Towards Better Explainable AI Through Genetic Programming

Dr. Yi Mei
yi.mei@ecs.vuw.ac.nz
Evolutionary Computation Research Group
Victoria University of Wellington
Wellington, New Zealand

2021 IEEE Congress on Evolutionary Computation Tutorial

2021 IEEE
CONGRESS ON EVOLUTIONARY COMPUTATION
28.06–1.07.2021 • Kraków • POLAND

Outline

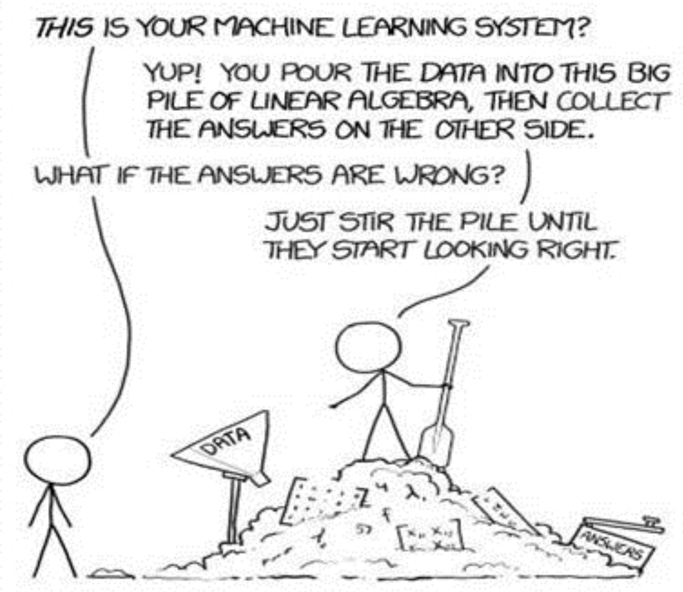
- Introduction to Explainable AI (XAI)
- Introduction to Genetic Programming (GP)
- Better Interpretability Through GP
- Challenges and Future Directions

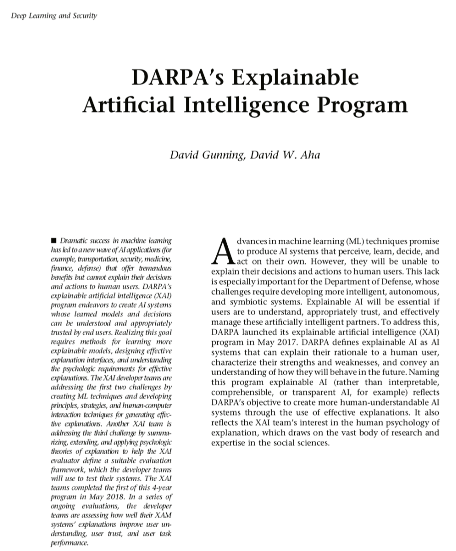
Outline

- **Introduction to Explainable AI (XAI)**
- Introduction to Genetic Programming (GP)
- Better Interpretability Through GP
- Challenges and Future Directions

Why Interpretability in AI

- Early logical and symbolic AI systems
  - Expert systems, manually design the logic and rules
  - Easy to understand and explain
  - Not effective enough, brittle against real-world complex problems
- Recent AI successes
  - Machine learning, deep learning, automatically learn relationships
  - High performance, but too complex and opaque





Deep Learning and Security

DARPA's Explainable Artificial Intelligence Program

David Gunning, David W. Aha

■ *Dramatic success in machine learning has led to a new wave of AI applications (for example, transportation, security, medicine, finance, defense) that offer tremendous benefits but cannot explain their decisions and actions to human users. DARPA's explainable artificial intelligence (XAI) program endeavors to create AI systems whose learned models and decisions can be understood and appropriately trusted by end users. Realizing this goal requires methods for learning more explainable models, designing effective explanation interfaces, and understanding the psychologic requirements for effective explanations. The XAI developer teams are addressing the first two challenges by creating ML techniques and developing principles, strategies, and human-computer interaction techniques for generating effective explanations. Another XAI team is addressing the third challenge by summarizing, extending, and applying psychologic theories of explanation to help the XAI evaluator define a suitable evaluation framework, which the developer teams will use to test their systems. The XAI teams completed the first of this 4-year program in May 2018. In a series of ongoing evaluations, the developer teams are assessing how well their XAM systems' explanations improve user understanding, user trust, and user task performance.*

Advances in machine learning (ML) techniques promise to produce AI systems that perceive, learn, decide, and act on their own. However, they will be unable to explain their decisions and actions to human users. This lack is especially important for the Department of Defense, whose challenges require developing more intelligent, autonomous, and symbiotic systems. Explainable AI will be essential if users are to understand, appropriately trust, and effectively manage these artificially intelligent partners. To address this, DARPA launched its explainable artificial intelligence (XAI) program in May 2017. DARPA defines explainable AI as AI systems that can explain their rationale to a human user, characterize their strengths and weaknesses, and convey an understanding of how they will behave in the future. Naming this program explainable AI (rather than interpretable, comprehensible, or transparent AI, for example) reflects DARPA's objective to create more human-understandable AI systems through the use of effective explanations. It also reflects the XAI team's interest in the human psychology of explanation, which draws on the vast body of research and expertise in the social sciences.

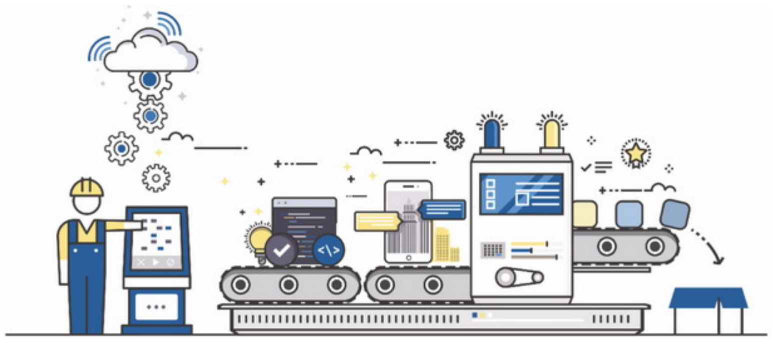
Gunning, D., & Aha, D. (2019). DARPA's explainable artificial intelligence (XAI) program. AI Magazine, 40(2), 44-58.

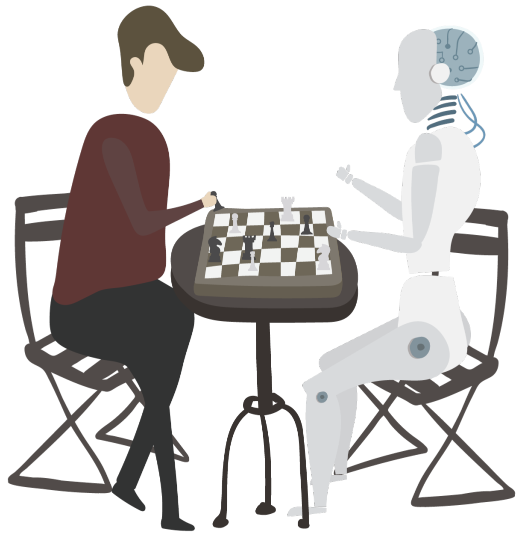
Why Interpretability in AI

- Identify "Clever Hans" Predictors
- Enhance Trust and Confidence from Users
- Provide New Insights (AlphaGo)
- Legislation
    - EU's General Data Protection Regulation (GDPR) requires the ML model to be able to be explained

(a) Husky classified as wolf (b) Explanation

Ribeiro et al. 2016

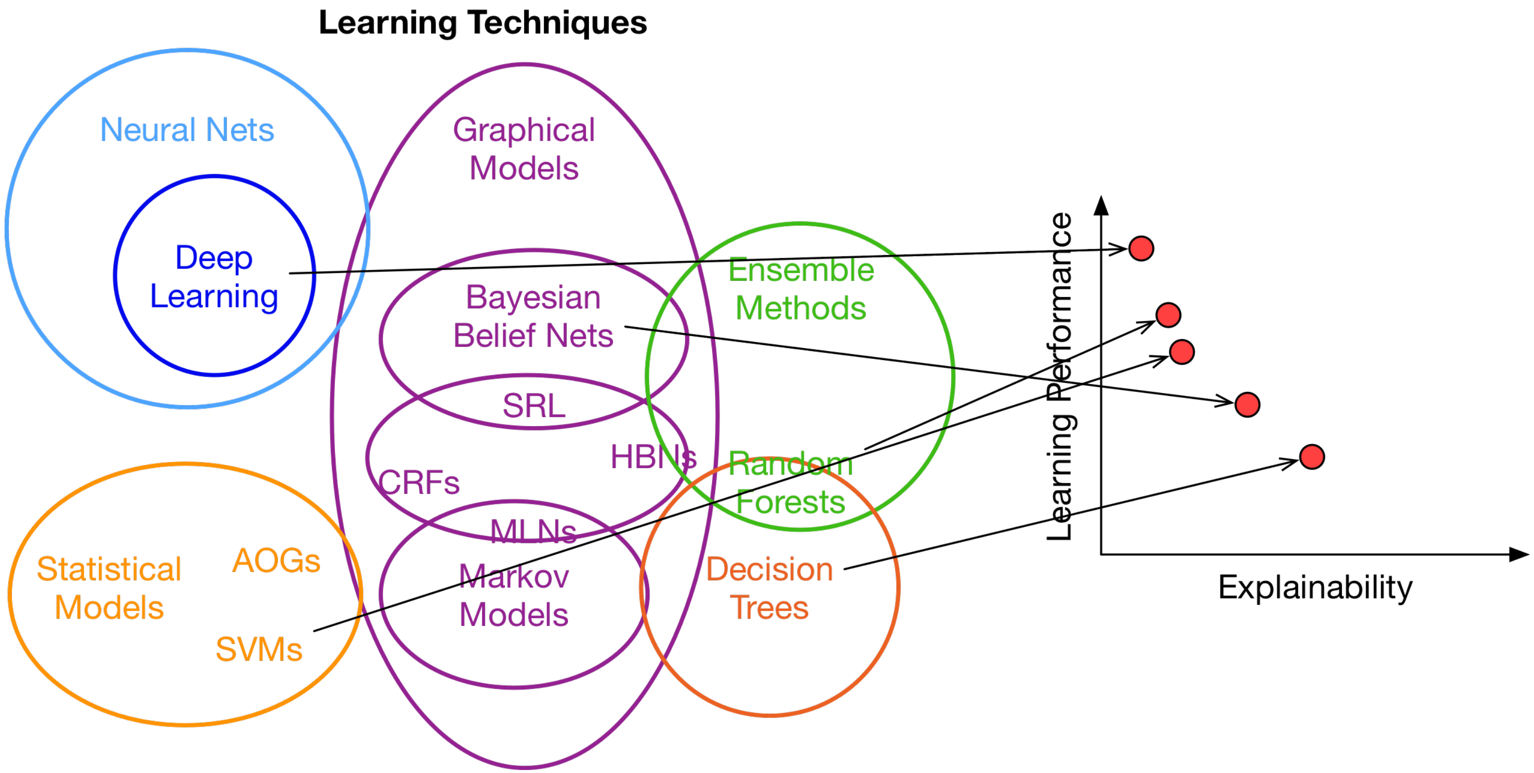




Samek, W., & Müller, K. R. (2019). Towards explainable artificial intelligence. In Explainable AI: interpreting, explaining and visualizing deep learning (pp. 5-22). Springer, Cham.

Learning Performance vs Explainability

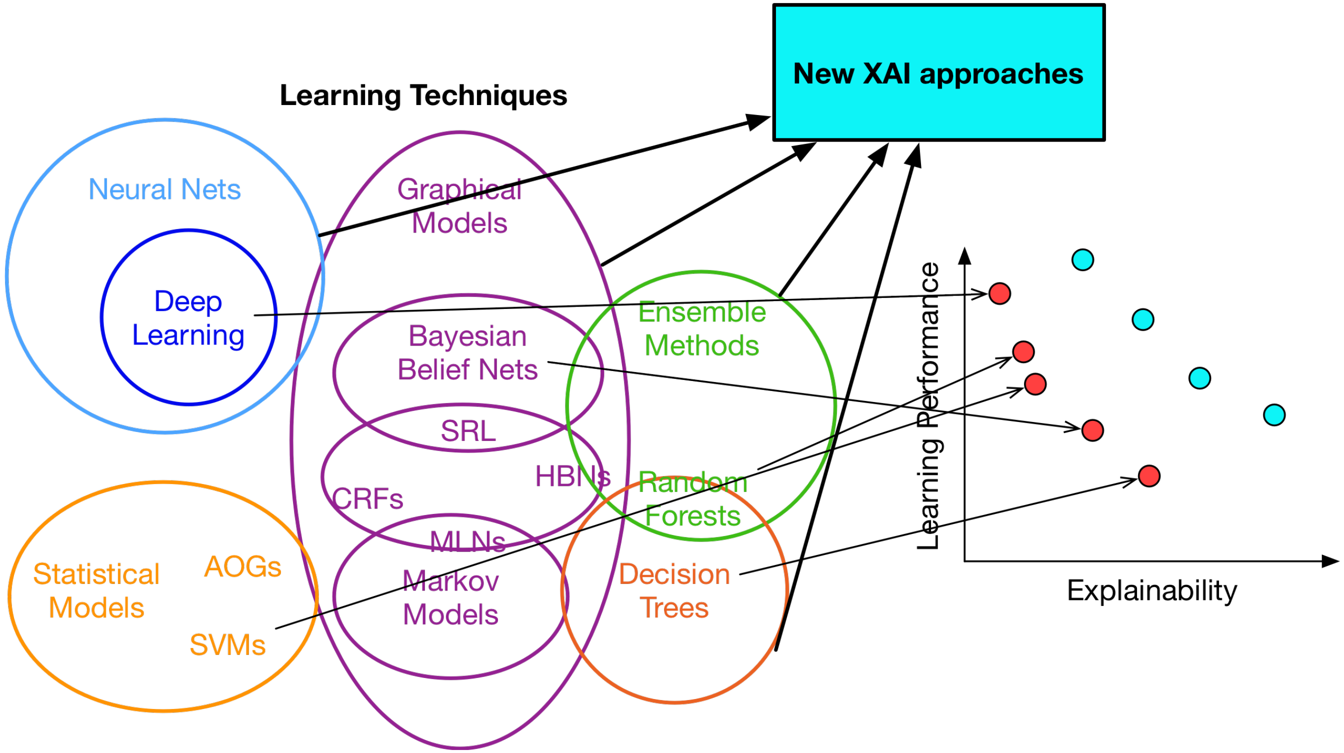
- There is a **trade-off** between performance and explainability
  - **Deep learning:** very good performance, but hard to explain
  - **Decision tree:** relatively easy to explain, but not as effective



Gunning, D., & Aha, D. (2019). DARPA's explainable artificial intelligence (XAI) program. AI Magazine, 40(2), 44-58.

Improve the Trade-Off by XAI

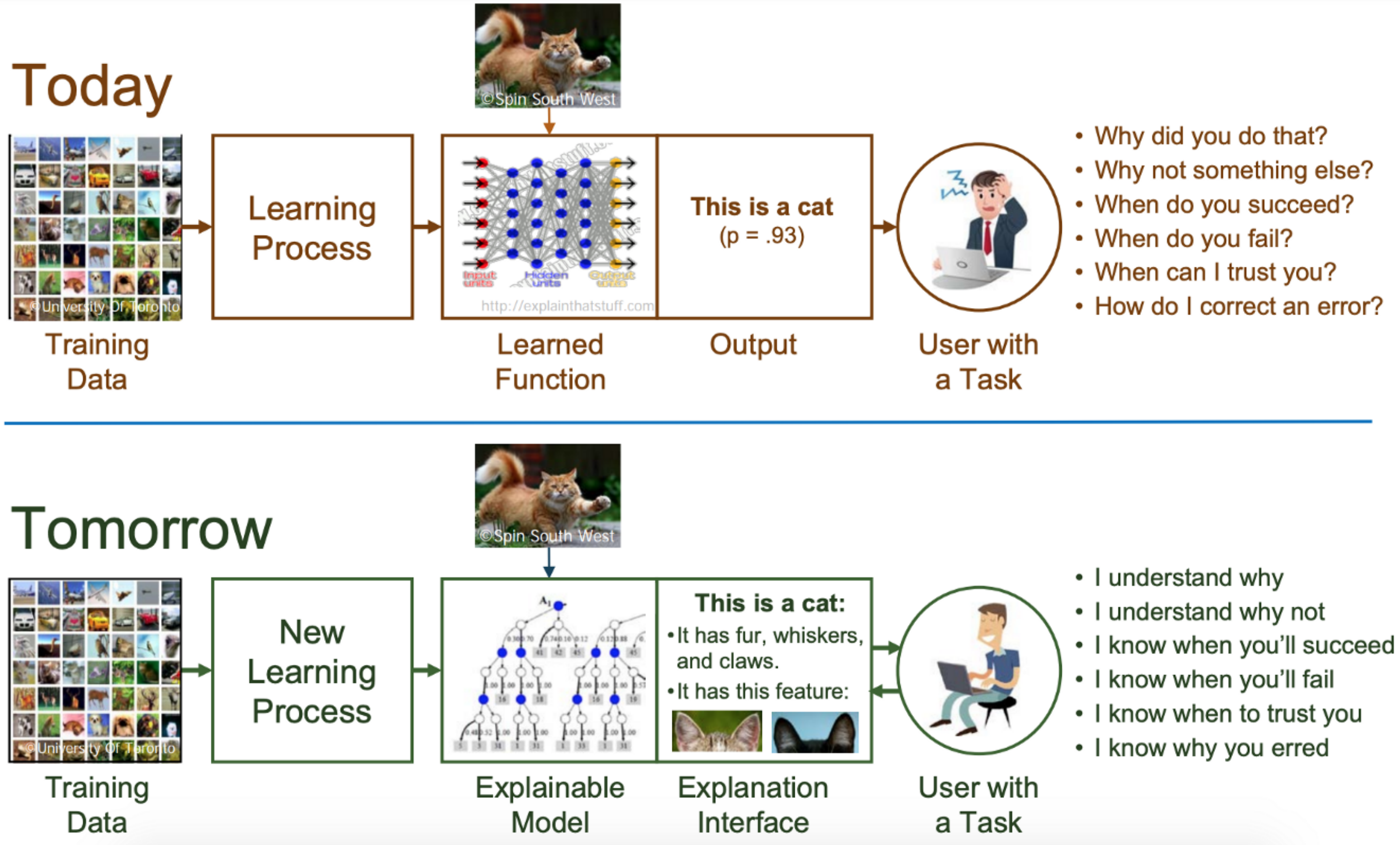
- **Deep explanation**: Modify DL methods to learn more explainable features or representations
- **Interpretable Models**: Techniques to learn more structured, interpretable or causal models
- **Model Induction**: Techniques that infer an approximate explainable model for a complex model, analysing the input-output behaviour of a black box model



Gunning, D., & Aha, D. (2019). DARPA's explainable artificial intelligence (XAI) program. AI Magazine, 40(2), 44-58.

