



# Solving impossible problems

With evolutionary computation

Sondre A. Engebråten – PhD Candidate, University of Oslo.  
Researcher at the Norwegian Defence Research Establishment

# What are evolutionary methods?

# How do infants solve problems?

No background knowledge



No way of asking for help

New and unknown problem

# Trial and error problem solving

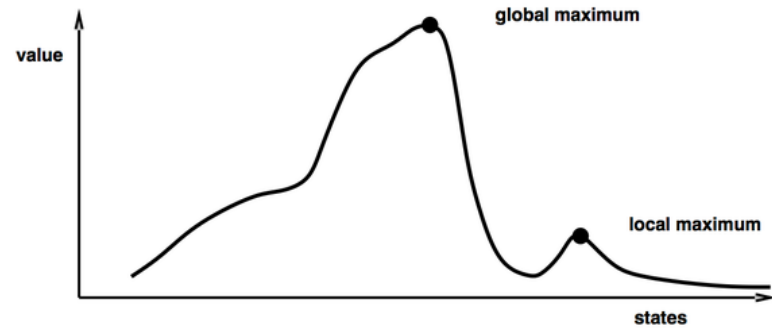
# What is trial and error?

- Trying many solutions
- Keeping good solutions
- Discarding bad solution
- Keep tweaking good solutions



# What is evolutionary computation?

- A way to solve complex problems
- A function optimizer
- A heuristic search method
- A hill climber optimization method



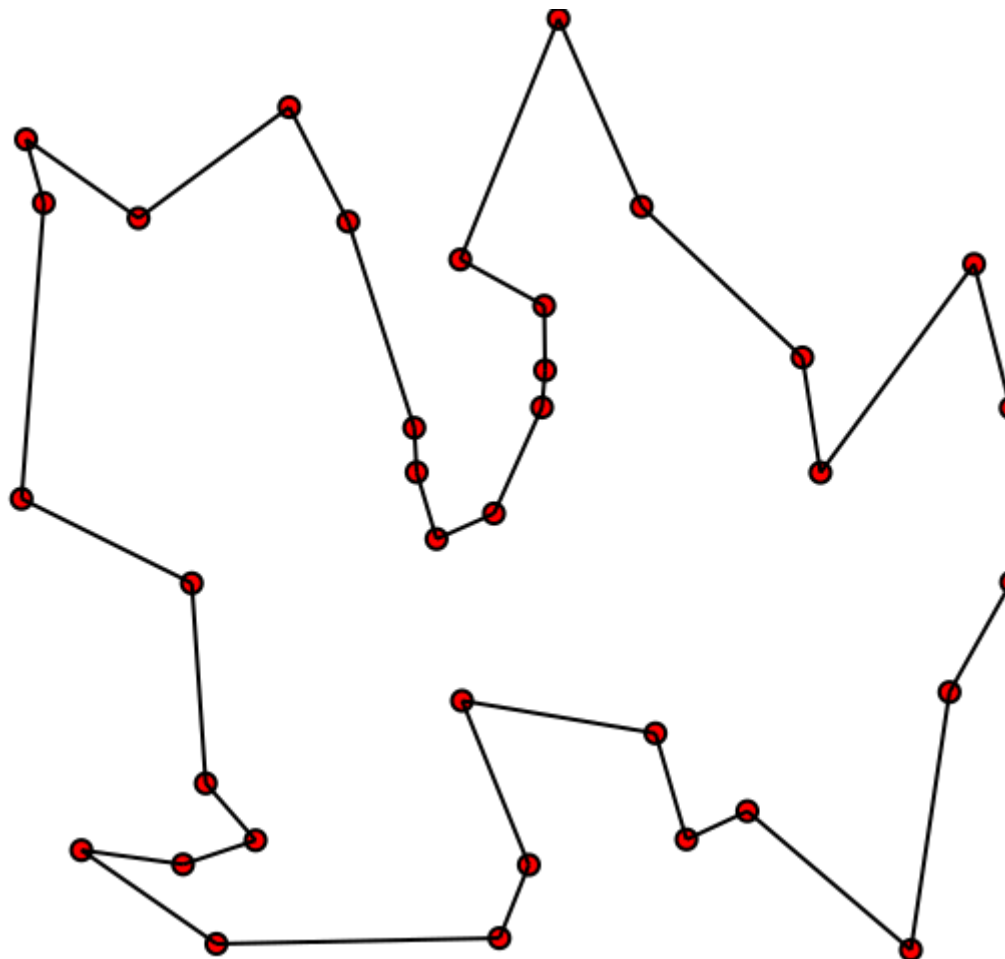
# What is not evolutionary computation?

- A no prior information, ultimate solver for all problems
- A free lunch
- A way to evolve little creatures inside a computer



What has evolutionary computation been used for?

# Travelling salesman problem TSP (NP-hard)



$N = 35$      $N! = 10333\ 147966386\ 144929666\ 651337523\ 200000000$



# NASA evolved antenna

- Antenna design is hard
- Simulating antenna is also hard
- Describing how, or more importantly why antennas work is hard

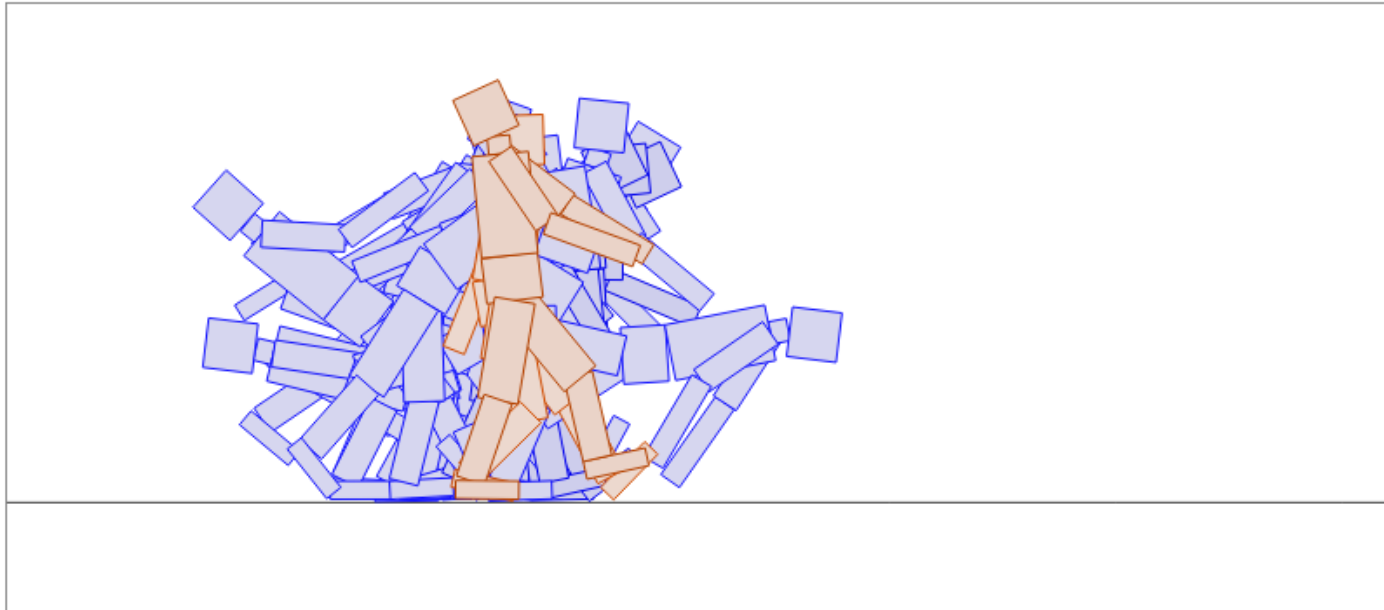


<http://alglobus.net/NASAwork/papers/Space2006Antenna.pdf>

# Evolving locomotion patterns

## Genetic Algorithm Walkers

"The Walking Fad"



