

Modern Game AI Algorithms

Learning and Optimization

Mike Preuss | LIACS



Universiteit
Leiden
The Netherlands



roadmap

lecture plan (Thursdays 14:15), https://smart.newrow.com/room/nr2/?room_id=stn-974

Feb	18	introduction	Apr	8	procedural content generation II
	25	learning and optimization		15	player experience modeling
Mar	4	procedural content generation I	Apr	22	Georgios: spatial data analysis
	11	Monte Carlo tree search		29	believable behavior
	18	experimentation / ANN / GAN	May	6	team AI and mass behavior
	25	Vanessa: 1 year game industry...		13	realtime strategy AI
Apr	1	learning from pixels	May	20	free project presentations 1

labs plan (Thursdays 16:15)

Feb	18	no lab	Apr	8	free projects (assignment 3)
	25	single assignment: map generation		15	free projects (assignment 3)
Mar	4	single assignment (week 2)	Apr	22	free projects (assignment 3)
	11	single ass. deadline / start ass 2		29	free projects (assignment 3)
	18	assignment 2	May	6	free projects (assignment 3)
	25	assignment 2		13	deadline free projects
Apr	1	deadline ass. 2 / start free proj.	May	20	free project presentations 2

assessment

what is graded:

- single assignment 25%, smaller group assignment 30%, free project assignment 45% (including presentation), no written exam

the single assignment is issued today (shall be on Brightspace already):

- simple programming task requiring a bit creativity, Python

smaller group project (largely March):

- group work, most likely in a competition-based fashion on the GVGAi (new version)

free group project (free means you choose the topic):

- working in small groups (4-5 people, exceptions possible)
- during the second half of the tutorials, and in your preparation time



topics of today

- learning
- representation
- utility
- Evolutionary Algorithm primer



picture from Michal Jarmoluk on Pixabay

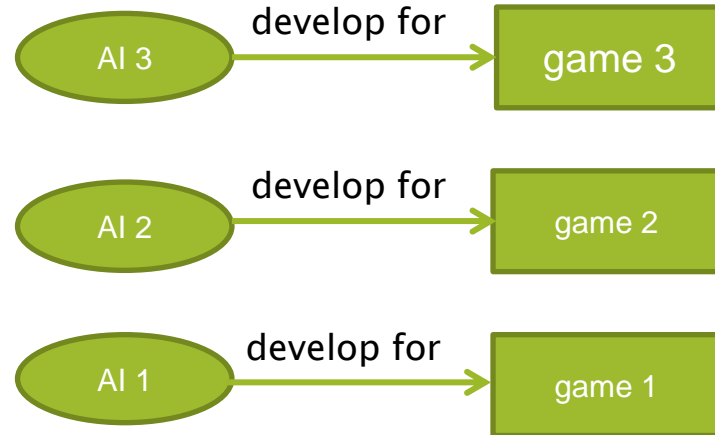
artificial stupidity vs. artificial intelligence

as already reported, traditional game AI consists mainly of:

- Finite State Machines (FSM)
- Hierarchical Finite State Machines (HFSM)
- scripting

reusability is very limited:

- specifically crafted AI is only intelligent in its design context
- in another context, it will most likely behave stupid
- how do we make it more adaptable?



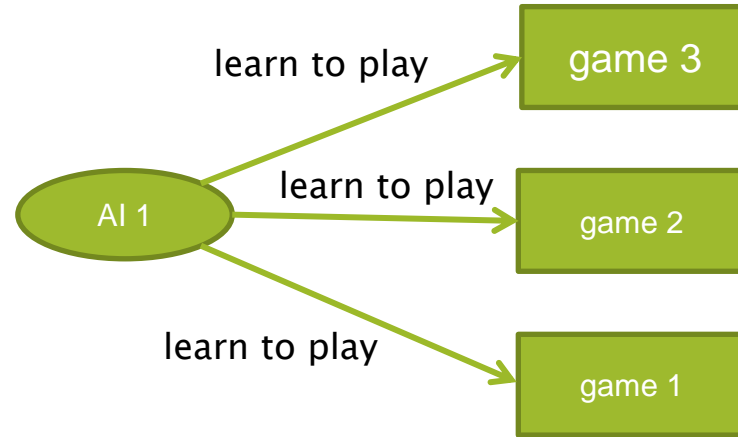
learning

- learning is the answer
- you can send humans to school/university
- they learn how to adapt to new situations (learn to learn)
- but how can we do that with game AIs?



learning prerequisites

this is what we want:



but how do we get there? we need:

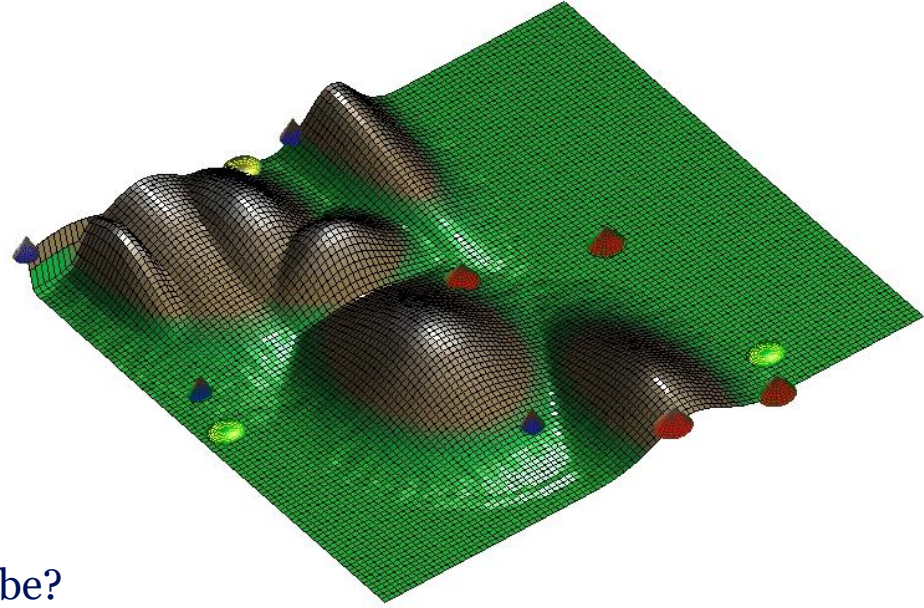
- to represent games/levels/policies/actions
- to measure utility – how good was that move?
or how “good” is the current game situation?
- a learning algorithm

machine learning and optimization

- **unsupervised learning** try to detect structure in the data
- **supervised learning** – usually static
 - train with data and value pairs to reproduce underlying relationship
 - implicit model building
 - important: good prediction quality on unseen data is the task
- **reinforcement learning** – dynamic
 - learn behavior in a changing environment
- **optimization**
 - process of improving towards a given target
 - often used as driving force for all types of learning
 - reproduction of **known** data is the task
- dangers:
 - overfitting (focus too much on replicating existing data)
 - model selection (do not enforce a structure that is not in the data)

what is: representation?