XAI Concept

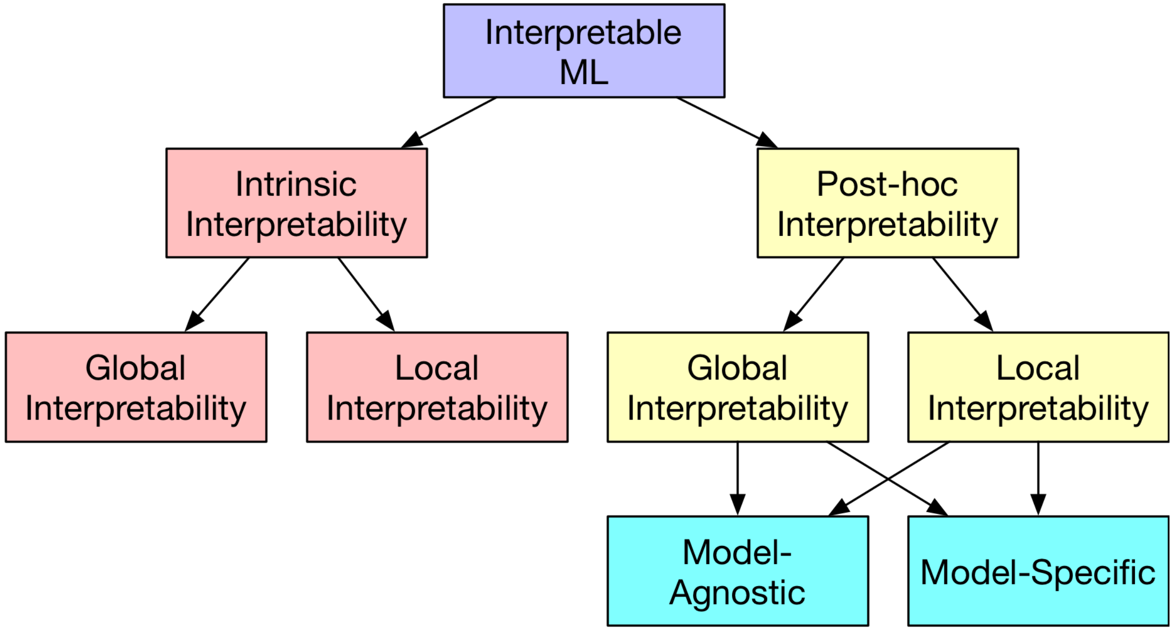
- Current AI vs XAI
  - XAI enables users to understand the system's overall strengths and weaknesses, how it will behave in future, perhaps correct its mistakes



Gunning, D., & Aha, D. (2019). DARPA's explainable artificial intelligence (XAI) program. AI Magazine, 40(2), 44-58.

Techniques for Interpretable ML

- Intrinsic interpretability (Global and Local)
  - Constructing self-explanatory models, such as decision tree, rule-based model, linear model, …
- Post-hoc interpretability (Global and Local)
  - Creating a second (interpretable) model to provide explanations for an existing (black-box) model
- Global interpretability: understand the overall model structure/behaviour
- Local interpretability: understand how/why the model makes one prediction/decision



Du, M., Liu, N., & Hu, X. (2019). Techniques for interpretable machine learning. Communications of the ACM, 63(1), 68-77.

Intrinsic Global Interpretability

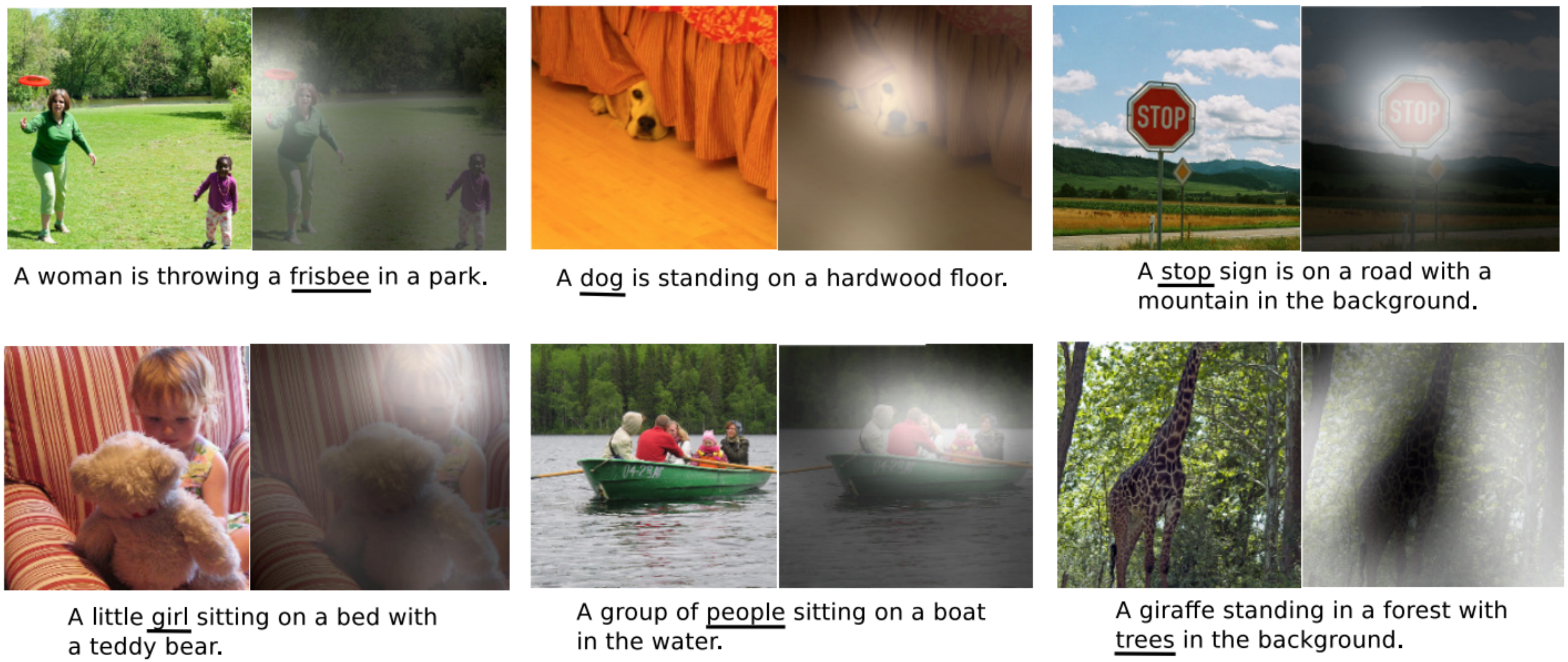
- Train self-explanatory models directly
  - Linear models
  - Rule-based systems
  - Decision trees
  - **Genetic programs (Syntax trees/graphs, …)**

  - Add interpretability constraints
    - Number of features used in the model
    - The used features must have monotonic relations with the prediction
    - Trade-off between accuracy and interpretability
  - Multi-objective training
    - Accuracy and interpretability metrics
    - Number of features used
    - Model complexity
    - …

Du, M., Liu, N., & Hu, X. (2019). Techniques for interpretable machine learning. Communications of the ACM, 63(1), 68-77.

Intrinsic Local Interpretability

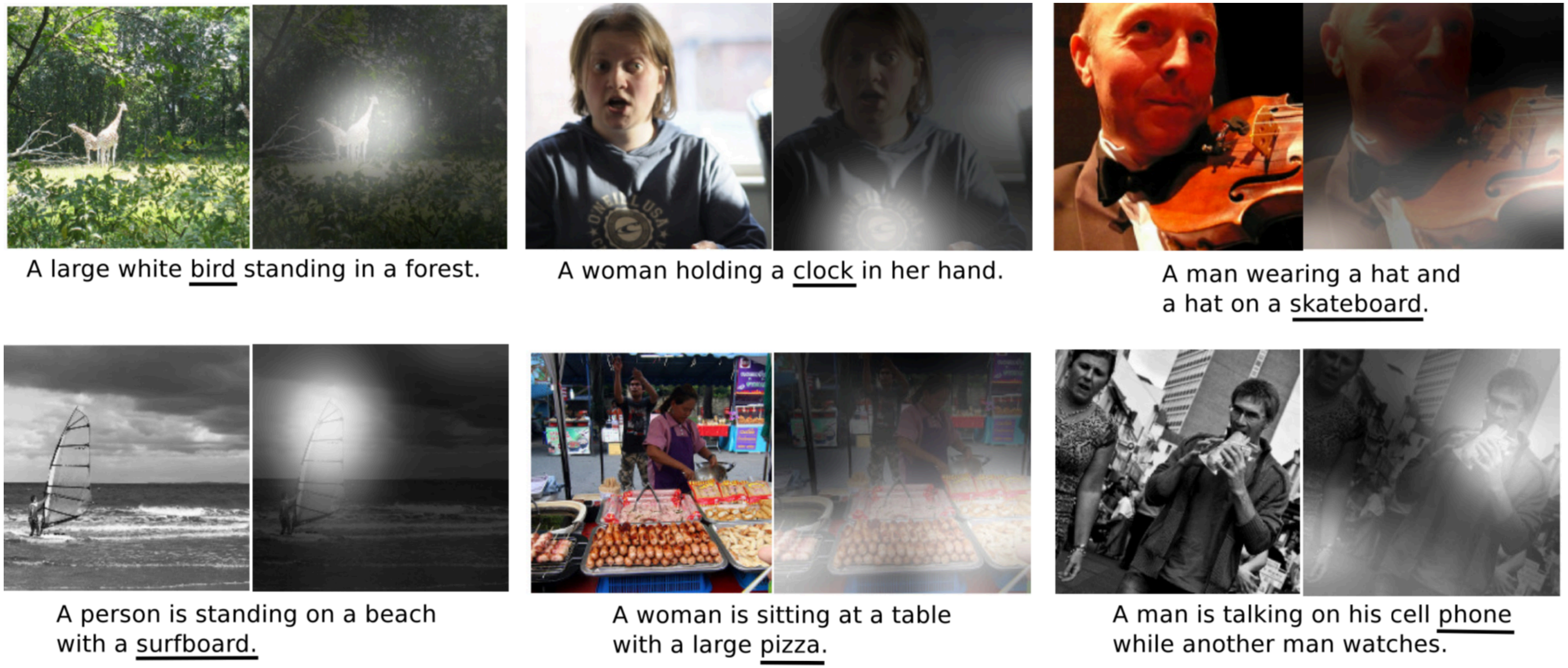
- Example: Employ attention mechanism in RNNs
  - Learn to describe the content of images: caption generation
- Visualise the attention weight matrix for each individual prediction



Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International conference on machine learning (pp. 2048-2057). PMLR.

Intrinsic Local Interpretability

- The attention can tell which mistakes the model made



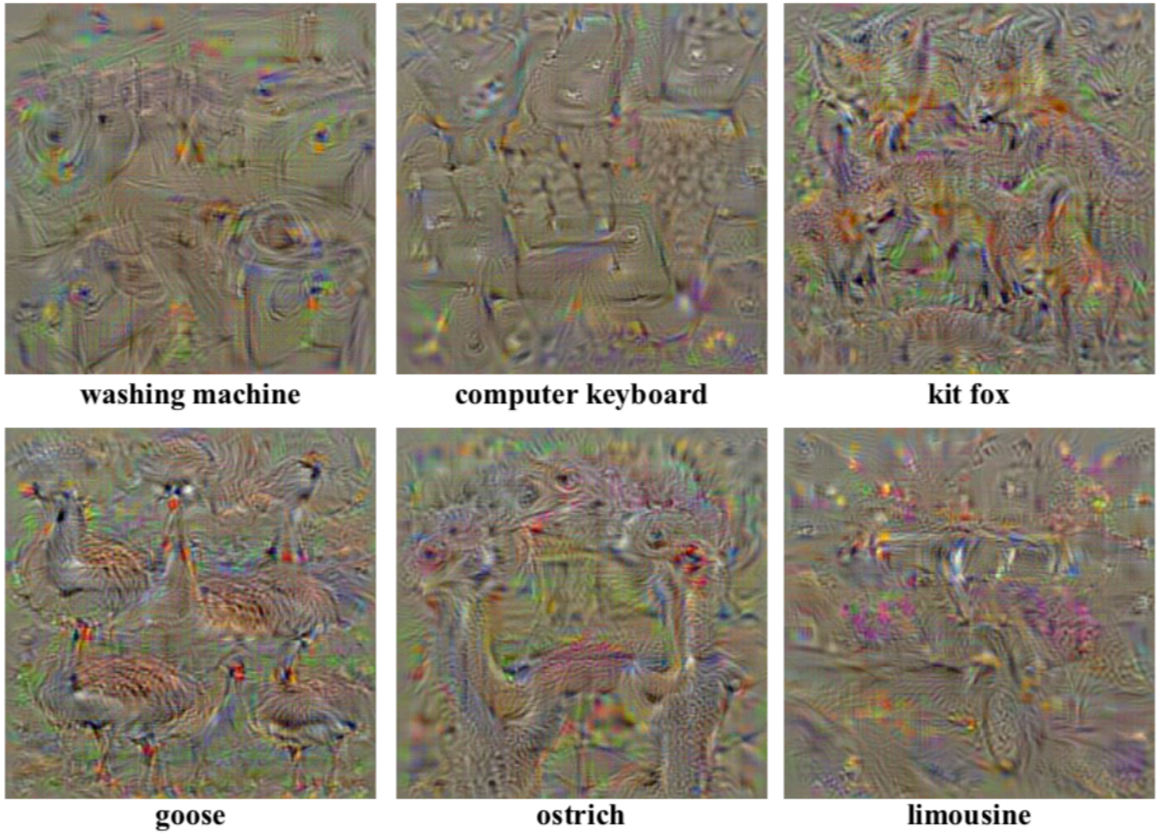
Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International conference on machine learning (pp. 2048-2057). PMLR.

Post-Hoc Interpretability

- Model-Specific
  - Designed for some specific model, e.g., deep learning
- Model-Agnostic
  - Can interpret/explain ANY model
  - Model simplification
  - Feature relevance/importance
  - Visualisation

Post-Hoc Global Interpretability

- **DNN-specific explanation**
  - Visualisation for class labels: generate a fake image I
  - $I = \arg\max_I S_c(I) - \lambda||I||_2^2$, where $S_c(I)$ is the score of class c by the classification layer



Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.

Post-Hoc Global Interpretability

- Use the input-output data predicted by the black-box model
- Train a simple model (e.g., decision tree, rules)
- **Model Agnostic**

- E.g., Use a decision tree to approximate
  - Can get promising accuracy – even better than the baseline
  - Cart pole policy explained by the decision tree:
    - To the right **if (pole velocity ≥ −0.286) ∧ (pole angle ≥ −0.071)**

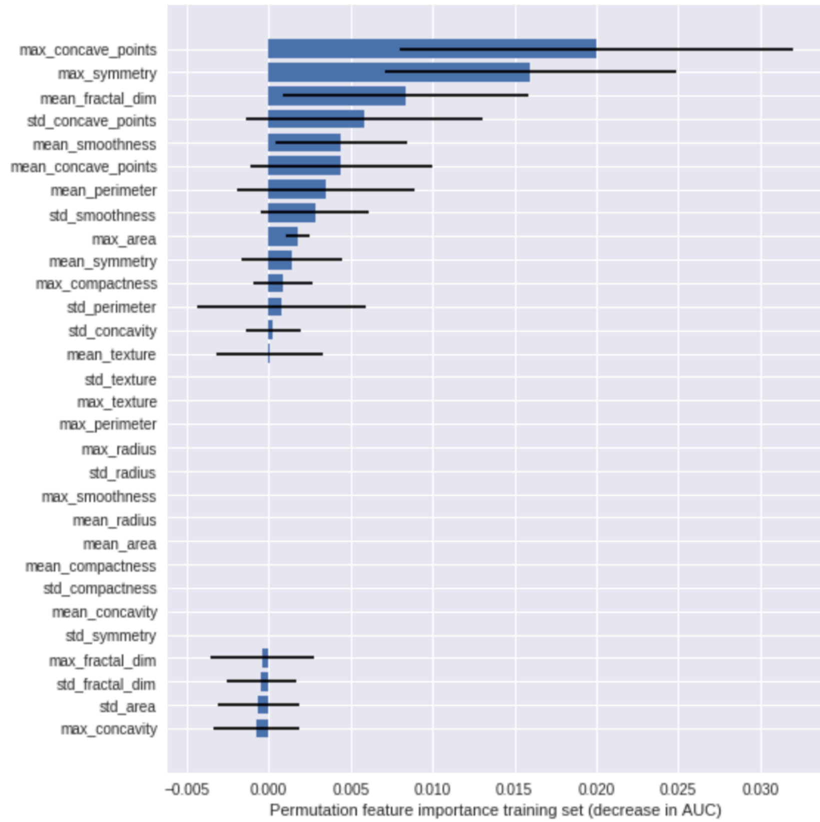
Description of Problem Instance					**Absolute**			**Relative**	
Dataset	Task	Samples	Features	Model	f	T	T_{base}	T	T_{base}
breast cancer [31]	classification	569	32	random forest	0.966	**0.942**	0.934	**0.957**	0.945
student grade [9]	regression	382	33	random forest	4.47	**4.70**	5.10	**0.40**	0.64
wine origin [14]	classification	178	13	random forest	0.981	**0.925**	0.890	**0.938**	0.890
wine origin [14]	classification	178	13	neural net	0.795	**0.755**	0.751	**0.913**	0.905
cartpole [5]	reinforcement learning	100	4	control policy	200.0	**190.0**	35.6	**86.8%**	83.8%

Bastani, O., Kim, C., & Bastani, H. (2017). Interpretability via model extraction. arXiv preprint arXiv:1706.09773.