Current Generation (18)	
Name	Score
Sosaa0 Sitono	103.26
Hogilo Viyupe	103.13
Hudaa0 Yitipu	2.33
Rerufe Uikuni	2.59
Segiao Oitone	1.27
Nogaao Siwipe	2.34
Husaeo Sitape	103.09
Hagoeo Vazupe	2.58
Hudoka Yayuke	2.36
Hokalo Vikupe	103.03

Record History		
Gen	Name	Score
0	Yodubu Dauuba	4.49
2	Qoduba Daxube	4.75
3	Poduhi Lixine	107.03
8	Voqube Recuna	107.89
14	Hosuee Yakupe	205.07
15	Hosuee Yakupe	205.68
16	Hogilo Viyupe	206.85

**Controls**

Gene mutation probability  
10% ▼

Gene mutation amount  
50% ▼

Champions to copy ▼

Motor noise 5% ▼

Round length Regular ▼

Animation quality 60 fps ▼

Simulation speed 60 ▼

Lower the animation quality

### About

#### What the hell is this?

This observational pastime hopes to evolve walking creatures through genetic algorithms.

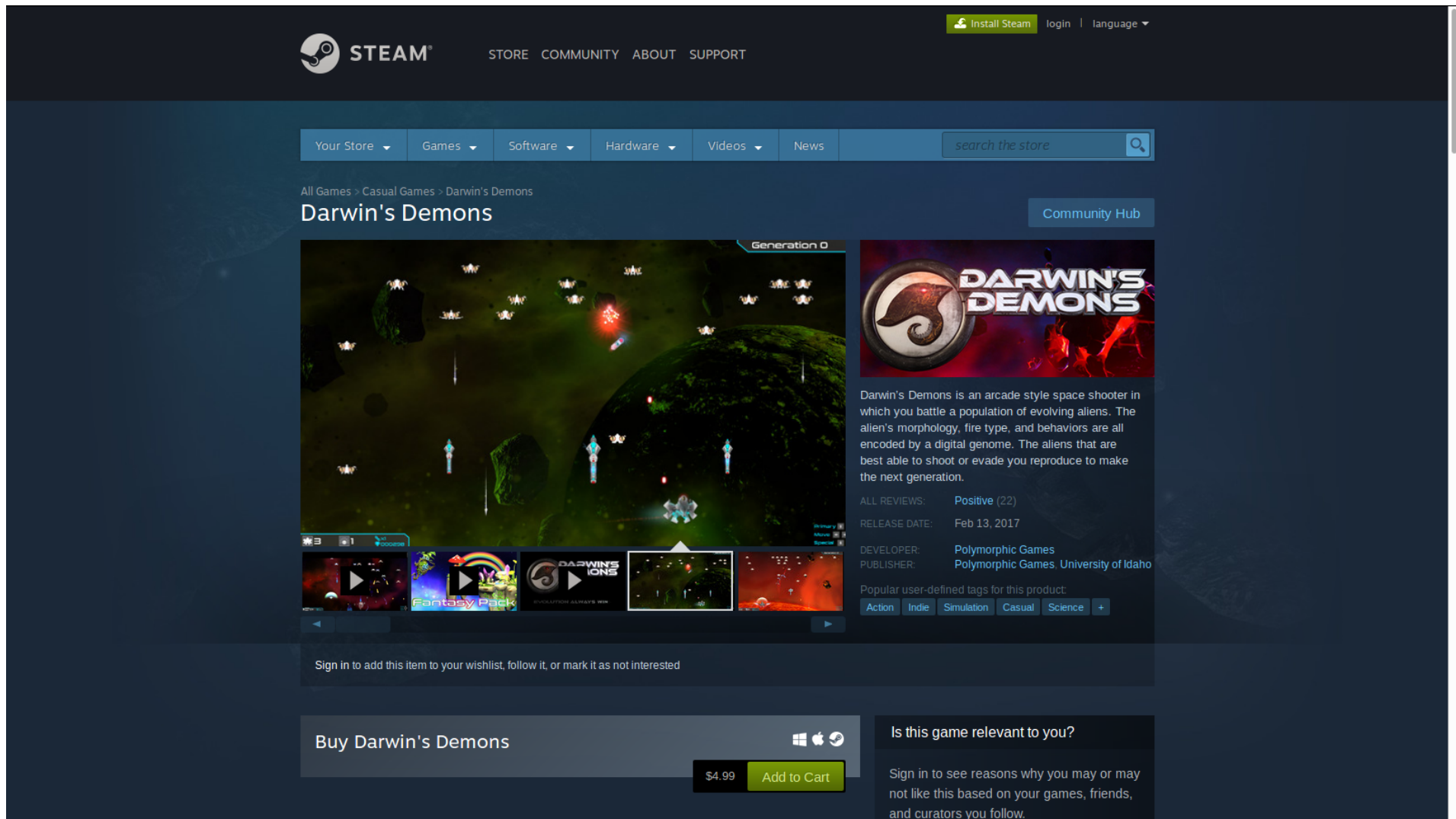
#### Who comes up with these names?

The names are generated based on each creature's genome. Since the genetic

Walkers: [http://rednuht.org/genetic\\_walkers/](http://rednuht.org/genetic_walkers/)

Cars: [http://rednuht.org/genetic\\_cars\\_2/](http://rednuht.org/genetic_cars_2/)

# Evolving game playing opponents



The screenshot shows the Steam store page for the game **Darwin's Demons**. The page layout includes a top navigation bar with the Steam logo, links to STORE, COMMUNITY, ABOUT, and SUPPORT, and user options like 'Install Steam', 'login', and 'language'. Below this is a secondary navigation bar with categories like 'Your Store', 'Games', 'Software', 'Hardware', 'Videos', and 'News', along with a search bar. The main content area features the game's title 'Darwin's Demons' and a breadcrumb trail 'All Games > Casual Games > Darwin's Demons'. A large gameplay screenshot is displayed, showing a space shooter environment with various alien enemies and a player ship. To the right of the screenshot is a description: 'Darwin's Demons is an arcade style space shooter in which you battle a population of evolving aliens. The alien's morphology, fire type, and behaviors are all encoded by a digital genome. The aliens that are best able to shoot or evade you reproduce to make the next generation.' Below the description are details: 'ALL REVIEWS: Positive (22)', 'RELEASE DATE: Feb 13, 2017', 'DEVELOPER: Polymorphic Games', and 'PUBLISHER: Polymorphic Games, University of Idaho'. A section for 'Popular user-defined tags for this product' lists 'Action', 'Indie', 'Simulation', 'Casual', and 'Science'. At the bottom, there is a 'Buy Darwin's Demons' button, a price tag of '\$4.99', and an 'Add to Cart' button. A sidebar on the right asks 'Is this game relevant to you?' and prompts the user to 'Sign in to see reasons why you may or may not like this based on your games, friends, and curators you follow.'

STEAM® STORE COMMUNITY ABOUT SUPPORT

Install Steam login language

Your Store Games Software Hardware Videos News search the store

All Games > Casual Games > Darwin's Demons

Darwin's Demons Community Hub

Generation 0

DARWIN'S DEMONS

Darwin's Demons is an arcade style space shooter in which you battle a population of evolving aliens. The alien's morphology, fire type, and behaviors are all encoded by a digital genome. The aliens that are best able to shoot or evade you reproduce to make the next generation.