- a map of the world
 - it stores knowledge about the world
 - a machine is able to process it!
 - data structure
- key importance for problem solving in AI
 - humans and knowledge storage
- key questions
 - how do people represent knowledge?
 - what is the nature of knowledge?
 - how generic can a representation scheme be?



types of representation

- grammars
 - grammatical evolution
- graphs
 - finite state machines, probabilistic models
- trees
 - decision trees, behaviour trees, genetic programming
- connectionism
 - e.g. artificial neural networks
- boolean/numerical vectors
 - e.g. evolutionary computation
- fuzzy
 - e.g. fuzzy sets
- table
 - e.g. temporal difference learning

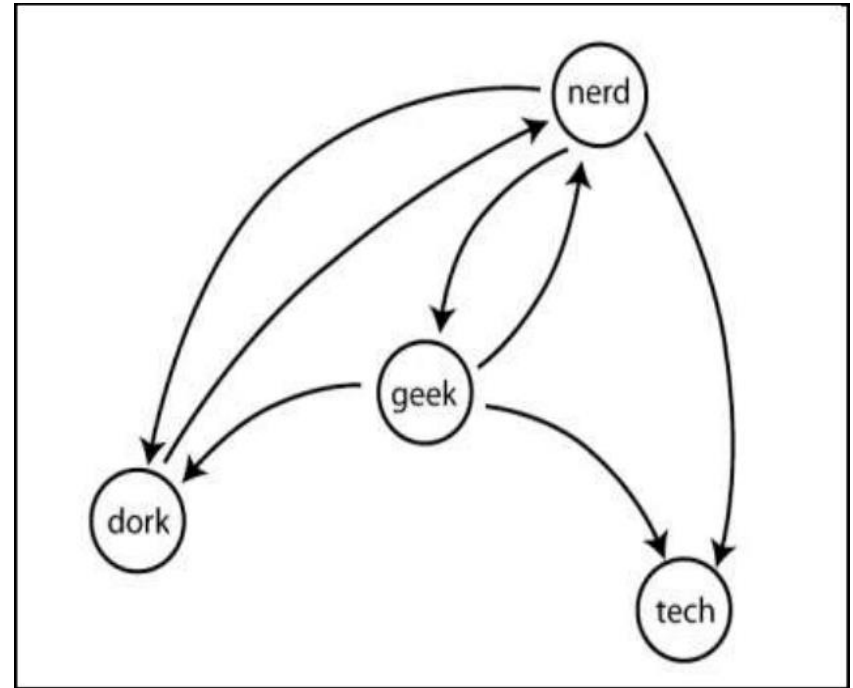


picture from S. Hermann & F. Richter on Pixabay

important: our choice of representation determines how well we can learn

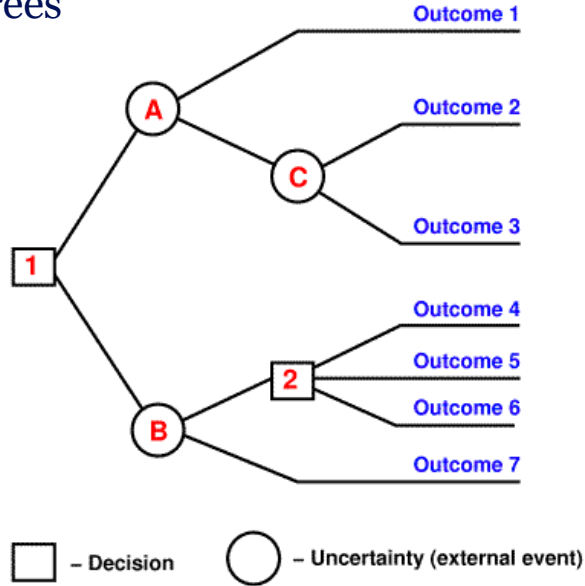
graph-based representation

- abstract representation of an interconnected set of objects/symbols/functions
- vertices (nodes) embed some mathematical abstraction
- edges represent relationship (conditional, direct, probabilistic,...)
- game examples:
 - decision making
 - control
 - story plot-point graph
- popular game AI technique
 - finite state machines

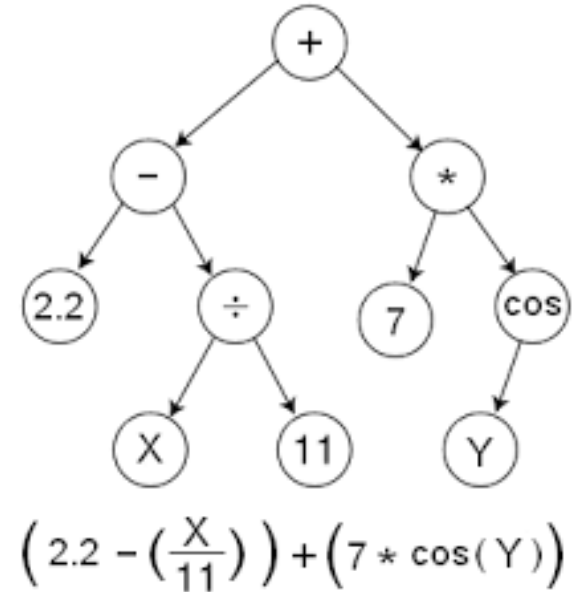


tree-based representation

decision trees



genetic programming



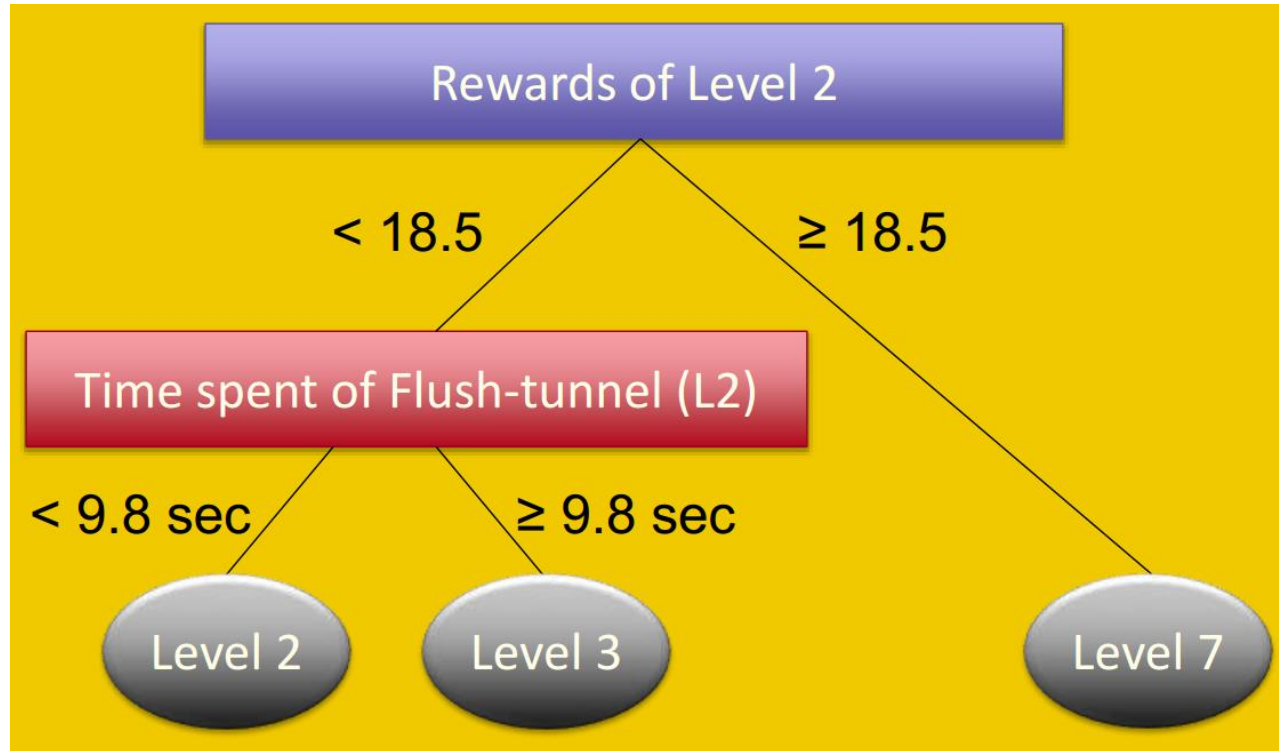
game examples:

