WCCI-CEC 2020 Tutorial, 19 - 24th July, 2020, Glasgow (UK)

Visualising the search process of EC algorithms

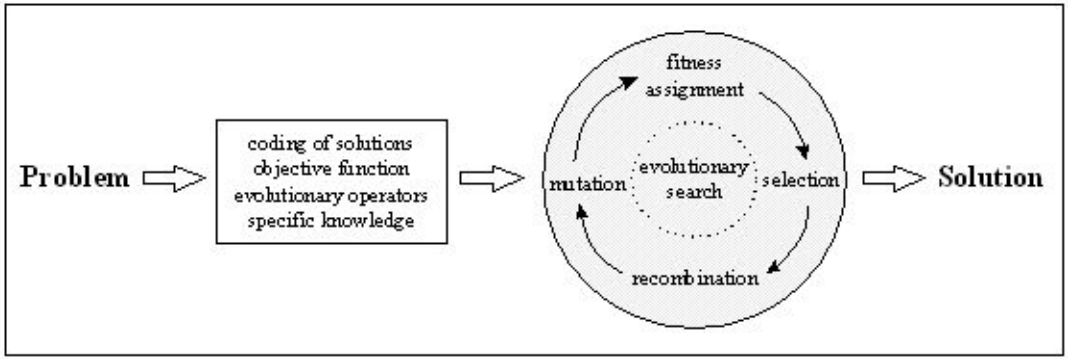
Su Nguyen, Yi Mei, Mengjie Zhang

Outline

- Motivations
- Review of DataViz
- DataViz for EC algorithms
- AI-based visualisation (AIV)
- Case studies
- From AIV framework to people-centric evolutionary systems

Evolutionary Computation

- Evolutionary computation (EC) is a computational intelligence approach inspired from natural evolution.
- Many applications
  - engineering,
  - finance,
  - supply chain management
  - etc.
- Thousands of algorithms
  - GP, GA, ES
  - PSO, DE
  - ACO
  - etc.



https://www.doc.ic.ac.uk/project/examples/2005/163/g0516312/images/a.gif

Why do we need to invent new EC algorithms?

- To find more effective algorithms
- To cope with new problems/applications
- To improve the scalability of EC algorithms
  - Cope with computationally expensive problems
  - Cope with high-dimensional problems
- To cope with dynamic changes and uncertainties
- To cope with multiple conflicting objectives
- To cope with multiple tasks
- To handle complex constraints
- To incorporate users' preferences

Visualisation in EC

- Visualisation is a popular approach to summarise the large amount of data generated in EC experiments:
  - Compare the final solutions obtained by different EC algorithms or configurations
  - Visualise the individual evolved solutions
  - Visualise the genetic or perturbation operators
  - Monitor the progress of EC algorithms
    - Fitness
    - Diversity
    - Size (e.g. with genetic programming)
    - Trajectory/Path through the search space
    - Check if the algorithms behave as conceptually designed
- Challenges: high-dimension, many aspects (fitness, diversity, etc.)

Need advanced visualisation techniques

- Dimension reduction
- Topological representation of evolved solutions
- Systematic combination of visual variables
- Scalability
- Flexibility
- Robustness
- (Interactivity)

DataViz

Theories

Visual concepts

Chart selection

Examples

Review of DataViz

- Theories
- Visual concepts
- Chart selection

Noted: this is not a comprehensive review of data visualisation but rather used to refresh your knowledge about data visualisation. Ones who are really interested in data visualisation may find the below materials useful:

- Fundamentals of Data Visualization (Claus Wilke)
- The Data Visualisation Catalogue
- Visualization with Matplotlib

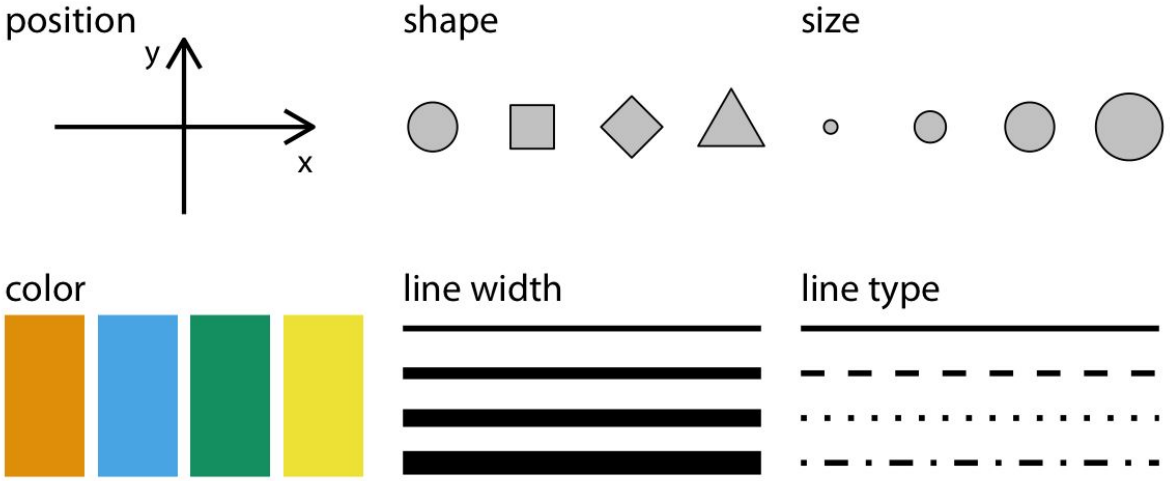
Visualisation steps

Systematic steps to develop a visualisation.

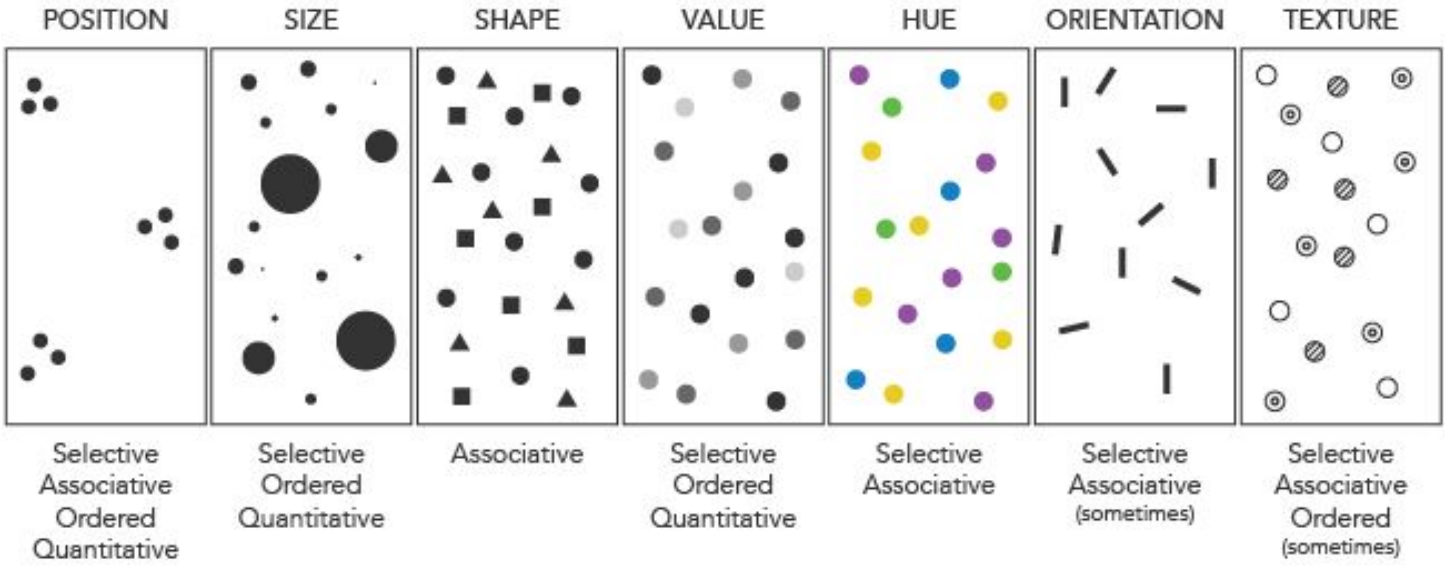
1. Determine what you are trying to show
2. Choose a visualisation method
3. Create the visualisation
4. Distribute your work

Visual concepts

Different visual variables can be used to visualise the data of interest. Some basic visual variables from “Fundamentals of Data Visualization”:

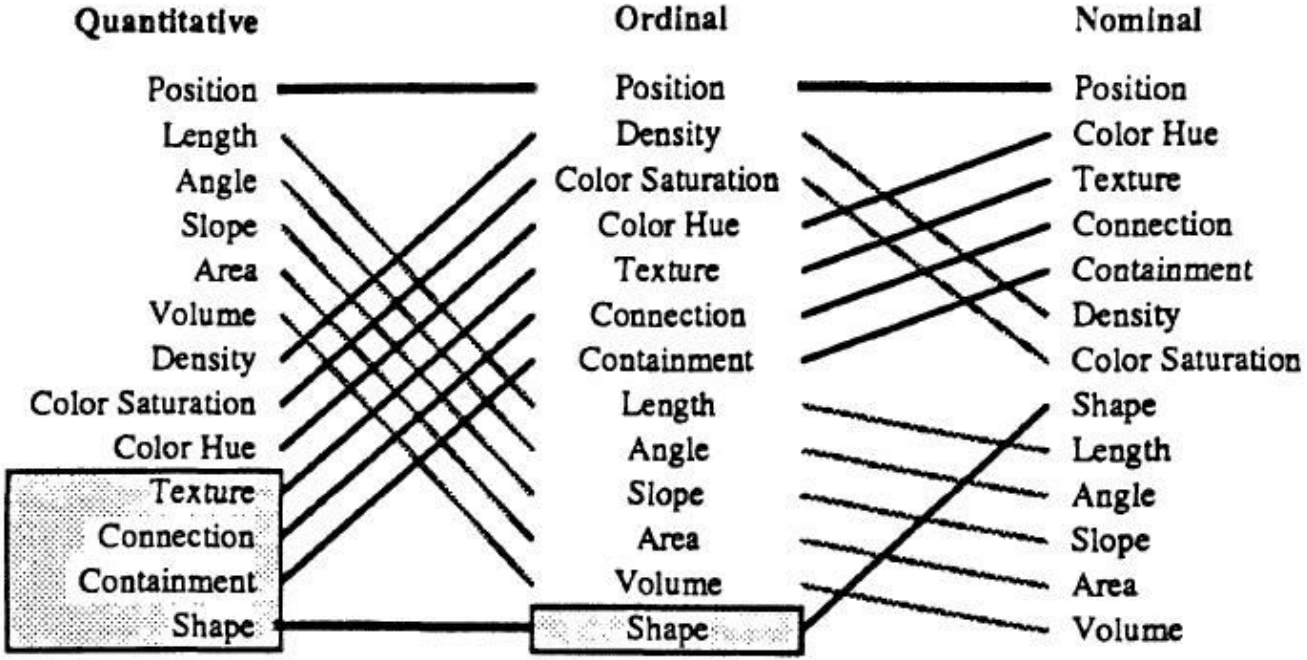


Bertin's visual variables

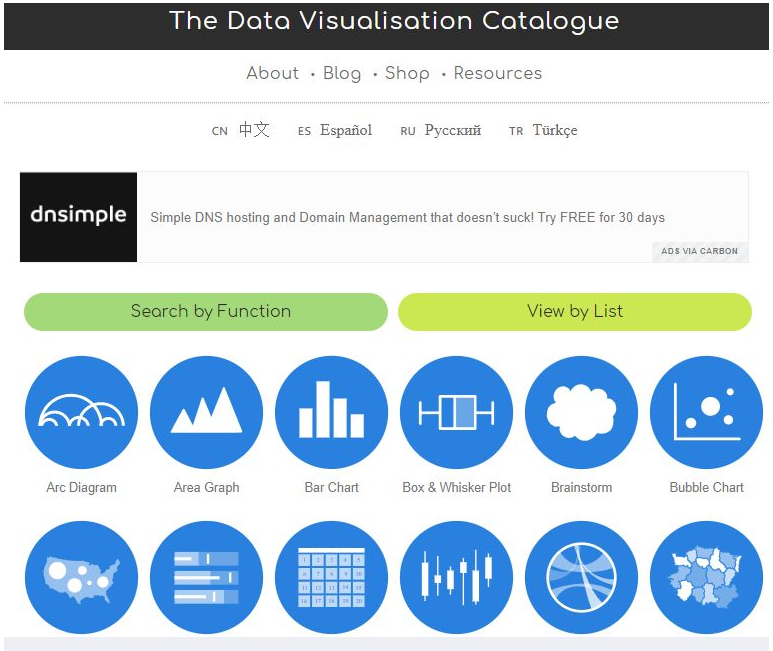


https://www.axismaps.com/guide/general/visual-variables/visual_variables.png

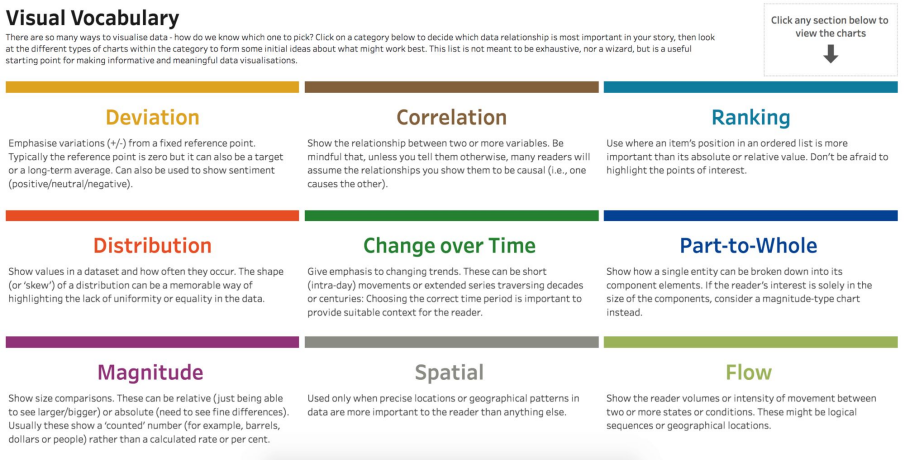
MacKinlay's ranking of visual elements/variables



Choose chart types by based on data relationship



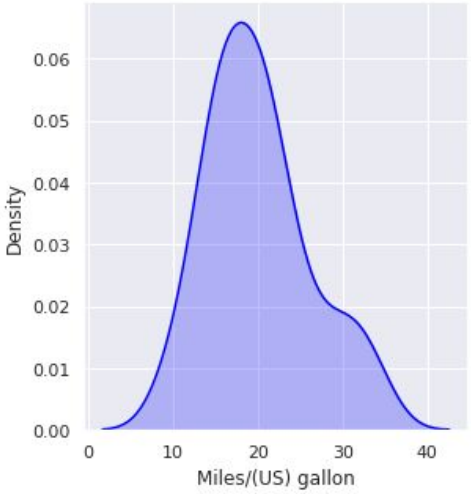
https://datavizcatalogue.com/



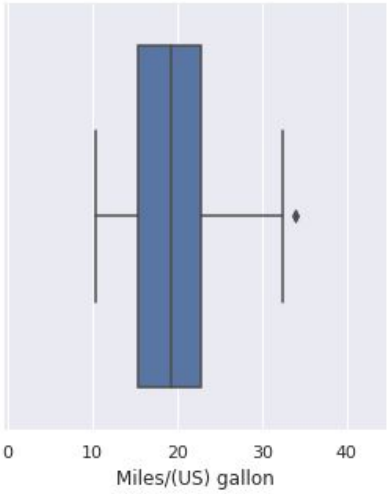
https://www.tableau.com/solutions/gallery/visual-vocabulary