ALL REVIEWS: Positive (22)

RELEASE DATE: Feb 13, 2017

DEVELOPER: Polymorphic Games

PUBLISHER: Polymorphic Games, University of Idaho

Popular user-defined tags for this product:

Action Indie Simulation Casual Science +

Sign in to add this item to your wishlist, follow it, or mark it as not interested

Buy Darwin's Demons

\$4.99 Add to Cart

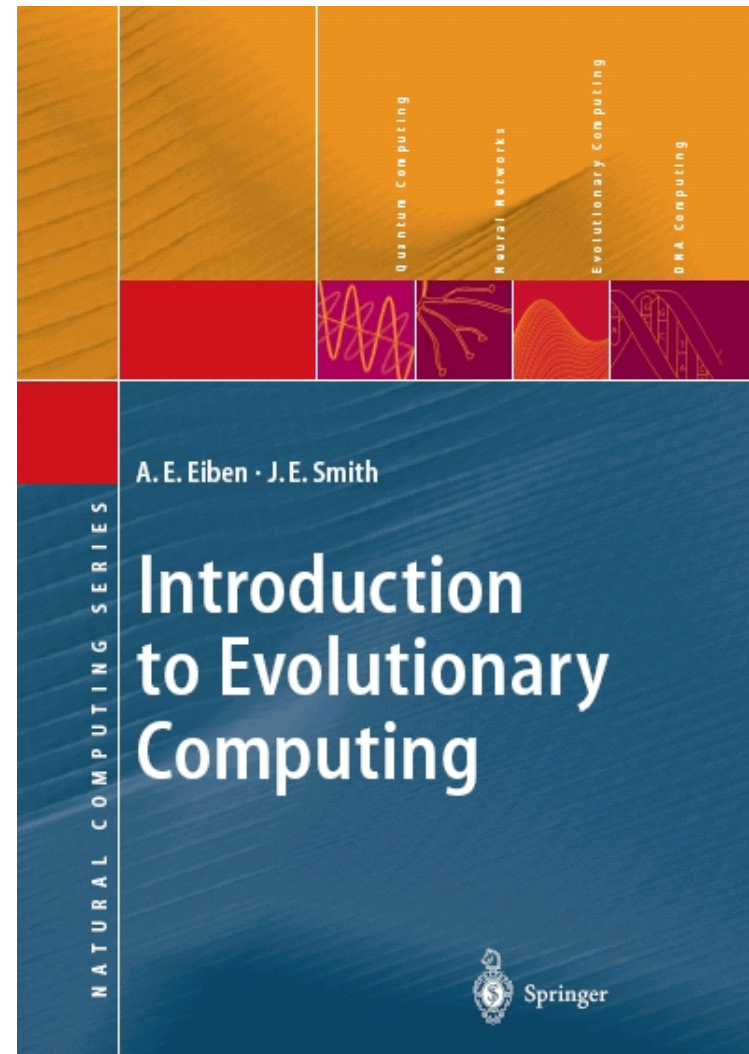
Is this game relevant to you?

Sign in to see reasons why you may or may not like this based on your games, friends, and curators you follow.

Paper: Darwin's Demons: Does Evolution Improve the Game?

# Variations of evolutionary methods

- Classical taxonomy:
  - Genetic Algorithm (GA)
  - Evolutionary Strategies (ES)
  - Evolutionary Programming (EP)
  - Genetic Programming (GP)
- More recent work:
  - Divergent search
  - Novelty search





# Genetic algorithm

- Simulates evolution
- A population
- Full solutions
- Solution space
- Survival of the fittest
- Not guaranteed optimal
- General algorithm, not problem specific

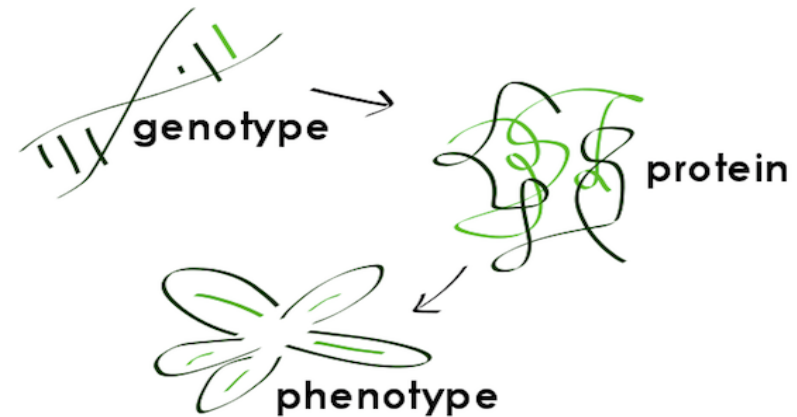


## Basic outline of a genetic algorithm

- 1) Create a random population of individuals
- 2) Calculate the fitness of the individuals in the population
- 3) Select some individuals in the population for mating; these are considered the parent.
- 4) Create a new set of individuals based on the ones previously selected (the children)
- 5) Mutate the children based on some (low) probability
- 6) Calculate the fitness of the new individuals and combine the children with the parent generation
- 7) If the desired performance of the population as a whole has been reached stop executing, else go back to step 3.

# Adapting the genetic algorithm to a problem

- Genotype and phenotype



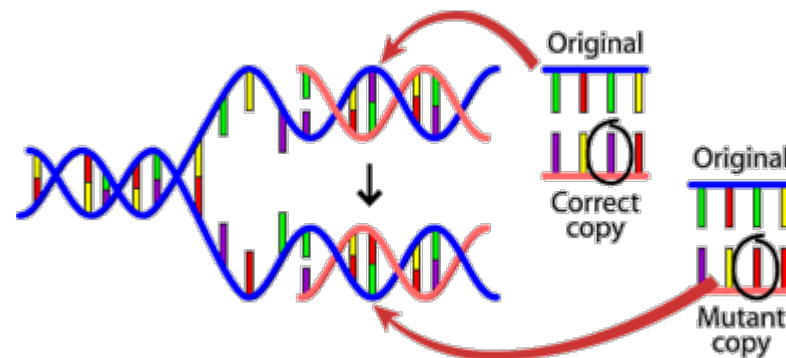
- Fitness-function

$$\text{function name } f(x) = \underbrace{x^2}_{\text{what to output}}$$

The diagram shows the mathematical representation of a fitness function. The function name  $f(x)$  is shown in blue, with an arrow pointing to the 'function name' label. The input  $x$  is shown in purple, with an arrow pointing to the 'input' label. The output  $x^2$  is shown in orange, with an arrow pointing to the 'what to output' label.