- story (nodes: story events/triggers/user decisions)
- decision making
- control

popular game AI techniques:

- decision trees
- behavior trees

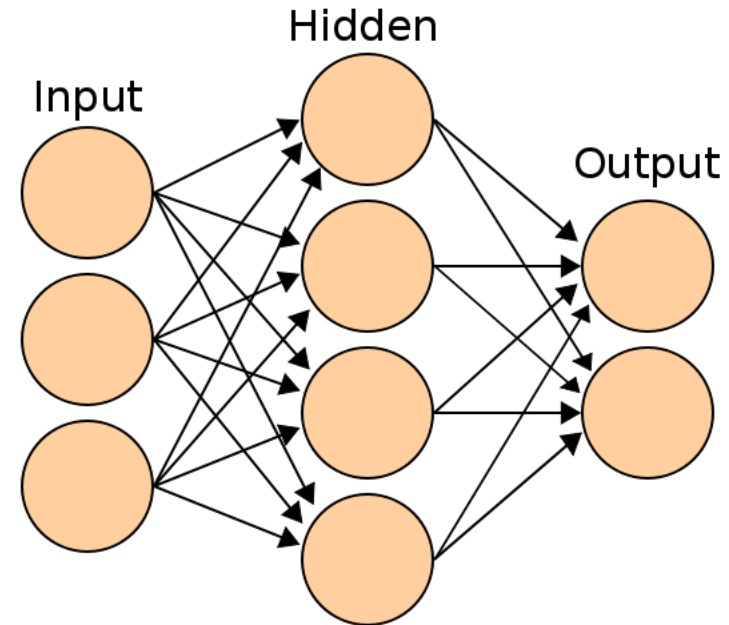
decision tree – tomb raider underworld example



from: Mahlmann, T., Drachen, A., Togelius, J., Canossa, A., Yannakakis, G.N., "Predicting player behavior in Tomb Raider: Underworld," in *Computational Intelligence and Games (CIG) 2010*

connectionist representation

- bio-inspired: bio neural networks
- network of interconnected units
- unit: basic processing structures
- connections: strength of the inter-relationship between units
- black-box model
- game examples:
 - decision making (when to buy or sell?)
 - control (aiming, car driving, ...)
- popular game AI techniques
 - artificial neural networks
 - self-organizing feature maps

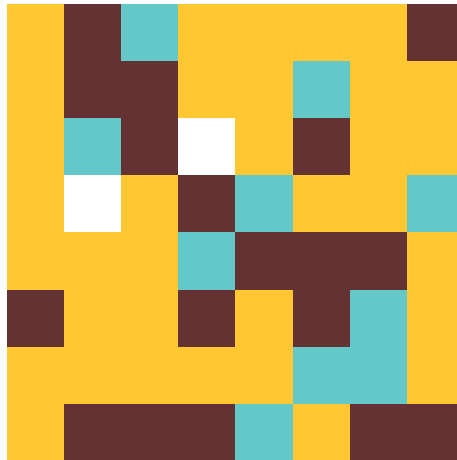


boolean/numeric/string representation

- bio-inspired
 - a string/vector of “values”
 - binary, discrete, real
 - or string
- genotype vs. phenotype
- game examples:
 - evolve non-scripted behaviors
 - emergent plot/story/gameplay
 - strategies – tactics
 - real-time adaptation

example: StarCraft map representation

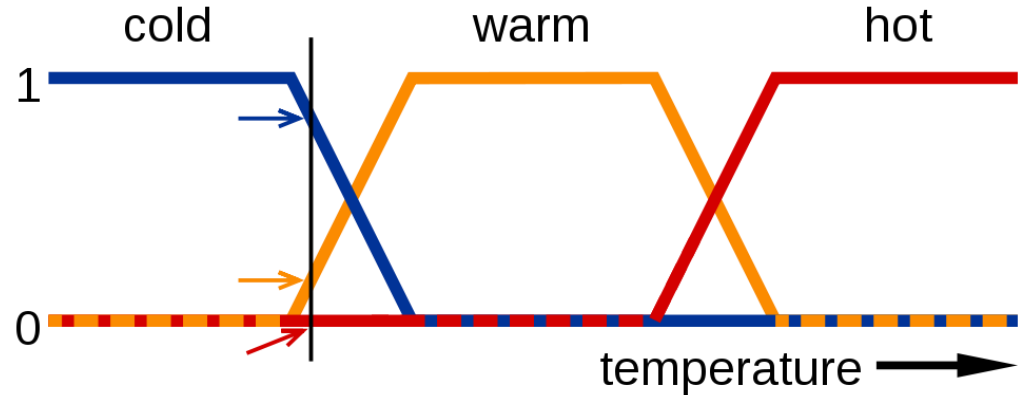
```
##...##.;  
.#.#.r..;  
...#.#...;  
#..#r..B;  
#..#r.#.;  
...####.;  
#r.r..#r;  
#Br#...##;
```



- popular game AI techniques:
 - evolutionary algorithms
(evolution strategies, genetic algorithms, learning classifier systems)

representation: fuzzy

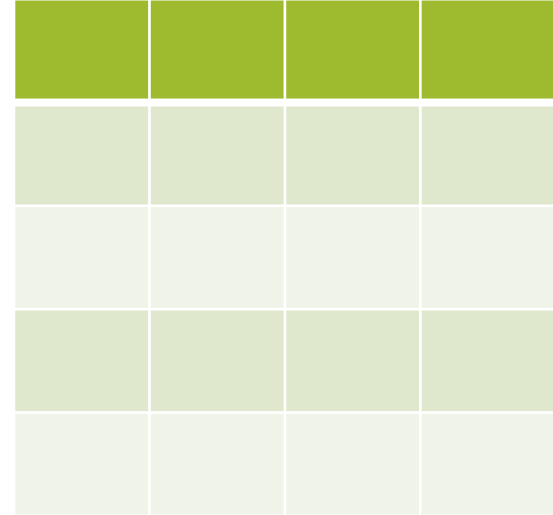
- linguistic expressions (“very tall”) and whole sentences
(e.g. IF Georgios makes a short pause THEN applaud.)
convert to real-value representations
- popular game AI techniques
 - fuzzy logic
- game examples
 - anything from decision making to control



picture from fullofstars on Wikimedia commons

table representation

- knowledge stored in look-up tables
 - n-dimensional
 - mapping between state and action's desirability
- table values infer a policy (strategy)
 - transition between table cells
- popular game AI techniques
 - temporal difference (Q-learning)
- game examples:
 - strategy/decision making (e.g. fight, tactics)



choosing representation

- no free lunch
- simplest possible (not easy)
 - detailed comes with computational effort
 - over simplistic comes with a sub-solution
- smallest possible (not easy either)
- **major** impact to the performance of your algorithm
- there is **black art** involved....



picture from Илия Илиев on Pixabay

learning approaches

how will your AI learn?