Example: Visualise distribution

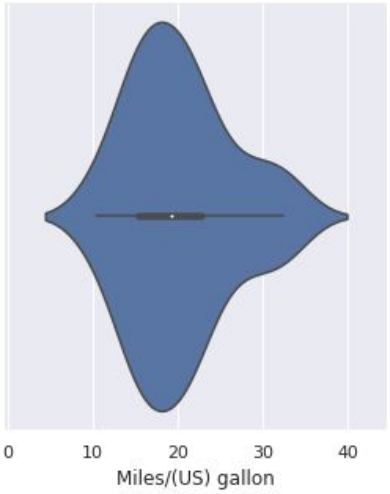
mtcars
dataset



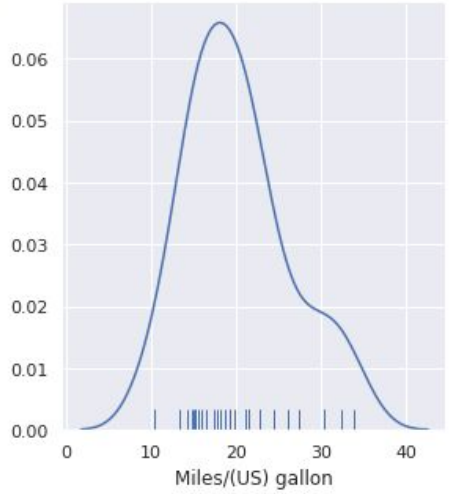
Density plot



Boxplot



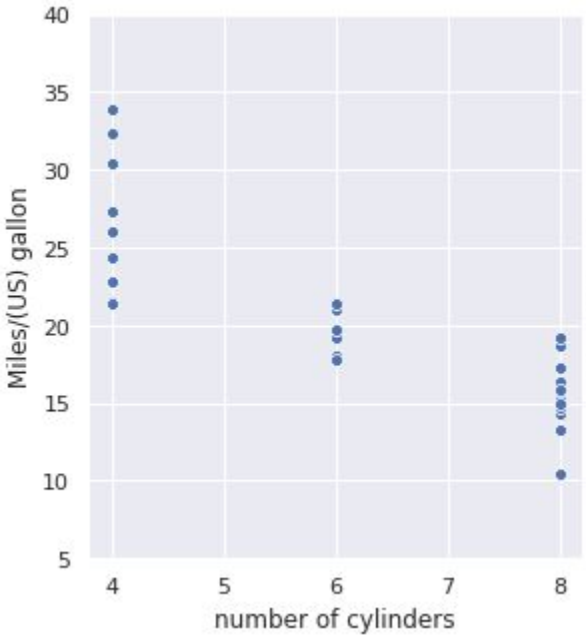
Violin plot



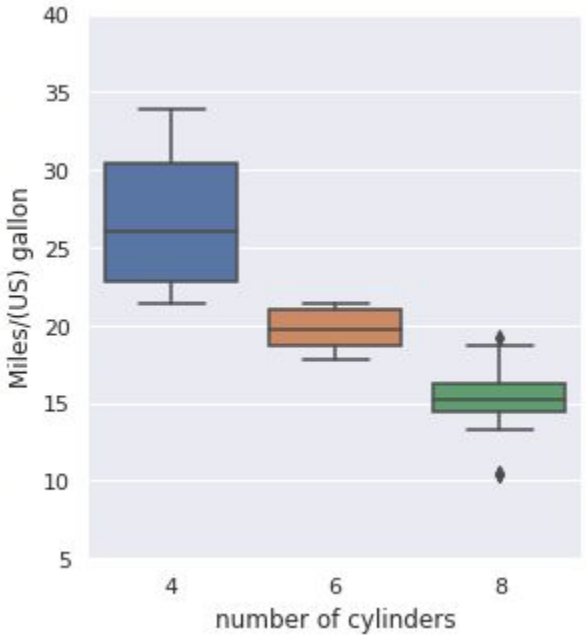
Hybrid plot

Example: Compare distributions

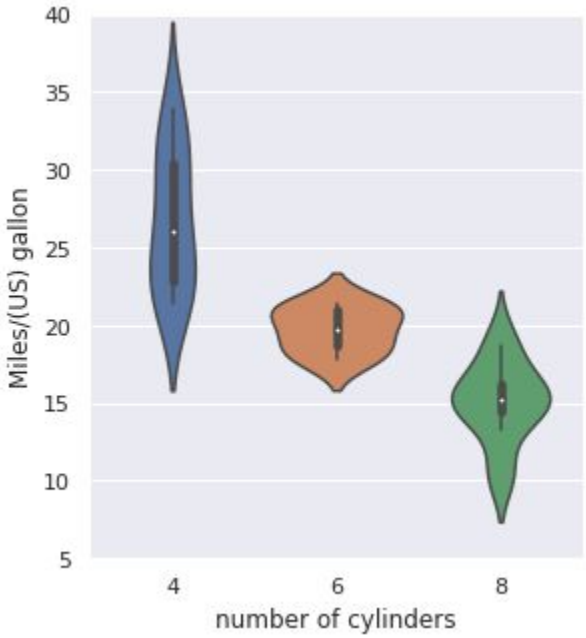
mtcars
dataset



Scatter plot



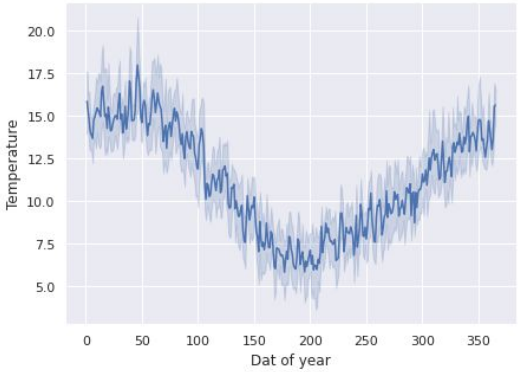
Boxplot

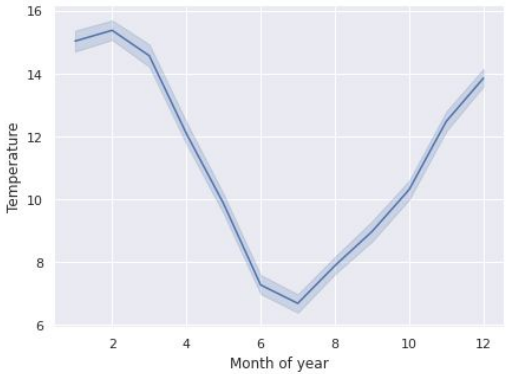


Violin plot

Example: Visualise changes over time

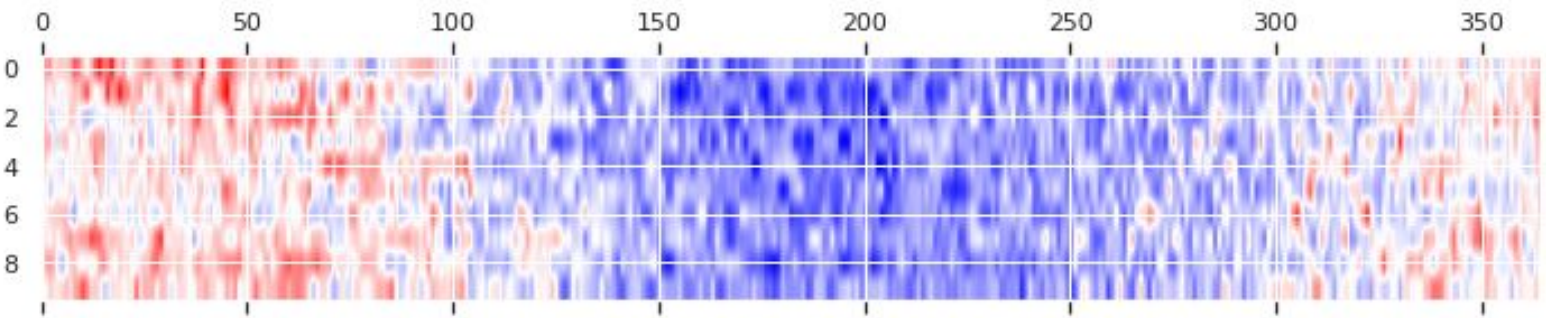
Daily Temperature dataset





Line charts with error bands

Time-serie Heatmap



How to cope with high-dimensional data

Correlation matrix

Useful but may lose a lot of information!

Information overload!

Scatter plot matrix

Apply dimension reduction - PCA

Principal Component Analysis

Apply dimension reduction - UMAP

Uniform Manifold Approximation and Projection

Visualisation for EC

Compare the final solutions obtained by different EC algorithms or configurations

Monitor the progress of EC algorithms

Python scripts

- All source codes to generate all visualisations here can be found at:

  https://github.com/nguyensu/visevo/tree/master/notebooks

- Required packages
  - **deap** for EC algorithms
  - **pandas** for data management
  - **seaborn** and **matplotlib** for visualisation
  - **statannot** for statistical tests and annotations
- The Python notebooks can be opened with **Google Colab**

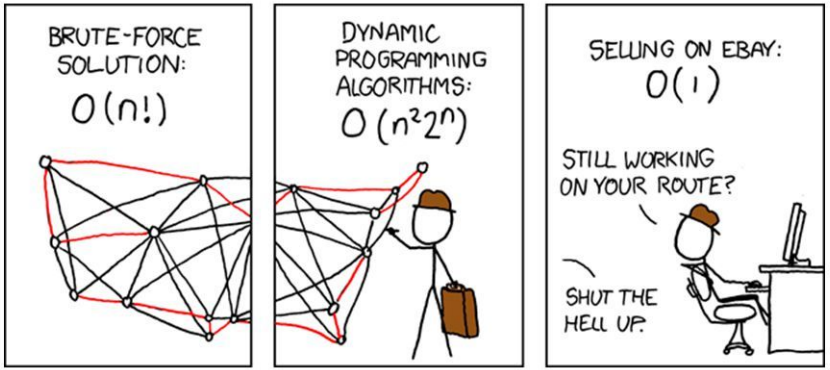
Genetic algorithm

1. Initialisation
2. Evaluation
3. Selection
4. Crossover and mutation
5. If not terminated, return to step 2; otherwise, stop

OneMax problem

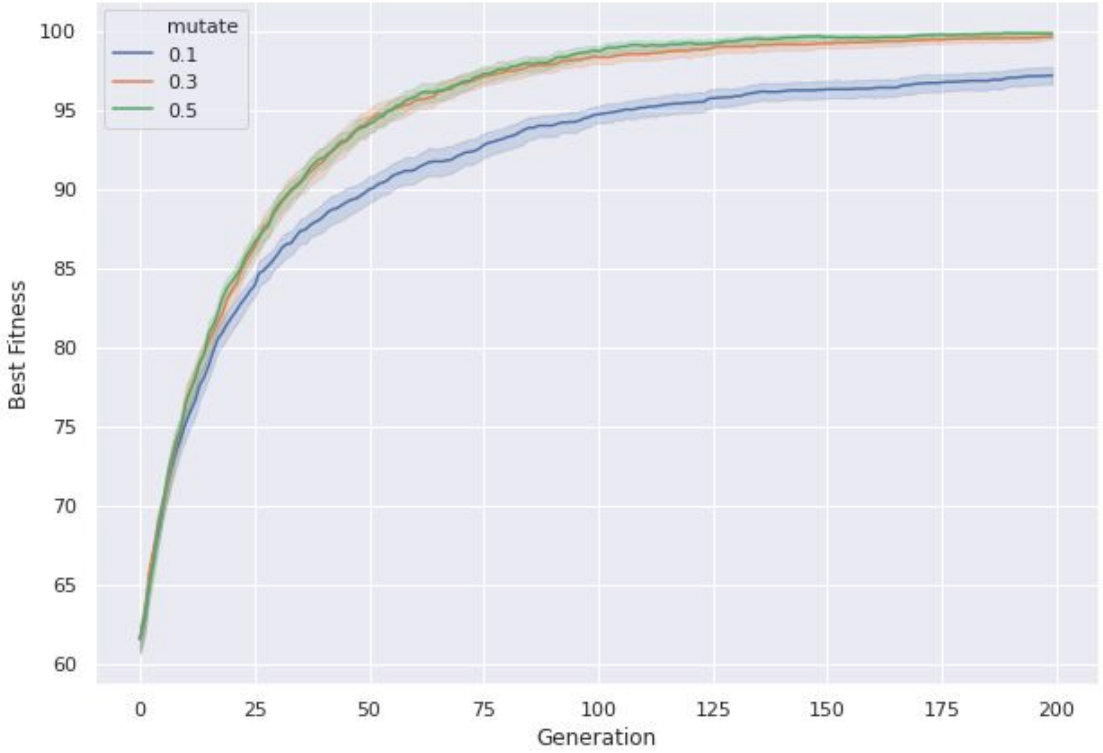
The **OneMax Problem** (or BitCounting) is a simple **problem** consisting in maximizing the number of ones of a bitstring

