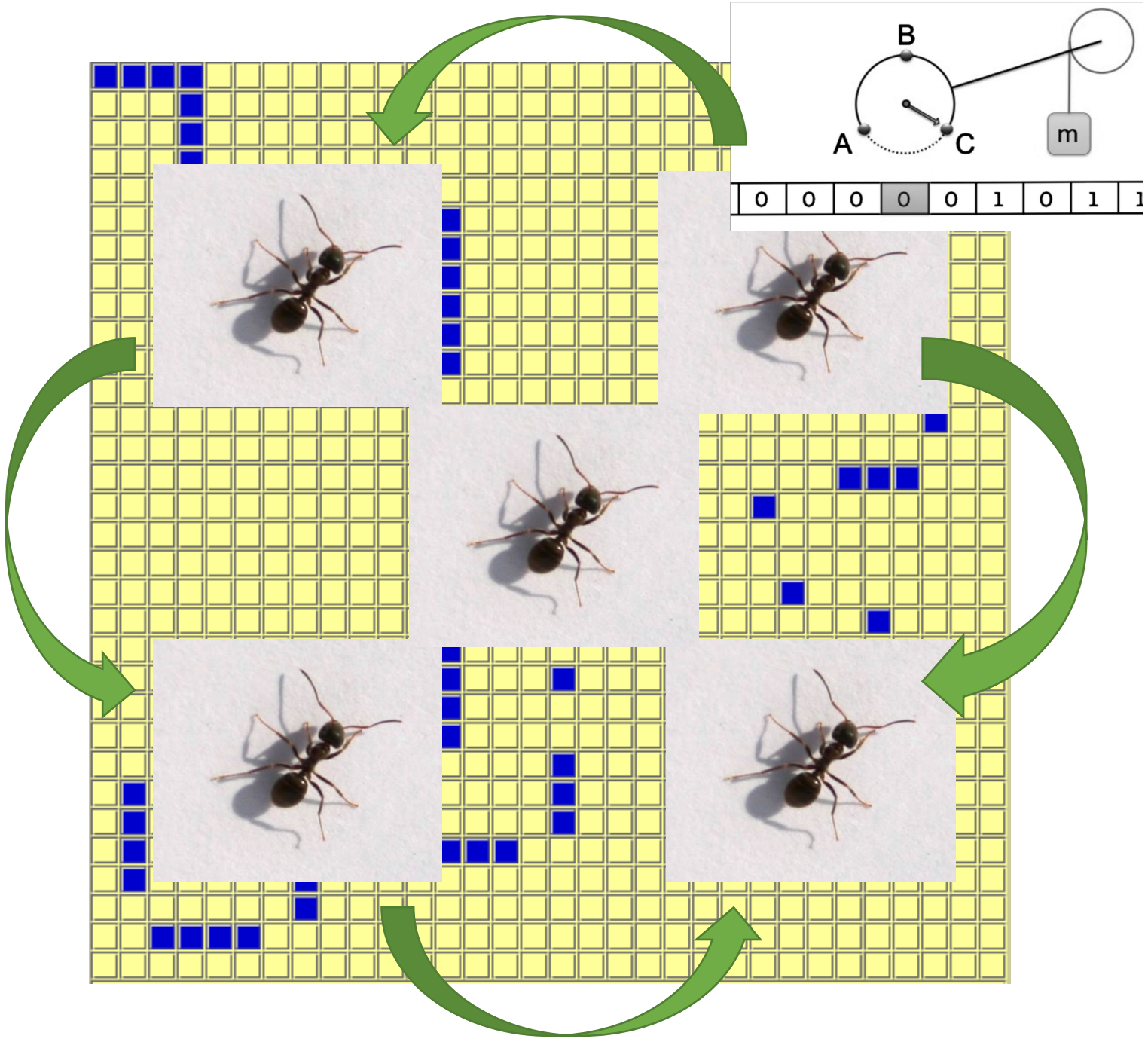
# For this presentation, I will be borrowing heavily from the following resources