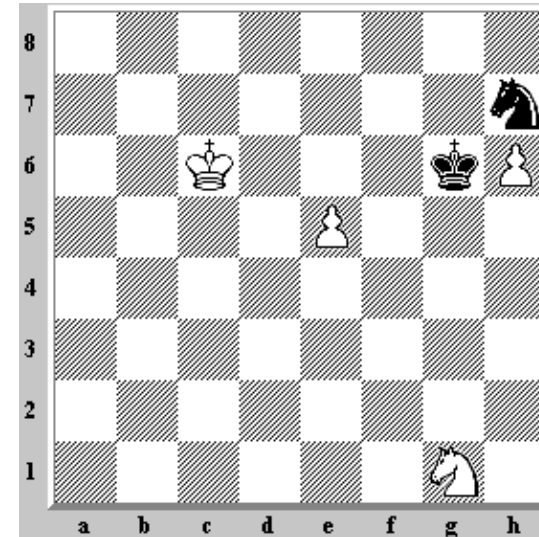
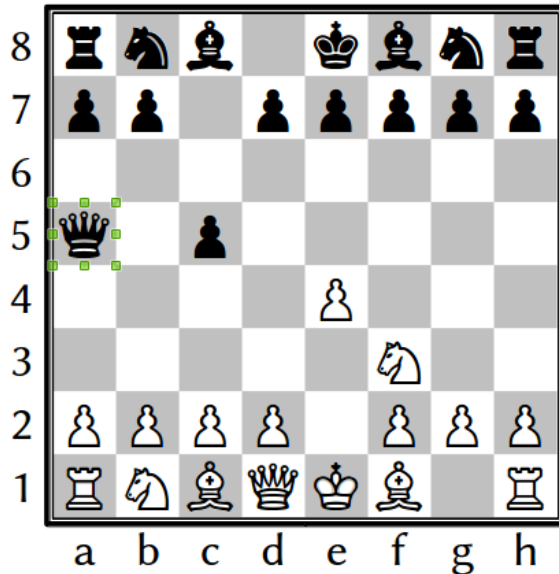
- it won't learn (expert-knowledge systems)
 - you know what good is -> heuristics
 - do you?
- supervised learning
 - do you have the data?
 - is it appropriate? indicative of what you want to learn?
 - you can test how good you are (training and test set, “ground truth”)
- reinforcement learning
 - does the environment provide all necessary rewards/fitness?
 - is it designed well?
- unsupervised learning
 - you cannot check how good you are
 - explore the data, clustering

what is utility?

- some call it objective function (operations research...)
- some call it cost function
 - fitness
 - reward
 - there may be more than one (multi-objective)
- bottom-line:
 - a heuristic of **goodness** of the learnt representation
- essential for learning!

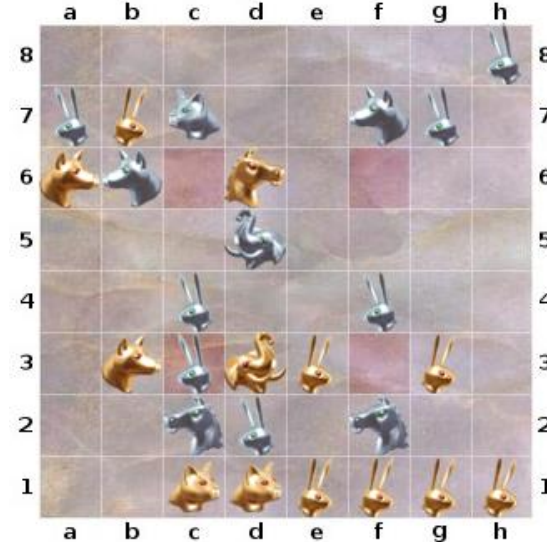
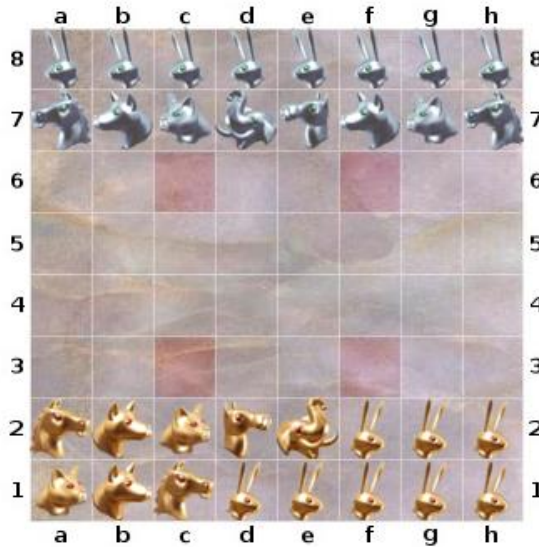
utility example: chess

- simple heuristics:
 - count pieces
 - weighted piece count (queen should be more important than pawn)
- more complex heuristics:
 - also consider position on the board



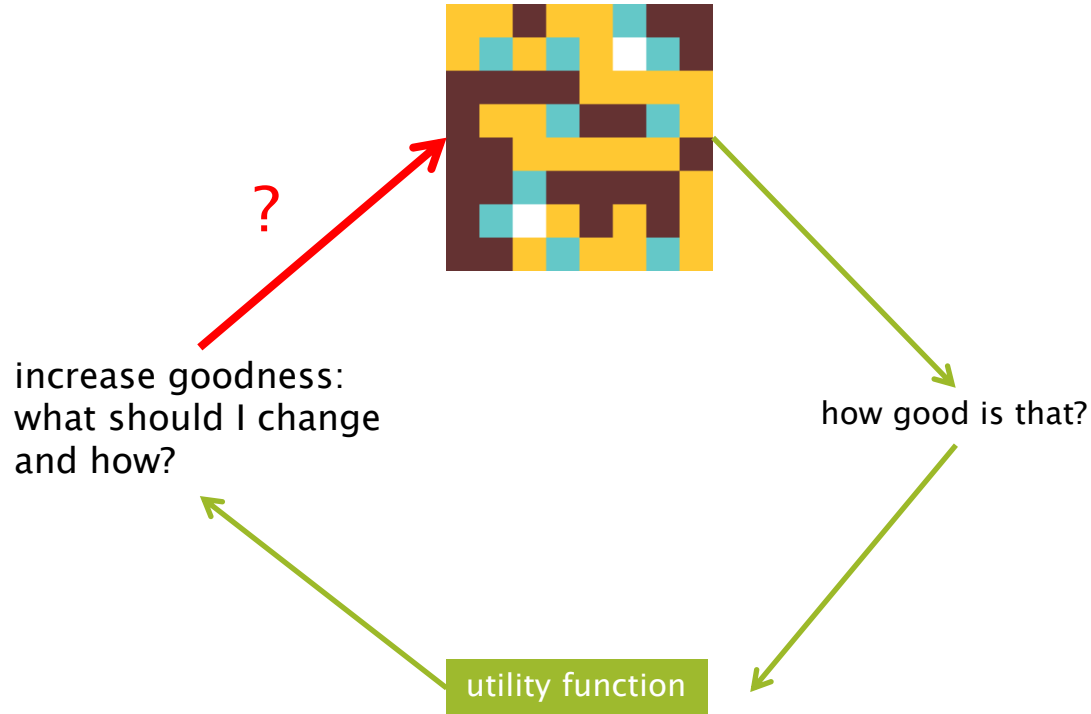
ARIMAA

- designed to be difficult for computers but easy for humans
- why? we now test-play (software for Windows on blackboard):
<http://arimaa.com/arimaa/>



- and now we look at some evaluation functions:
<http://arimaa.janzert.com/eval.html>

representation + utility = ?



representation + utility = learning

- how do I build an artificial brain?
- basic
 - I build it from scratch based on my brain!
- advanced
 - I let it learn based on samples of good behavior
 - I let it learn by rewards/penalties
 - I let it learn by itself!