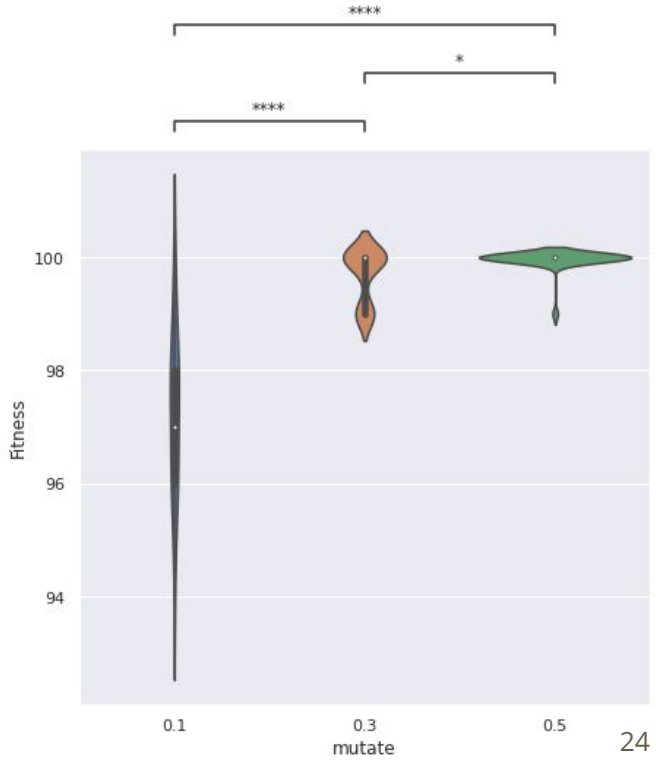
Travelling salesman problem



Genetic algorithm for solving OneMax problem

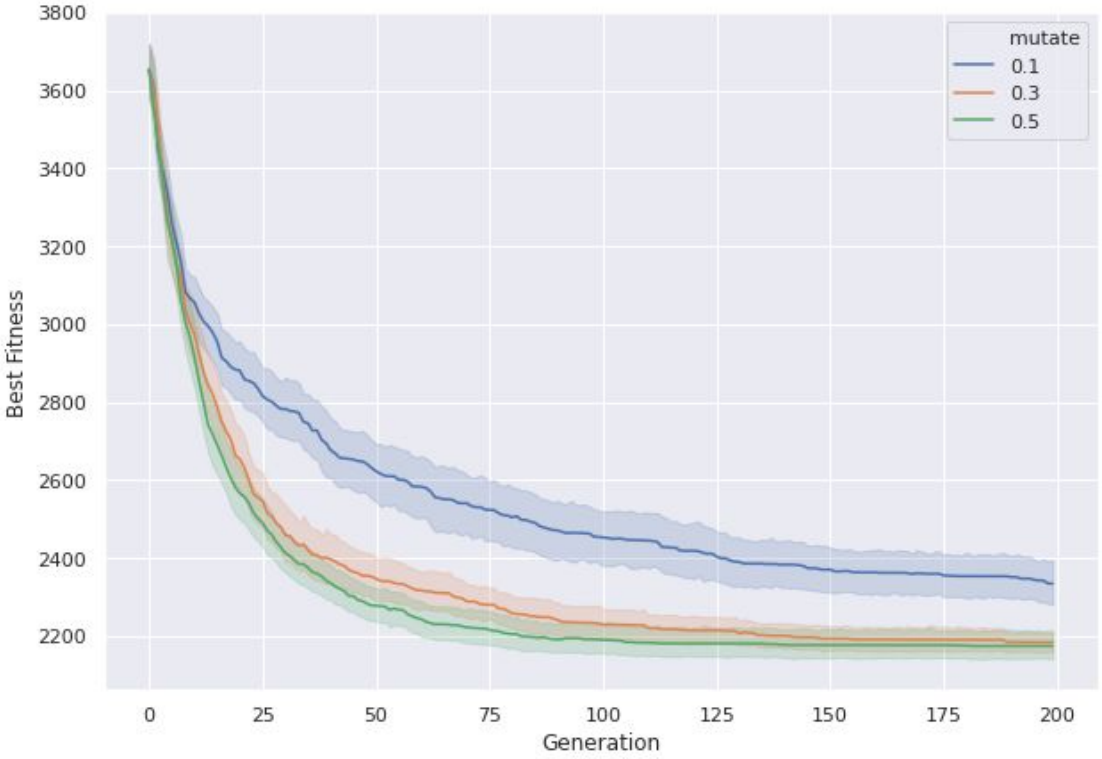
OneMax
n=100

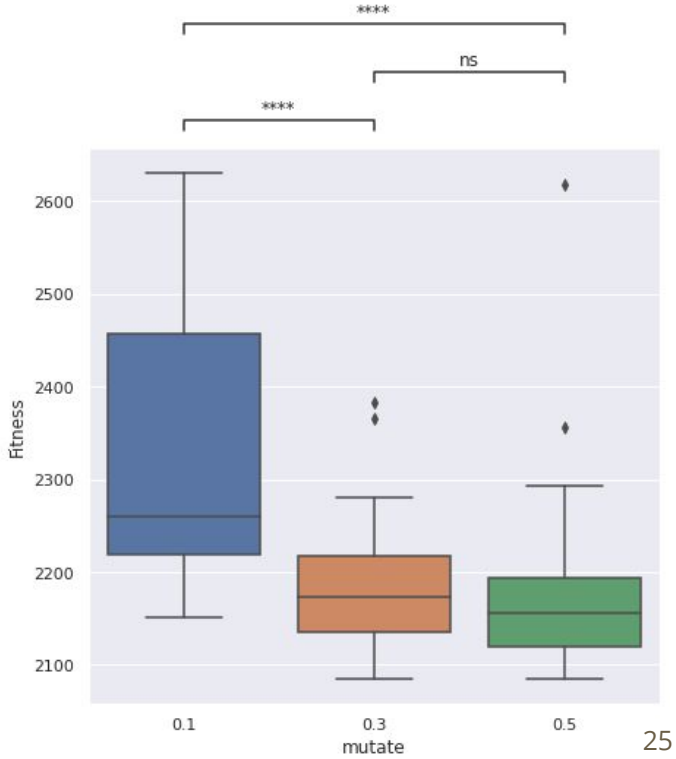




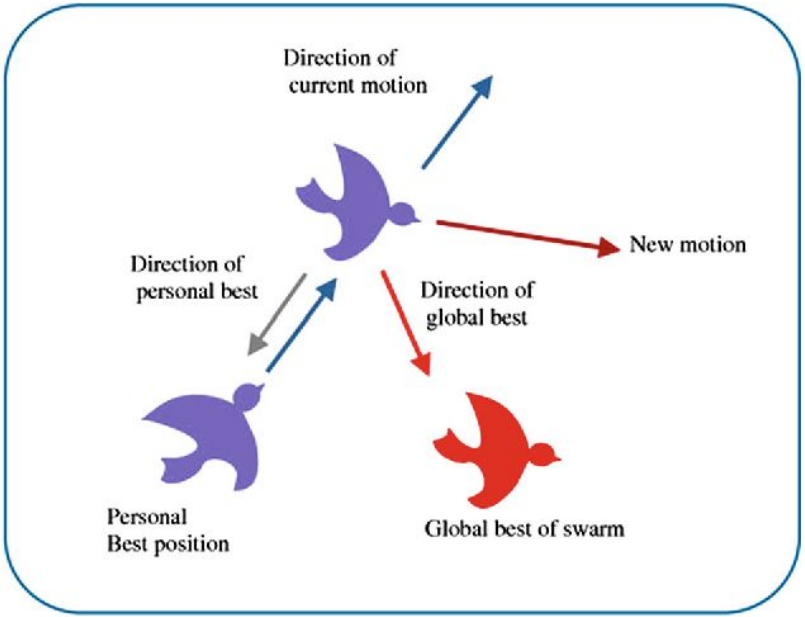
GA for Travelling Salesman Problem (TSP)

TSP
gr17
instance

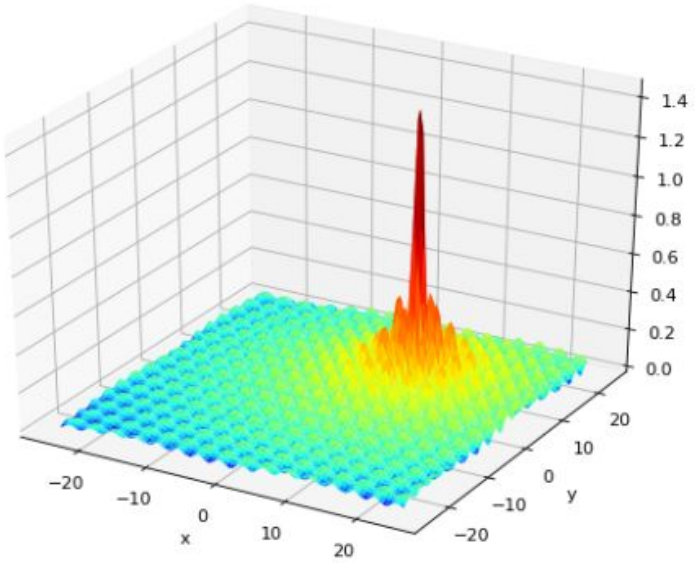




Particle Swarm Optimisation for Function Optimisation



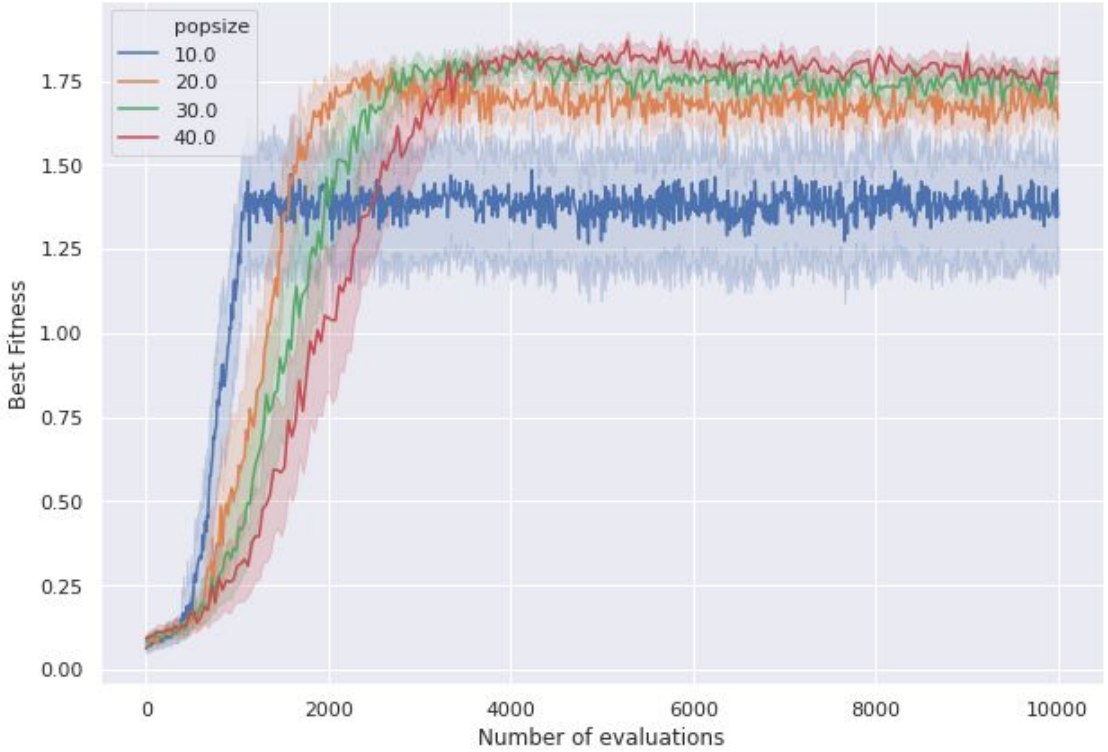
$$f(\mathbf{x}) = \frac{\sin(x_1 - \frac{x_2}{8})^2 + \sin(x_2 + \frac{x_1}{8})^2}{\sqrt{(x_1 - 8.6998)^2 + (x_2 - 6.7665)^2} + 1}$$

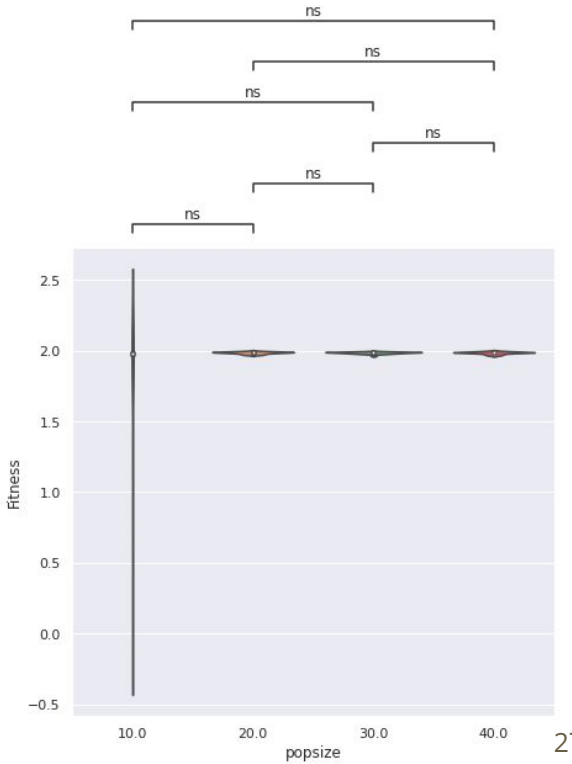


https://deap.readthedocs.io/

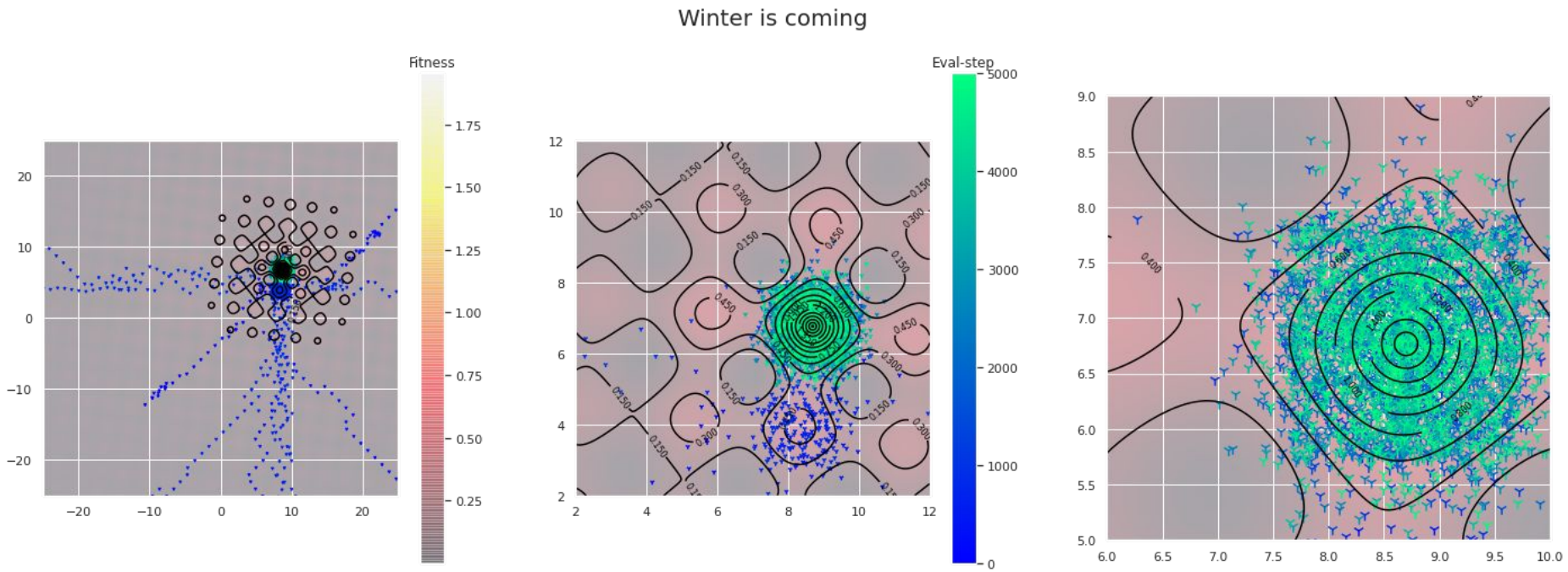
Esmin, Ahmed Ali Abdalla et al. “A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data.” Artificial Intelligence Review 44 (2013): 23-45.

Particle Swarm Optimisation for Function Optimisation

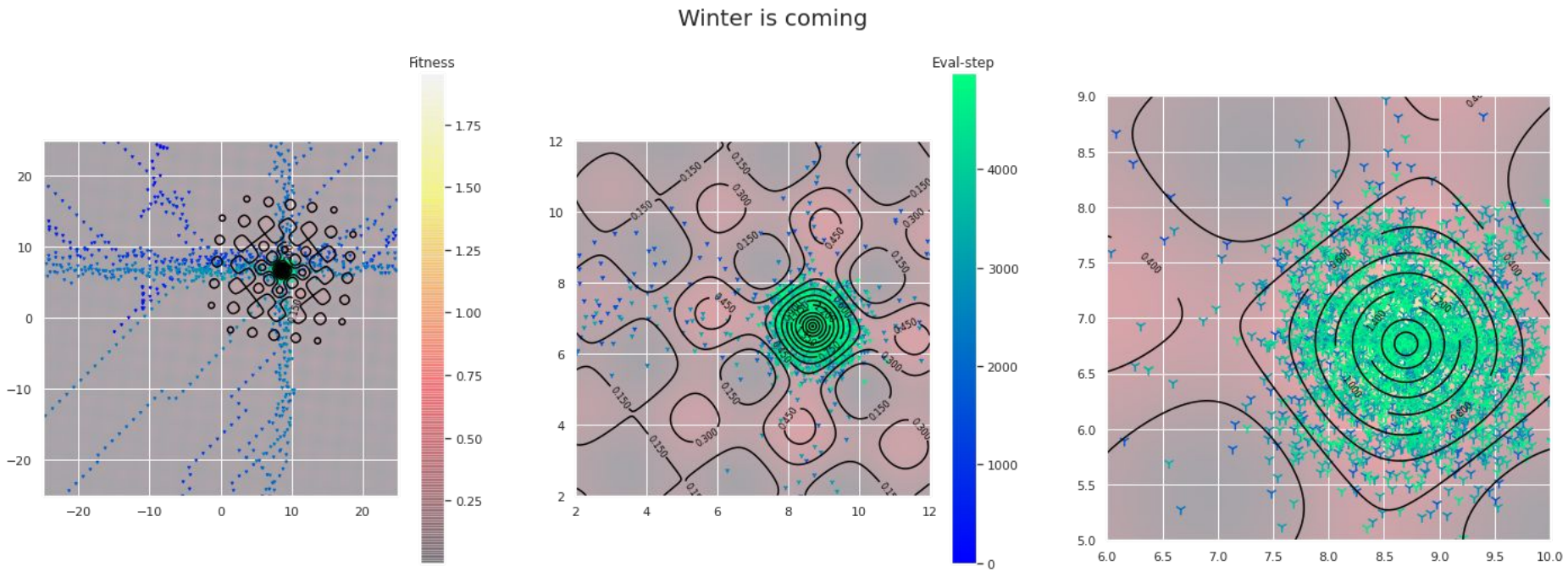




How the swarm explore the search space? (1)



How the swarm explore the search space? (2)



Visualise search process with high-dimensional problems

Symbolic Regression

Binary classification

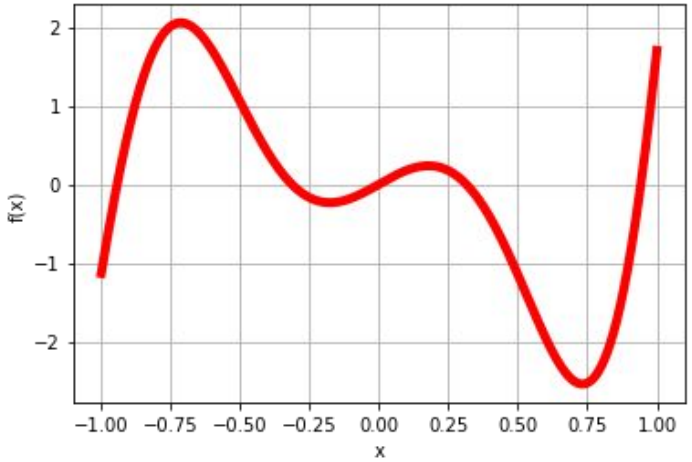
What information are we interested?