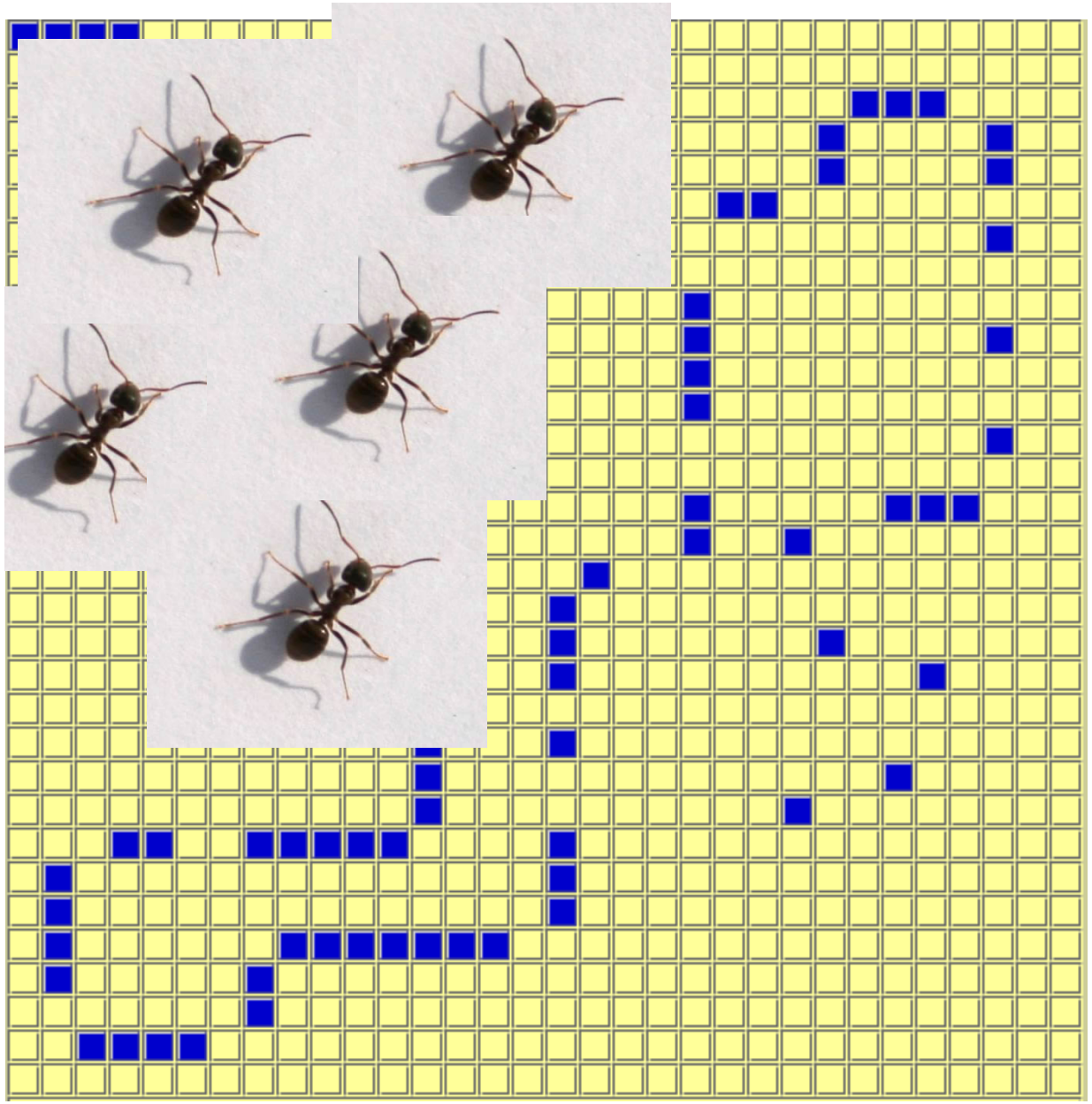
- https://link.springer.com/chapter/10.1007/11666806_20
- https://www.semanticscholar.org/paper/Particle-Swarm-Optimization-Kennedy/20a5cda34f158ace5881d3f6635c3eb5b531c199
- https://pdfs.semanticscholar.org/2eec/dd4f717cdb46f38b3c77c50e700a946fdecb.pdf
- https://pdfs.semanticscholar.org/b007/d1e0940baff34282e3c45c9843eb09a1239e.pdf
- https://www.sciencedirect.com/science/article/pii/B9780123850850000105
- https://medium.com/analytics-vidhya/implementing-particle-swarm-optimization-pso-algorithm-in-python-9efc2eb179a6