Post-Hoc Global Interpretability

- Permutation **feature importance (Model Agnostic)**
  1. Calculate the baseline accuracy of the model on test dataset
  2. Permute the values of a feature on the test set, calculate the new accuracy on the modified dataset
  3. Repeat the permutation for all features, set the feature importance score as the accuracy reduction

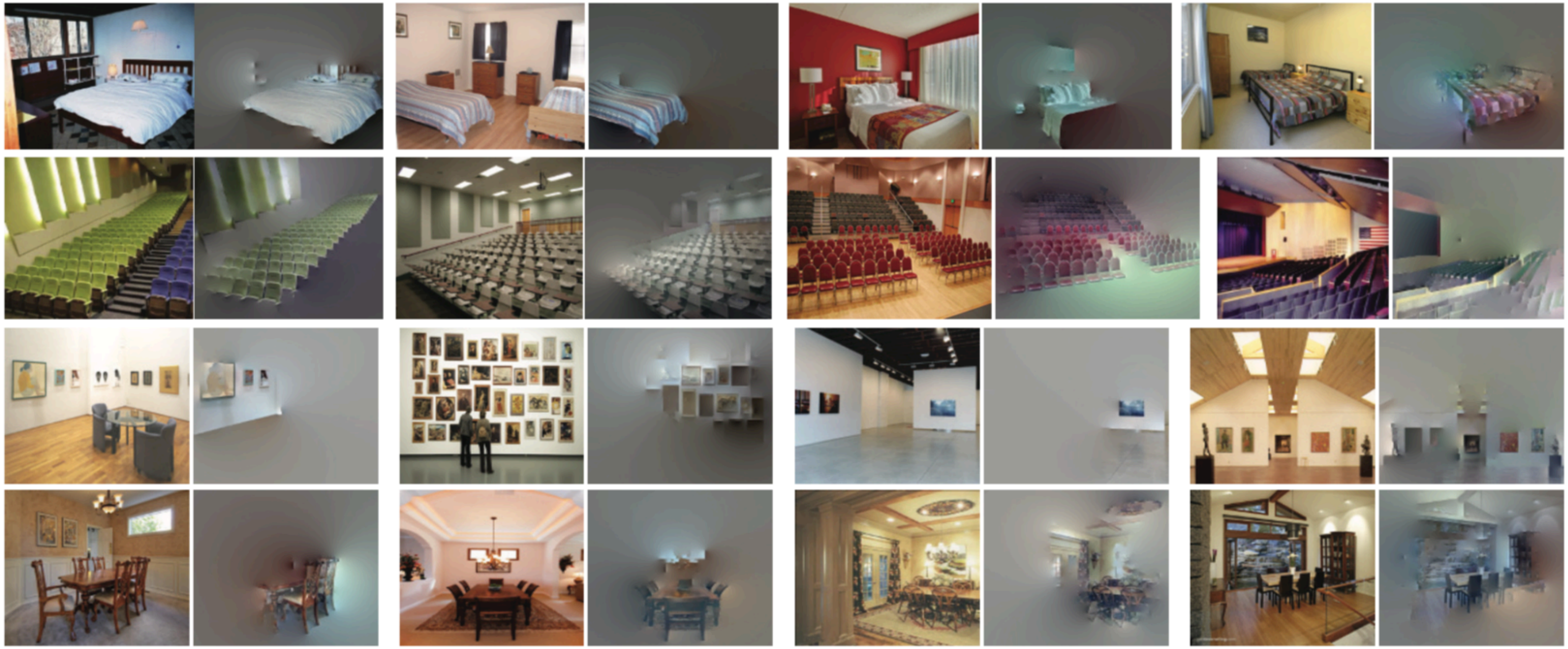
- Wisconsin breast cancer data
- Random forest



Du, M., Liu, N., & Hu, X. (2019). Techniques for interpretable machine learning. Communications of the ACM, 63(1), 68-77.

Post-Hoc Local Interpretability

- **DNN-specific explanation**
  - Simplify image:
    - Segment the image, and remove each component until it is misclassified by the CNN



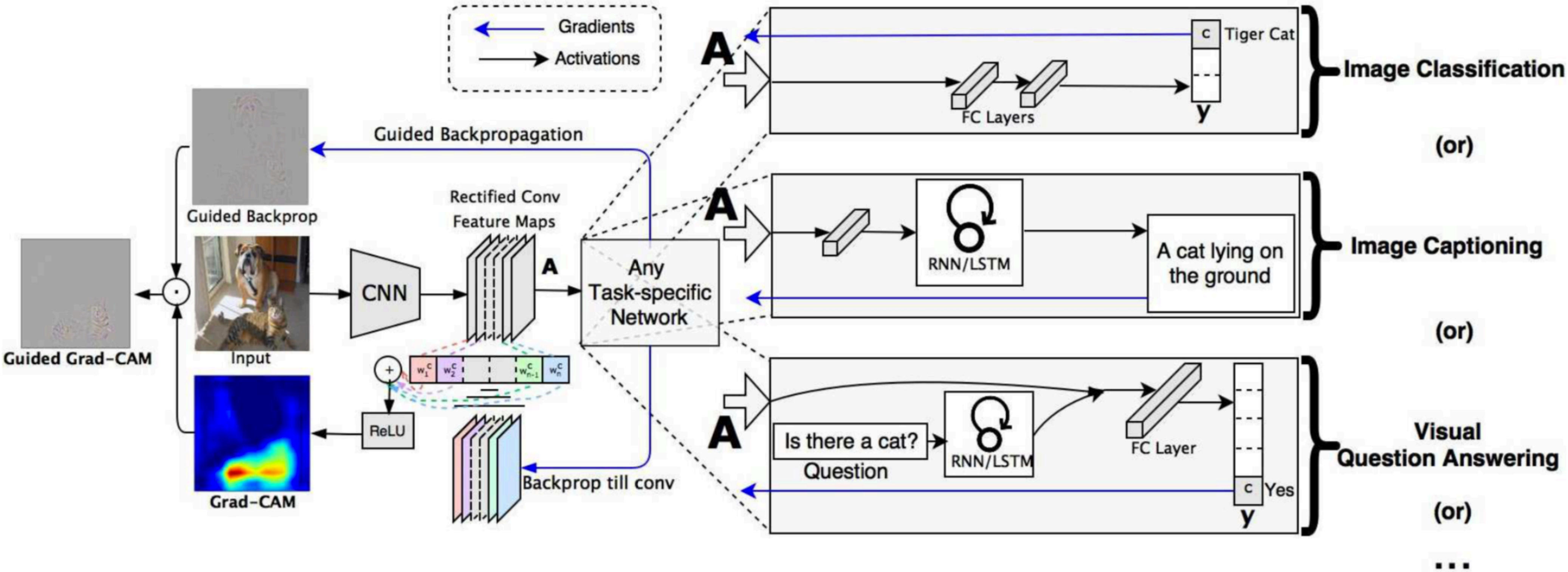
Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2014). Object detectors emerge in deep scene cnns. arXiv preprint arXiv:1412.6856.

Post-Hoc Local Interpretability

- **DNN-specific explanation**
  - **Grad-CAM**: use gradient

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

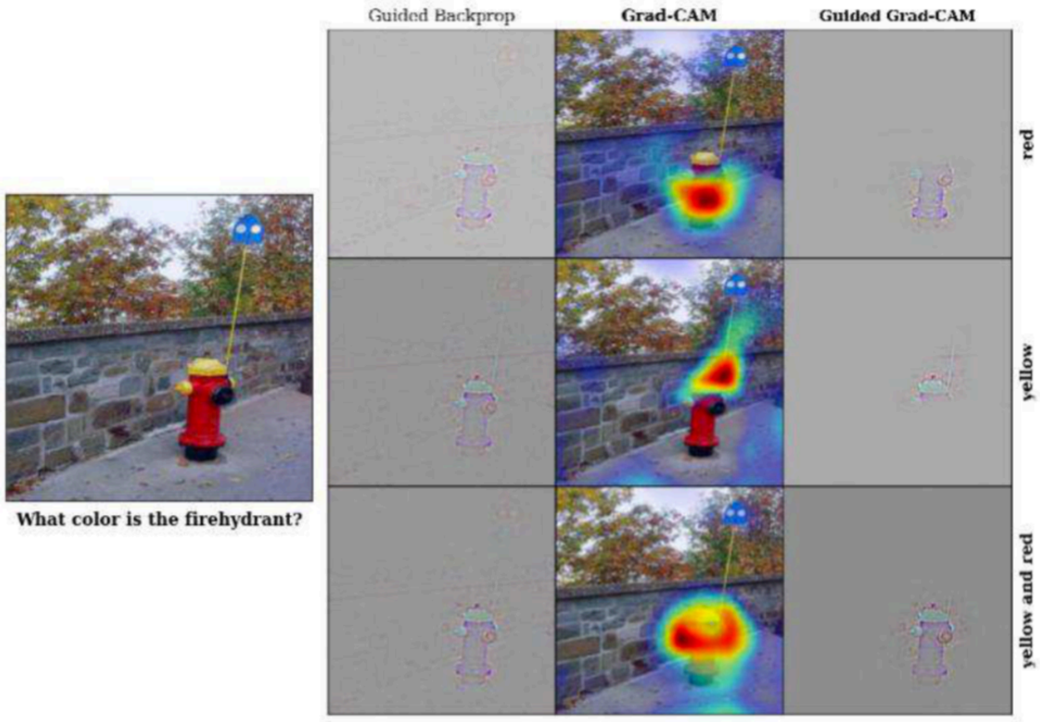
$$L_{\text{Grad-CAM}}^c = ReLU \underbrace{\left(\sum_k \alpha_k^c A^k \right)}_{\text{linear combination}}$$

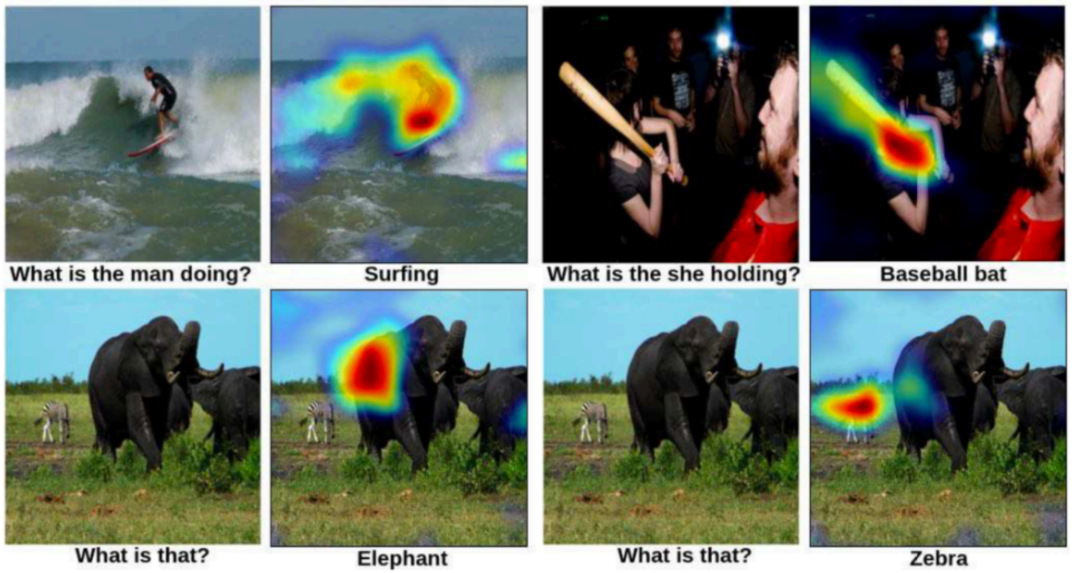


Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626).

Post-Hoc Local Interpretability

- **DNN-specific explanation**
  - **Grad-CAM**: show the important regions clearly

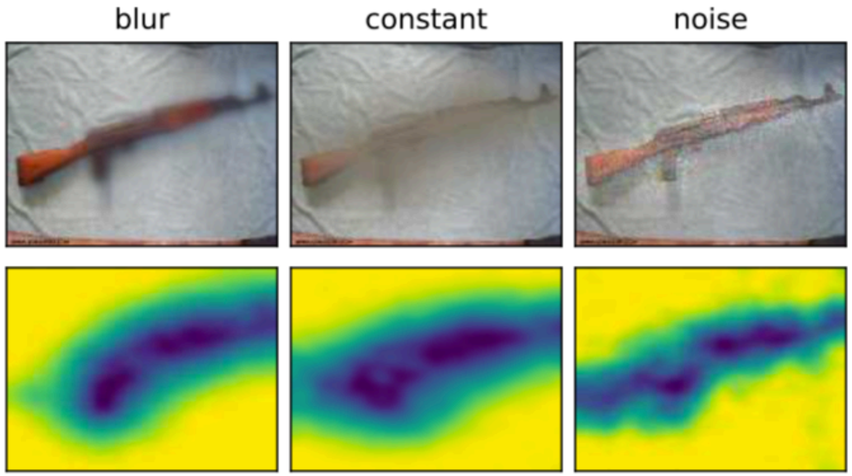




Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626).

Post-Hoc Local Interpretability

- **DNN-specific explanation**
  - Learn a mask



- Loss function
  - Regularisation (simply masks)
  - Classification drop significantly with the mask

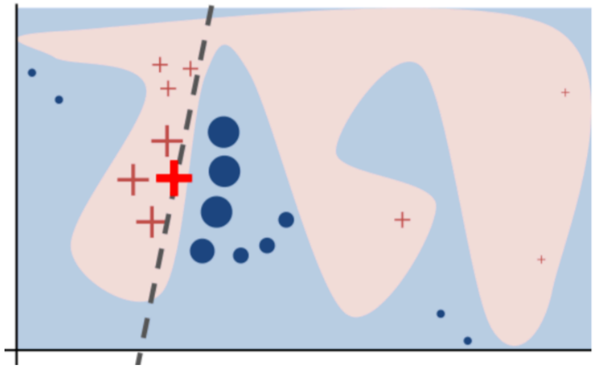
$$m^* = \operatorname*{argmin}_{m \in [0,1]^\Lambda} \lambda \|\mathbf{1} - m\|_1 + f_c(\Phi(x_0; m))$$

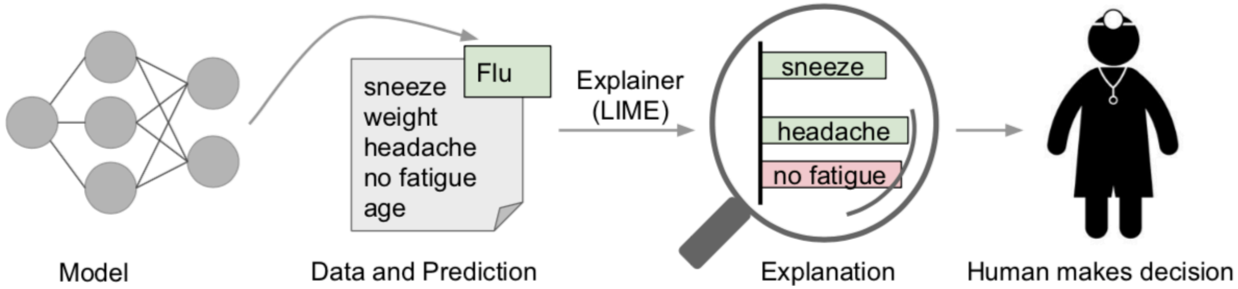
$$[\Phi(x_0; m)](u) = \begin{cases} m(u)x_0(u) + (1 - m(u))\mu_0, & \text{constant}, \\ m(u)x_0(u) + (1 - m(u))\eta(u), & \text{noise}, \\ \int g_{\sigma_0 m(u)}(v - u)x_0(v)\, dv, & \text{blur}, \end{cases}$$

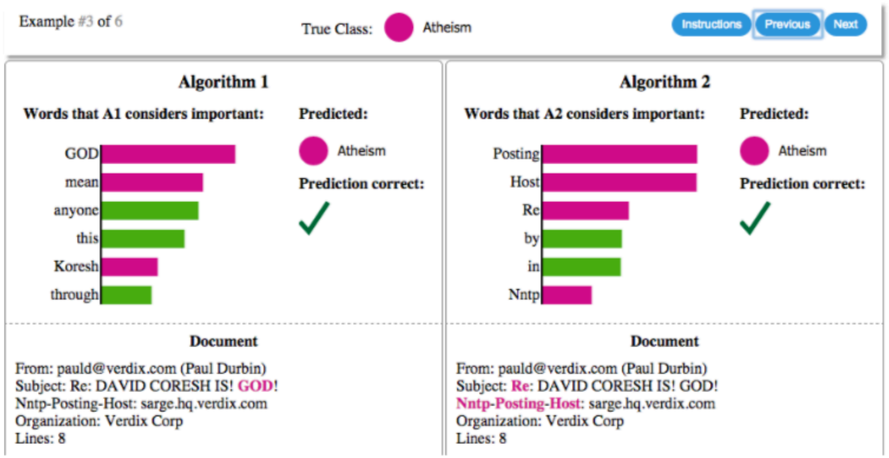
Fong, R. C., & Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. In ICCV (pp. 3429-3437).