(do you recognize the types of learning?)

utility = training signal

- where does it come from?
- I have a good idea of what good is – don't need to learn a heuristic!
 - expert-knowledge systems (FSMs, behavior trees, fuzzy rule-based systems).
no learning!
- is it sampled from data (good input-output patterns)?
 - supervised learning (backpropagation – artificial neural networks, decision trees)
- is it provided by the environment (rewards for doing something well) ?
 - reinforcement learning (brute force, evolutionary algorithms, temporal difference learning)
- is it provided internally (within the representation) ?
 - unsupervised learning (self-organization, competitive learning)
- is it provided as pairwise preferences or rankings ?
 - preference learning (support vector machines, neuro-evolution, back propagation)

what does our training signal measure?

- in games, the signal is usually not static but depends on the current state (as in dynamic optimization)
- uncertainty may be involved (non-determinism)
- partial observability
- our heuristic may be wrong
- or incomplete



picture from Michael Misof on Pixabay

utility: hints + tips

- no free lunch
- difficult to design an appropriate heuristic
- sometimes it is impossible...
- simplicity pays off
- completeness as well
- remember: this is the driving force of learning
- again, there is **black art** involved...



picture from Stefan Keller on Pixabay

Evolutionary Algorithm primer

a type of black-box optimization algorithms:

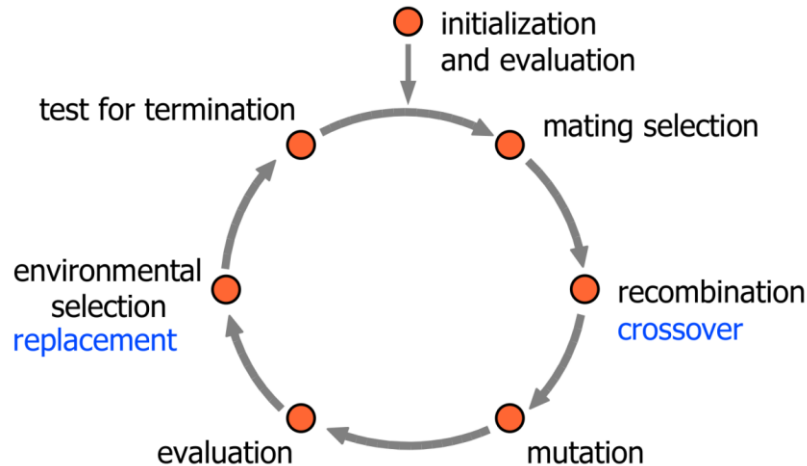
- usually population based (a “population” of candidates is evolved simultaneously)
- evaluate the “fitness” of a solution candidate by means of a target function (usually “black box”, not tractable analytically)
- remove the worst candidates
- replace these with copies of the best candidates or mix new candidates from several good ones
- mutate (variate) the copies
- important:
variation randomized, selection deterministic (usually)



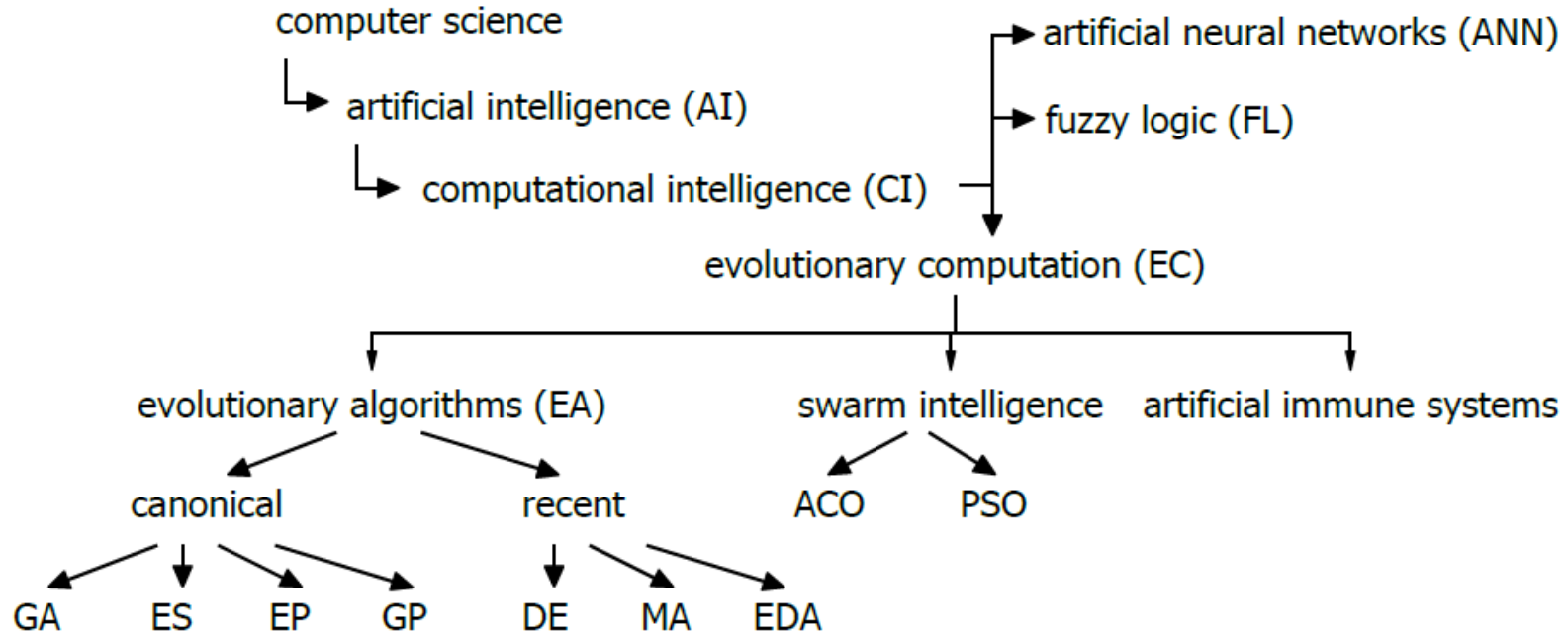
picture from Karsten Paulick on Pixabay

evolutionary cycle

- start solutions needed: existing solutions or randomly generated ones
- target function shall measure quality we want to improve
- matching variation operators are necessary