- Crossover and mutation-operators

- Problem specifics

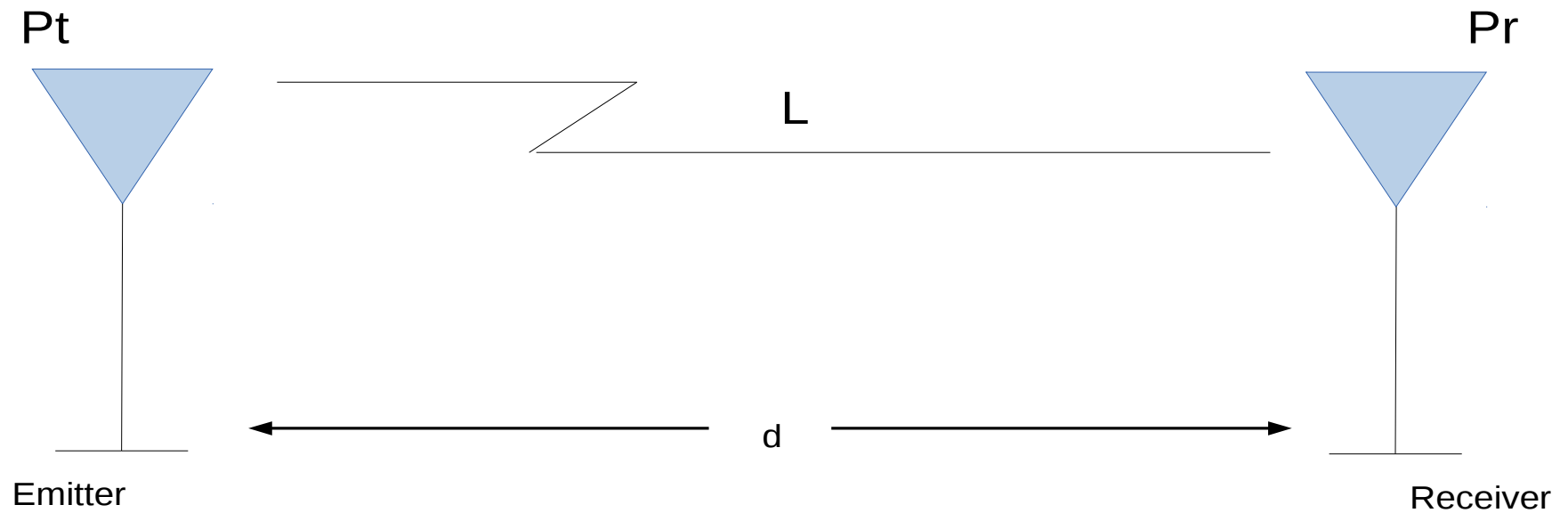


# What have I used evolution for?

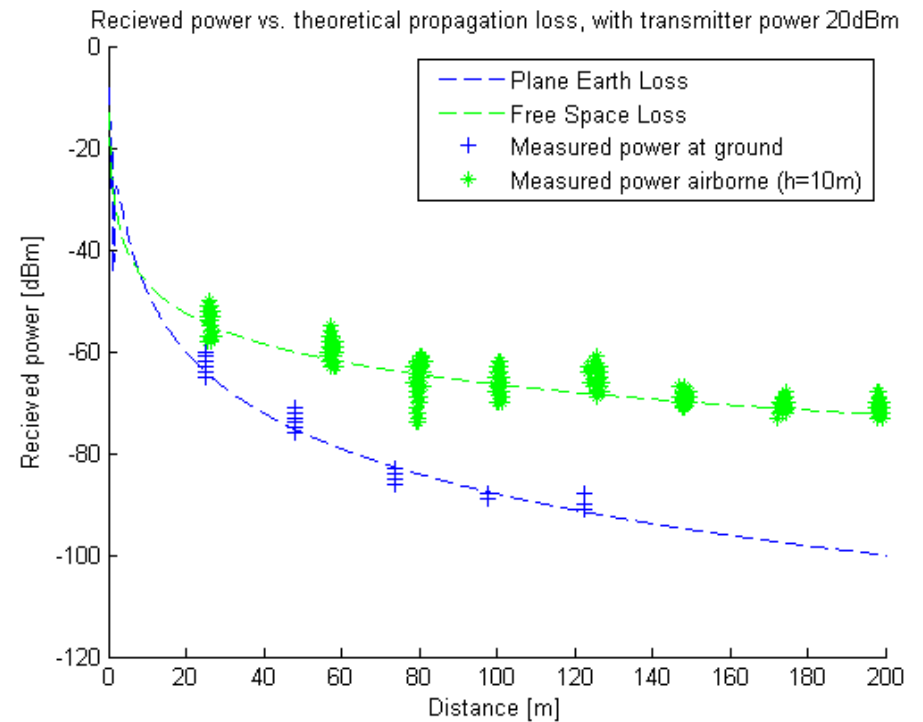
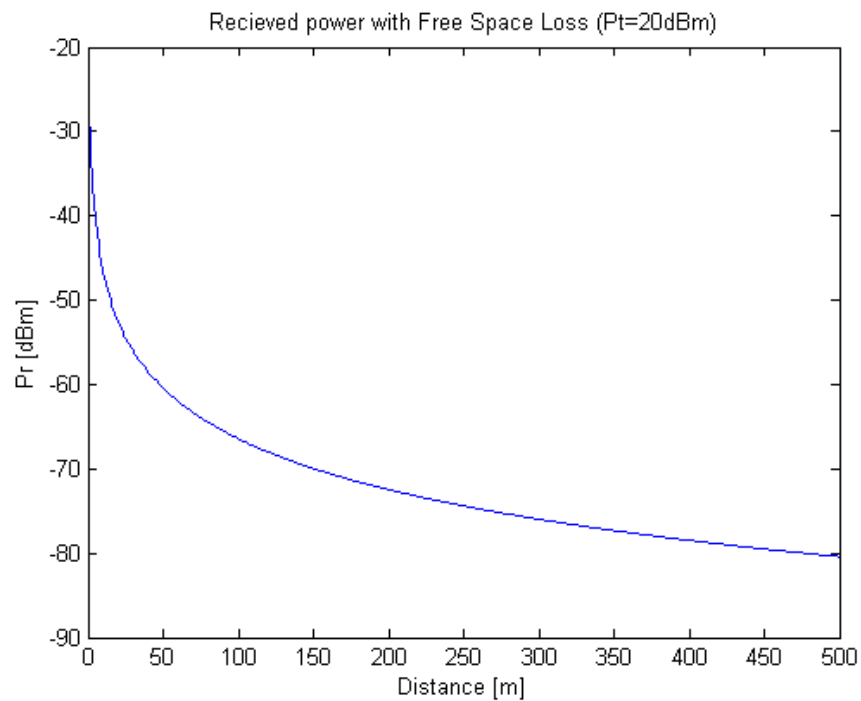
- 1) Path planning for UAVs
- 2) Multilateration for RF emitters
- 3) (Now) Evolving swarm behaviors