LIME for Post-Hoc Local Interpretability

- **Model Agnostic**: show which features were most important for the model to make the decision
- Local fidelity vs interpretability: $\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$
  - G: class of linear models
  - $\pi_x(z) = e^{-\frac{D(x,z)^2}{\sigma^2}}$
  - $\mathcal{L}(f, g, \pi_x) = \sum_{z,z' \in Z} \pi_x(z)\left(f(z) - g(z')\right)^2$
  - $\Omega(g)$ is the task-specific interpretability measure (e.g., limiting number of words in text mining, number of super-pixels in image processing)







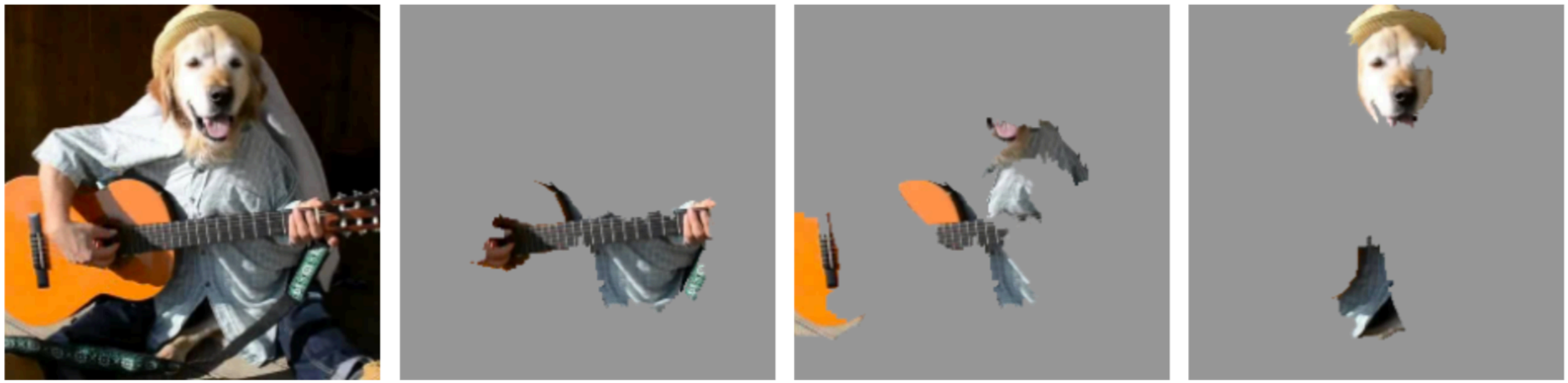
Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should I trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

LIME for Post-Hoc Local Interpretability

- Use K-Lasso to get the top k most important features
- Can show interpretable super-pixels contributing to the prediction

Algorithm 1 Sparse Linear Explanations using LIME

Require: Classifier f, Number of samples N
Require: Instance x, and its interpretable version x'
Require: Similarity kernel π_x, Length of explanation K
  $\mathcal{Z} \leftarrow \{\}$
  for $i \in \{1, 2, 3, ..., N\}$ **do**
    $z'_i \leftarrow sample\_around(x')$
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \langle z'_i, f(z_i), \pi_x(z_i) \rangle$
  end for
  $w \leftarrow$ K-Lasso($\mathcal{Z}, K$) ▷ with z'_i as features, $f(z)$ as target
  return w



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

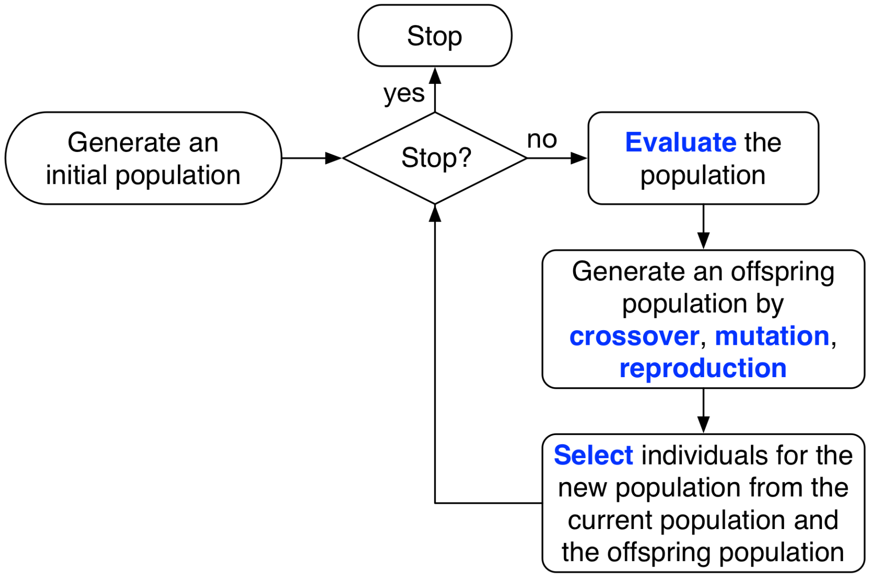
Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should I trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

Outline

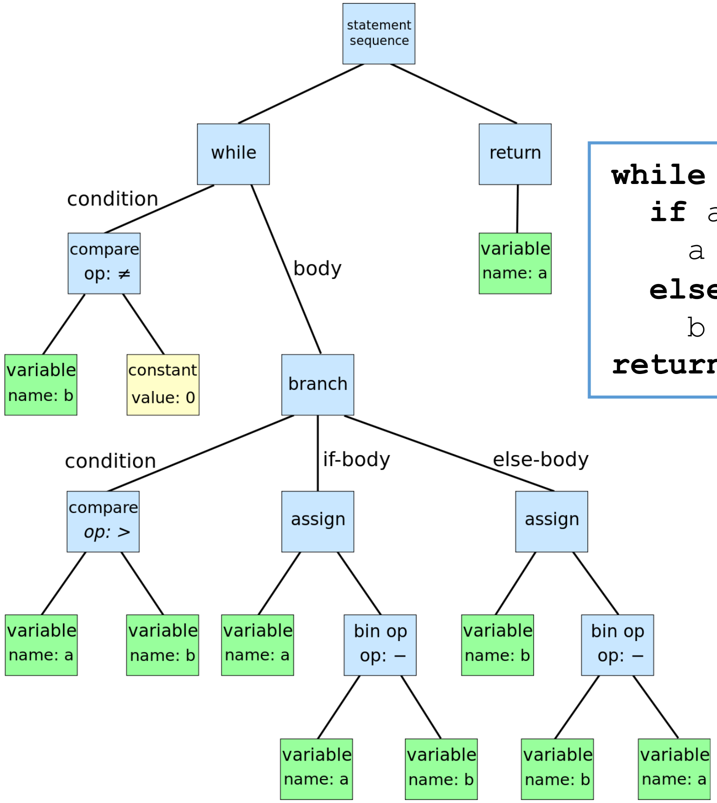
- Introduction to XAI
- **Introduction to GP**
- Better Interpretability Through GP
- Challenges and Future Directions

Genetic Programming (GP)

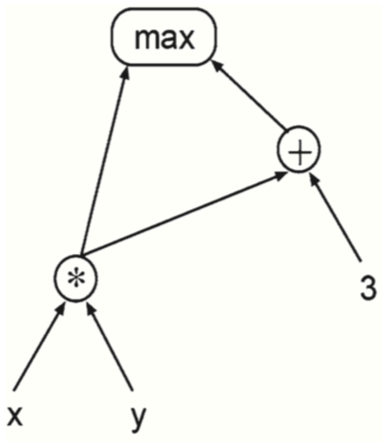
- A type of evolutionary algorithm
  - Evolve **computer programs** rather than solutions



- **Representation** of computer programs
  - Tree-like, graph-like, linear, …




```
while b ≠ 0
  if a > b
    a := a − b
  else
    b := b − a
return a
```

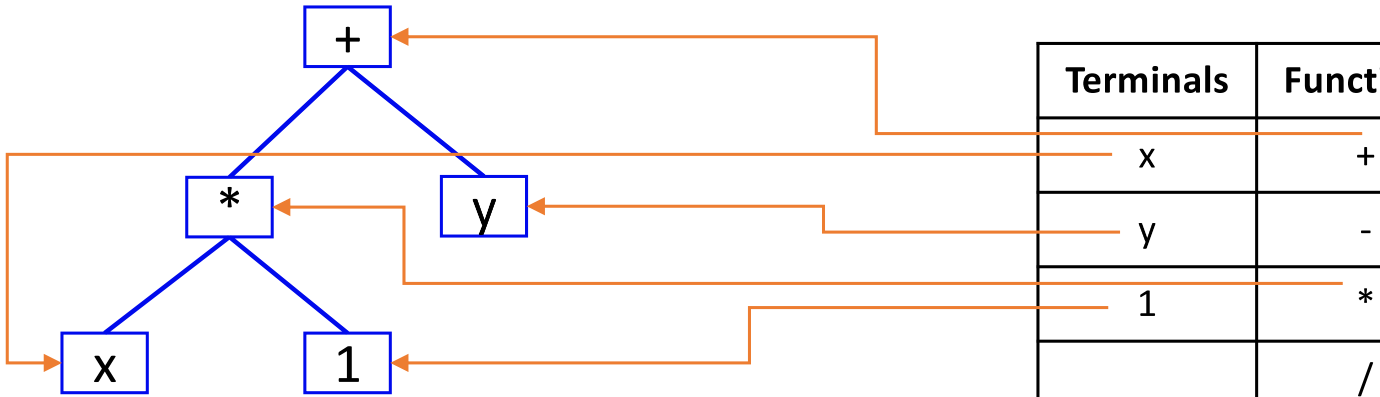


$\max(x * y, x * y + 3)$

```
r[3] = r[1] / 1.3;
r[1] = r[2] * -5.5;
r[0] = sqrt(10);
r[3] = r[1] + r[1];
r[1] = log(r[3]);
r[1] = r[3] >= r[0];
r[0] = abs(r[2]);
r[0] = if r[1] < 0 then
          r[0] else r[3];
```

Genetic Programming (GP)

- Individual generation (Tree-based representation)
  - **Terminal set**: inputs of the program and constants, no argument, form the leaf nodes,
  - **Function set**: operators to the inputs and intermediate results of the program (e.g. +, -, max, …), form the non-leaf nodes
- Start from the **root** node
- For each node, **randomly sample from the terminal/function set**
  - If sampling from the terminal set, then stop this branch
  - If sampling from the function set, create the child nodes, and **recursively sample** the child nodes



Terminals	Functions
x	+
y	-
1	*
	/

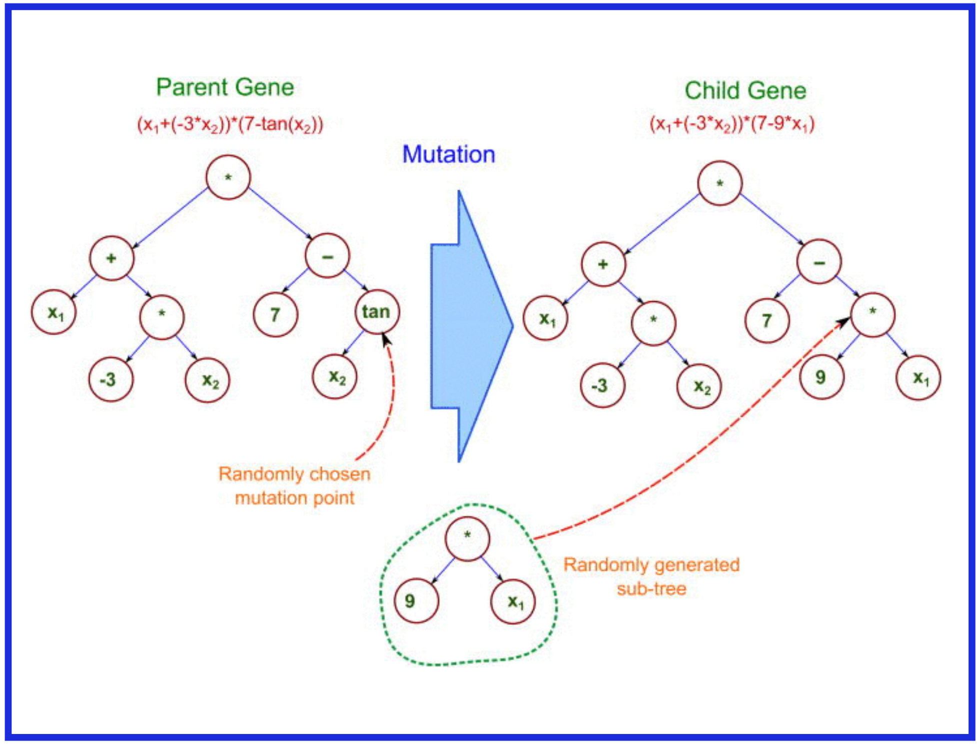
$$x * 1 + y$$

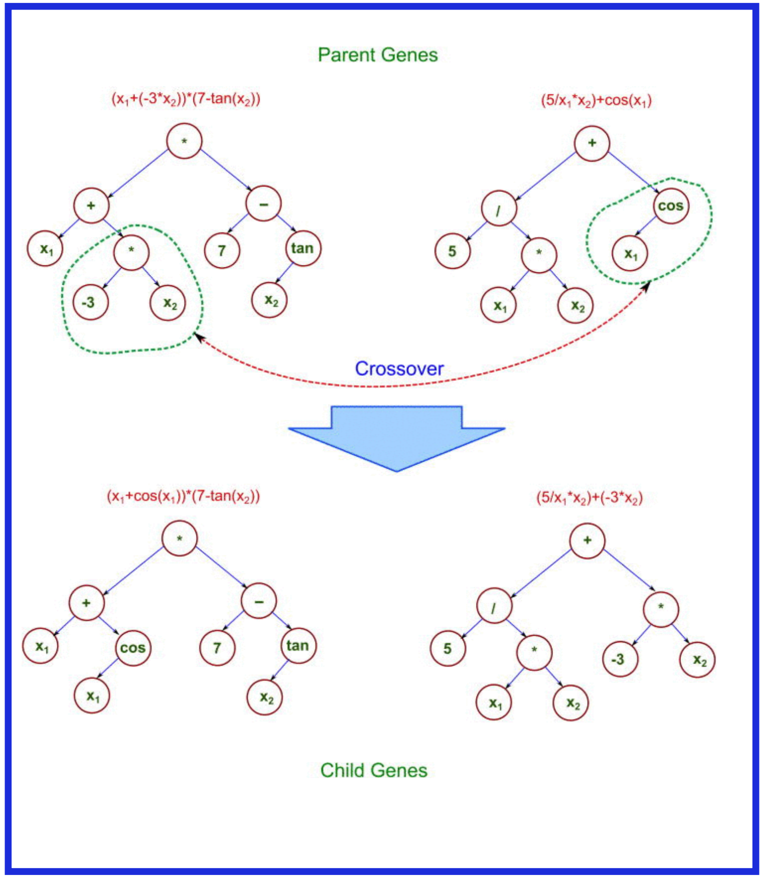
Genetic Programming (GP)

- **Sufficiency** and **Closure** for success of GP
  - Criteria for selecting the terminal and function sets
- Sufficiency: There must be some combination of terminals and function symbols that can solve the problem
  - If the target program is to calculate $\log(x) + 2^y$, but the function set is $\{+, -, *, /\}$, then not sufficient
- Closure: Any function can accept any input value returned by any function (and any terminal).
  - If the function set includes $AND(boolean, boolean)$ and $+$, then not closure, since we may have AND taking the real-value inputs.

Genetic Programming (GP)

- GP genetic (Crossover/Mutation) operators depends on representation





GP vs GA

Genetic Algorithm

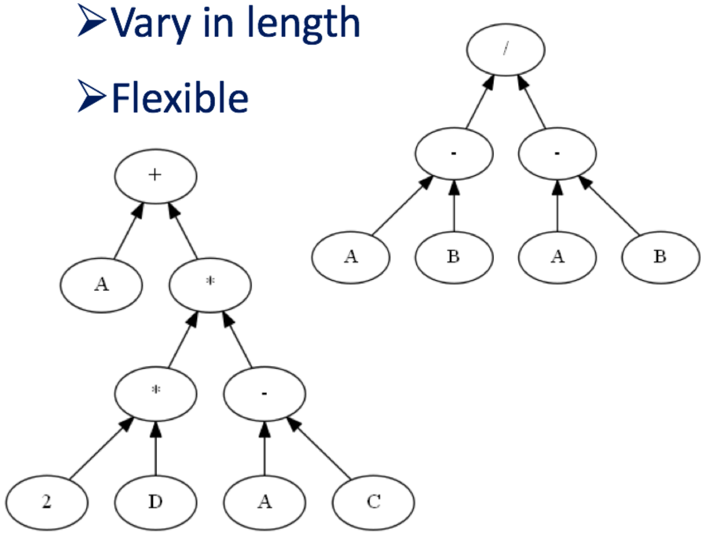
- Bit string representation
- Fixed in length
- Inflexible

1	0	1	1	0	1	0	1	1	0

0	0	0	1	0	1	1	1	0	1

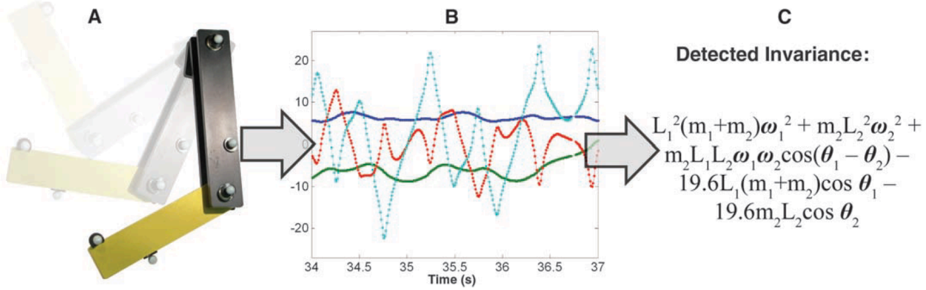
Genetic Programming

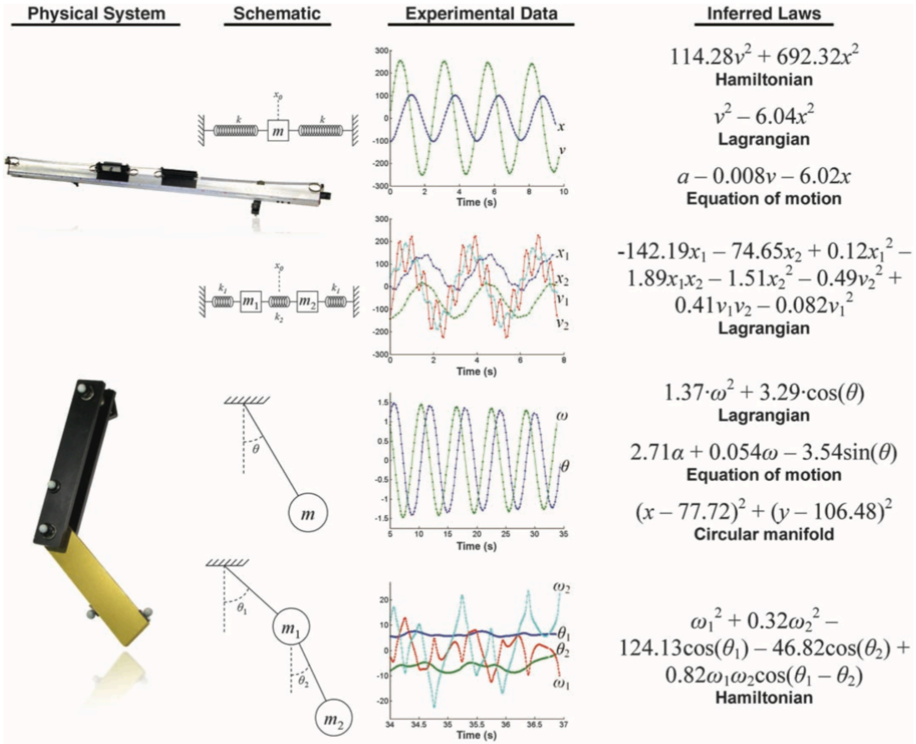
- Tree-like structure
- Vary in length
- Flexible



GP for Symbolic Regression

- In real world, the relationship structure between variables are usually unknown
- Symbolic regression is to learn both the **model structure** and **coefficients**
  - Can be very helpful for natural law discovery



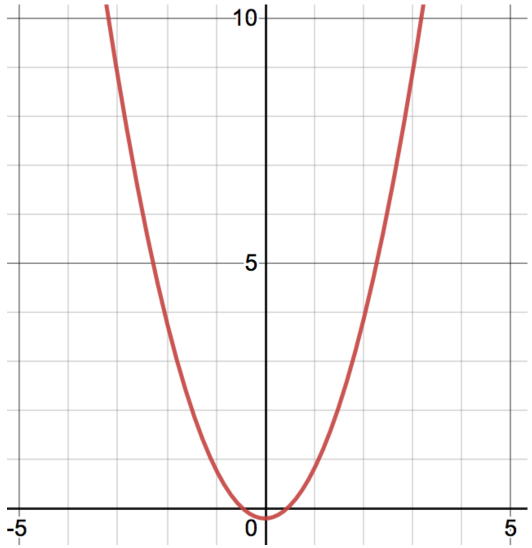


Schmidt, M., & Lipson, H. (2009). Distilling free-form natural laws from experimental data. science, 324(5923), 81-85.