- What solutions have been generated?
- What are the relationships between generated solutions?
- What are the diversity of generated solutions?
- Which are the areas that the algorithms have explored?
- How do the algorithms explore the search space?
  - Overly explore a specific area (trapped in a local optimum)?
  - Randomly explore the search space?
  - Balance between exploration and exploitation?
  - etc.

Genetic programming for symbolic regression

- Apply GP to evolve a regressor
- Discover a function based on the data generated by:

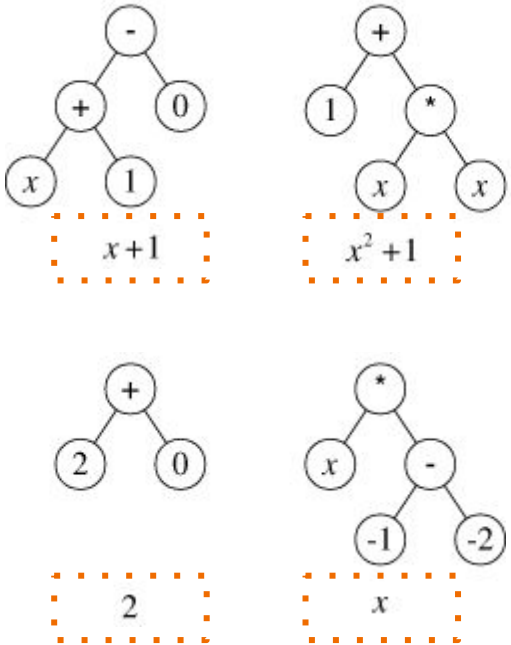
f(x) = (x**4 + 2*x + 3*x**3)*cos(x*5)



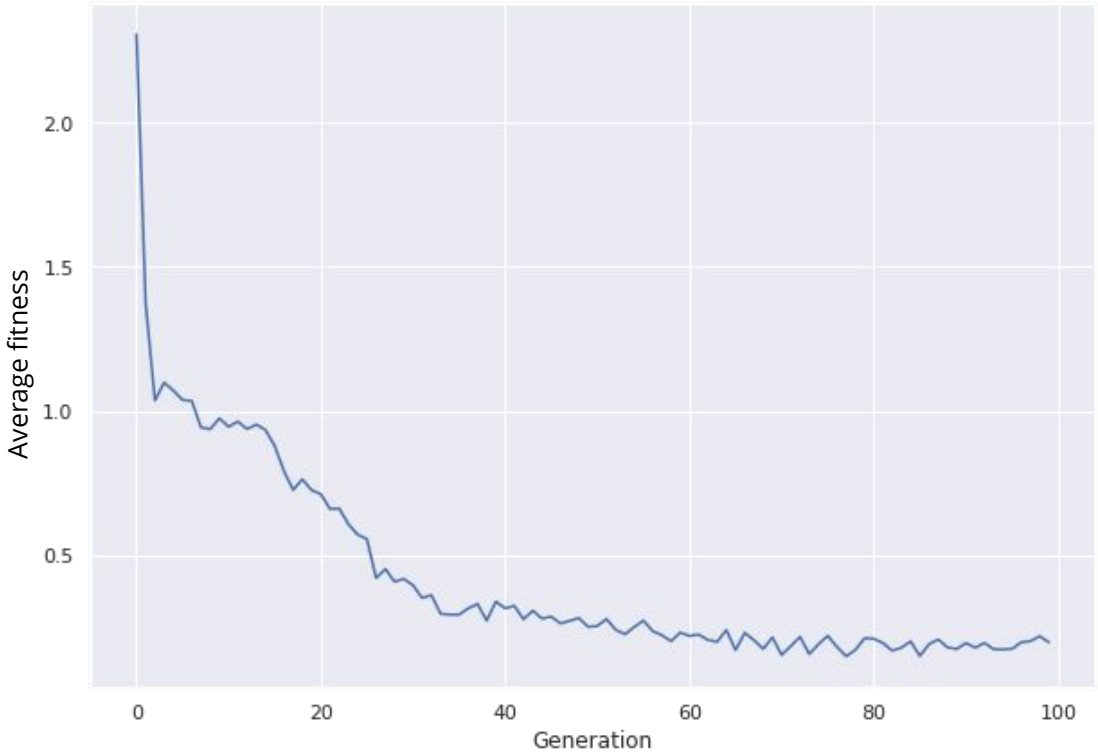
x	f(x)
-1.	-1.13464874
-0.99979998	-1.1304163
-0.99959996	-1.12618549
...	...

Parameter settings

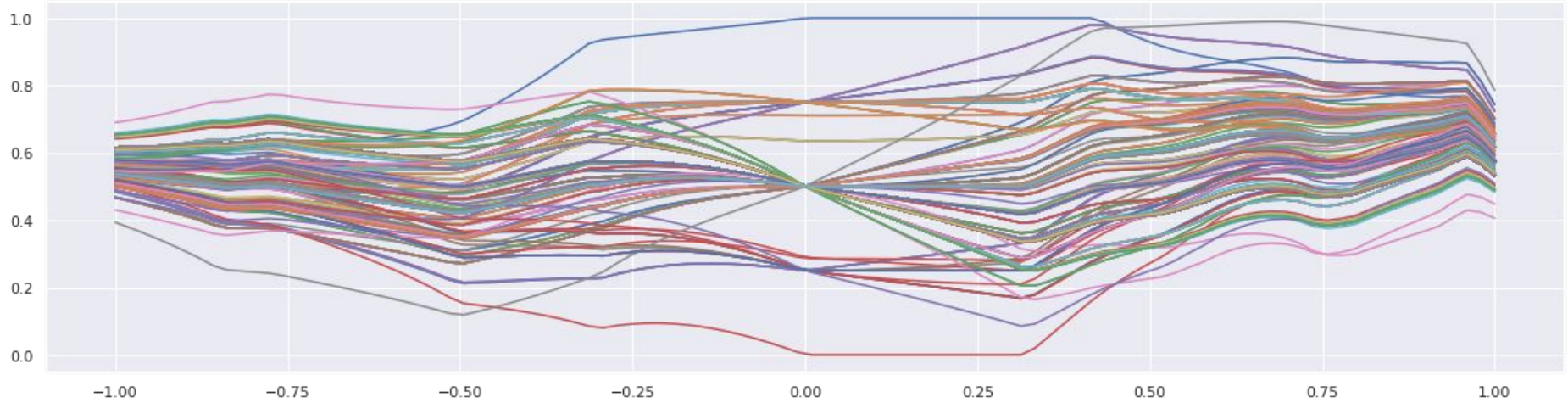
- Representation: tree structure
- Terminal set: x
- Function set: +, -, *, protected division, sin, cos
- Population size = 1000
- Number of generation = 100
- Subtree-mutation prob. = 0.1
- Subtree-crossover prob. = 0.9
- Tournament selection (size = 5)
- Fitness function: Mean Squared Error (MSE)
- Phenotype vector (dim=100):
  - g(x') - f(x') with x' sampled from the dataset
  - Data subset used for phenotype calculation is pre-determined
  - This phenotype shows how well the evolved function fits the true function f(x)



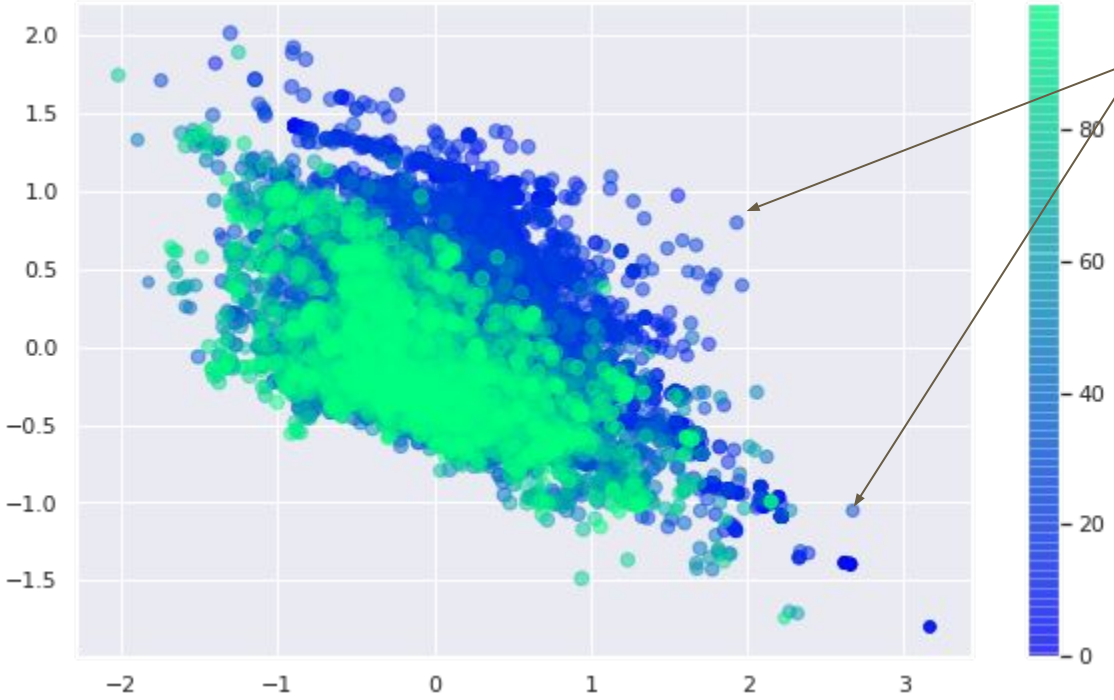
GP progress



Recorded phenotypes



Visualise the search process with PCA

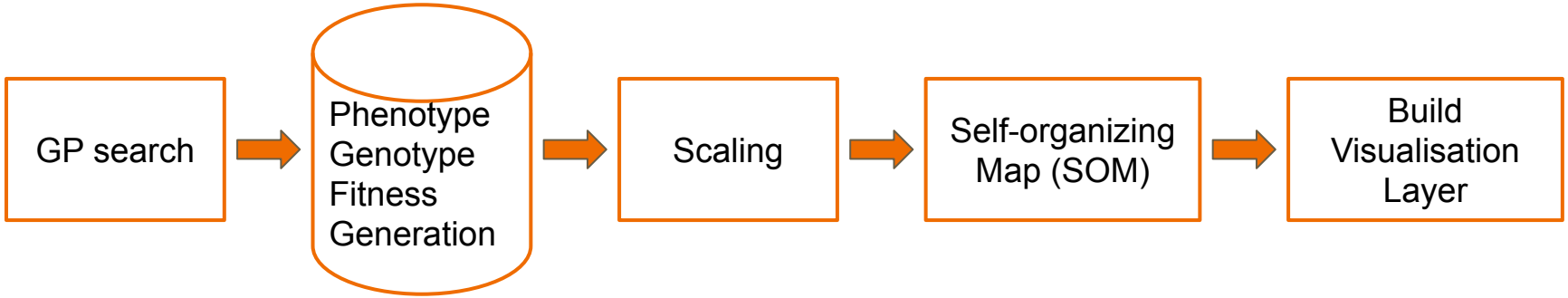


Principal Component Analysis

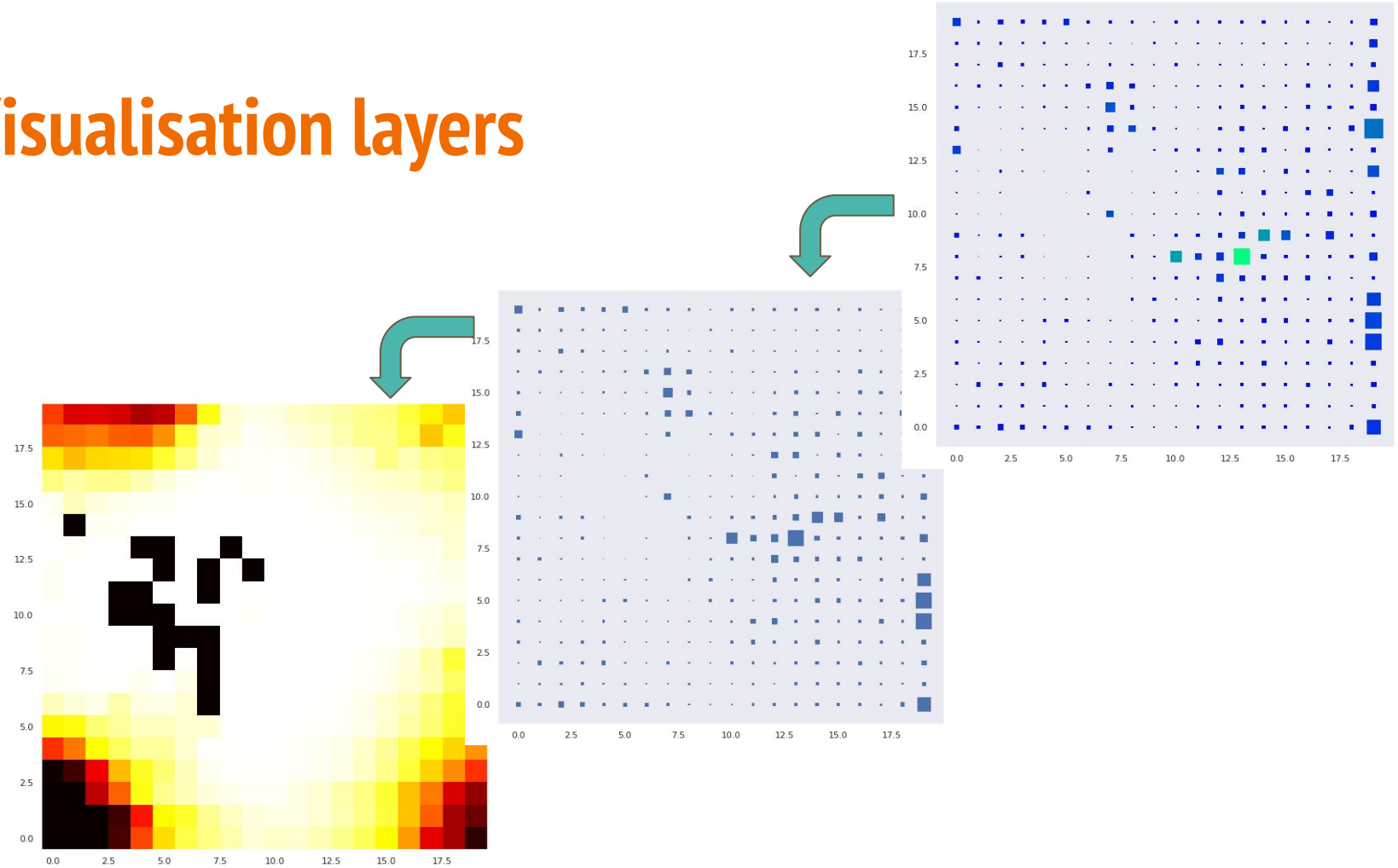
Each point represents a specific behaviour (or curve/shape)

Programs with different genotypes can have similar or the same phenotypes, e.g.
(x - 1)*x = x*x - x