Computational Intelligence

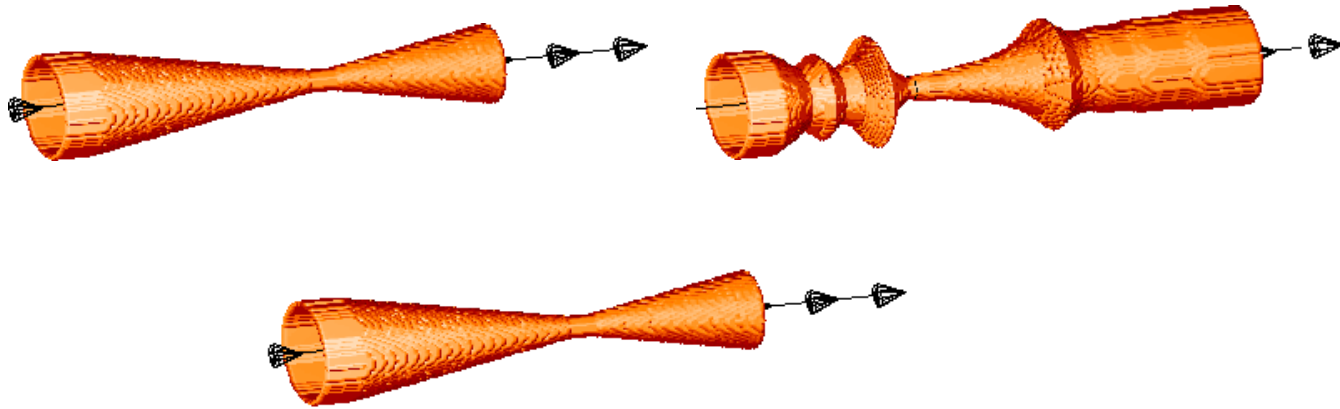


GA = genetic algorithm, ES = evolution strategy, GP = genetic programming,
DE = differential evolution, PSO = particle swarm optimization, ACO = ant colony optimization

the very first evolution strategy

- "experimentelle optimierung einer 2-phasendüse"

original experiment with 1 cm discs: Hans-Paul Schwefel (1970)



- population: 1 individual, per generation one offspring, (1+1)-ES

<https://ls11-www.cs.uni-dortmund.de/people/schwefel/EADemos/>

pros and cons of evolutionary algorithms

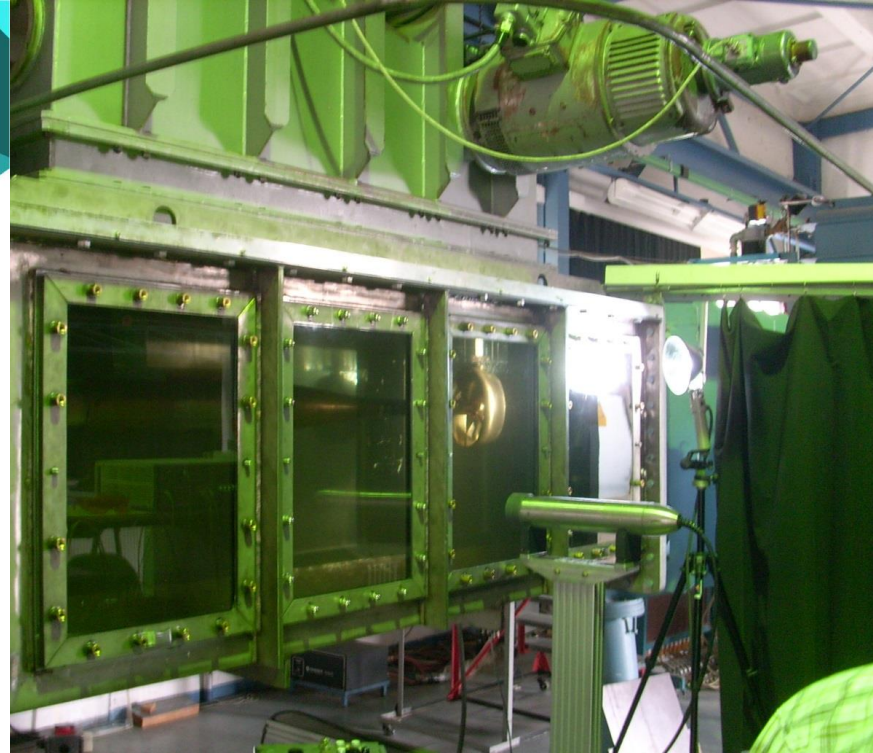
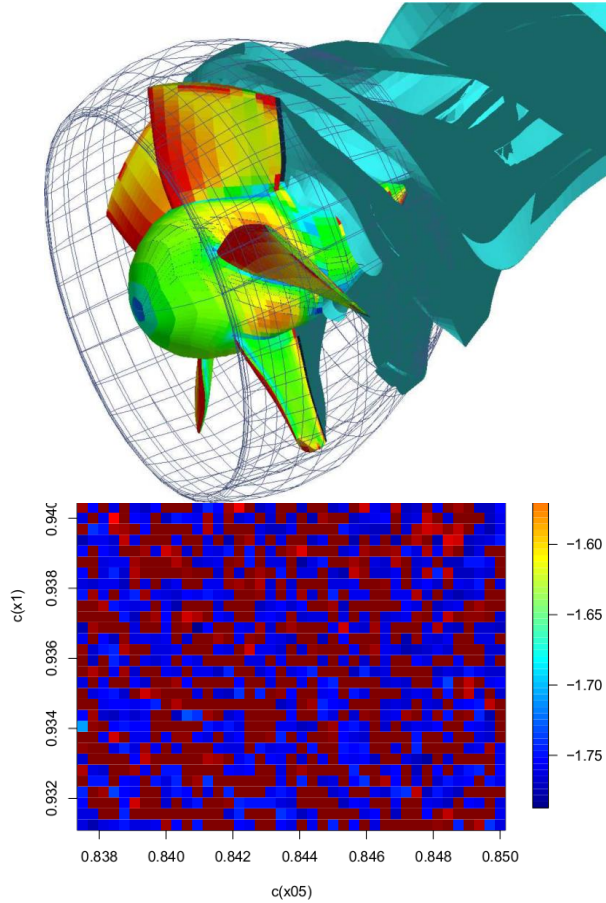
pros:

- flexible, mainly needs utility/target/fitness (synonyms) function
- quite insensitive concerning noise or errors
- anytime algorithm: you get a result whenever you want

cons:

- optimality cannot be guaranteed (only in very special cases)
- stagnation phases of unpredictable duration (when to stop?)
- depends on good representation and matching operators
- too slow on simple problems

industrial application: linear jet (ship engine)