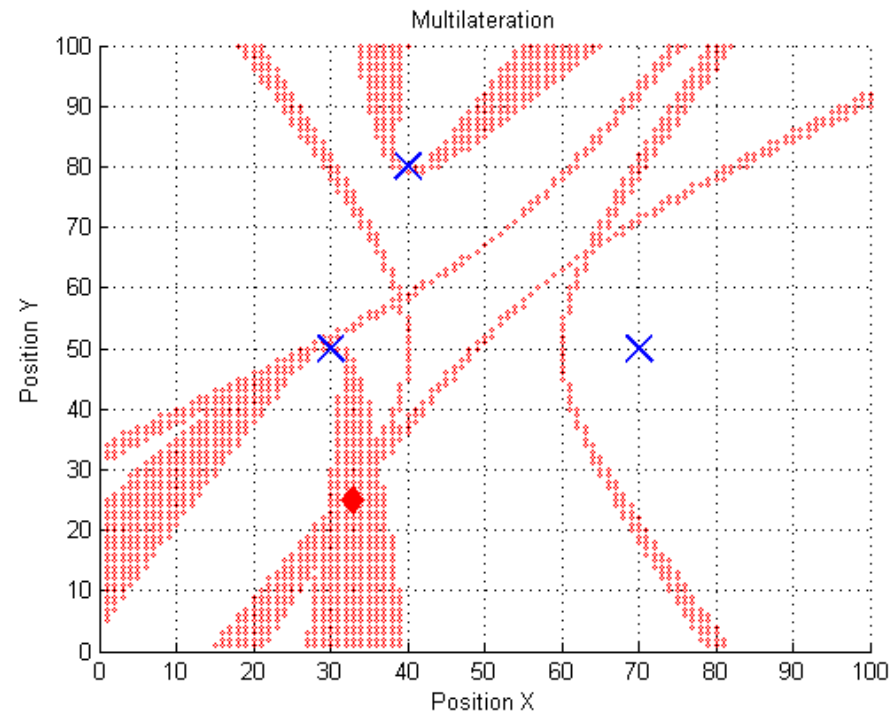
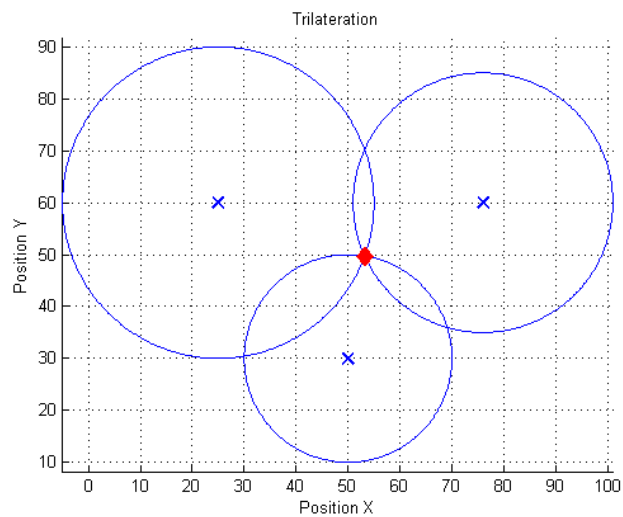
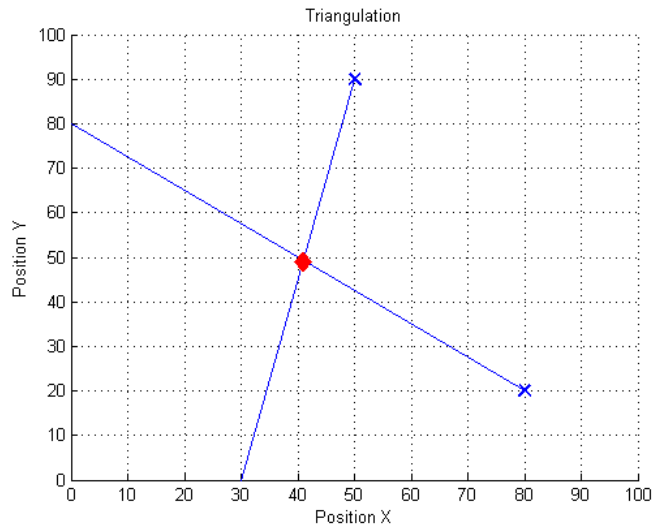
## Path loss between emitter (sender) and receiver



# Path loss between emitter (sender) and receiver



# Methods of locating a radio frequency emitter



## Need to solve a non-linear error function

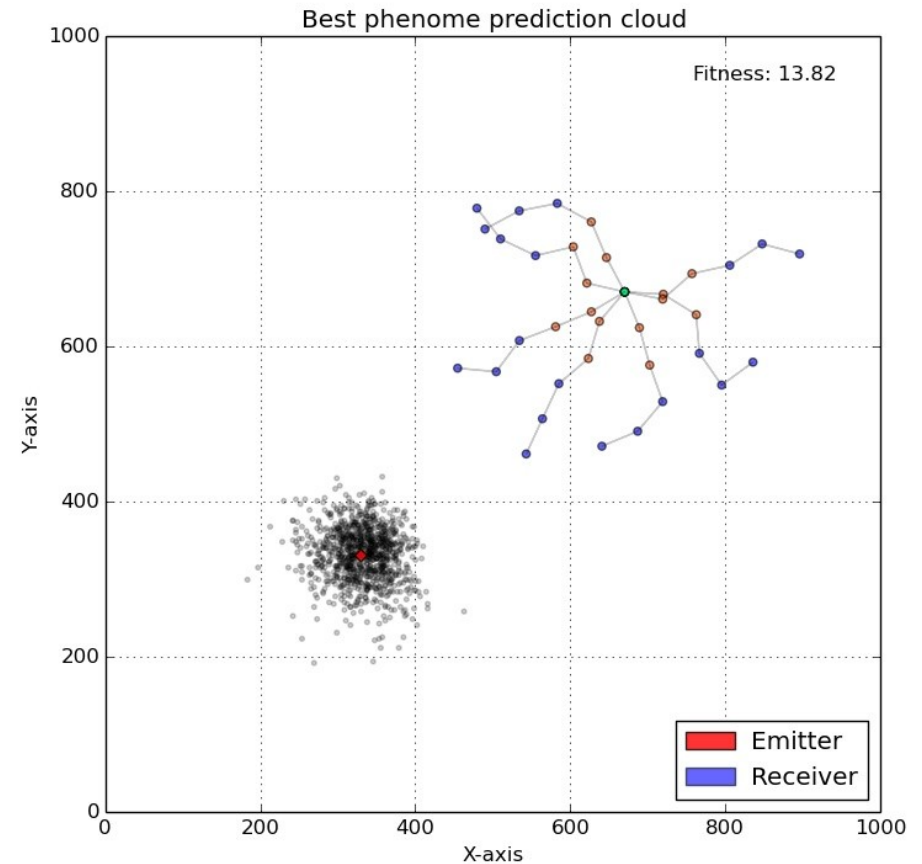
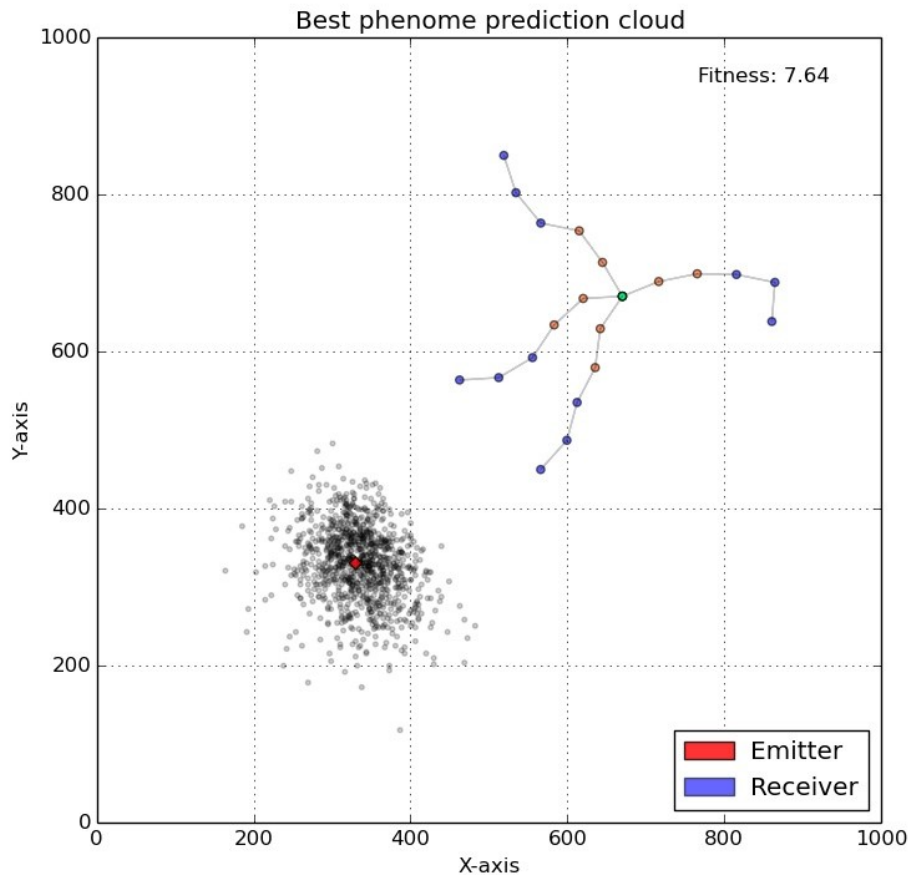
$$P_{kl} = P_k - P_l \quad (2.8)$$