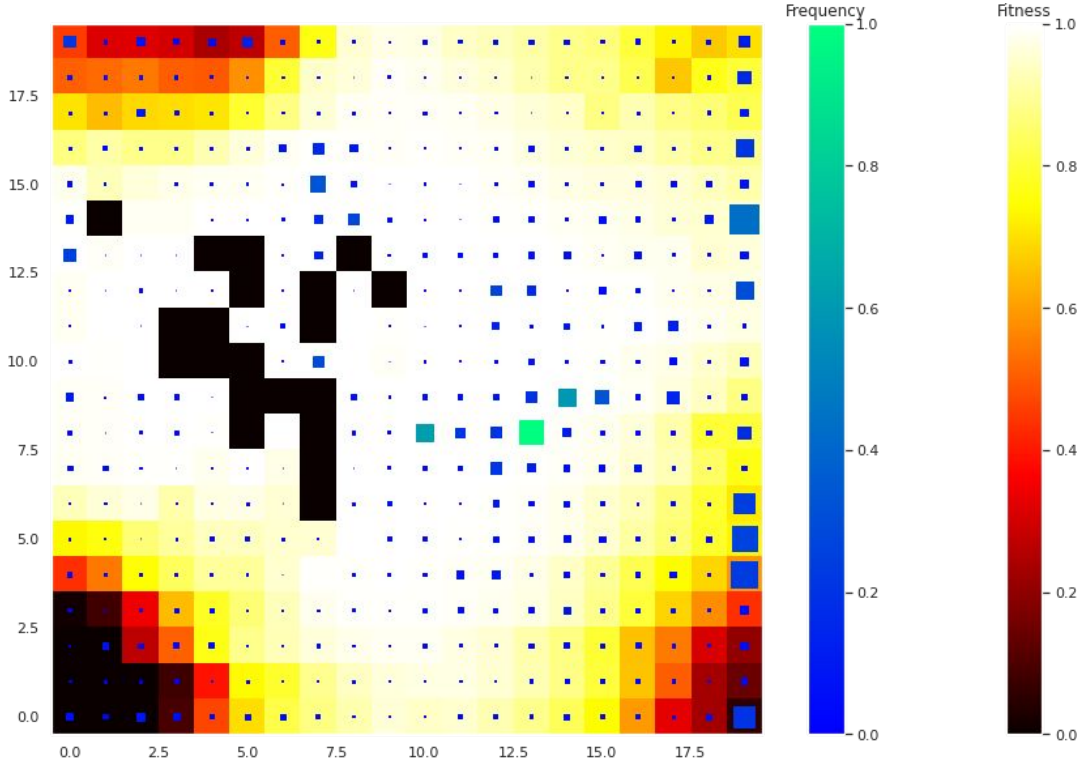
Visualise the search process with SOM



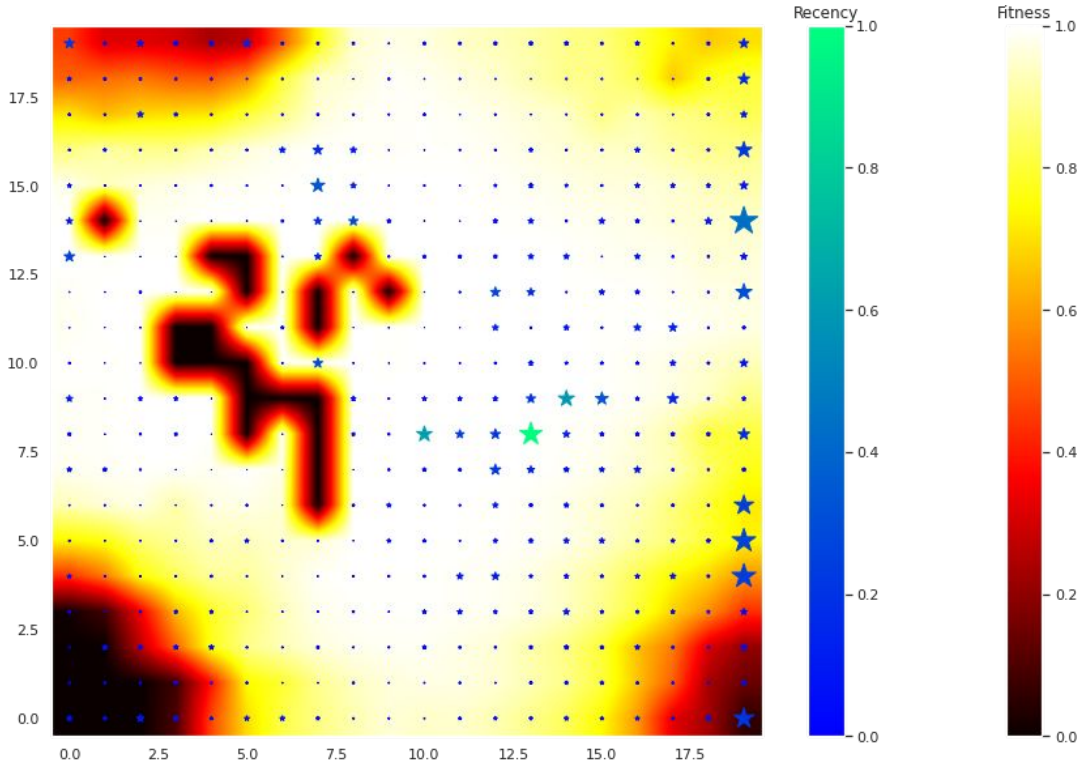
Visualisation layers



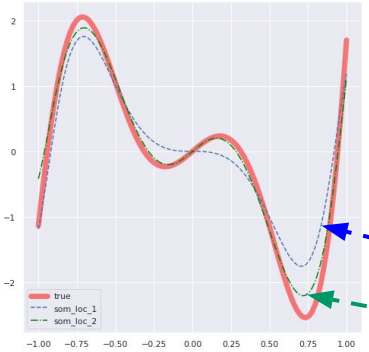
Visualisation with SOM

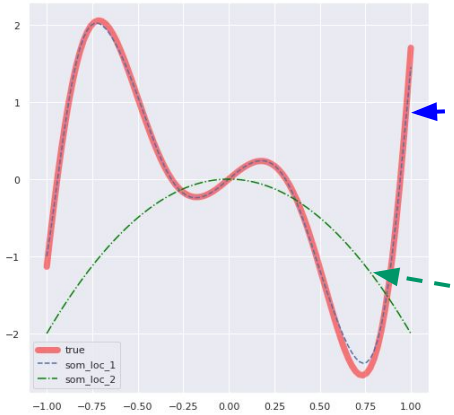


Visualisation with SOM - Final touch



Check if SOM really works



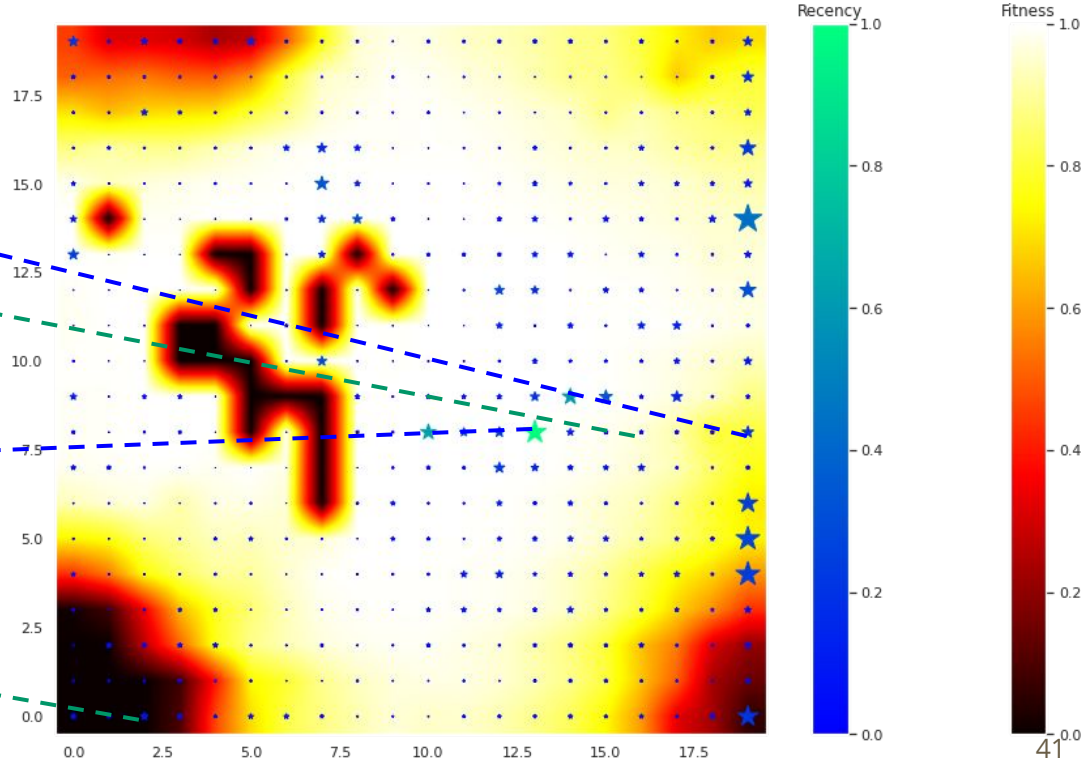


(19,8)

(16,8)

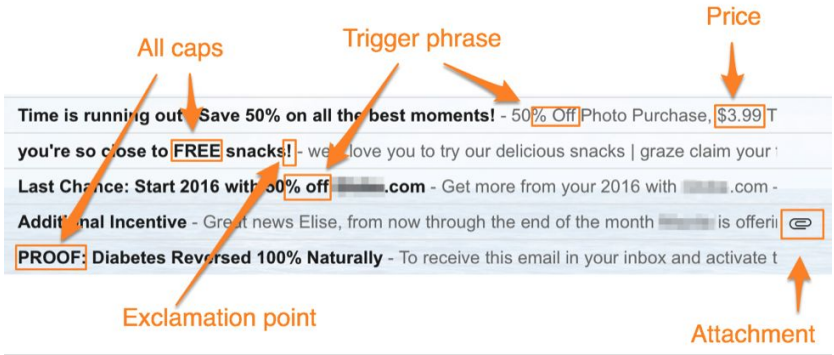
(13,8)

(2,0)



Genetic programming for binary classification

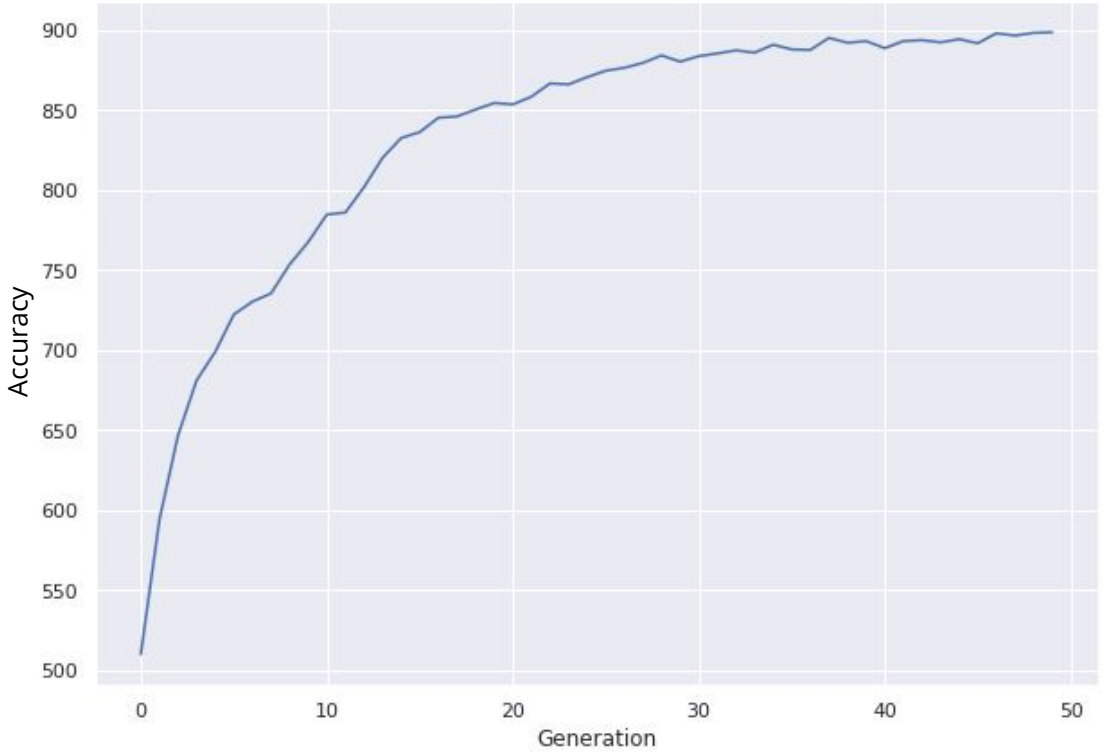
- Apply GP to evolve a symbolic classifier for spambase dataset
- Different from symbolic regression, the outputs of the evolved classifiers are boolean, i.e. class label, instead of float
- Terminal set: textual features (e.g. frequency of words) , True, False
- Function set: and, or, not, if_then_else, less_than, equal_to, +, -, *, protected division, sin, cos
- Fitness function: Accuracy



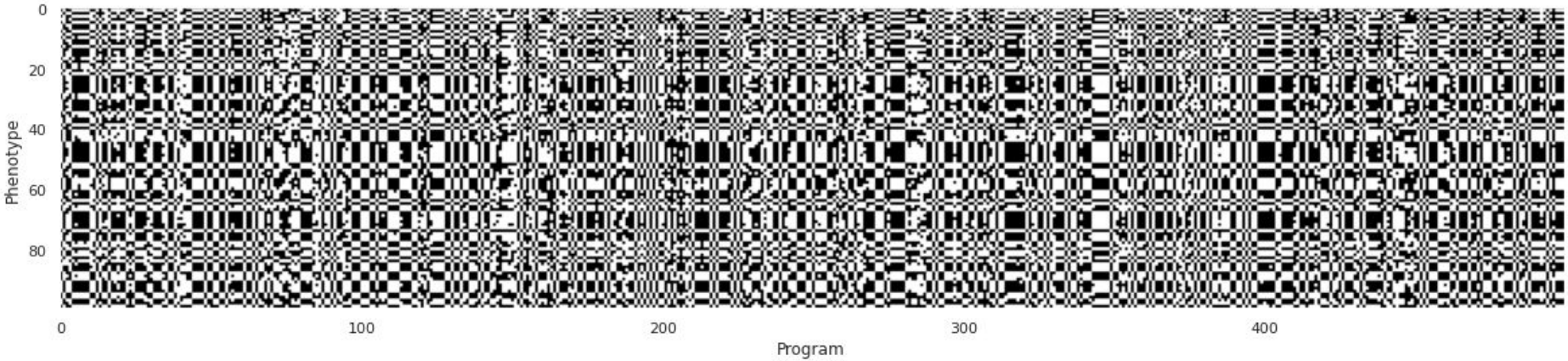
Parameter settings

- Population size = 1000
- Number of generation = 100
- Mutation prob. = 0.1
- Crossover prob. = 0.9
- Tournament selection (size = 5)
- Phenotype vector (dim=100):
  - Prediction == Label: 1 if the prediction is correct; 0 otherwise
  - Data subset used for phenotype calculation is pre-determined
  - This phenotype shows how the evolved classifiers make decisions in different situations