B
A
C
m
0 0 0 0 0 1 0 1 1



# A Brief Tour of Evolutionary Computation

- *Evolutionary computation:* Machine learning optimization and classification paradigms roughly based on mechanisms of evolution such as biological genetics and natural selection
- EC paradigms utilize a population of points (potential solutions) in their search
- EC paradigms use direct "fitness" information instead of function derivatives or other related knowledge
- EC paradigms use probabilistic, rather than deterministic, transition rules

https://pdfs.semanticscholar.org/2eec/dd4f717cdb46f38b3c77c50e700a946fdecb.pdf

# Pros / Motivation

- Naïve first approach to a problem
- Gradient-free approach
- Analytic, closed form for objective function not required
- Perfect for parallel implementations
- ~~Give outsiders a false impression that this is biophysics related~~

1)GA
2)MOGA
3)PSO

# Genetic Algorithm 101

# Encoding

## Chromosome

$$[p_1^i, p_2^i, \cdots, p_N^i]$$

## Fitness function

$$fitness = f_i(p_1^i, p_2^i, \cdots, p_N^i)$$

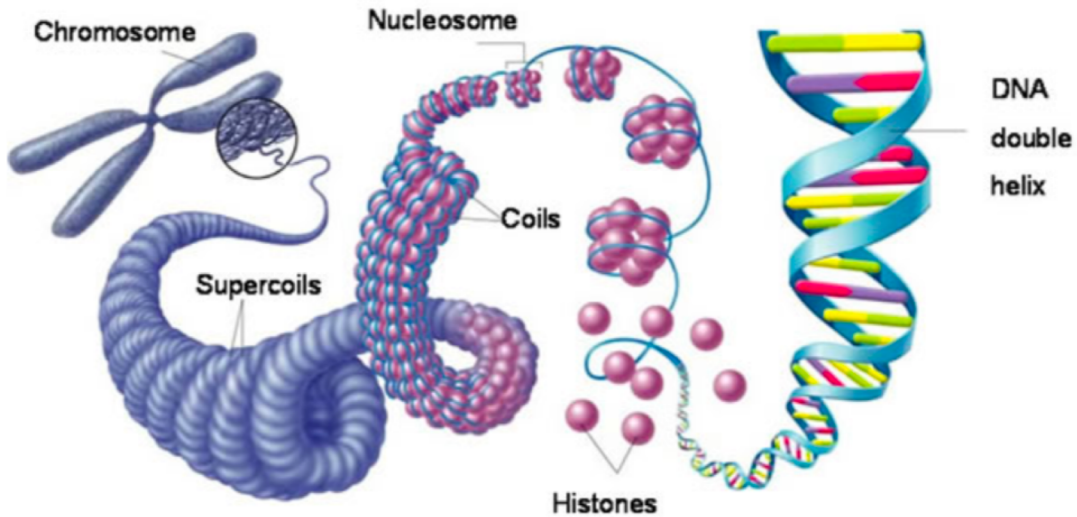



Figure: https://pmgbiology.files.wordpress.com/2015/10/00-eukaryotic-chromosomes.jpg

# Natural selection

## Survival of the fittest

- Selection rate (0.5?)
- Threshold

## Parent selection

- Tournament
- Rank weighting
- Fitness weighting

$$P_n^r = \frac{N' + 1 - n}{\sum_{n'=1}^{N'} n'} \quad ; \quad P_j^f = \frac{f_j}{\sum_i f_i}$$

No sorting is needed for both **Threshold** and **Tournament**.