$$Q(x, y) = \sum_{k < l} [P_{kl} - 5\alpha \log(\frac{(x - x_l)^2 + (y - y_l)^2}{(x - x_k)^2 + (y - y_k)^2})]^2 \quad (2.9)$$

$$\min_{0 \leq x \leq m} (\min_{0 \leq y \leq n} Q(x, y)) \quad (2.10)$$



# Results of evolving path plans for UAVs





# Meta-heuristics for improved RF emitter localization

S. Engebråten, J. Moen and K. Glette (sondre.engebraten@ffi.no)

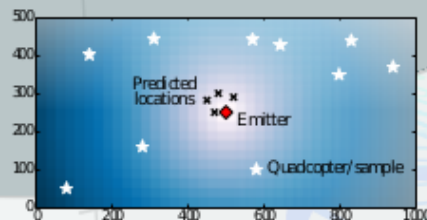
Making PDOA viable for small UAV platforms through meta-heuristics

## Background

### RF emitter localization

One method of locating an RF emitter is Power Difference of Arrival (PDOA). PDOA relies on measuring the differences in received signal strength at multiple points in space. Given a guess for the emitter location, it is possible to calculate an error measurement using the free space path loss model.

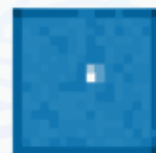
An estimate of the emitter location can be found by solving the non-linear error function for the emitter position. Non-linearity makes this a time consuming optimization. In this paper, meta-heuristics are applied to solve this optimization problem faster, and more precisely, than conventional methods



### Non-linear optimization problem

$$P_{kl} = \frac{P_k - P_l}{\sum_{k < l} \left[ P_{kl} - 5\alpha \log \left( \frac{(x - x_l)^2 + (y - y_l)^2}{(x - x_k)^2 + (y - y_k)^2} \right) \right]^2}$$

### Preselect NM



The figures are histograms of how each algorithm explores a single problem instance, accumulated over several runs

### Challenges

Multi-modality/deceptiveness makes this problem challenging. Instances of the optimization problem often has multiple local optima. These are caused by noisy sensor measurements and the local optima may often be far away from the true emitter location

Strict time constraints are enforced as the goal of this paper is to enable the use of PDOA on resource restricted platforms. The search space is smooth, which may allow for use of local information.

Lack of optimality guarantees combined with strict time constraints make this a challenging problem for meta-heuristic search.

## Methods

PSO



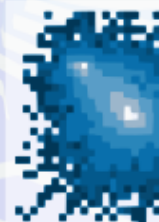
NM



Diff. Evo.



CMA-ES

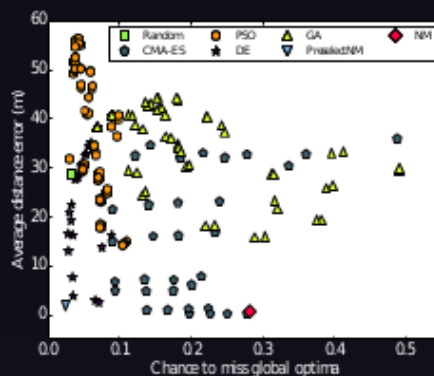


GA



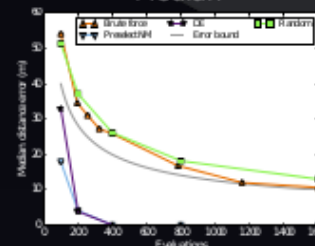
## Results and conclusion

### Multi-objective performance

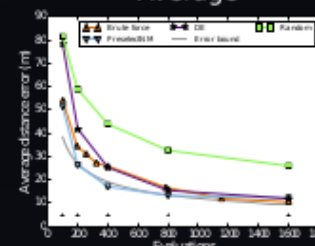


### Unreliability affects average performance

Median



Average



### Conclusion

A no-cost speedup up 2x can be achieved using Preselect NM

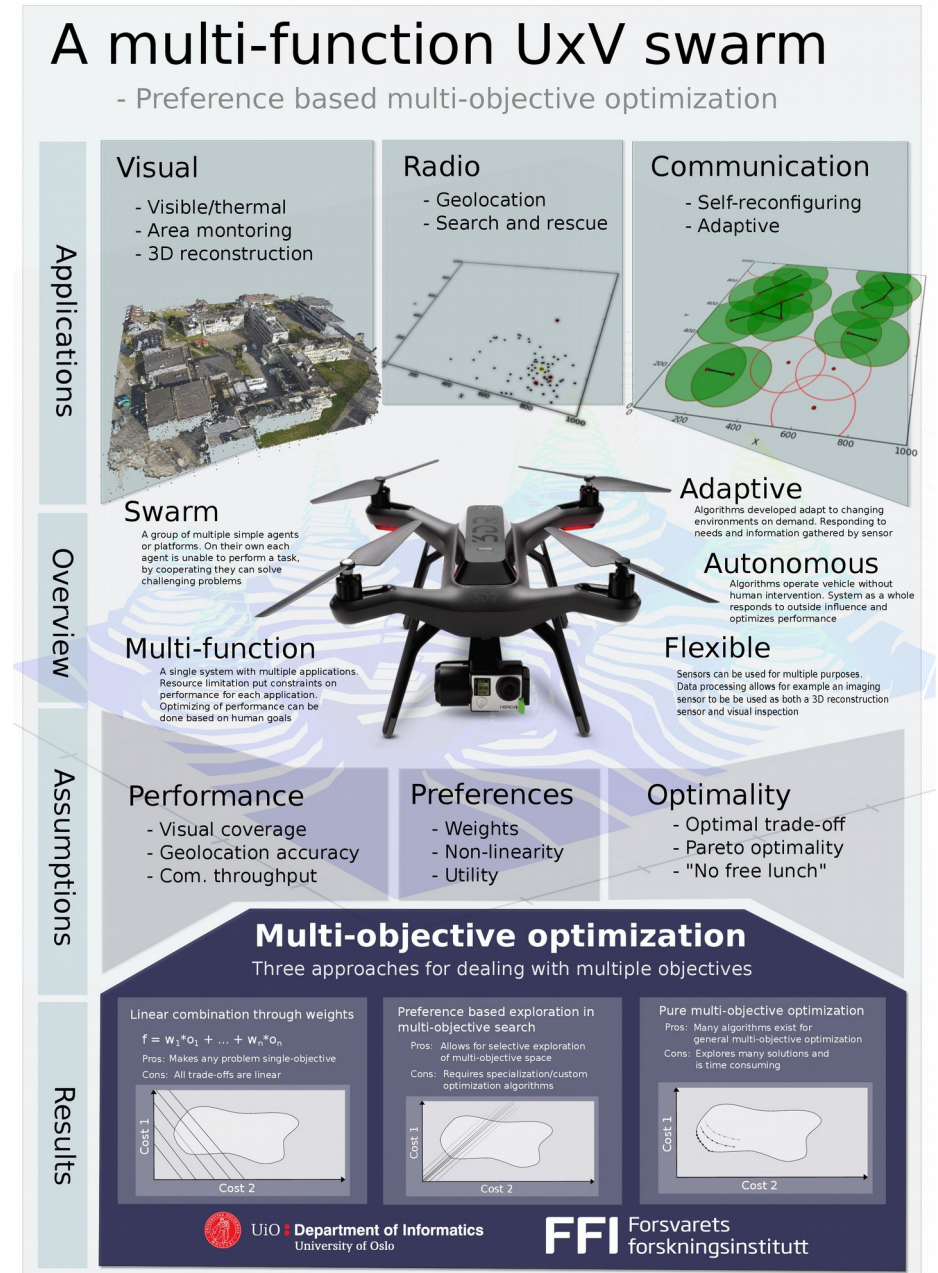
Using median metric (instead of average), 400 evaluations with a meta heuristic is comparable to 40 000 evaluations (100x) using "brute force"

For quick convergence and hill climbing, an adaptive mutation/permutation step is essential.