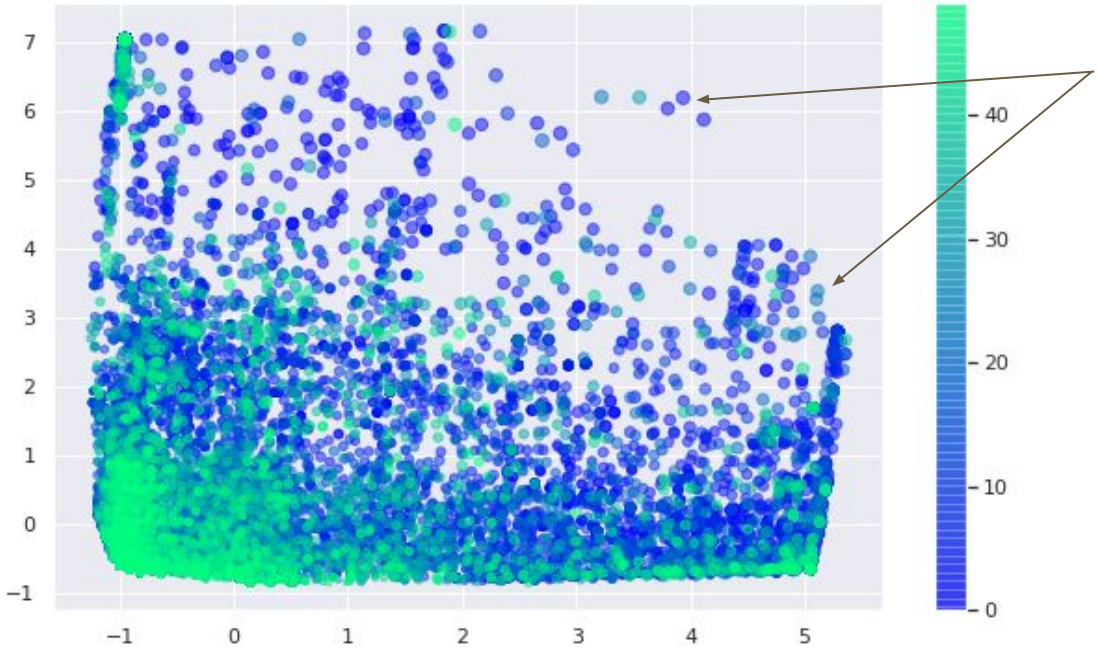
GP progress


900
850
800
750
700
650
600
550
500
Accuracy
0
10
20
30
40
50
Generation

Recorded phenotypes



Visualise the search process with PCA

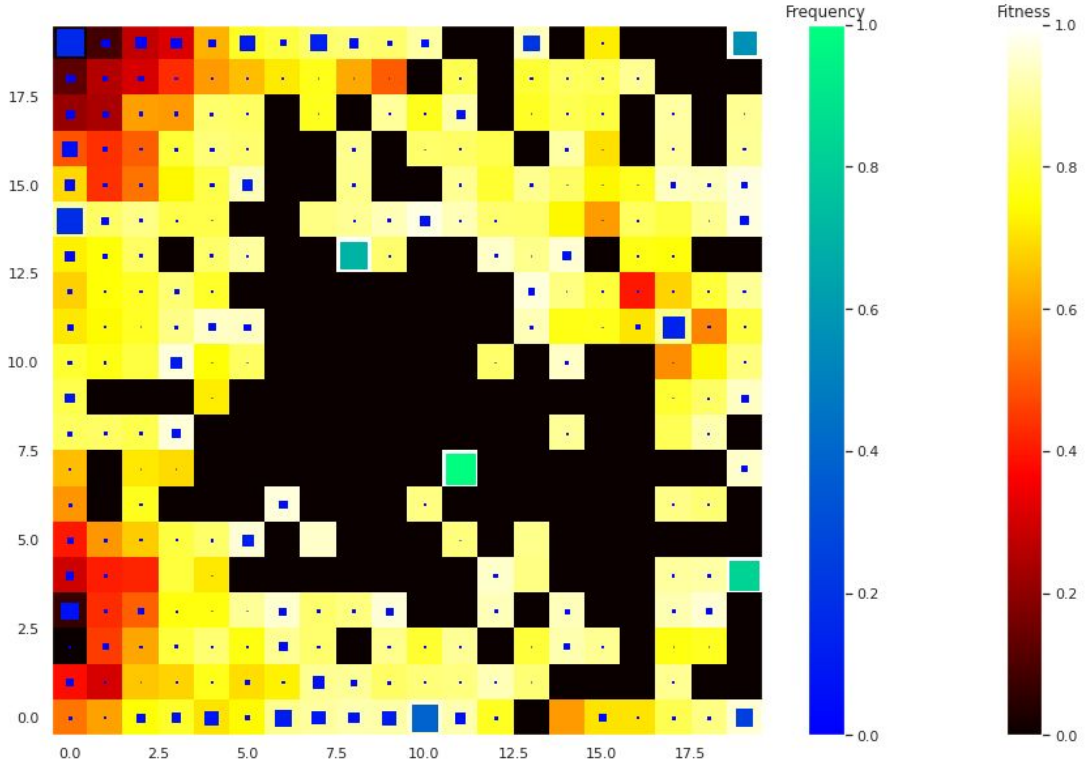


Each point represent a specific behaviour

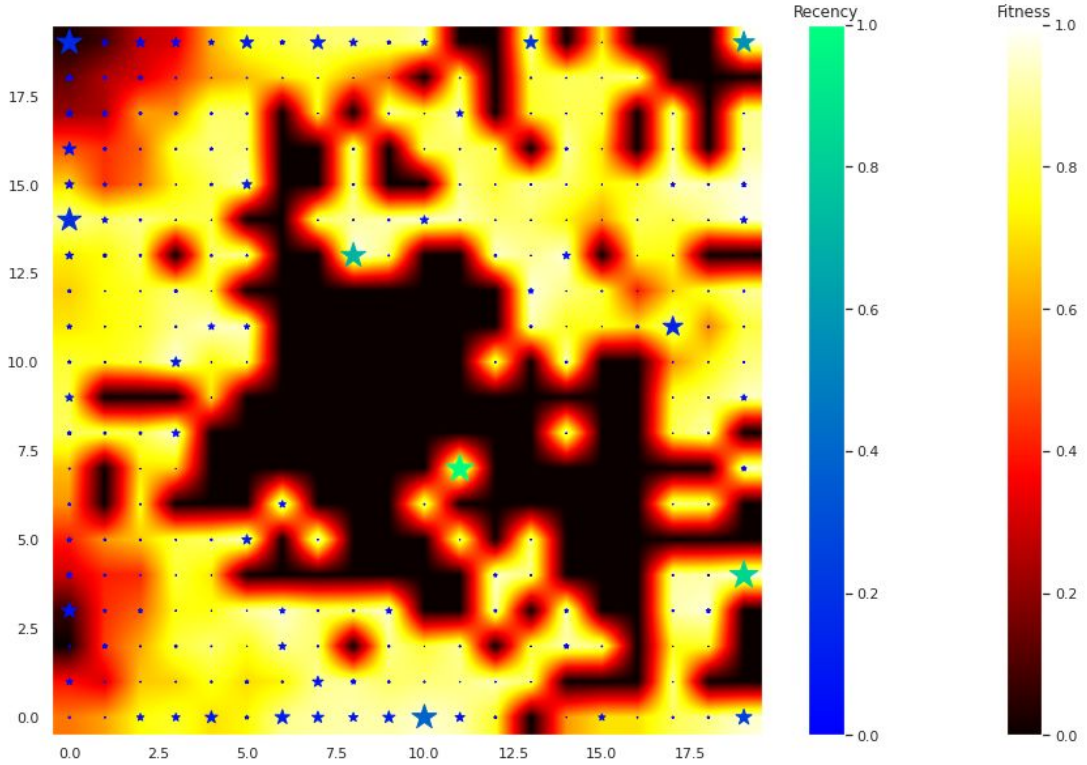
Programs with different genotypes can have similar or the same phenotypes

Principal Component Analysis

Visualisation with SOM



Visualisation with SOM - Final touch



People-centric Evolutionary System

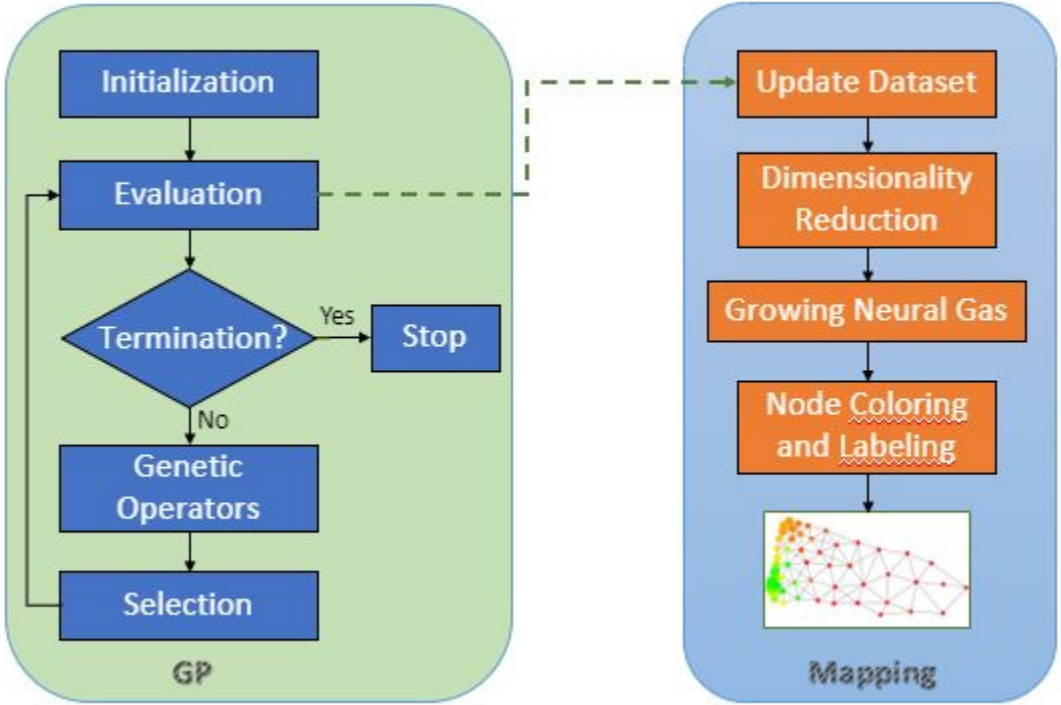
Dimension reduction

Topological learning

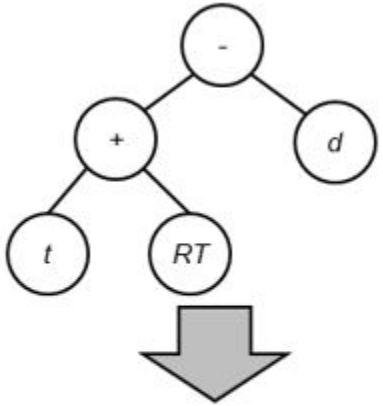
Users’ preferences

Case study: Dynamic Production Scheduling

Visualising Program Evolution



Representations of Scheduling Rules



Priority = - [*d* - (*t* + *RT*)]
→ Priority = - [**due date** –(**current time** + **remaining processing time**)]

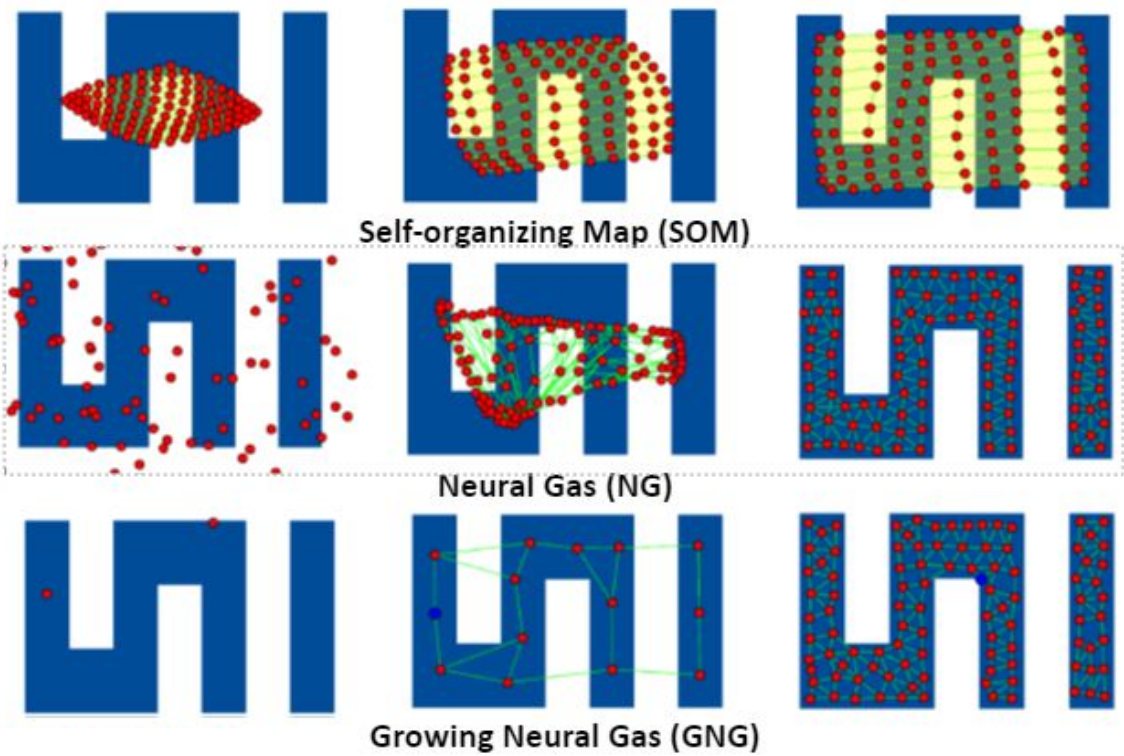
Phenotypes

DECISION SITUTATION (DISPATCHING RULES)	ATTRIBUTE SET			REFERENCE RULE W/PT	GENERATED DISPATCHING RULE #1 W*W/PT		GENERATED DISPATCHING RULE #2 PT/W	
	W	...	PT	RANKING	RANKING	DECISION VECTOR	RANKING	DECISION VECTOR
1	2	...	10	3	2	1	1	3
1	1	...	4	2	3		2	
1	4	...	15	1	1		3	
2	2	...	6	1	1	1	2	2
2	1	...	7	2	2		1	
...	...	...	...	...	...	...	...	...
D1	4	...	9	2	1	2	3	4
D1	1	...	4	4	4		1	
D1	2	...	5	3	2		2	
D1	1	...	2	1	3		4	

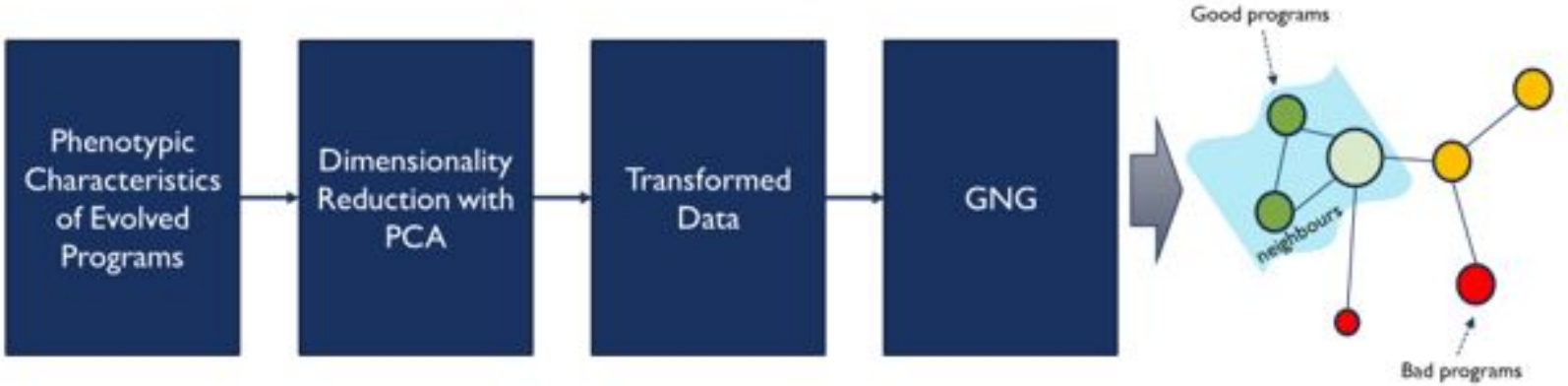
DECISION SITUTATION (ROUTING RULES)	ATTRIBUTE SET			REFERENCE RULE -WLN	GENERATED ROUTING RULE #1 -WLN/NOP		GENERATED ROUTING RULE #2 2*WLN	
	WLN	...	NOP	RANKING	RANKING	DECISION VECTOR	RANKING	DECISION VECTOR
1	50	...	3	2	1	2	1	2
1	20	...	1	1	2		2	
2	100	...	4	3	3	1	1	3
2	10	...	2	1	1		3	
2	20	...	1	2	2		2	
...	...	...	...	...	...	...	...	...
D2	40	...	2	2	2	3	2	3
D2	30	...	1	1	3		3	
D2	45	...	3	3	1		1	

*W is the job weight, PT is the job processing time.
**WLN is the workload, NOP is the number of jobs waiting at a machine.