# Mating

## Single Point Crossover

0000000000 0001111111

1111111111 1110000000

## Double Point Crossover

0000000000 0001111100

1111111111 1110000011

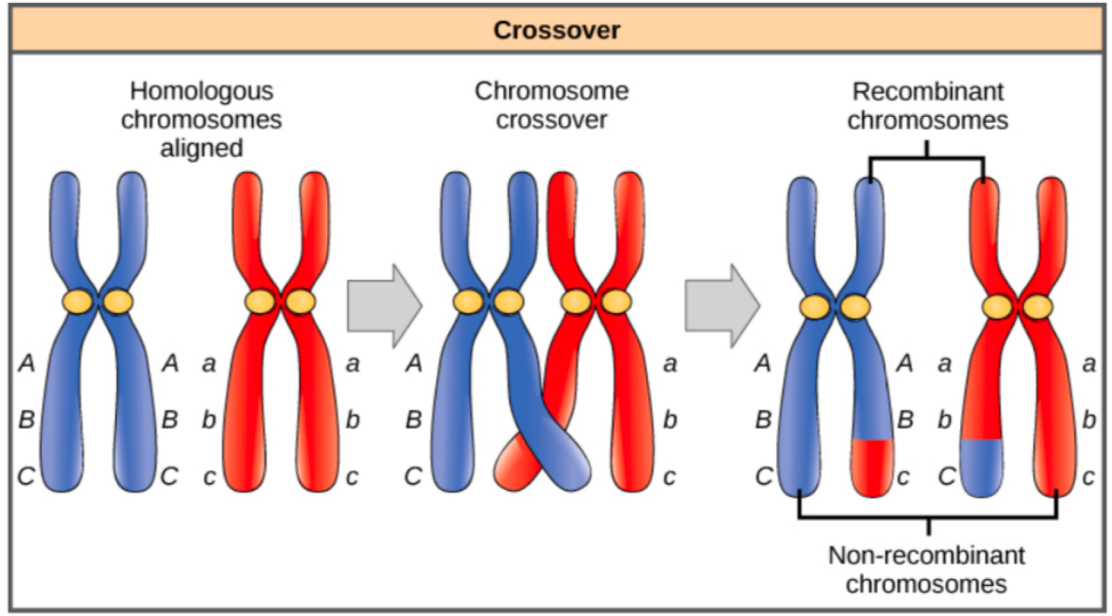



Figure: https://s-media-cache-ak0.pinimg.com/originals/9d/7c/d9/9d7cd91abd65e9c7fa857cf6c6d13