GP for Symbolic Regression

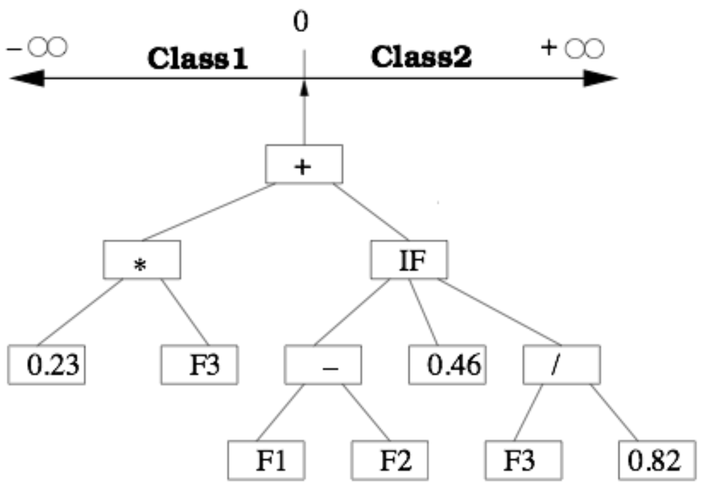
- Given a set of training data $(x_1, x_2, \ldots, x_n, y)$
  - Define terminal set $\{x_1, x_2, \ldots, x_n, w\}$
  - Define function set $\{+, -, *, /, \log, \ \ldots\}$
  - Define the fitness function
    - Mean squared error $\sum_i (gp(\overrightarrow{x_i}) - y_i)^2$
    - Can consider regularisation (generalisation performance)
- **Initialisation**
  - Use the program generation (grow, full, ramp-half-and-half)
- **Breeding**
  - Elitism: select the top individuals directly
  - Tournament selection to select parents
  - Tree-based crossover and mutation, reproduction
  - Directly copy the generated offspring to the next population

x	y
1	0.8
2	3.8
3	8.8
…	…



GP for (Binary) Classification

- Given a set of training data (feature vector and class label)
- Evolve GP program in the same way as regression
- Translate the final real-valued output into class prediction



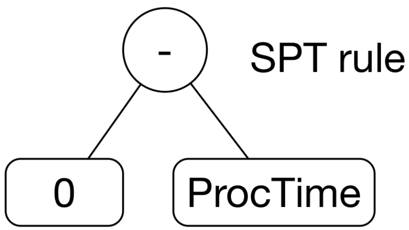

```
Genetic Program: (+ (* 0.23 F3)
                    (IF (- F1 F2) 0.46 (/ F3 0.82))
                 )
```

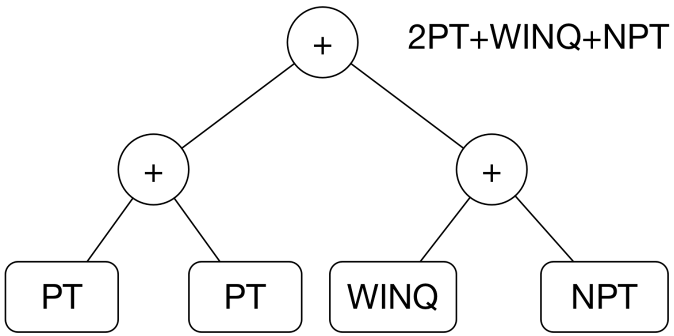
```
if ProgOut < 0  then  Class1  else   Class2;
```

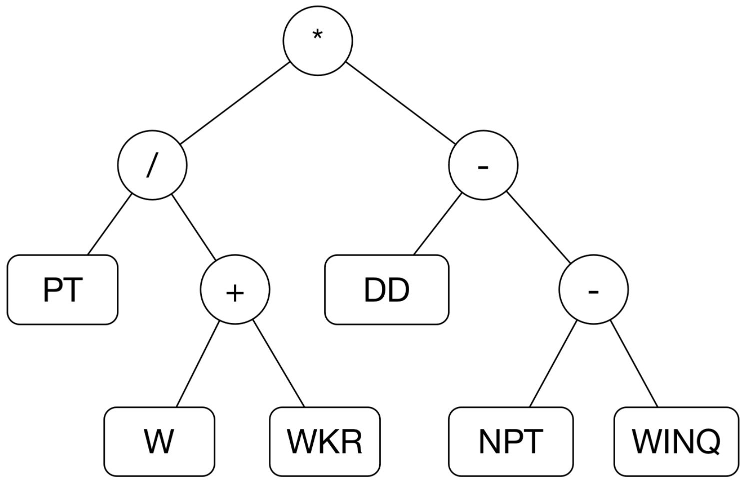
GP for Learning Decision Making Policy

- Dispatching rules in job shop scheduling
  - When a machine becomes idle, select the next job in the queue
  - E.g. first-come-first-serve, shortest processing time, …
- Routing policy in vehicle routing
  - When a vehicle becomes idle, select the next customer to serve
  - E.g. nearest neighbour, path scanning, saving, …

- Use GP to learn a priority function of the candidates (jobs, customers, …)
- Calculate the priority of the candidates
- Select the next candidate based on priority

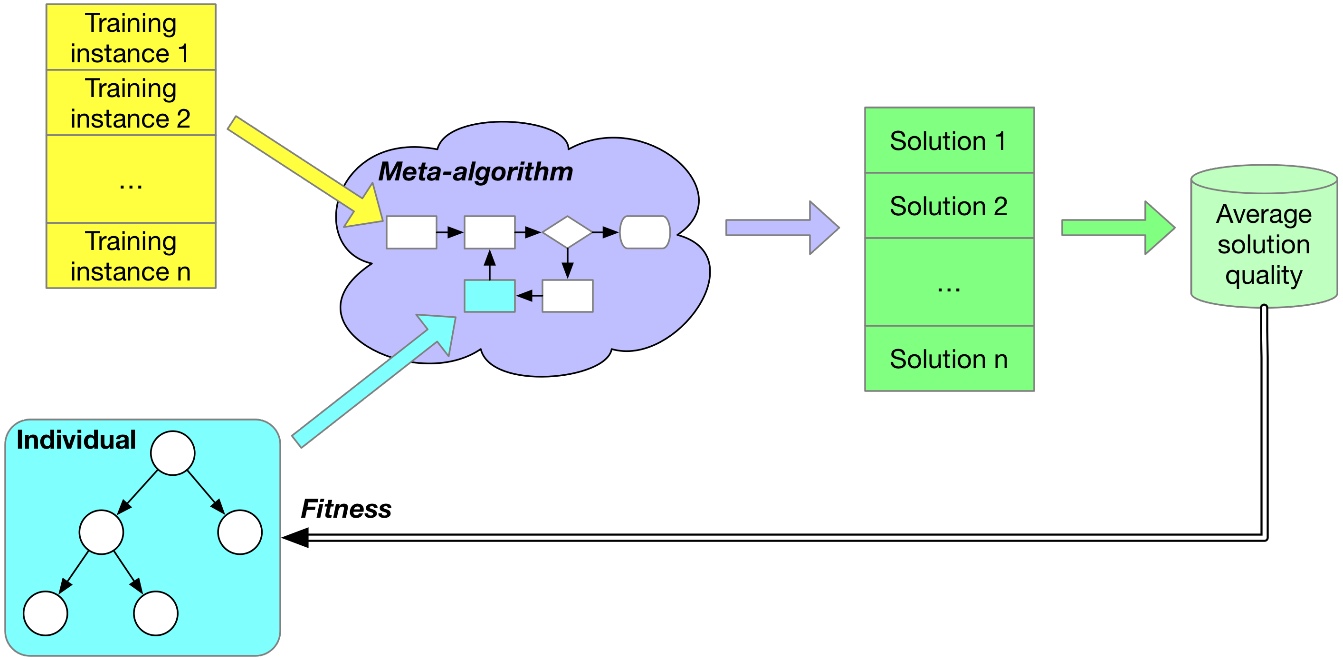






GP for Learning Decision Making Policy

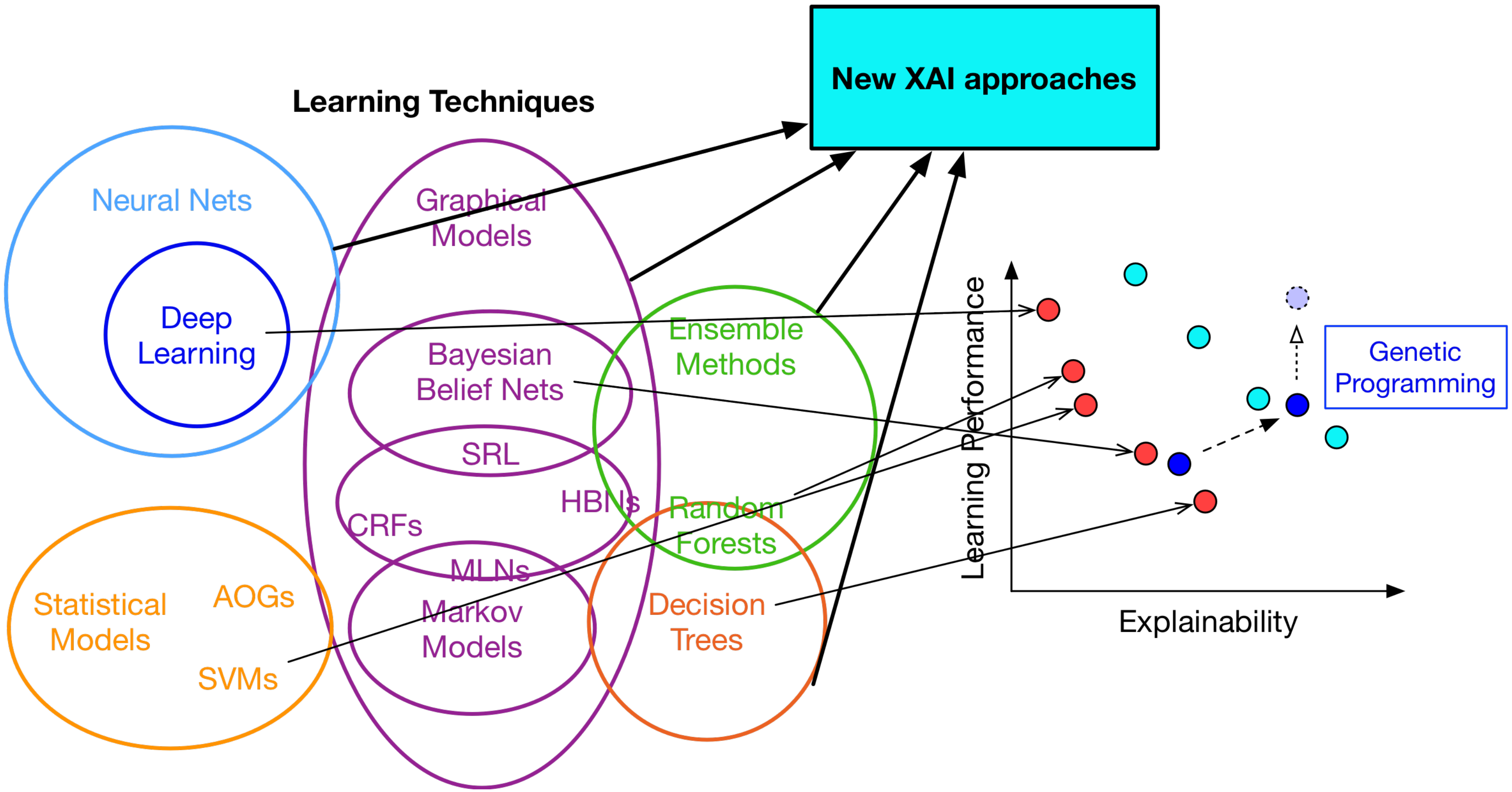
- Given a set of training data (**problem instances**)
  - Define terminal set: the state attributes/features, constants
  - Define function set: $\{+, -, *, /, \log, \max, \min, \dots\}$
  - Define the fitness function
    - For each training instance, run a simulation (meta-algorithm) using this GP rule, get a solution to the instance
    - Calculate the objective value of the obtained solution
    - Fitness can be set to the average normalised objective value of the solutions



Outline

- Introduction to XAI
- Introduction to GP
- **Better Interpretability Through GP**
- Challenges and Future Directions

Learning Performance vs Explainability with GP



Better Interpretability Through GP

- Improve the **interpretability of GP-evolved models**
  - Consider model size (e.g., number of nodes): bloat control
  - Consider number of features used in the model: feature selection
  - Consider model complexity (e.g., non-linear operators are more complex)
  - Consider physical meanings (e.g., time cannot be added with length)

  - Constrained GP (penalise less interpretable models)
  - Multi-objective GP (accuracy vs interpretability measures)
  - Simplification (e.g., tree pruning)
  - Different GP representations (e.g., strongly-typed, grammar-guided, ensemble/multi-tree)
  - Visualisation
- Use **GP to interpret other** complex models
  - Post-hoc local interpretability
  - Visualisation

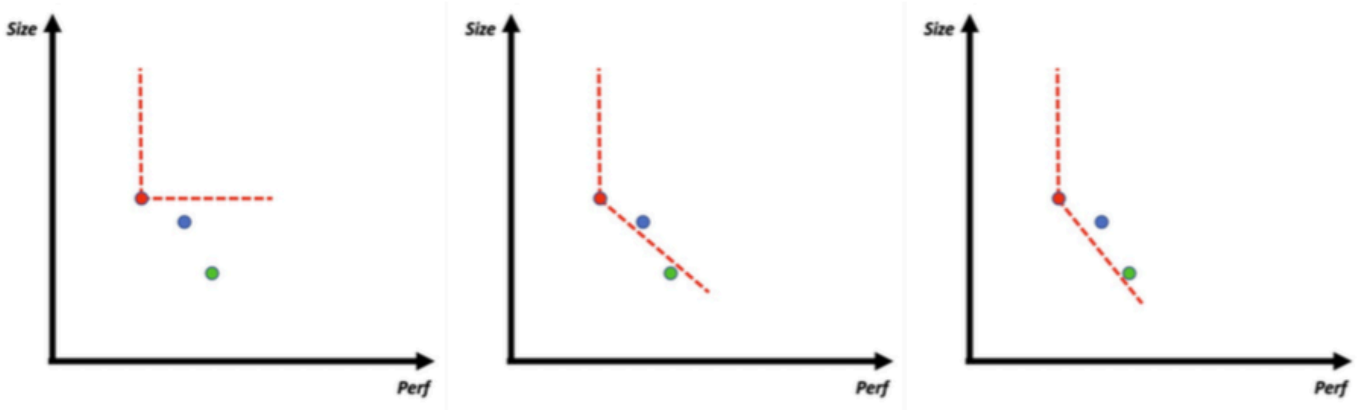
Accuracy vs Model Size: Bloat Control

- Tarpeian Method (Penalisation): if an individual is too large (above average size), then assign a very bad fitness to it
- Parsimony Pressure
  - Linear: $fit = obj + \alpha * size$
  - Lexicographic: divide the individuals into different buckets based on fitness, and select the individuals first based on the rank of bucket, second based on size
- Double Tournament
  - First tournament selects candidates based on fitness
  - Second tournament selects the parent from the first tournament winners based on size
- Waiting room
- Operator equalisation
  - Try to make a flat distribution of program size, reduce crossover bias
- …

Luke, S., & Panait, L. (2006). A comparison of bloat control methods for genetic programming. Evolutionary Computation, 14(3), 309-344.