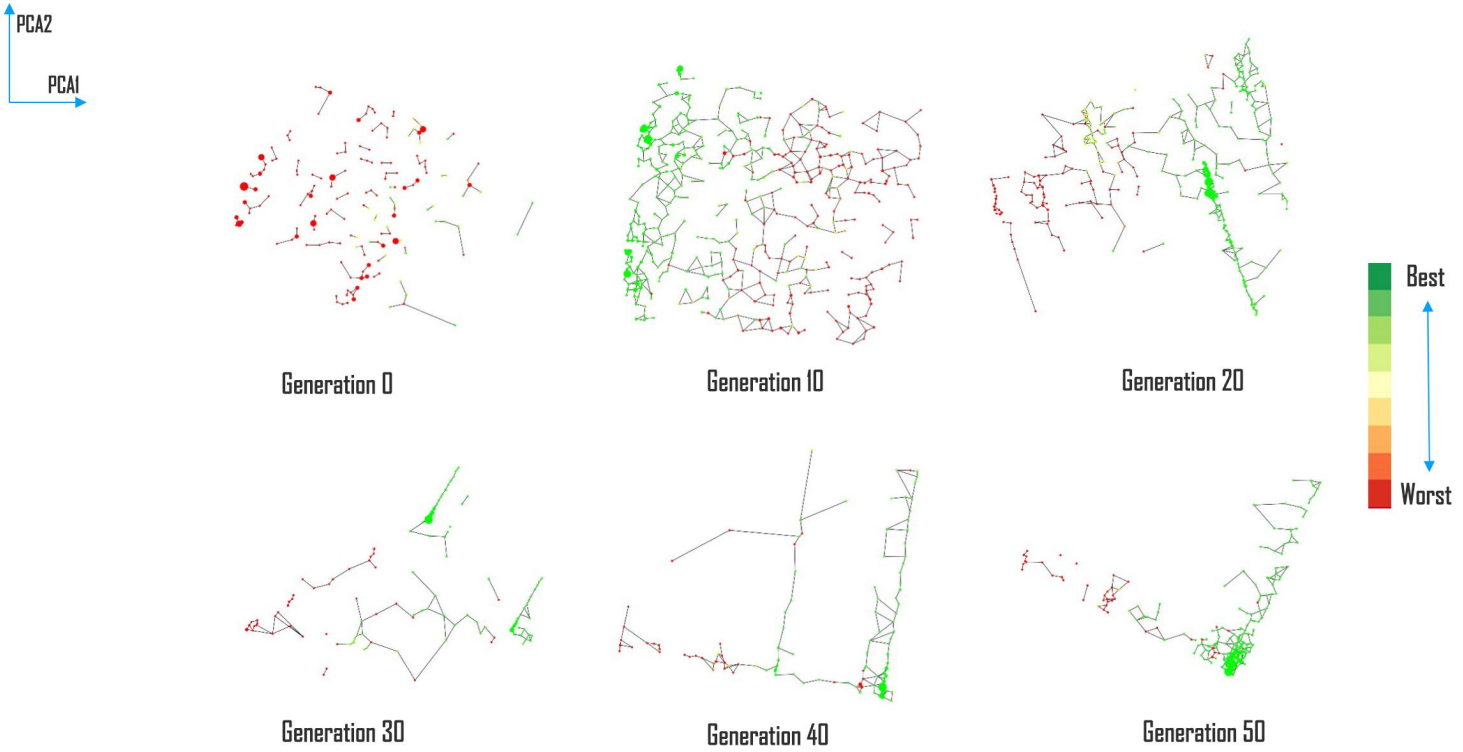
Growing neural gas (GNG)



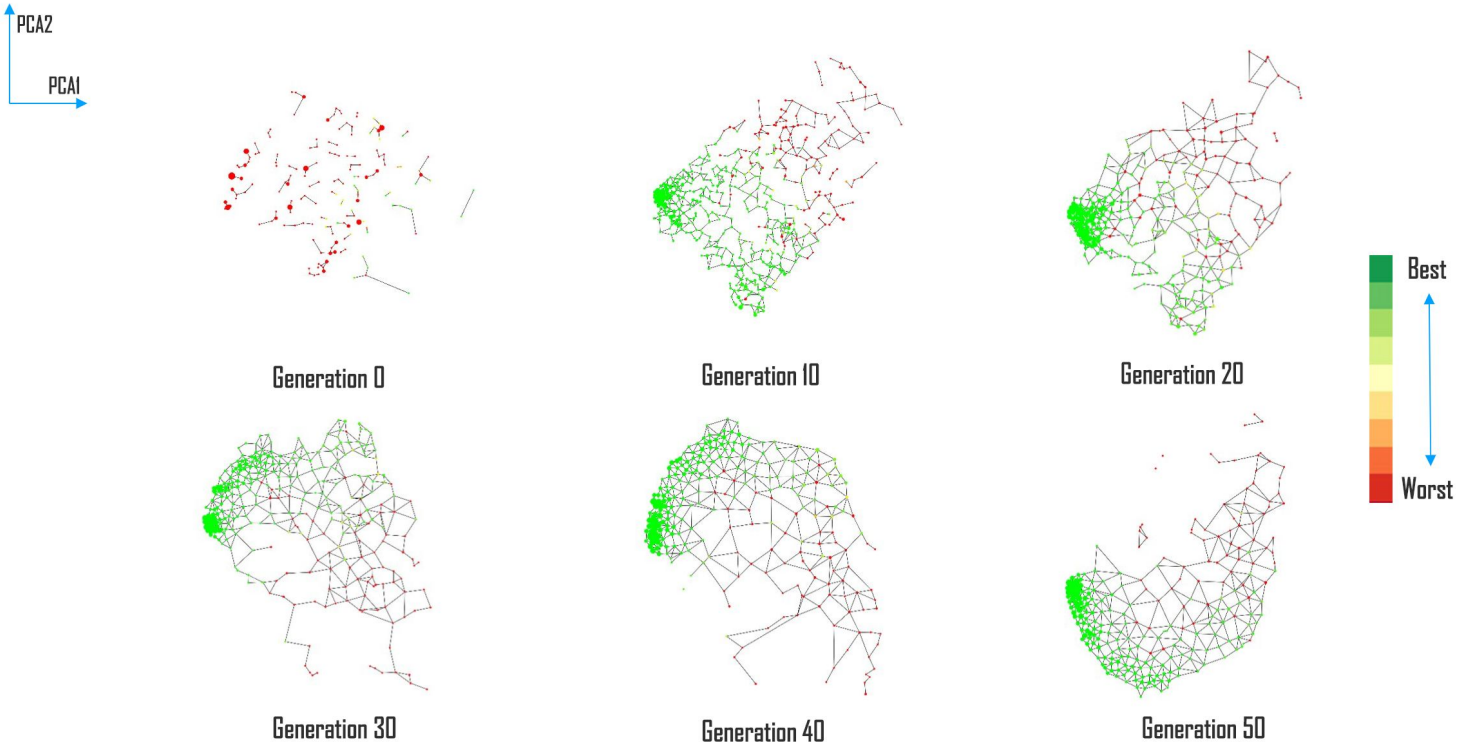
Topological relations of evolved scheduling heuristics



Visualising and adapting evolutionary process – Simple GP



Visualising and adapting evolutionary process – Surrogate GP



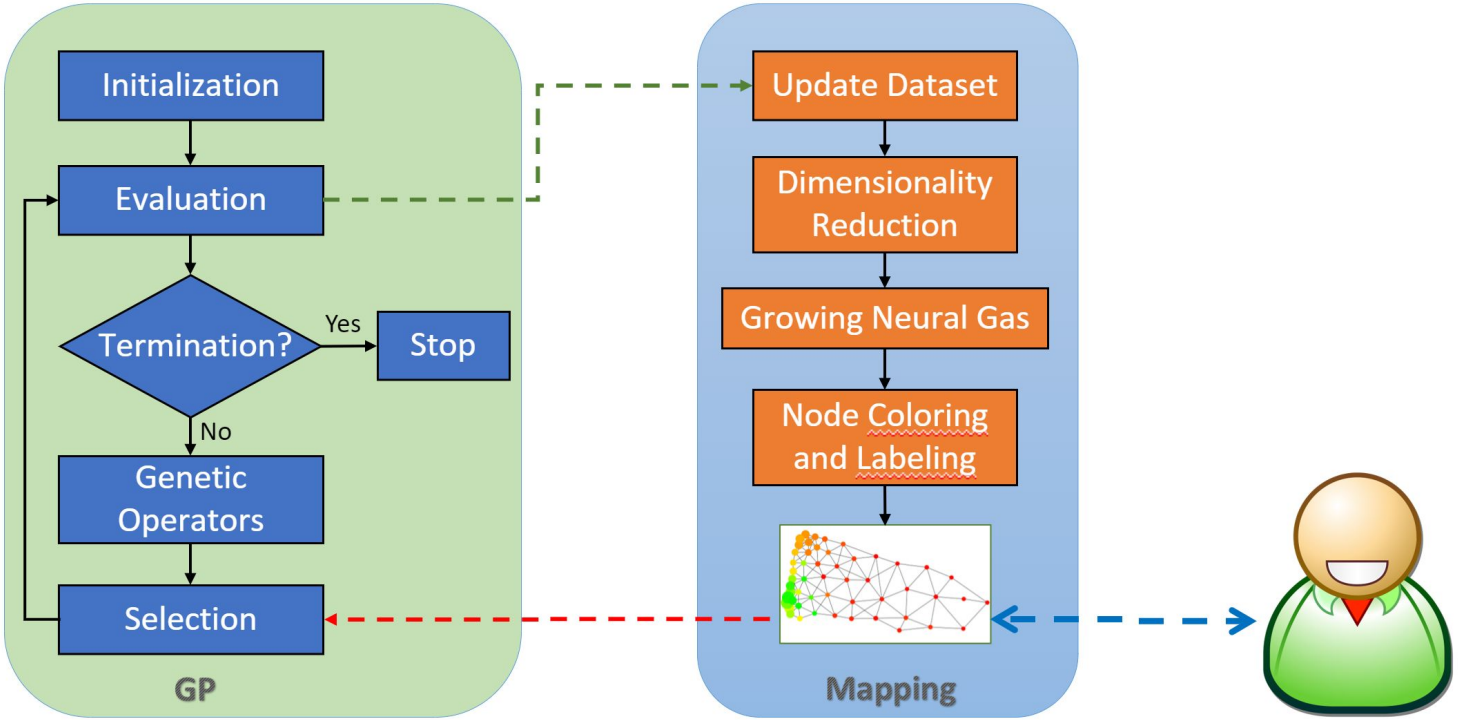
Common Observed Patterns

Pattern	Behaviour	Algorithm					
		TGP-C	TGP-M	TGP-L	TGP-H	SGP	MGP
Large number of components and nodes with different colours	initialisation, random search	○	○	○	○	○	○
Large nodes with different colours scattered randomly over the network	uninformed or ill-informed exploration	○◐	○◐⊙	○◐⊙	○	○	○
Network rotation	genetic drift	○◐	○◐⊙◑	○◐⊙◑	○◐	○◐⊙	○◐⊙◑
Large nodes in a narrow "greener" area of the network and only a few "red" nodes	exploitation, convergence	⊙◑	⊙◑	◑	◐⊙◑	◐⊙◑	◐⊙◑
Large "green" nodes in a wide area of the network	exploitation, exploration	⊙			◐	◐⊙◑	◐⊙◑
Large nodes forming multiple "green" clusters in the network	exploitation, exploration						◐⊙

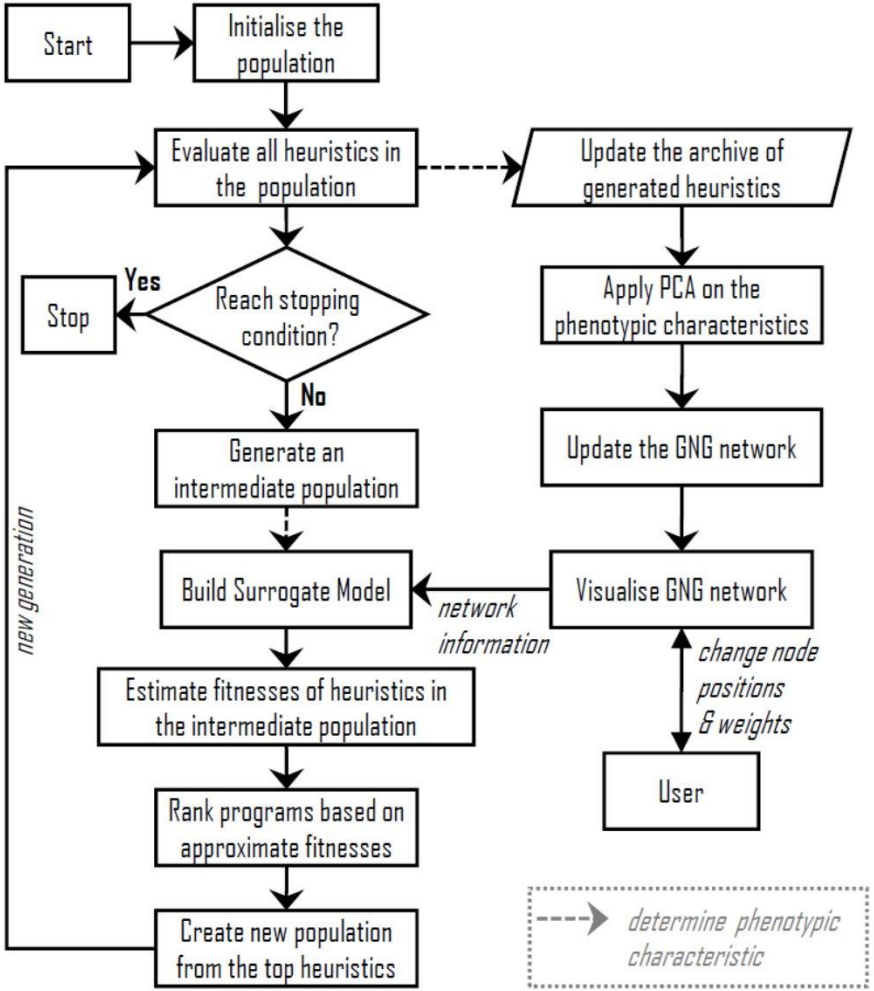
○: first generations; ◐: early generations; ⊙: mid-generations; ◑: later generations

Visualizing the Evolution of Computer Programs for Genetic Programming [Research Frontier]
S Nguyen, M Zhang, D Alahakoon, KC Tan
IEEE Computational Intelligence Magazine 13 (4), 77-94

People-centric Evolutionary System (PES)



PES Flowchart



Su Nguyen, Mengjie Zhang, Damminda Alahakoon, Kay Chen Tan, “People-Centric Evolutionary System for Dynamic Production Scheduling”, IEEE Trans. On Cybernetics, 2019.