# Mutation

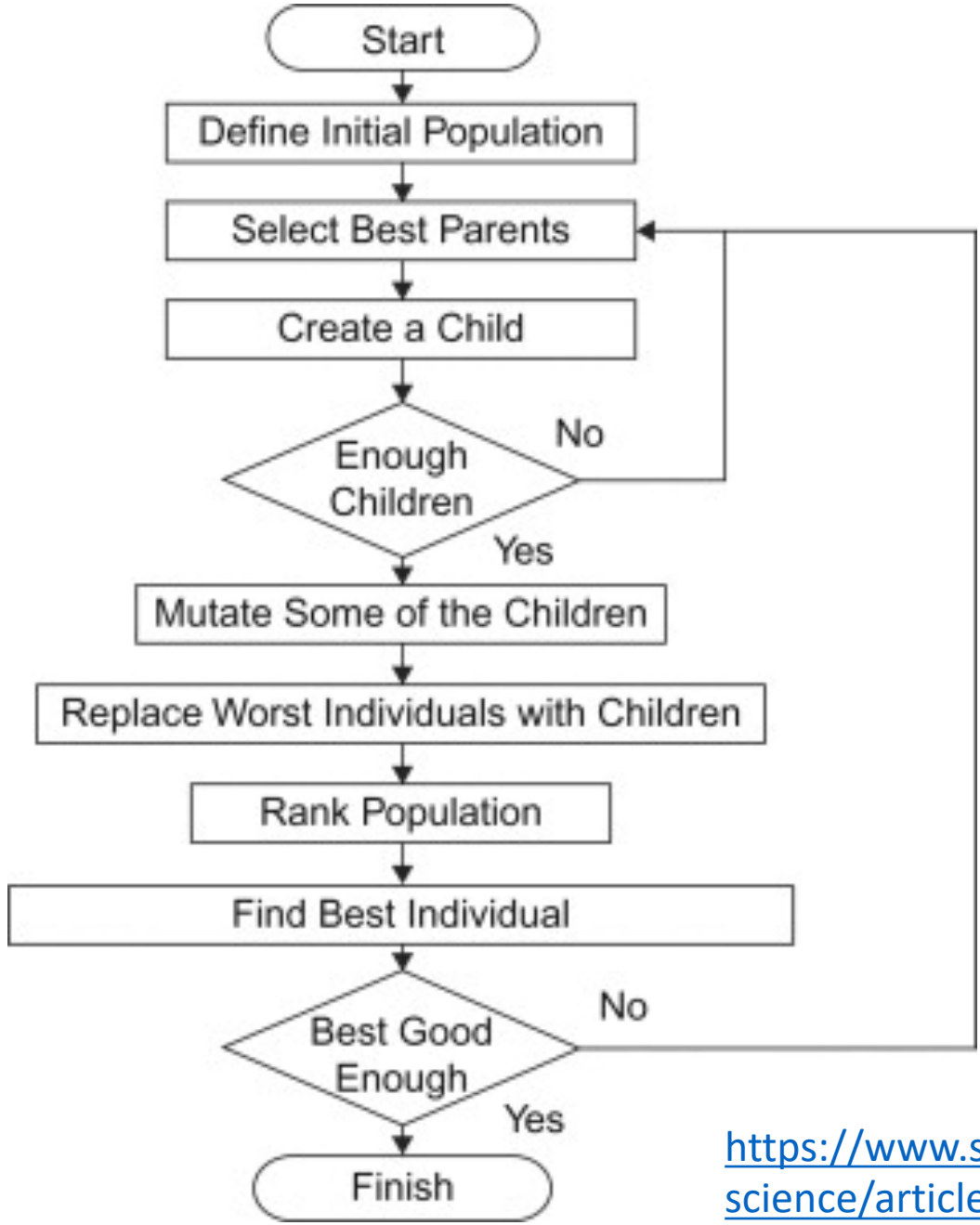
- Elitism (mutation rate = 0.2)

| | |
|---|---|
| 00000 | 00000 |
| 00000 | 01100 |
| 00000 | 00000 |
| 00000 | 00001 |
| 00000 | 00100 |
| 00000 | 00100 |

- Normal distribution

$$p'_n = p_n + \sigma N_n(0, 1)$$

- Uniform deviate
- Mutation only genetic algorithm (MOGA)




https://www.sciencedirect.com/science/article/pii/B9780123850850000105

# Mutation-Only Genetic Algorithm (MOGA)

Disclaimer: This is an advertisement

ACO-SI - Ant Colony Optimization and Swarm Intelligence

CS - Complex Systems (Artificial Life/Artificial Immune Systems/ Robotics/Evolvable Hardware)