Multi-Objective GP with α-dominance

- It is hard to balance effectiveness (e.g., accuracy) and size during the MOGP search
- If not evolve properly, the population can be easily biased to small but bad individuals, and lose exploration ability
- Use α-dominance to adjust the balance between effectiveness and size
  - $\alpha = 0$: normal dominance relationship
  - $\alpha = \infty$: single objective with only effectiveness

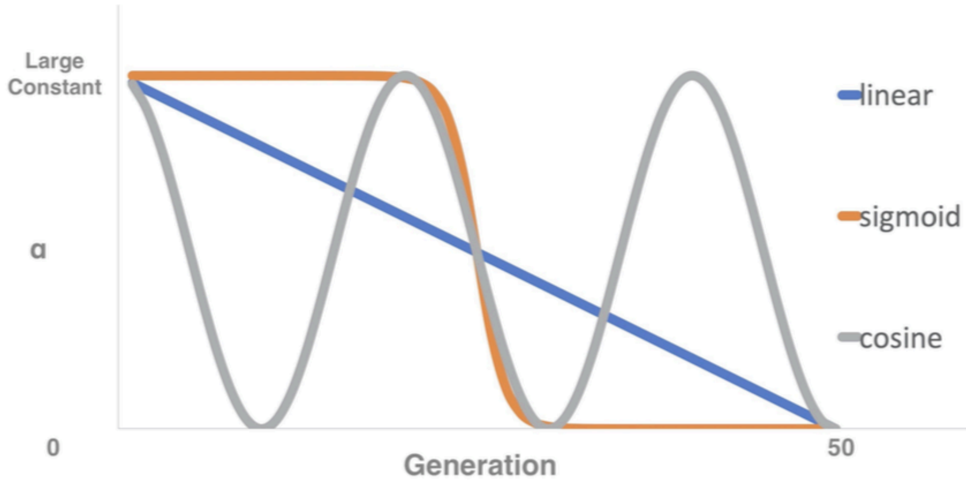


(a) $\alpha = 0$ (b) $\alpha = 10$ (c) $\alpha = 100$

Wang, S., Mei, Y., & Zhang, M. (2020, July). A multi-objective genetic programming hyper-heuristic approach to uncertain capacitated arc routing problems. In 2020 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

Multi-Objective GP with α-dominance

- Test on uncertain arc routing, to evolve routing policies
- Use NSGA-II + GP
- Different α adaptation schemes
  - **Linear**: gradually shift from effectiveness to size
  - **Sigmoid**: focus on effectiveness first, then quickly shift to size
  - **Cosine**: focus on effectiveness first, shift to size, then back to effectiveness, and back and forth



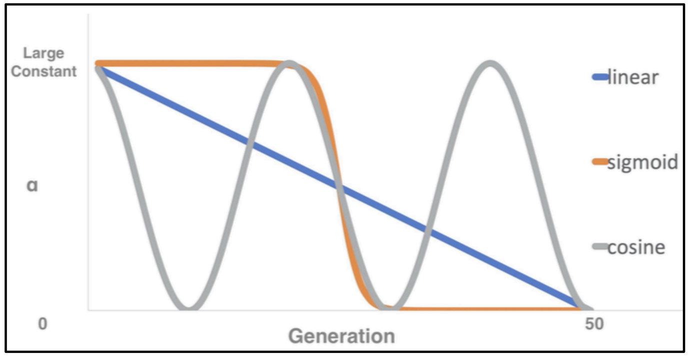
Wang, S., Mei, Y., & Zhang, M. (2020, July). A multi-objective genetic programming hyper-heuristic approach to uncertain capacitated arc routing problems. In 2020 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

Multi-Objective GP with α-dominance

- Much better than normal MOEAs
- The sigmoid adaptation seems better than linear and cosine

HV value

Instance	NSGA-II	SPEA2	α-MOGP-l	α-MOGP-s	α-MOGP-c
ugdb1	0.9071	0.8645	**0.9389**	**0.9427**	**0.9423**
ugdb2	0.9153	0.8894	**0.9395**	**0.9572**	**0.9423**
ugdb8	0.9142	0.8625	**0.9404**	**0.9505**	**0.9427**
ugdb23	0.8889	0.8738	**0.9295**	**0.9416**	**0.9341**
uval9A	0.9756	0.9577	**0.9781**	**0.9853**	**0.9811**
uval9D	0.9190	0.8528	**0.9393**	**0.9581**	**0.9480**
uval10A	0.9736	0.9534	**0.9832**	**0.9905**	**0.9859**
uval10D	0.9302	0.8986	**0.9518**	**0.9724**	**0.9630**



$$RP = \max(S_1, S_2),$$
$$S_1 = DC * CFH + CTT1,$$
$$S_2 = \frac{DEM1}{DC - CFR}.$$

Wang, S., Mei, Y., & Zhang, M. (2020, July). A multi-objective genetic programming hyper-heuristic approach to uncertain capacitated arc routing problems. In 2020 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

Multi-Objective GP with α-dominance

- Select the most effective rule from the Pareto front, compare its effectiveness and size
  - Traditional MOEAs biased too much to the small but bad individuals
  - α-MOGP can obtain similar effectiveness with much smaller size

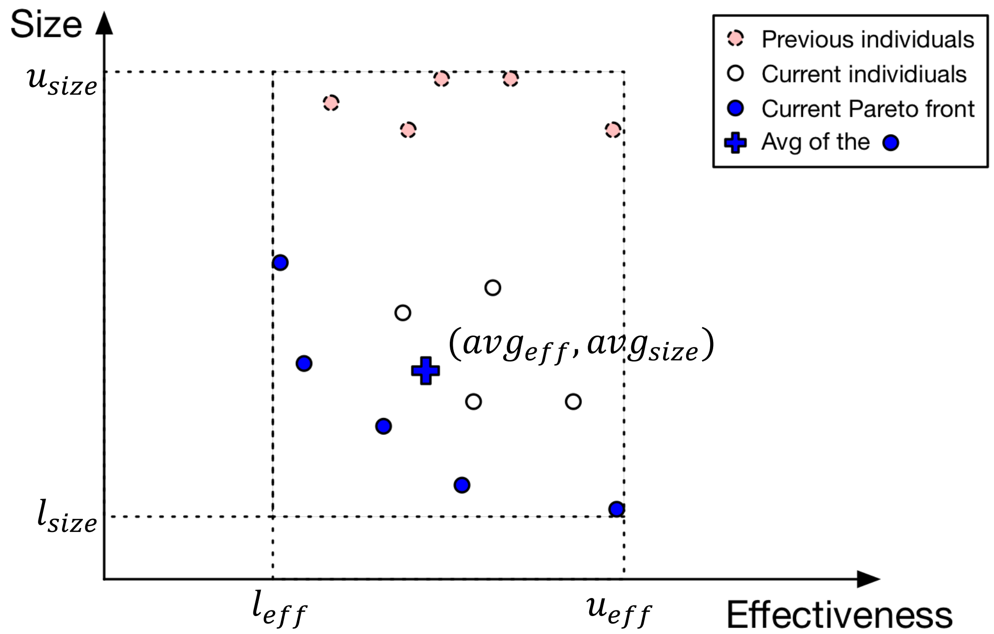
Instance	SO-GP		NSGA-II		SPEA2		α-MOGP-s	
	tc	size	tc	size	tc	size	tc	size
ugdb1	355.47	74.6	373.2	10.0	389.15	8.4	358.4	17.33
ugdb2	371.72	71.93	392.8	6.93	404.08	5.73	370.8	26.53
ugdb8	463.34	65.47	476.3	7.07	509.82	5.6	448.9	30.87
ugdb23	252.47	71.8	260.2	8.27	262.35	8.6	252.0	33.07
uval9A	335.13	56.93	351.3	9.73	371.24	8.53	336.4	26.07
uval9D	478.14	69.27	522.7	10.33	586.58	7.53	479.3	37.0
uval10A	439.41	60.47	460.0	8.07	481.59	4.53	440.9	15.73
uval10D	620.91	65.33	668.9	8.93	699.8	9.2	622.2	34.13

Wang, S., Mei, Y., & Zhang, M. (2020, July). A multi-objective genetic programming hyper-heuristic approach to uncertain capacitated arc routing problems. In 2020 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

Multi-Objective GP with α-dominance

- Learn to adjust the α value online, rather than setting it manually
  - If there are many small but bad individuals in the population, increase α
  - If there are many large individuals in the population, decrease α
- Calculate the boundaries found so far
  - u_{eff} and l_{eff} for effectiveness
  - u_{size} and l_{size} for size
  - Find the current pareto front
  - Calculate its average avg_{eff} and avg_{size}

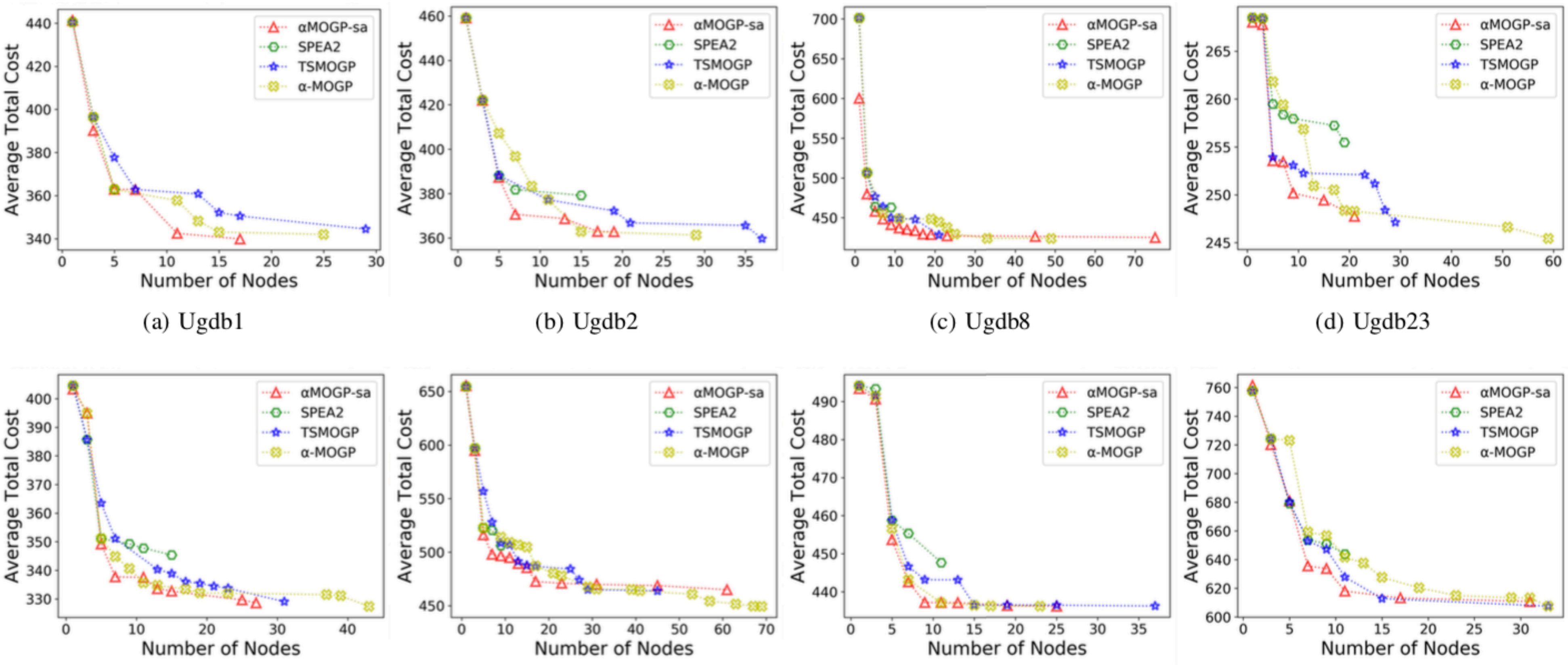
If $avg_{eff} < \frac{u_{eff}+l_{eff}}{2}$ and $avg_{size} > \frac{u_{size}+l_{size}}{2}$ **then**
$\alpha = \alpha - \Delta$
If $avg_{eff} > \frac{u_{eff}+l_{eff}}{2}$ and $avg_{size} < \frac{u_{size}+l_{size}}{2}$ **then**
$\alpha = \alpha + \Delta$



Wang, S., Mei, Y., & Zhang, M. (2020, July). A multi-objective genetic programming hyper-heuristic approach to uncertain capacitated arc routing problems. In 2020 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

Multi-Objective GP with α-dominance

- Better Pareto front on different instances



(a) Ugdb1 (b) Ugdb2 (c) Ugdb8 (d) Ugdb23

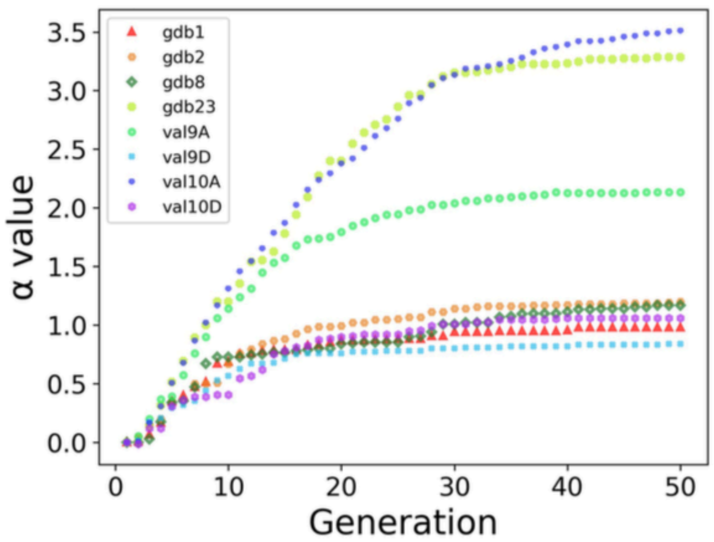
(e) Uval9A (f) Uval9D (g) Uval10A (h) Uval10D

Wang, S., Mei, Y., & Zhang, M. (2021). A Multi-Objective Genetic Programming Approach with Self-Adaptive α Dominance to Uncertain Capacitated Arc Routing Problem. In 2021 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

Multi-Objective GP with α-dominance

- Select the most effective rule from the Pareto front, compare its effectiveness and size
  - Slightly better than $\boldsymbol{\alpha}$-MOGP
  - Better than SO-GP and SPEA2
  - Different $\boldsymbol{\alpha}$ adaptation for different instances

Instance	SO-GP		SPEA2		α-MOGP		α-MOGP-sa	
	tc	size	tc	size	tc	size	tc	size
ugdb1	355.47	74.6	389.15	8.4	354.77	27.67	351.82	34.47
ugdb2	371.72	71.93	404.08	5.73	370.17	28.8	372.92	28.07
ugdb8	430.34	65.47	509.82	5.6	441.05	51.67	433.73	51.2
ugdb23	252.47	71.8	262.35	8.6	250.46	47.53	251.23	37.07
uval9A	335.13	56.93	371.24	8.53	336.03	28.4	333.87	32.27
uval9D	478.14	69.27	586.58	7.53	474.24	58.0	477.67	40.7
uval10A	439.41	60.47	481.59	4.53	440.51	19.27	438.18	25.8
uval10D	620.91	65.33	699.8	9.2	619.58	47.67	620.21	42.4



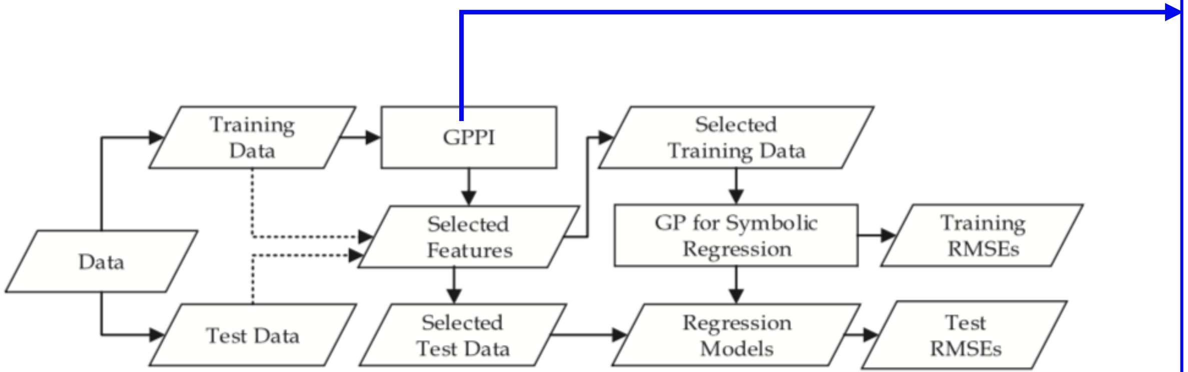
Wang, S., Mei, Y., & Zhang, M. (2021). A Multi-Objective Genetic Programming Approach with Self-Adaptive α Dominance to Uncertain Capacitated Arc Routing Problem. In 2021 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

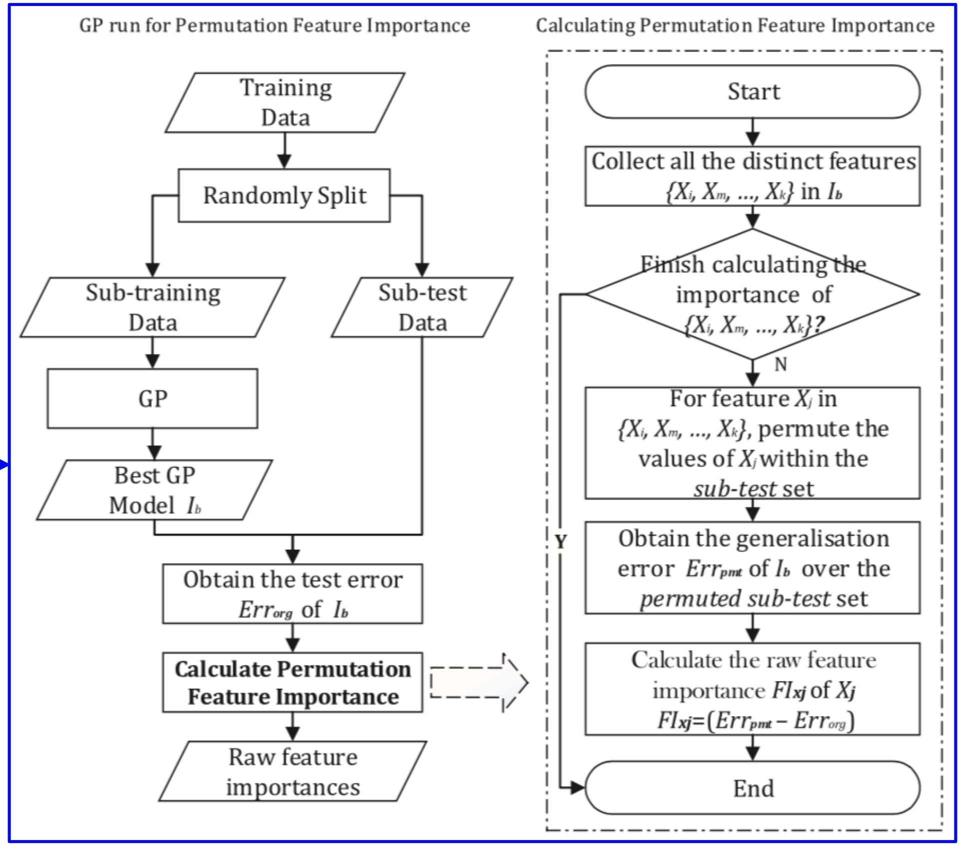
GP with Feature Selection

- Reducing the number of used features in the final evolved GP model can improve interpretability
- Although GP can naturally do feature selection, we may need stronger and explicit feature selection for complex problems
- The key issue is to estimate the feature importance based on the information collected during the GP process
  - Offline
  - Online