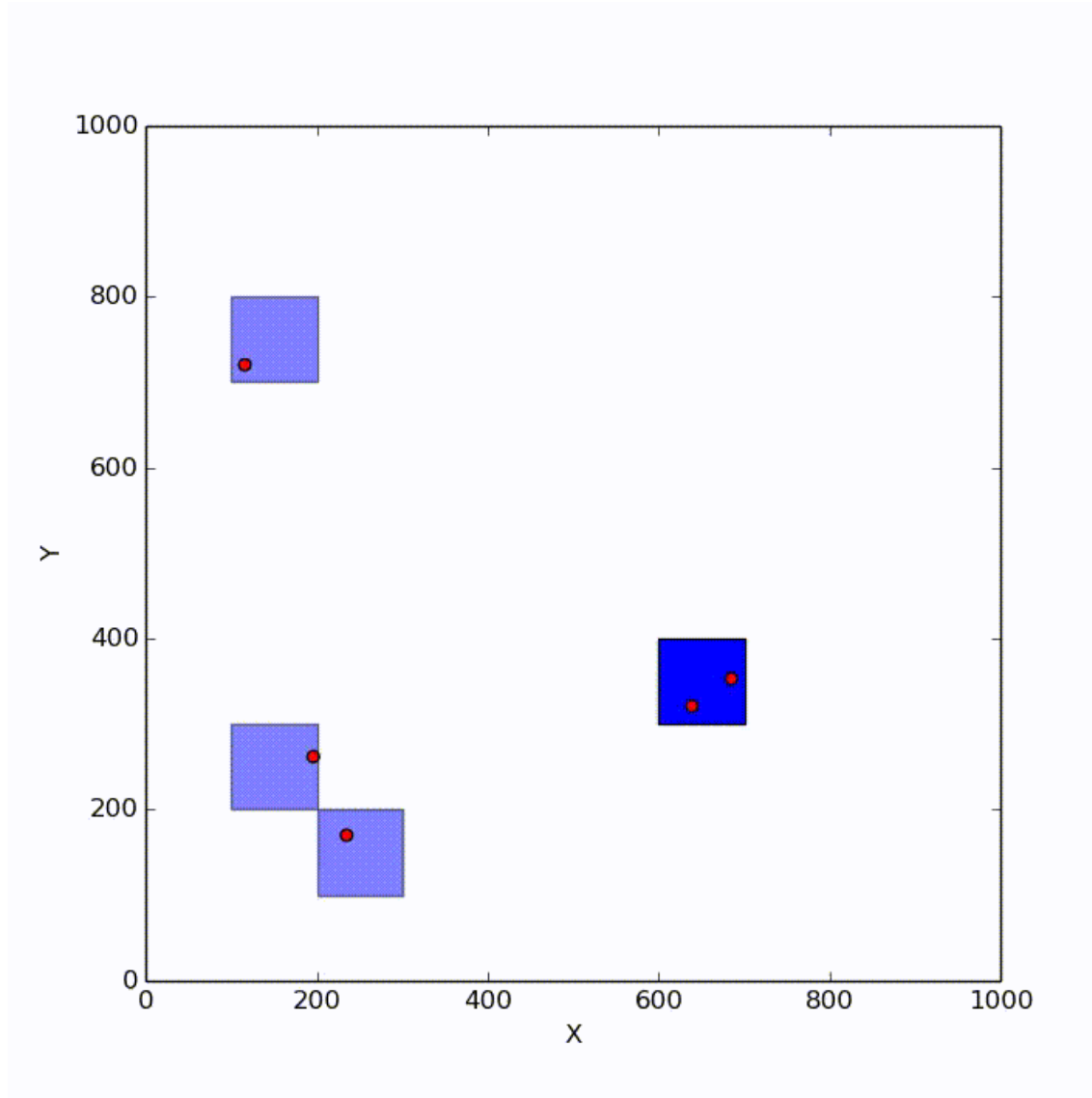


# Multi-function swarm

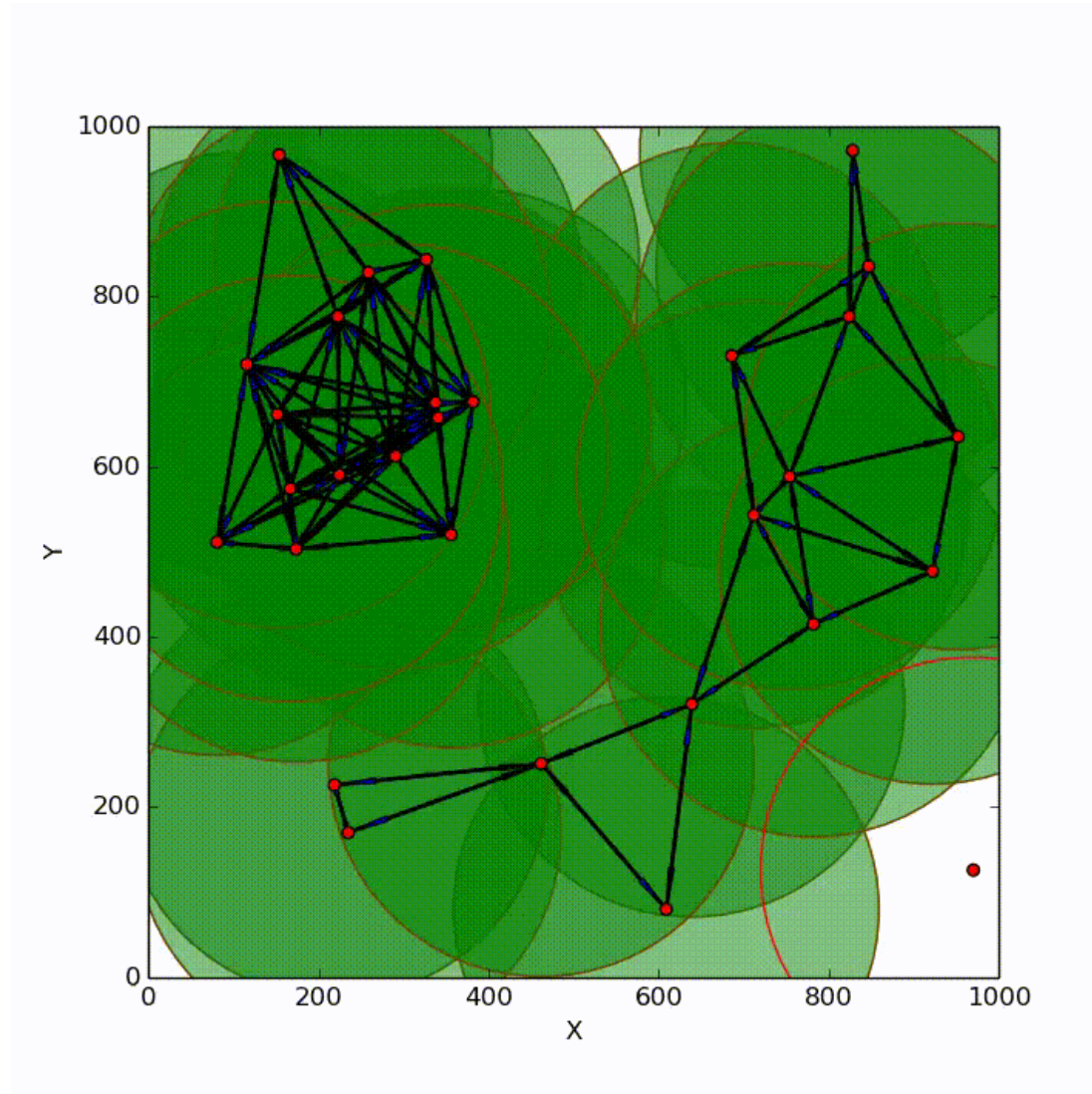
- **What** is a multi-function swarm?
  - One system that solves many tasks
  - Many agents/units that cooperate
  - A terminology to optimize resource use
- **Why** work on a multi-function swarm?
  - Capable system
  - Connects activities with a common goal
- **How** work towards a multi-function swarm?
  - Identify applications
  - Handle/maximize user utility