preparation of the measurement

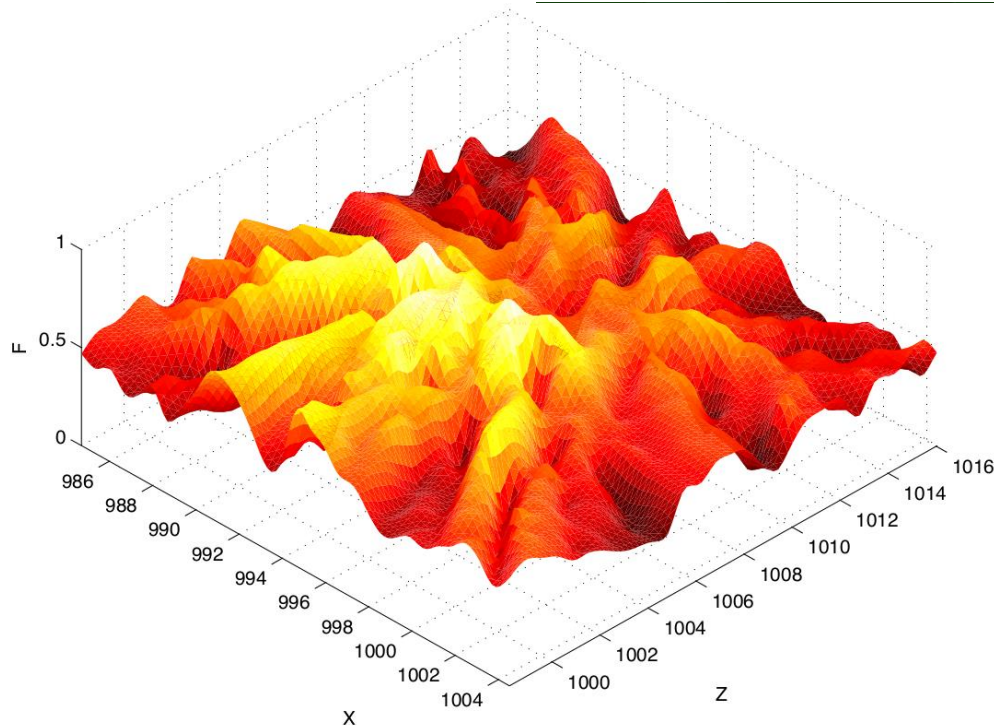


cavitation



fitness landscapes

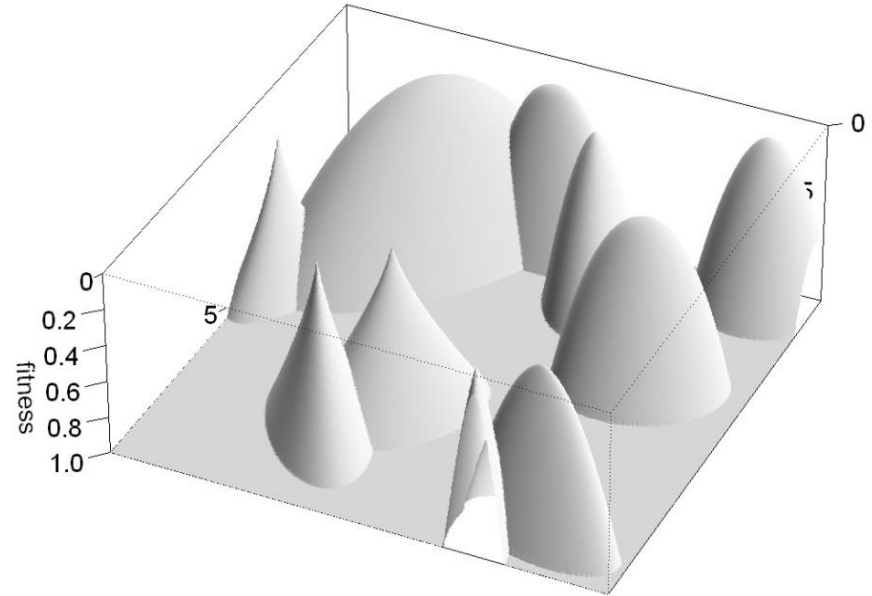
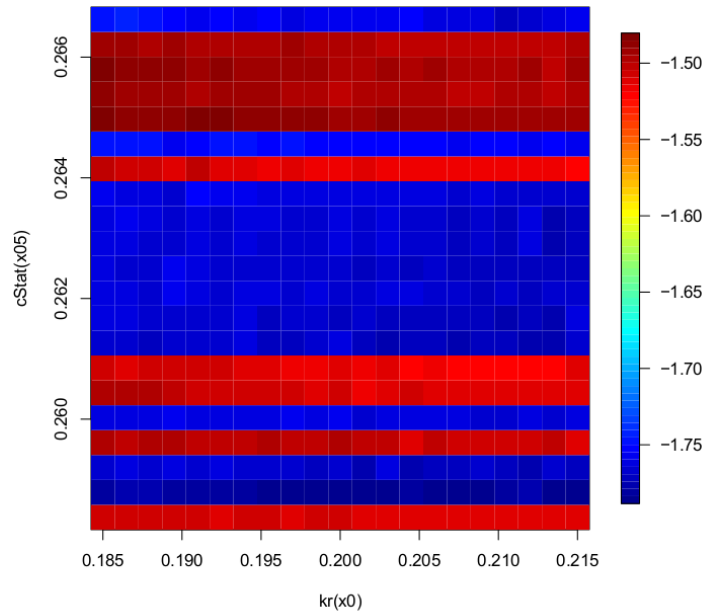
- (camera positioning problem)



Preuss, Burelli, Yannakakis: [Diversified Virtual Camera Composition](#).
EvoApplications 2012

the locality principle

- small movements in the search space lead to small changes in the target function
- cliffs and “needle in the haystack”-problems very difficult
- selection leads population to local optima (restarts help)



how do I set up my evolutionary algorithm?

necessary contents:

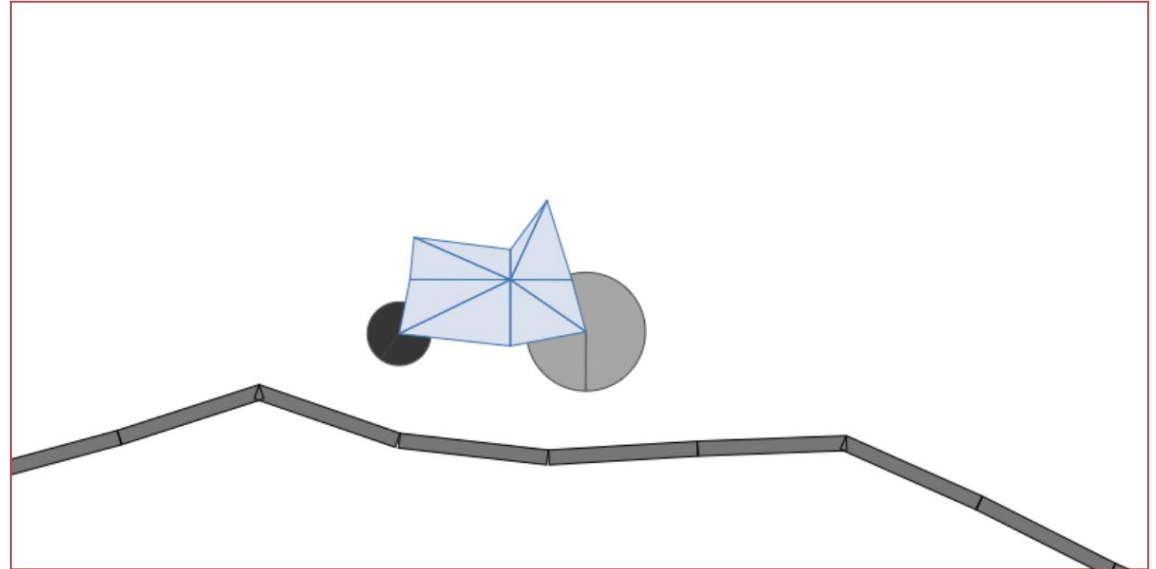
- a representation (usually numerical or binary string)
- a utility function
 - direct, based on simulation, or interactive (user)?
- variation operators
 - strength of variation should be controllable via parameters
 - every point of the search space shall be reachable
 - variation operators shall not be biased in one direction
- for uncommon representations: distance function useful
- termination criterion (run length, quality, stagnation,...)

a running example: genetic cars

start evolutionary algorithm: https://rednuht.org/genetic_cars_2/

questions:

- what is the representation?
- what is the utility function?
- which parameters control the algorithm?



generic EA

parameters:

population size μ , offspring size λ , problem dimension d

algorithm:

1. set up initial population randomly (μ arrays of size d)
2. evaluate initial population
3. for i in 1 to λ :
 4. randomly choose 2 parent individuals a and b
 5. create one offspring individual c (usually by mixing a and b)
 6. mutate offspring individual c by adding gaussian random number to every one of the d components
 7. evaluate individual c , add it to population
8. select best μ individuals as next parent population, remove the rest
9. if not termination, go to step 3

(many variants of this exist)

how to apply an evolutionary algorithm

1. choose the representation
2. define fitness
3. choose EA type (ES, GA, CMA-ES, EMOA)
4. identify EA parameters
5. define stopping criterion
6. go and have a
 - coffee
 - lunch
 - holiday!