GP with Permutation Importance for Symbolic Regression

- Run GP to get a good model
  - Best trained model in the run
- For each feature in the good model, do a permutation test to calculate the feature importance
  - $FI_{raw}(X_j, I_b) = Err_{pmt}(I_b) - Err_{org}(I_b)$
  - $FI_{sca}(X_j) = \frac{avg_i\left(FI_{raw}(X_j, I_{b,i})\right)}{\sigma/\sqrt{n}}$
- Select the features with large importance: $FI_{sca}(X_j) > 0$
- Run GP again with the selected features

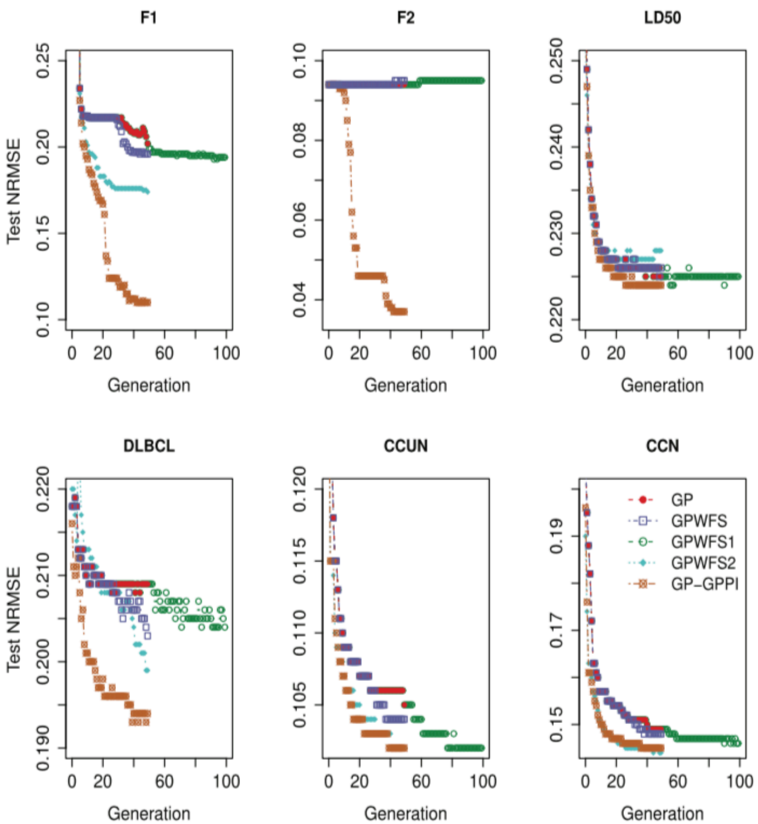


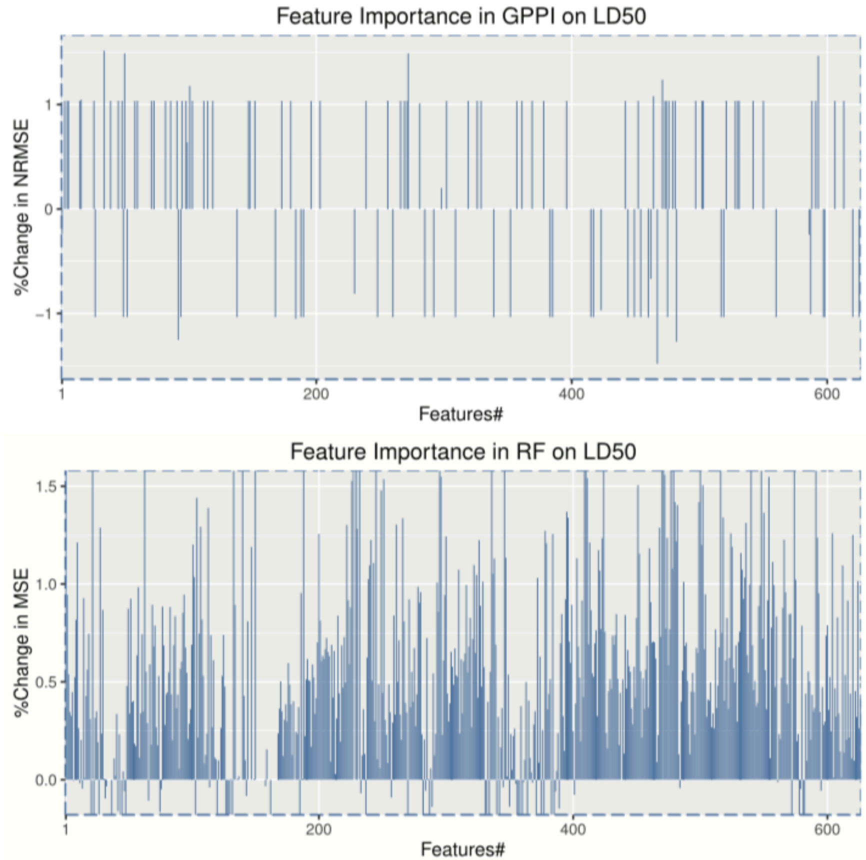


Chen, Q., Zhang, M., & Xue, B. (2017). Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression. IEEE Transactions on Evolutionary Computation, 21(5), 792-806.

GP with Permutation Importance for Symbolic Regression

- Much better generalisation/test performance (since the models are simpler)
- Can select much fewer features than existing methods

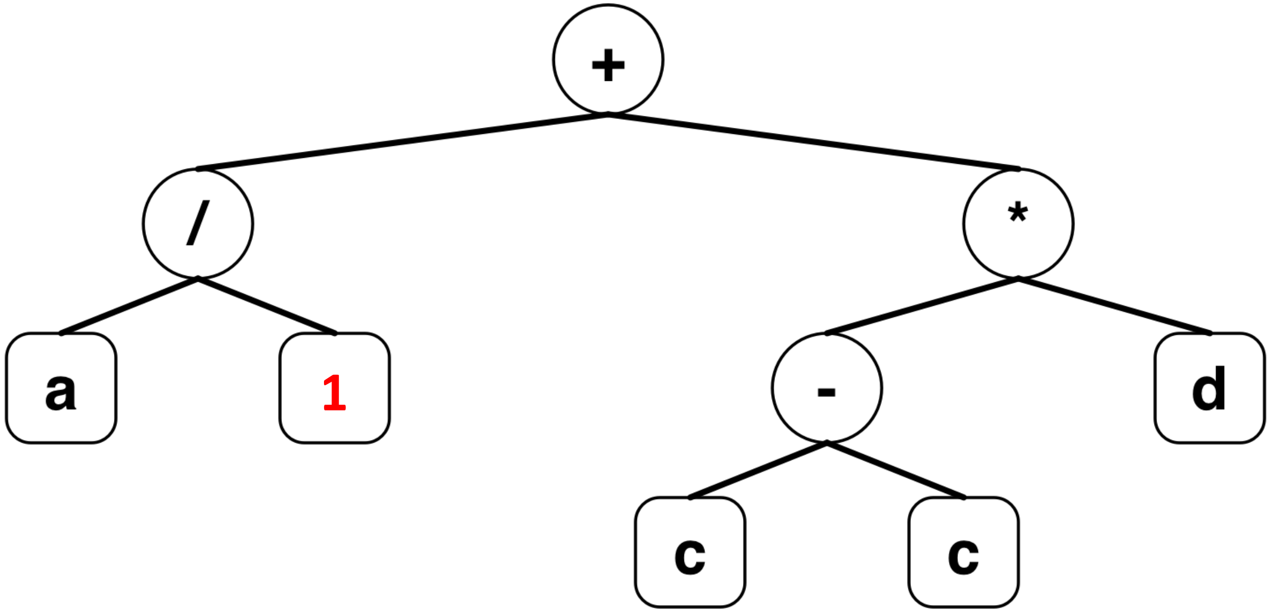




Chen, Q., Zhang, M., & Xue, B. (2017). Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression. IEEE Transactions on Evolutionary Computation, 21(5), 792-806.

GP with Feature Selection for Learning Scheduling Rules

- Run GP for 30 times, collect 30 best GP rules
- For each feature of each best rule, do a permutation test (set the feature to 1)
  - Calculate $obj(I_b)$: run simulations of the training set using I_b, calculate the objective values of the solutions
  - Calculate $obj(I_b|X_j = 1)$: replace all X_j to be 1 in I_b and rerun the simulations
  - $FI(X_j, I_b) = obj(I_b|X_j = 1) - obj(I_b)$
- Select the features with $FI(X_j, I_b) > 0$ for over 15 of the best GP rules
- Run GP again with the selected features



Fit(tree) = 0.9

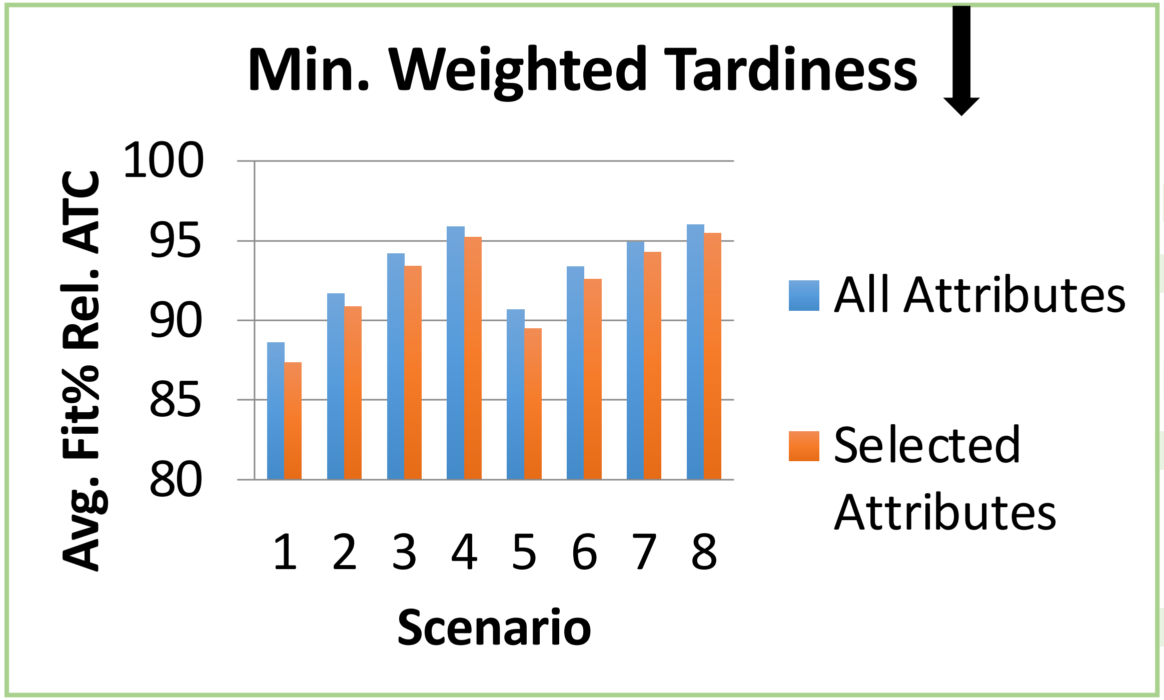
Fit(tree|b=1) = 1.1

Contribution(b) = 0.2

Contribution(c) = 0
Contribution(d) = 0

GP with Feature Selection for Learning Scheduling Rules

- Test on dynamic job shop scheduling problem
  - Minimise mean weighted tardiness
- Much best test performance
- Selected **6 out of the 16** features



Notation	Description
NOW	The current time.
PT	Processing time of the operation.
IPT	Inverse of the processing time.
NOPT	Processing time of the next operation.
ORT	Ready time of the operation.
MRT	Ready/Idle time of the machine.
NMRT	Ready time of the next machine.
WIQ	Work in the current queue.
WINQ	Work in the next queue.
NOIQ	Number of operations in the current queue.
NOINQ	Number of operations in next queue.
WKR	Work remaining (including the current operation).
NOR	Number of operations remaining.
FDD	Flow due date of the operation.
DD	Due date of the job.
W	Weight of the job.

Yi Mei, Mengjie Zhang, Su Nguyen, "Feature Selection in Evolving Job Shop Dispatching Rules with Genetic Programming," Genetic and Evolutionary Computation Conference (GECCO), Denver, USA, 2016.

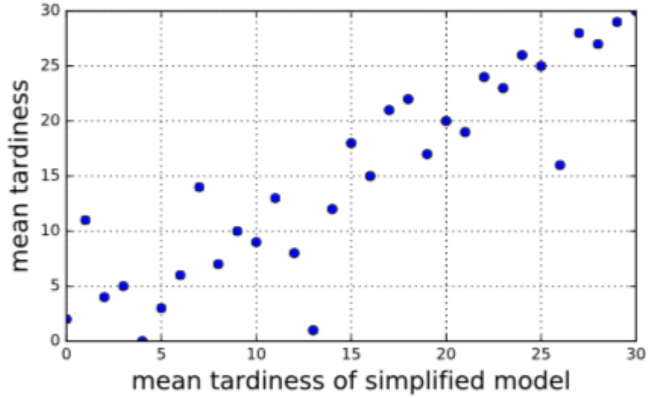
Two-Stage GP with Feature Selection

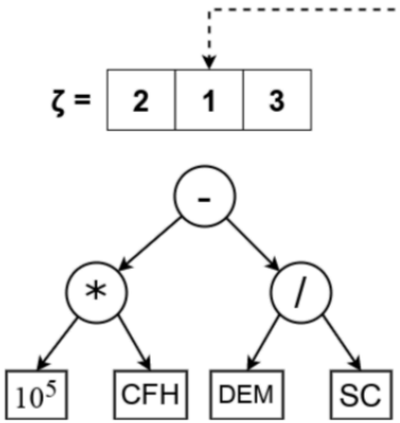
- Many GP runs are need to collect the data for feature selection
  - A diverse set of good GP models
- **Speed up** the process for data collection
  - A single run rather than multiple runs (use niching to obtain a diverse set)
  - Use surrogate (shorter simulations) to speed up evaluation
- **Stage 1**: run GP to get a diverse set of good models for feature selection
  - Use surrogate evaluation and niching
- Feature selection using the diverse set of good GP models
  - Permutation test
- **Stage 2**: run another GP with the selected features

Mei, Y., Nguyen, S., Xue, B., & Zhang, M. (2017). An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. IEEE Transactions on Emerging Topics in Computational Intelligence, 1(5), 339-353.

Two-Stage GP with Feature Selection

- Test on dynamic job shop scheduling problem
  - Minimise mean weighted tardiness
  - **Original** simulation: 2500 jobs, 10 machines
  - **Surrogate** simulation: 500 jobs, 5 machines
- Clearing for niching
  - Calculate a behaviour vector for each individual
  - Calculate distance between individuals based on their behaviour vector
  - For the individuals with the same behaviour vector, keep only the best-fit one (set others to worst fitness)



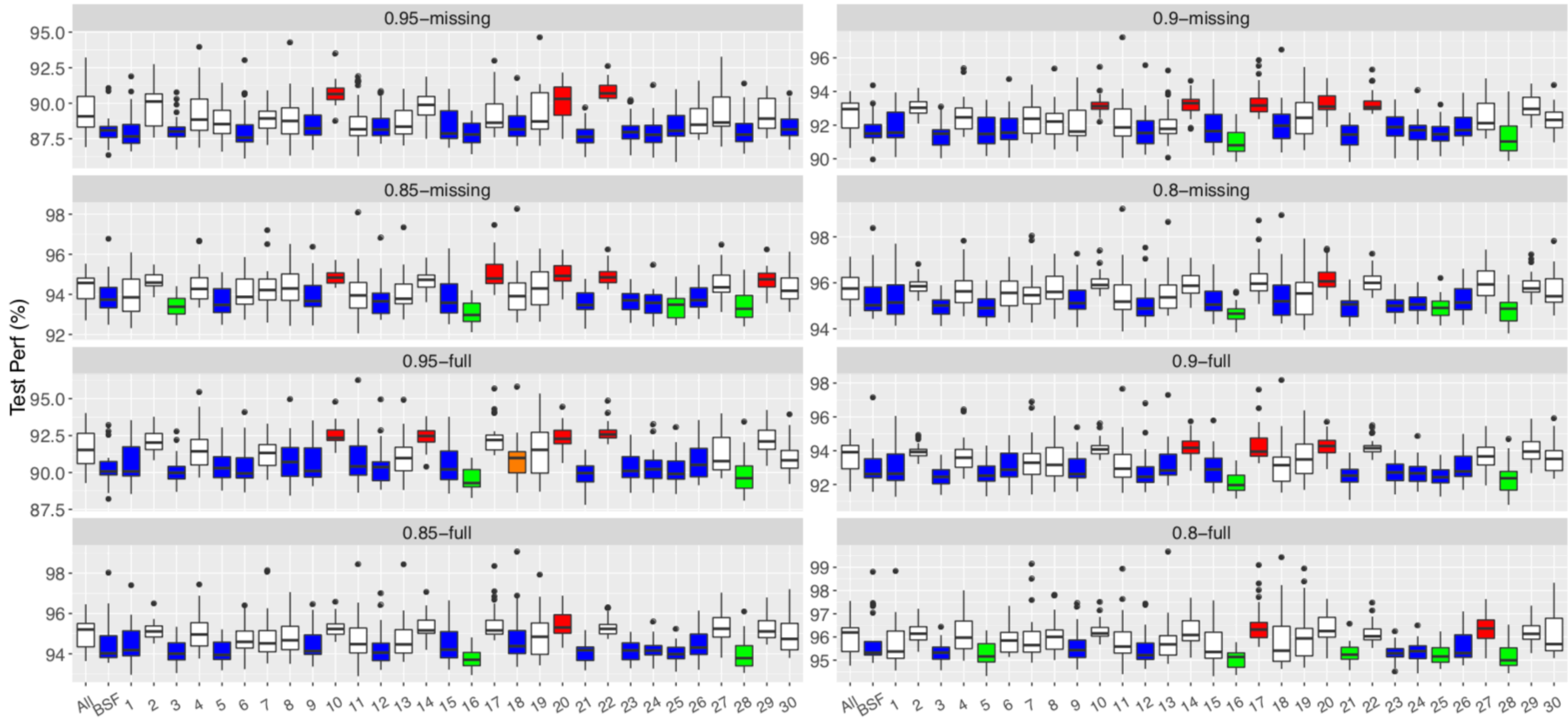


Tasks in Ω	Task Attributes *CFH*	*DEM*	*SC*	Ranking by Rule	Characterisation
Ω_{11}	5	2	8	2	2
Ω_{12}	9	2	3	**1**	
Ω_{21}	5	9	8	**1**	1
Ω_{22}	3	1	6	2	
Ω_{31}	6	4	9	3	3
Ω_{32}	7	5	1	2	
Ω_{33}	1	2	2	**1**	

Mei, Y., Nguyen, S., Xue, B., & Zhang, M. (2017). An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. IEEE Transactions on Emerging Topics in Computational Intelligence, 1(5), 339-353.