Adaptive Surrogate Models

- Existing surrogate model:

$$\hat{f}(\mathcal{H}) = fitness\left(\arg\min_{\mathcal{H}' \in \mathcal{A}} \|phenotype(\mathcal{H}) - phenotype(\mathcal{H}')\|\right)$$

Archive of evolved heuristics

- Adaptive surrogate model:

$$\hat{f}_a(\mathcal{H}) = \hat{f}(\mathcal{H}) \times control\_factor(\mathcal{H}, \mathcal{N})$$

in which:

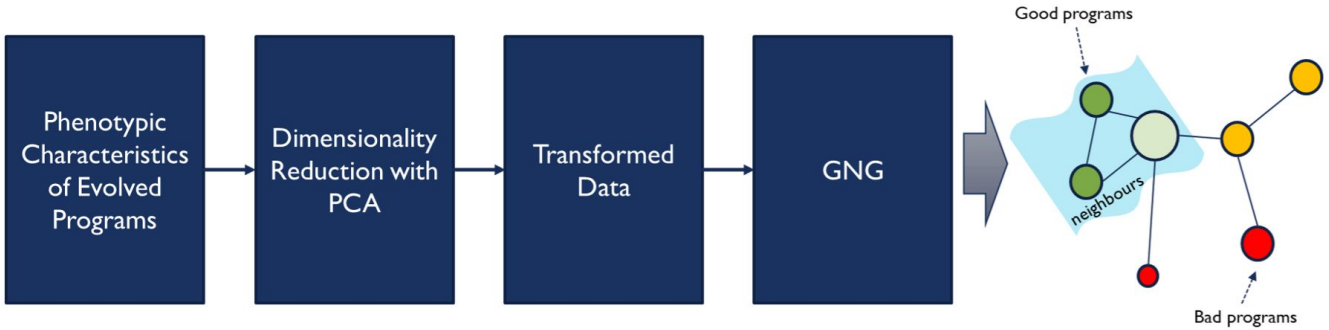
$$control\_factor(\mathcal{H}, \mathcal{N}) = df(\mathcal{H}, \mathcal{N}) \times \frac{bf(\mathcal{H}, \mathcal{N})}{pf(\mathcal{H}, \mathcal{N})}$$

bloat factor

Diversity factor

user-preference factor

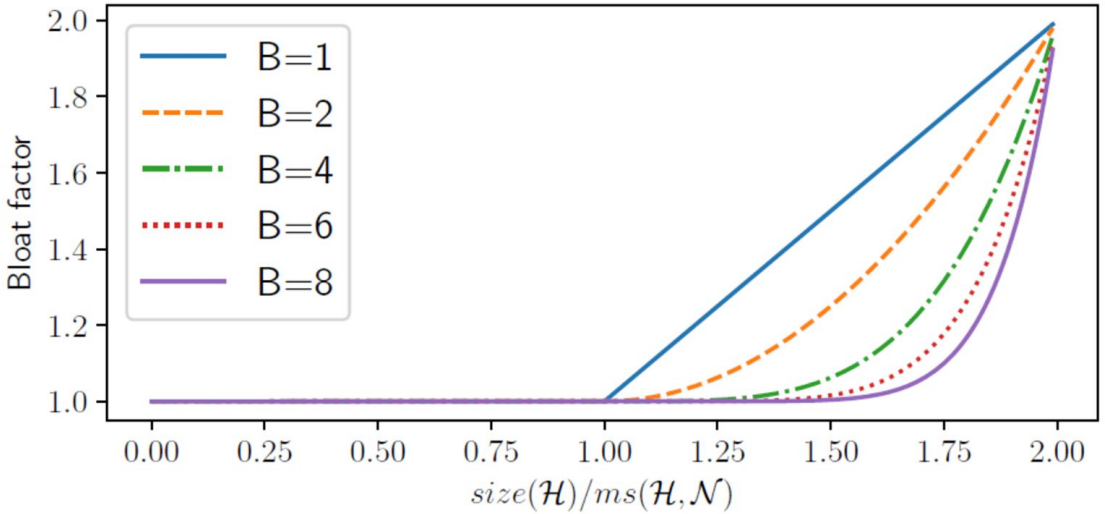
Diversity Factor

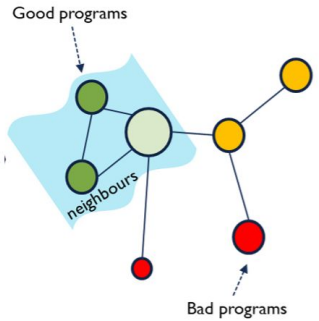


$$df(\mathcal{H},\mathcal{N}) = \begin{cases} \frac{nd(\mathcal{H},\mathcal{N})}{T} & nd(\mathcal{H},\mathcal{N}) > T \\ 1 & otherwise \end{cases}$$

$$nd(\mathcal{H},\mathcal{N}) = \frac{1}{|\mathcal{A}|}\left(freq(n_{\mathcal{H}}^{\mathcal{N}*}) + \sum_{n \in NB(n_{\mathcal{H}}^{\mathcal{N}*})} freq(n) \right)$$

Bloat Factor





$$bf(\mathcal{H},\mathcal{N}) = \begin{cases} 1 + \left(\frac{size(\mathcal{H})}{ms(\mathcal{H},\mathcal{N})} - 1\right)^B & \frac{size(\mathcal{H})}{ms(\mathcal{H},\mathcal{N})} > 1 \\ 1 & otherwise \end{cases}$$

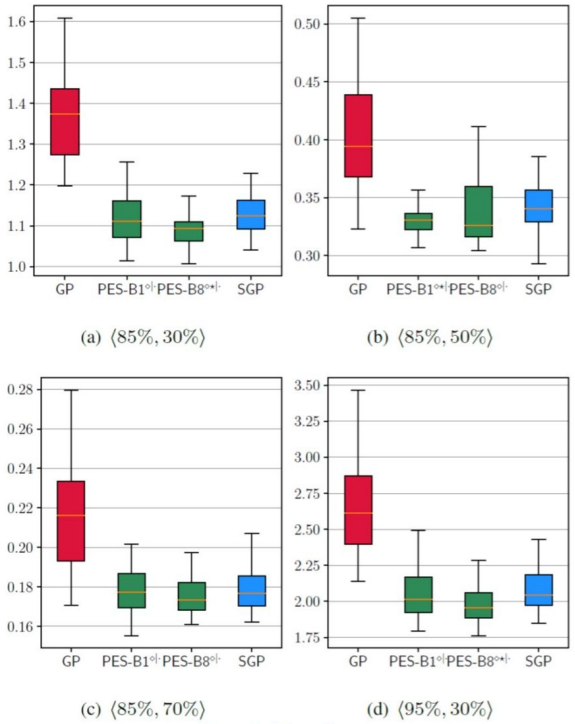
Maximum size of heuristics in neighbourhood

User Preference

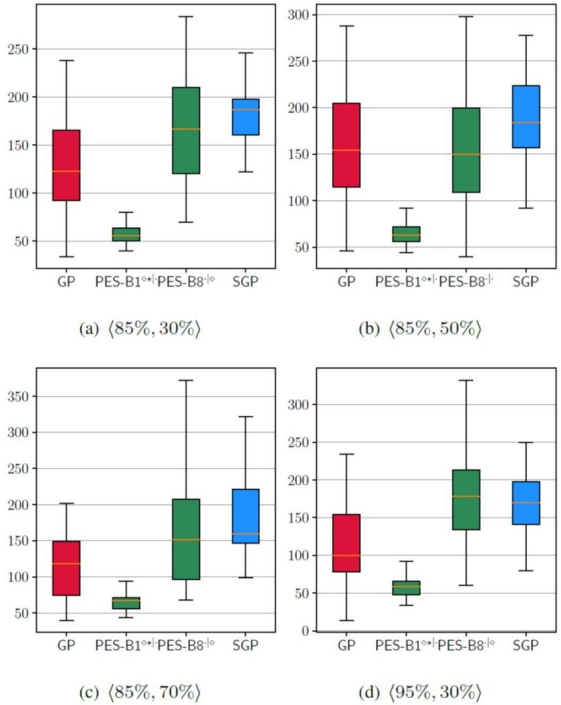
- User can interact with GNG network
  - Select nodes
  - Move nodes
  - Change preference values

$$pf(\mathcal{H},\mathcal{N}) = 1 + \frac{pref(n_{\mathcal{H}}^{\mathcal{N}*}) - 1}{maxp - 1}$$

Results

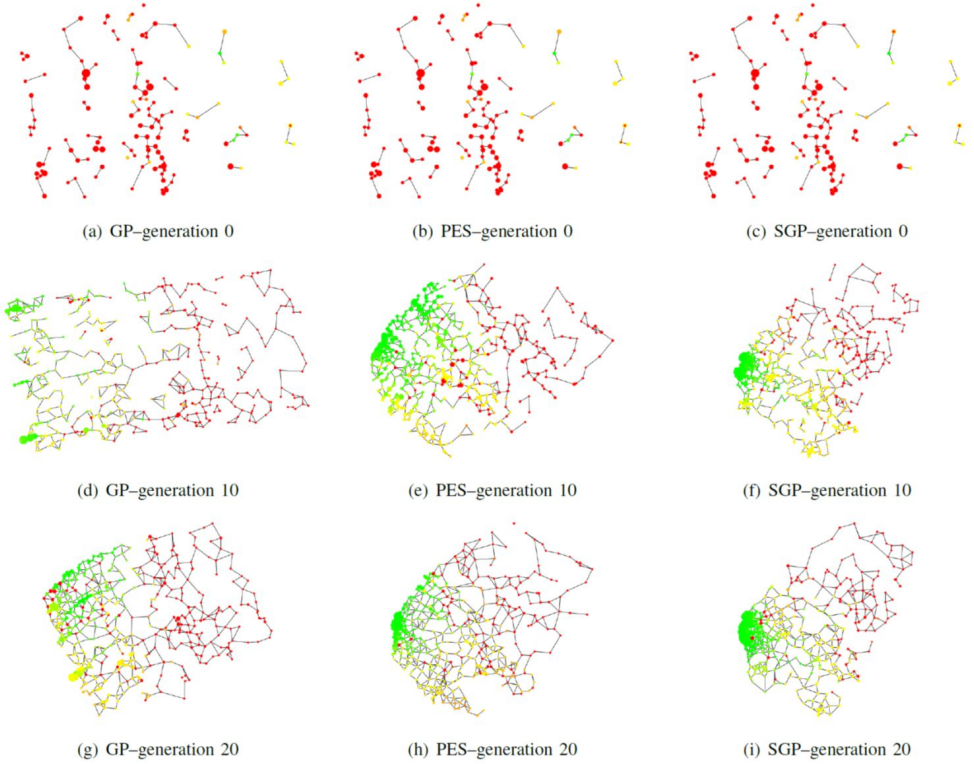


Test Performance



Program Sizes

Visualisation (1)

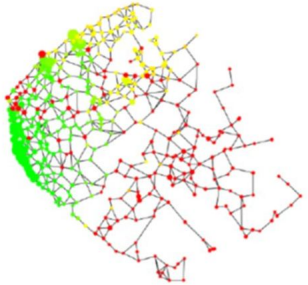


(a) GP–generation 0 (b) PES–generation 0 (c) SGP–generation 0

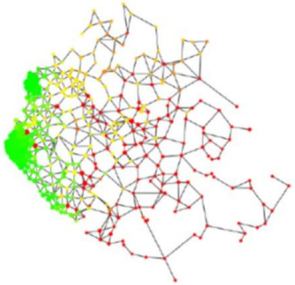
(d) GP–generation 10 (e) PES–generation 10 (f) SGP–generation 10

(g) GP–generation 20 (h) PES–generation 20 (i) SGP–generation 20

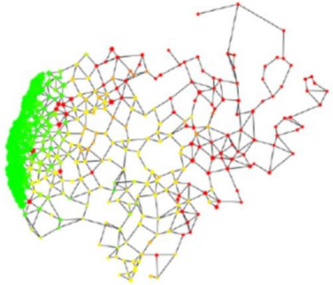
Visualisation (2)



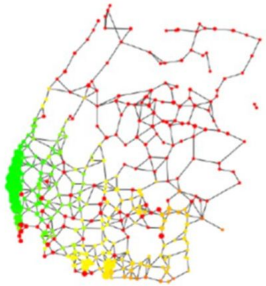
(j) GP–generation 30



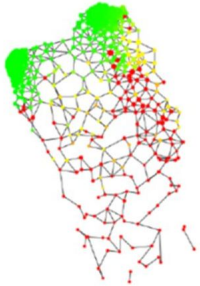
(k) PES–generation 30



(l) SGP–generation 30



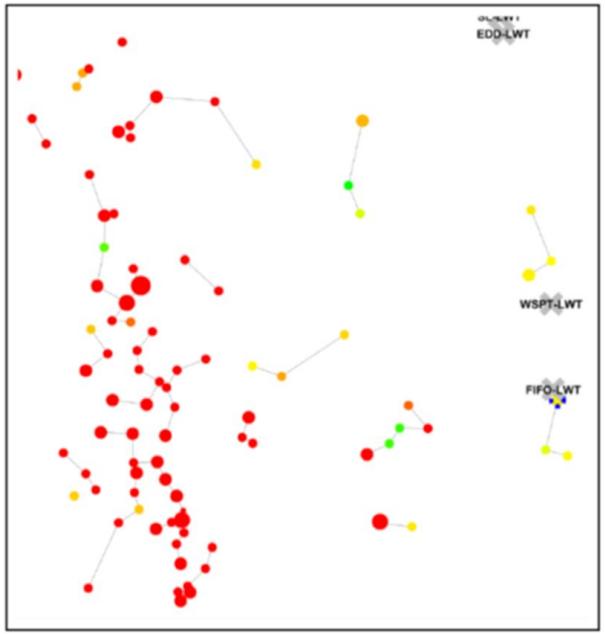
(m) GP–generation 40



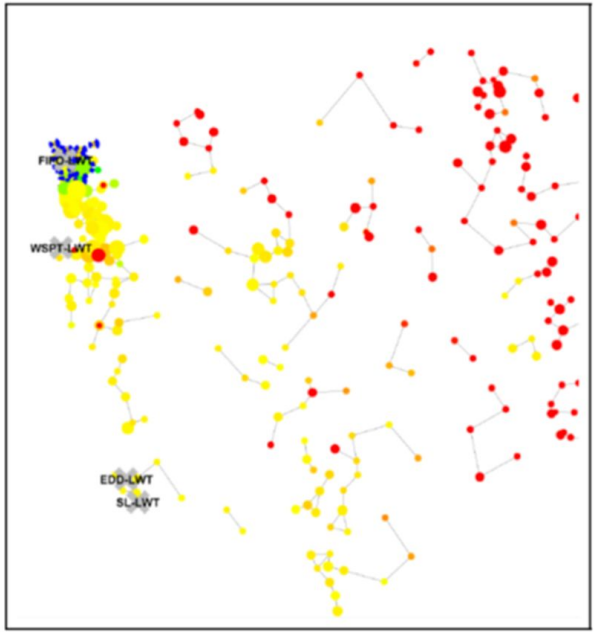
(n) PES–generation 40

(o) SGP–generation 40

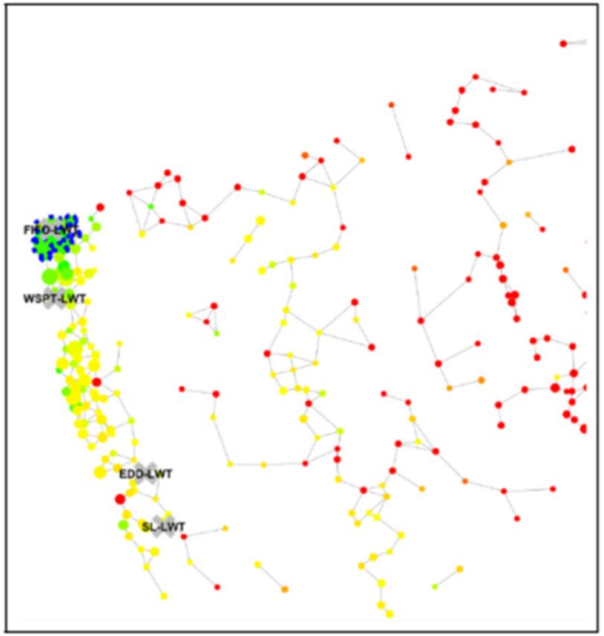
User Preferences



(a) Generation 0:
$\mathcal{P}_d = \max(rRJ, W)$
$\mathcal{P}_r = ((rPT - NON) - (NON - NON))$



(b) Generation 3:
$\mathcal{P}_d = W$
$\mathcal{P}_r = (rPT - WLN)$



(c) Generation 6:
$\mathcal{P}_d = (W * div(W, PR))$
$\mathcal{P}_r = (rPT - WLN)$

What Next?

- Improve mapping techniques
  - Growing SOM (no need to pre-determine the map size)
  - Grammar Variational Autoencoder -- to cope with genotype
  - Etc.
- Automatically detect and visualise evolution paths
- Enhance the robustness of SOM and related techniques in the incremental learning setting
- Examine supervised or (semi) supervised mapping
- Improve interactivity

My Email:
p.nguyen4@Latrobe.edu.au