DETA - Digital Entertainment Technologies and Arts

ECOM - Evolutionary Combinatorial Optimization and Metaheuris

EML - Evolutionary Machine Learning

EMO - Evolutionary Multiobjective Optimization

ENUM - Evolutionary Numerical Optimization

GA - Genetic Algorithms

GECH - General Evolutionary Computation and Hybrids

GP - Genetic Programming

RWA - Real World Applications

SBSE - Search-Based Software Engineering

THEORY - Theory

https://gecco-2019.sigevo.org/index.html/Program+Tracks

# Some definitions

- Define population $A_{ij}$(t), $i$=1,2,…,$N$; $j$=1,2,…,$L$
- Solution population size :$N$
- Binary encoding with length $L$
- Row mutation probability $a_i$, column mutation probability $b_j$
- Mutation Matrix: $M_{ij}(t) = a_i(t)b_j(t)$
- MOGAR vs MOGAC
- Investment portfolio ($\gamma$, 1- $\gamma$)

https://link.springer.com/chapter/10.1007/11666806_20

# Traditional Genetic Algorithm in MOGA formulation

- The population is partitioned into three categories
  - Fit survivors: $N_1=c_1N \rightarrow A_{ij}(t+1)=A_{ij}(t)$ for $i \in [1,N_1]$
  - Children: $N_2=c_2N$
  - Randomly generated chromosome: $N_3=N-N_1-N_2$
- Mutation Matrix is time independent
  - For i=1,…,$N_1$: $M_{ij}=0$
  - For i=$N_1$+1,…,$N_1$+$N_2$: $M_{ij}$=m ∈ (0,1)
  - For i = $N_2$+1,…,$N$: $M_{ij}=1$

# Mutation Probability

- Exploration vs Exploitation

$$a_i(t) = (i-1)/(N-1)$$

- Loci Statistics:

$$p_{jX} = \frac{\sum_{k=1}^{N}(N+1-k) \times \delta_{kj}(X)}{\sum_{m=1}^{N} m}$$

- Column mutation:

$$b_j = \frac{1 - |p_{j0} - 0.5| - |p_{j1} - 0.5|}{\sum_{j'=1}^{N} b_{j'}}$$

# MOGAR (MOGA by Row)

- Sort the locus (column in $A$) in descending order of $b_j$
- Flip $N$ coins to determine which chromosome to mutate based on $a_i$
- If rand()<$a_i$, flip the first $a_iL$ spins of the $i$-th row in $A$

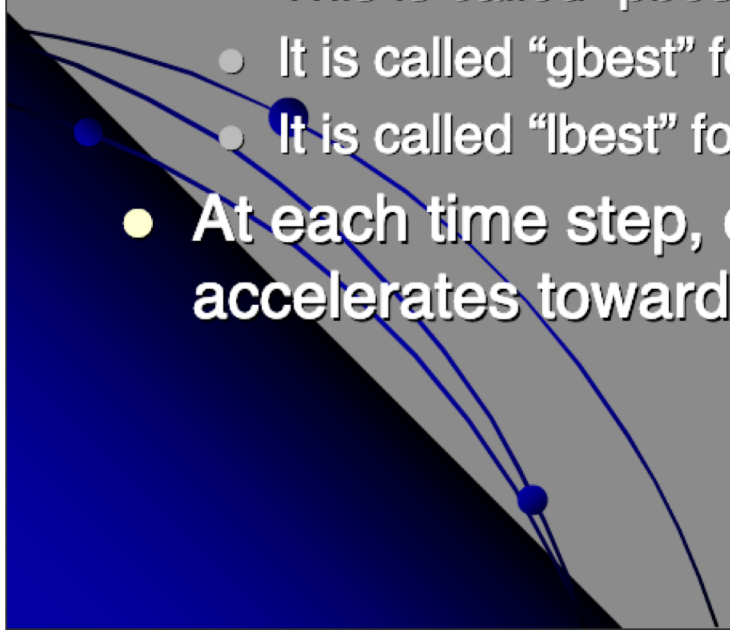
# MOGAC (MOGA by Column)

- Sort the chromosomes (rows in $A$) in descending order of $a_i$
- Flip $L$ coins to determine which locus to mutate based on $b_j$
- If rand()<$b_j$, flip the first $b_jN$ spins of the $j$-th column in $A$