ES = evolution strategy

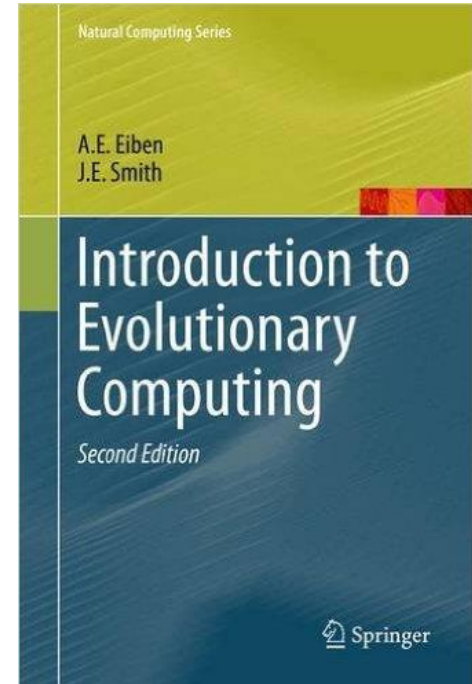
GA = genetic algorithm

CMA-ES = cumulative matrix adaptation ES

EMOA = evolutionary multi-objective
optimization algorithm

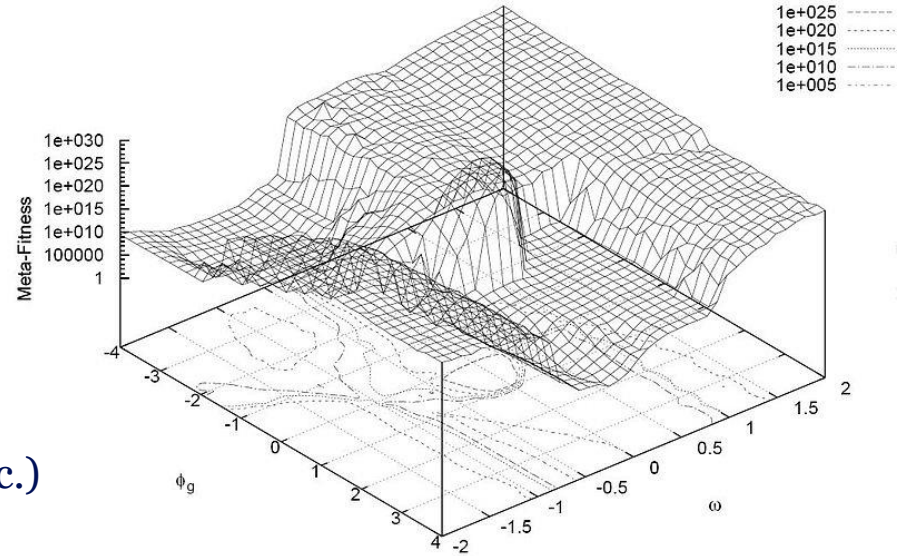
good reference:

Eiben, Smith: Introduction to Evolutionary Computing (2nd ed.), Springer, 2015



EA parameters / operators

- how large should the population be?
- this effects running times!
- large populations are slower
- but search highly parallelized
- can we do crossover / recombination?
difficult for many representations
- how strong shall mutation be?
- shall it change over time? (self-adaptation etc.)
- selection method:
plus-selection includes parents (safe)
comma-selection disregards parents (fast)



PSO meta-fitness:
effect of 2 parameters on aggregated
performance over 12 benchmark problems

EA (metaheuristics) vs brute force

we can apply these methods to every measurable problem, see the Blind Watchmaker Framework <http://watchmaker.uncommons.org/>

start in main directory:

```
“java -jar watchmaker-examples-0.7.1.jar salesman”
```

- try with 5 cities, compare EA and brute force solution times
- increase number of cities 1 by 1
- does the EA deliver the optimal solution?
- for how many cities does the EA get faster than brute force search?

Metaheuristics means:

- base algorithm that can be adapted to the problem
- adaptation by changing parameters and (variation) operators
- problem: difficult to make general statements about quality (which “instantiation” is meant, what parameters, which representation ?)

how to evolve behavior

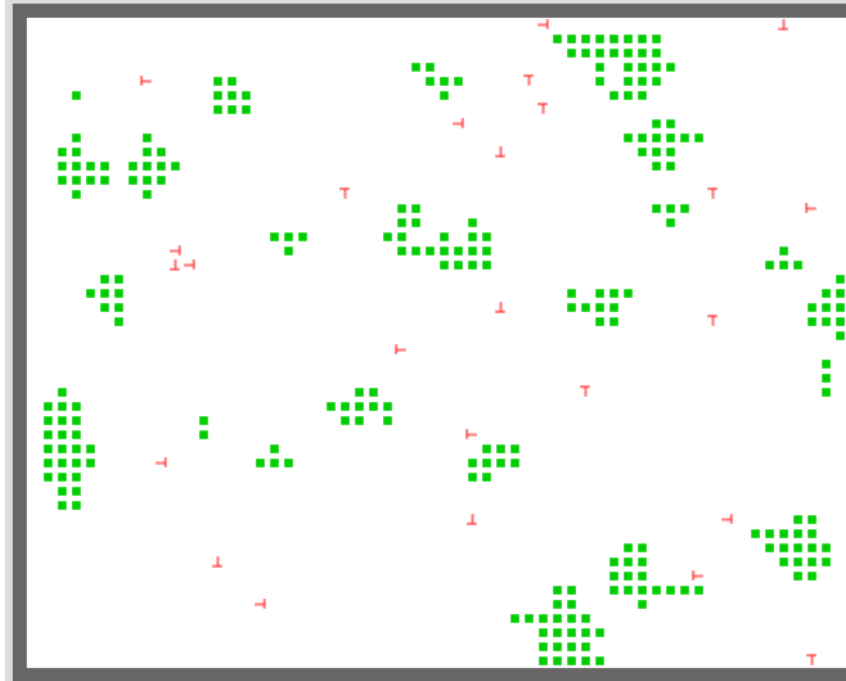
<http://math.hws.edu/eck/js/genetic-algorithm/GA.html>

how is behavior encoded?

Genetic Algorithms Demo

For more information about the genetic algorithm and this program, see [ga-info.html](#).

Starting with World No. 1! Click Run or Step.



GA-eaters

we have 4 possible actions:

1. move forward one square
2. move backwards one square
3. turn in place 90 degrees to the left
4. turn in place 90 degrees to the right

- every eater has 16 possible states
- it can see just the square in front of it (empty, wall, plant, eater)
- we encode rules: 16 states x 4 possible items in view = 64 rules
- every rule contains one action and one new state number
- how does the eaters behavior change over time?

take home

- we have different types of learning: unsupervised, supervised, reinforced
- representation determines what steps are possible for learning
- utility needs to provide learning signal
- optimization often used as tool for driving the learning process
- Evolutionary Algorithms simple and capable anytime black-box optimizers



picture from OpenClipart-Vectors on Pixabay