# Algorithms for perimeter surveillance



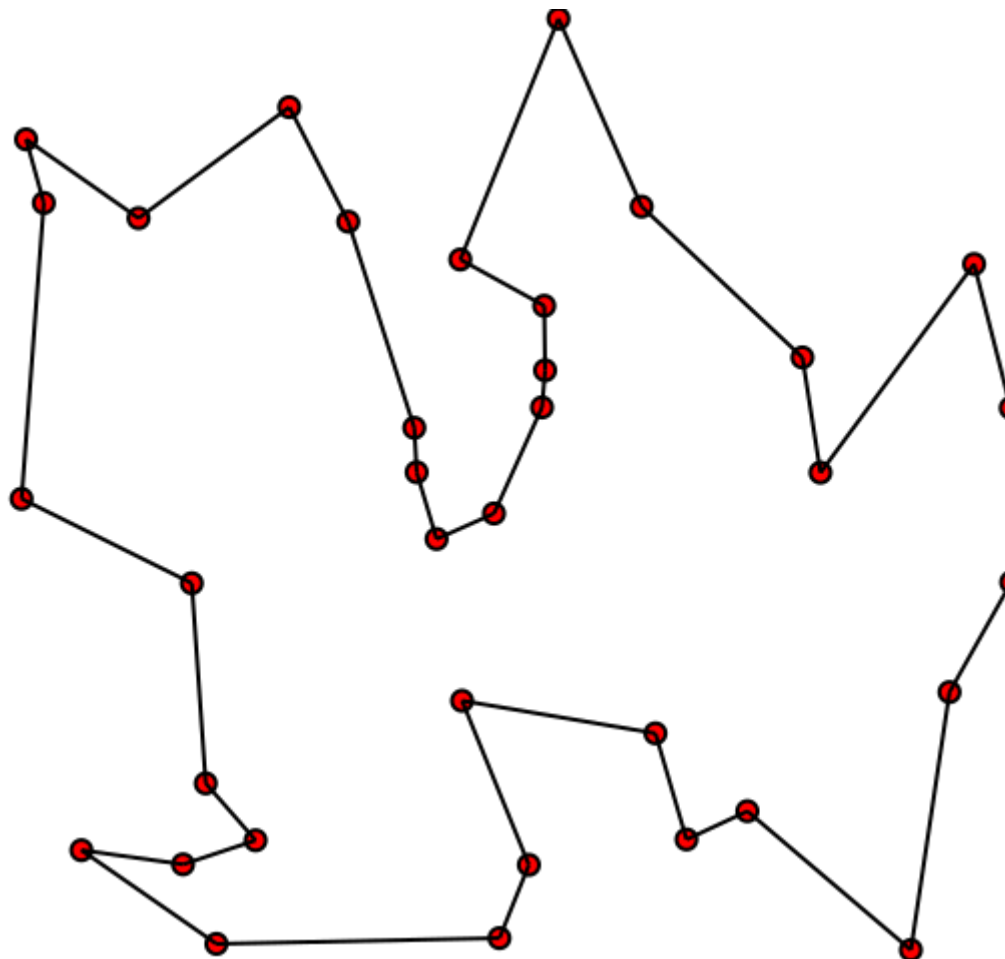
# Algorithm to construct communication network



## Impossible problems, why are they impossible?

Best algorithms to solve them take a really long time for large problems

# Travelling salesman problem TSP (NP-hard)



$N = 35$      $N! = 10333147966386144929666651337523200000000$

## Travelling salesman problem - scaling


1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800
11	39916800
12	479001600
13	6227020800
14	87178291200
15	1307674368000
16	20922789888000
17	355687428096000
18	6402373705728000
19	121645100408832000
20	2432902008176640000

## Travelling salesman problem - scaling

21 51090942171709440000  
22 1124000727777607680000  
23 25852016738884976640000  
24 620448401733239439360000  
25 15511210043330985984000000  
26 403291461126605635584000000  
27 10888869450418352160768000000  
28 304888344611713860501504000000  
29 8841761993739701954543616000000  
30 265252859812191058636308480000000  
31 8222838654177922817725562880000000  
32 263130836933693530167218012160000000  
33 8683317618811886495518194401280000000  
34 295232799039604140847618609643520000000  
35 1033314796638614492966651337523200000000  
36 371993326789901217467999448150835200000000  
37 13763753091226345046315979581580902400000000  
38 523022617466601111760007224100074291200000000  
39 20397882081197443358640281739902897356800000000  
40 815915283247897734345611269596115894272000000000



# DEAP – Distributed Evolutionary Algorithms in Python

 Project Homepage » DEAP 1.2.2 documentation »

next | modules | index

Next topic  
Overview

This Page  
Show Source

Docs for other versions  
DEAP 1.0 (Stable)  
DEAP 0.9

Other resources  
Downloads  
Issues  
Mailing List  
Twitter

Quick search  
  
Go

## DEAP documentation


DEAP is a novel evolutionary computation framework for rapid prototyping and testing of ideas. It seeks to make algorithms explicit and data structures transparent. It works in perfect harmony with parallelisation mechanism such as multiprocessing and [SCOOP](#). The following documentation presents the key concepts and many features to build your own evolutions.

- **First steps:**
  - [Overview \(Start Here!\)](#)
  - [Installation](#)
  - [Porting Guide](#)
- **Basic tutorials:**
  - [Part 1: creating types](#)
  - [Part 2: operators and algorithms](#)
  - [Part 3: logging statistics](#)
  - [Part 4: using multiple processors](#)
- **Advanced tutorials:**
  - [Genetic Programming](#)
  - [Checkpointing](#)
  - [Constraint Handling](#)
  - [Benchmarking Against the Bests \(BBOB\)](#)
  - [Inheriting from Numpy](#)
- [Examples](#)
- [Library Reference](#)
- [Release Highlights](#)
- [Contributing](#)
- [About DEAP](#)


### Getting Help

Having trouble? We'd like to help!

- Search for information in the archives of the [deap-users mailing list](#), or post a question.
- Report bugs with DEAP in our [issue tracker](#).



DISTRIBUTED  
EVOLUTIONARY  
ALGORITHMS IN  
PYTHON

 Project Homepage » DEAP 1.2.2 documentation »

next | modules | index

© Copyright 2009-2018, DEAP Project.  
Built on Jan 31, 2018. [Found a bug?](#)  
Created using [Sphinx](#) 1.6.5.

v: master ▼

Examples: <https://goo.gl/NLr9gY>

**FFI** Forsvarets  
forskningsinstitutt



# Hands on example: Solving TSP!

Link: <https://github.com/sondreengebraten/RobodojoImpossible2018>