# Particle Swarm Optimization (PSO)

# Features of Particle Swarm Optmization

- Population initialized by assigning random positions *and* velocities; potential solutions are then *flown* through hyperspace.
- Each particle keeps track of its "best" (highest fitness) position in hyperspace.
  - This is called "pbest" for an individual particle
  - It is called "gbest" for the best in the population
  - It is called "lbest" for the best in a defined neighborhood
- At each time step, each particle stochastically accelerates toward its pbest and gbest (or lbest).



https://pdfs.semanticscholar.org/2eec/dd4f717cdb46f38b3c77c50e700a946fdecb.pdf

# Particle Swarm Optimization: Swarm Types

- In his paper, [Kennedy, J. (1997), "The Particle Swarm: Social Adaptation of Knowledge", Proceedings of the 1997 International Conference on Evolutionary Computation, pp. 303-308, IEEE Press.]
- Kennedy identifies 4 types of PSO based on $\varphi 1$ and $\varphi 2$ .
- Given:

$$v_{id} = v_{id} + \varphi 1 * rnd() * (p_{id} - x_{id}) + \varphi 2 * rnd() * (p_{gd} - x_{id});$$

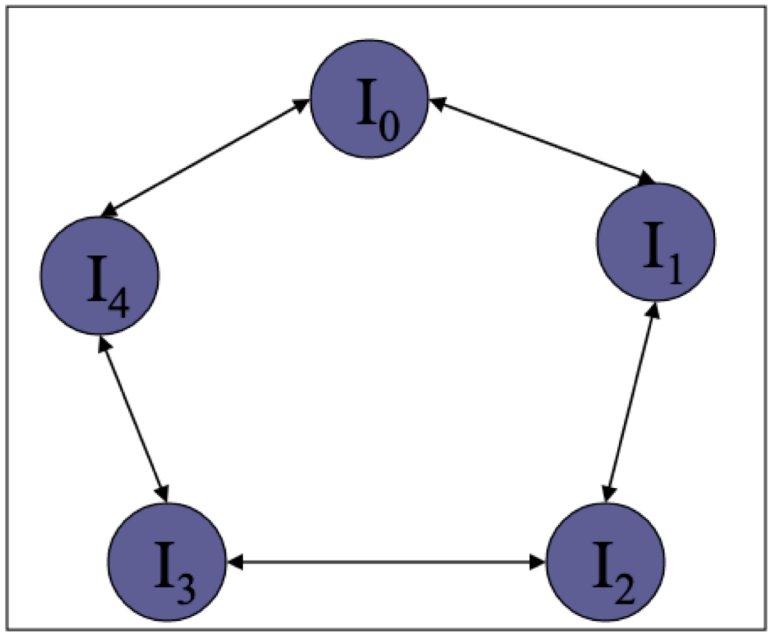
$$x_{id} = x_{id} + v_{id};$$

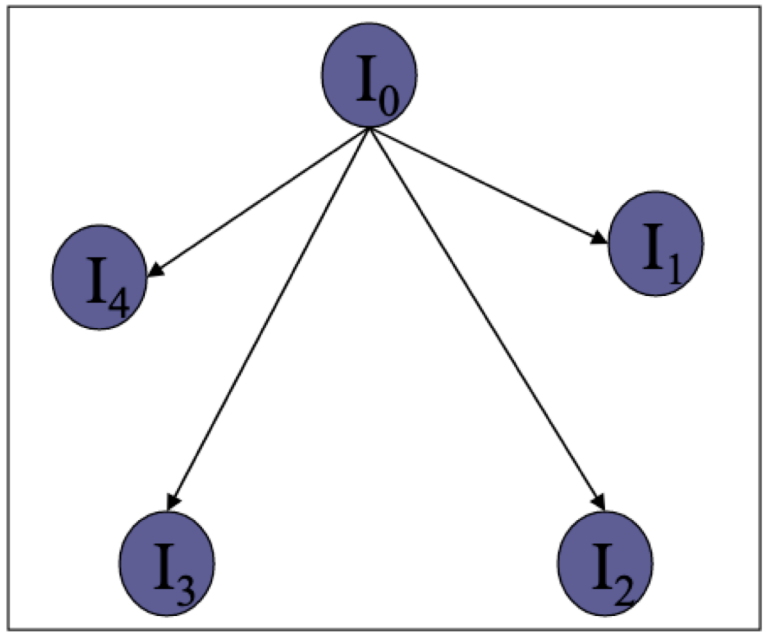
  - Full Model ($\varphi 1, \varphi 2 > 0$)
  - Cognition Only ($\varphi 1 > 0$ and $\varphi 2 = 0$),
  - Social Only ($\varphi 1 = 0$ and $\varphi 2 > 0$)
  - Selfless ($\varphi 1 = 0$, $\varphi 2 > 0$, and $g \neq i$)

https://pdfs.semanticscholar.org/b007/d1e0940baff34282e3c45c9843eb09a1239e.pdf

# Particle Swarm Optimization: Swarm Topology

- In PSO, there have been two basic topologies used in the literature
  - Ring Topology (neighborhood of 3)
  - Star Topology (global neighborhood)




https://pdfs.semanticscholar.org/b007/d1e0940baff34282e3c45c9843eb09a1239e.pdf

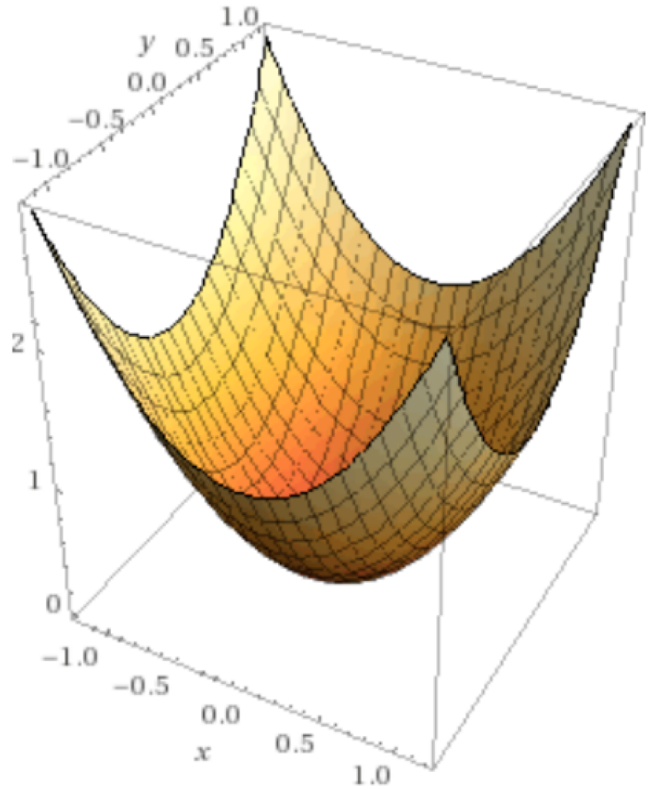
# Implementing the Particle Swarm Optimization (PSO) Algorithm in Python

Iran Macedo Follow

Dec 24, 2018 · 6 min read

https://medium.com/analytics-vidhya/implementing-particle-swarm-optimization-pso-algorithm-in-python-9efc2eb179a6

```
The best solution is: [ 5.06237850e-04 -1.14851873e-05] in
n_iterations: 22
```




3D graphic of $x^2+y^2$ (Wolframalpha)

Code:

https://gist.githubusercontent.com/IranNeto/21542660d740ac02dfce2f6aeb11ebeb/raw/09e718c23ce5c2f9180b3240d10c99c50a24d14f/pso.py

```
iran@iran-300E5M-300E5L:~/Documents/IA
Inform the number of iterations: 50
Inform the target error: 1e-6
Inform the number of particles: 30
```

# Where do we go from here?

- Hyperparameters
- Model training (ANN)
- Biophysics related applications

- https://www.sciencedirect.com/science/article/pii/S0926580507000064/#bbib12
- https://link.springer.com/chapter/10.1007/0-387-27705-6_6