Two-Stage GP with Feature Selection

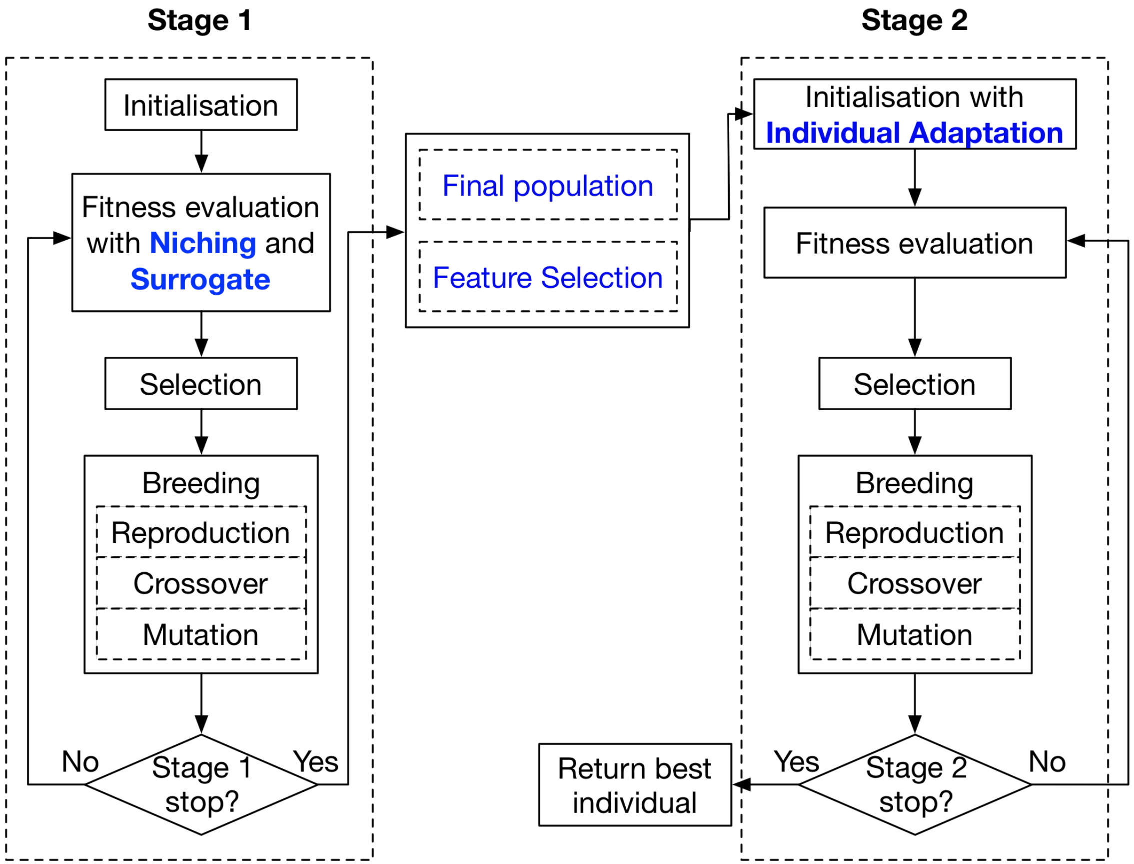
- Mostly better than "All features", no difference with "Best Feature Subset"
- Sometimes even better than the current "Best Feature Subset"
- Example selected set: {PT, NOPT, WINQ, NOINQ, W}



Mei, Y., Nguyen, S., Xue, B., & Zhang, M. (2017). An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. IEEE Transactions on Emerging Topics in Computational Intelligence, 1(5), 339-353.

Two-Stage GP with Feature Selection

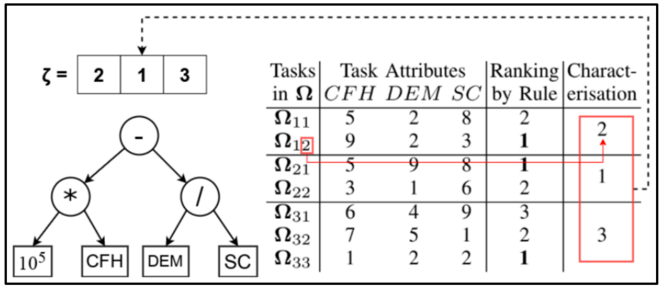
- Feature selection requires running GP to collect the good GP rules
- The good GP rules were **ONLY** used for calculating the feature importance, but ignored in the GP with the selected features
- **Stage 1**: run GP to get data for feature selection and final population
  - Niching and surrogate are used
- **Stage 2**: run another GP using the final population (adapt the individuals)

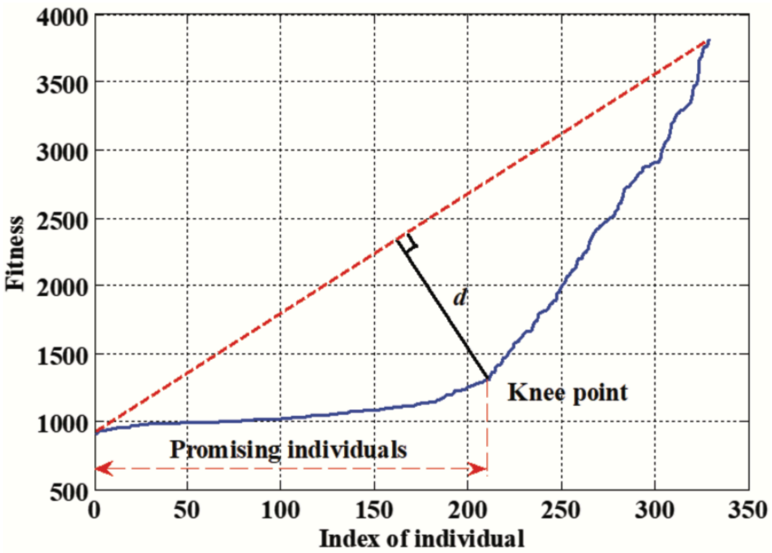


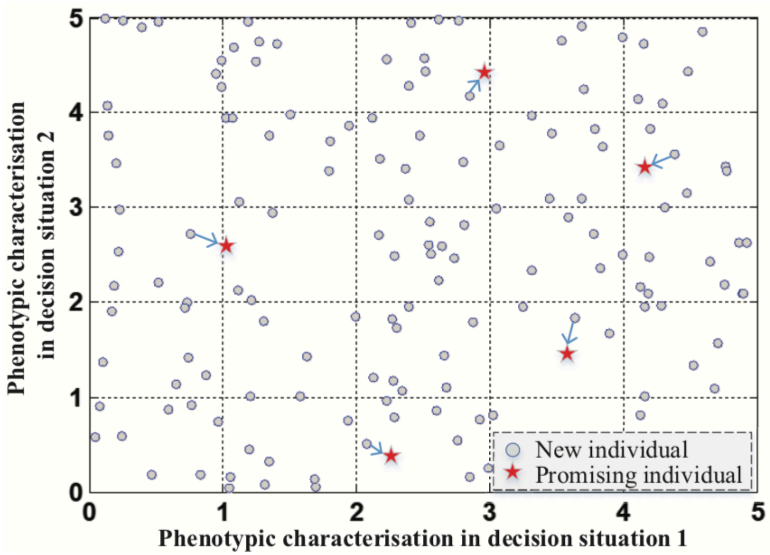
Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2020). Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. IEEE Transactions on Cybernetics.

Two-Stage GP with Feature Selection

- Stage 2 individual adaptation: adapt "promising" individuals, re-initialize the remaining
- Use "knee point" to detect "promising" individuals
- Two adaptation strategies
  - **Replace** the unselected features by 1
  - **Mimicking** behaviour
    - Randomly generate many individuals with the selected features
    - For each promising final individual, replace with the newly generated individual with the most similar behaviour



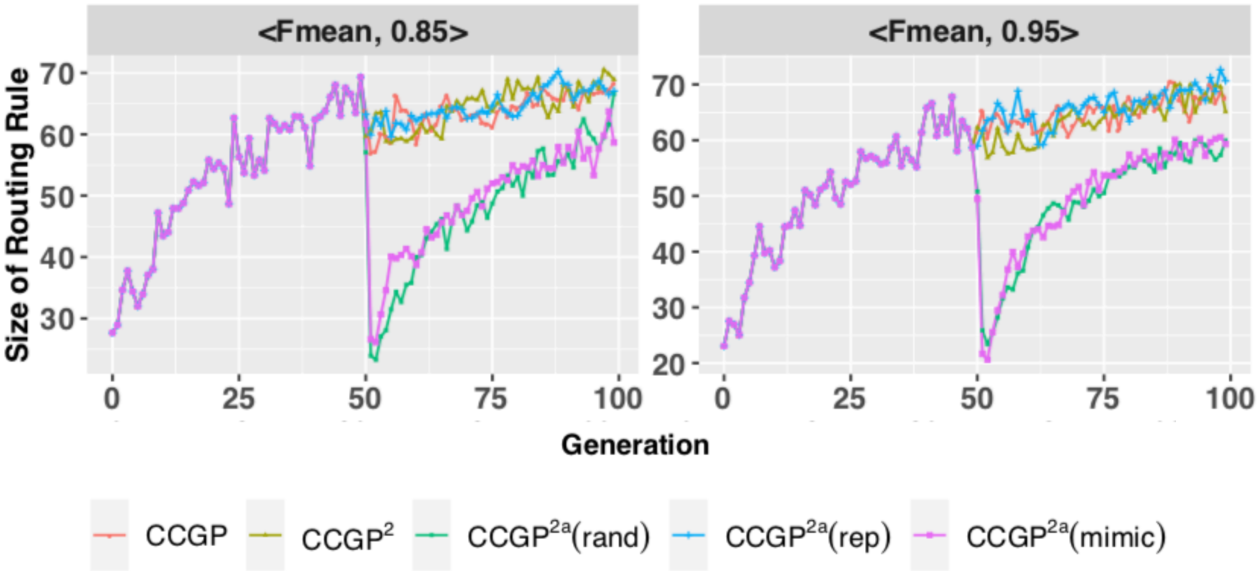




Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2020). Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. IEEE Transactions on Cybernetics.

Two-Stage GP with Feature Selection

- Test on dynamic flexible job shop scheduling
  - Cooperative Co-evolution GP to co-evolve routing and sequence rules
- Almost the same test performance
- Much smaller rule size and number of used features (mimic version)



#features used in sequencing rule

Scenario	CCGP	CCGP2(mimic)
Fmax,0.85	7.13	**5.20**
Fmax,0.95	7.40	**5.17**
Fmean,0.85	6.57	**3.70**
Fmean,0.95	6.90	**3.70**
WFMean,0.85	6.53	**4.00**
WFMean,0.95	6.80	**4.27**

Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2020). Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. IEEE Transactions on Cybernetics.

Two-Stage GP with Feature Selection

- Example evolved rules are simpler

With feature selection

$$R_1 = min\{2 * NIQ, max(\frac{NIQ * PT}{MWT}, PT * WIQ * min(WIQ, WKR))\} + NIQ * \frac{PT}{W} - MWT$$

$$S_1 = \frac{PT + WKR}{W} - \frac{W}{PT}(W * WIQ - W + WIQ) - \frac{W}{PT} * (W * WIQ + WKR + \frac{PT + WKR}{W * WIQ - W + WKR})$$

Without feature selection

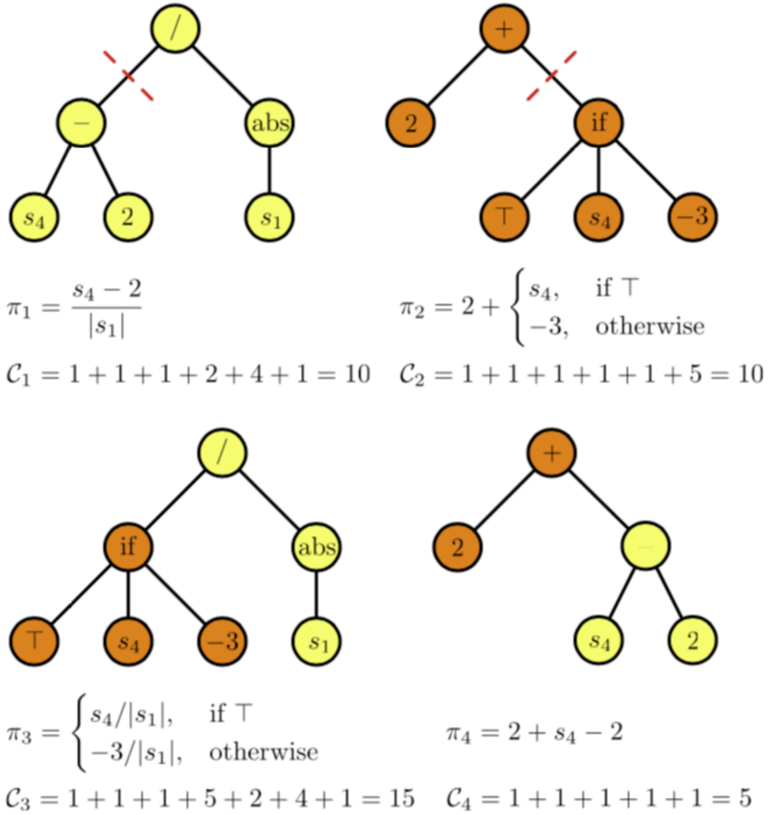
$$R_2 = max\{NIQ^2, (NIQ + NPT) * min(NIQ, NOR)\} - min(MWT, \frac{WKR}{MWT * WKR - 1}) * max\{WKR, -MWT * WKR + \frac{NIQ * PT * max\{WIQ, \frac{min(MWT, PT)}{(W + WKR)}\}}{max\{NIQ^2, \frac{NOR - W + WKR}{W}, \frac{NIQ + NPT}{(min(NIQ, NOR))^{-1}}\}}\}$$

$$S_2 = NIQ(PT - W)(PT + \frac{WKR}{W} * max\{PT, \frac{WIQ}{W}, \frac{max\{WIQ, \frac{WKR}{W}\}}{W}\}) + max\{\frac{WIQ}{W^2}, \frac{NIQ}{W^2} + WIQ\} * (MWT + W + max\{\frac{WKR}{W^2}, NIQ - 1\}) * \frac{max\{WIQ, \frac{WKR}{W}\}}{W}$$

Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2020). Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. IEEE Transactions on Cybernetics.

GP with Model Complexity

- Different operators/functions have different complexities
- Multi-objective GP (effectiveness vs complexity)



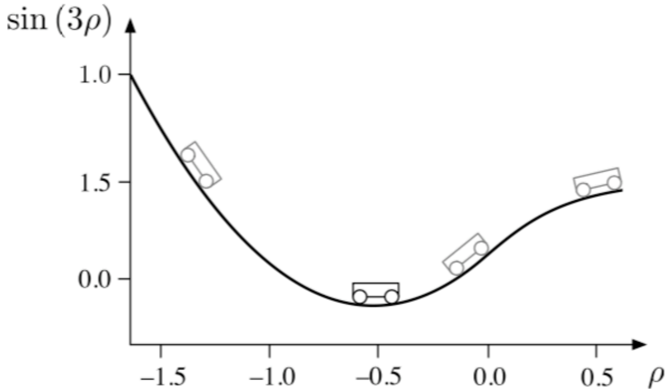
Complexities

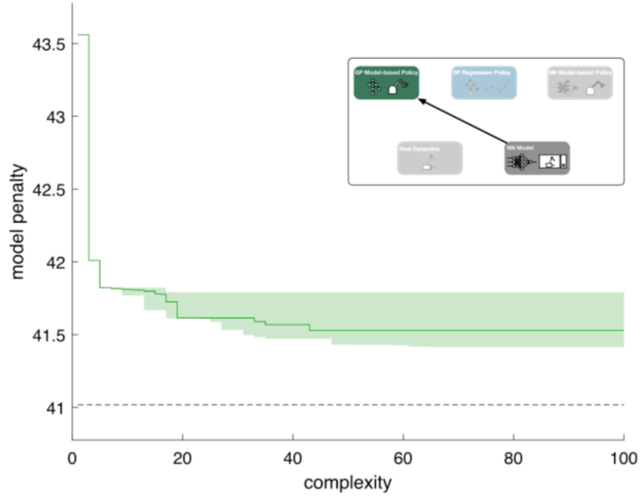
Terminals	1
+, -, x	1
/	2
AND, OR	4
Tanh, abs	4
If	5

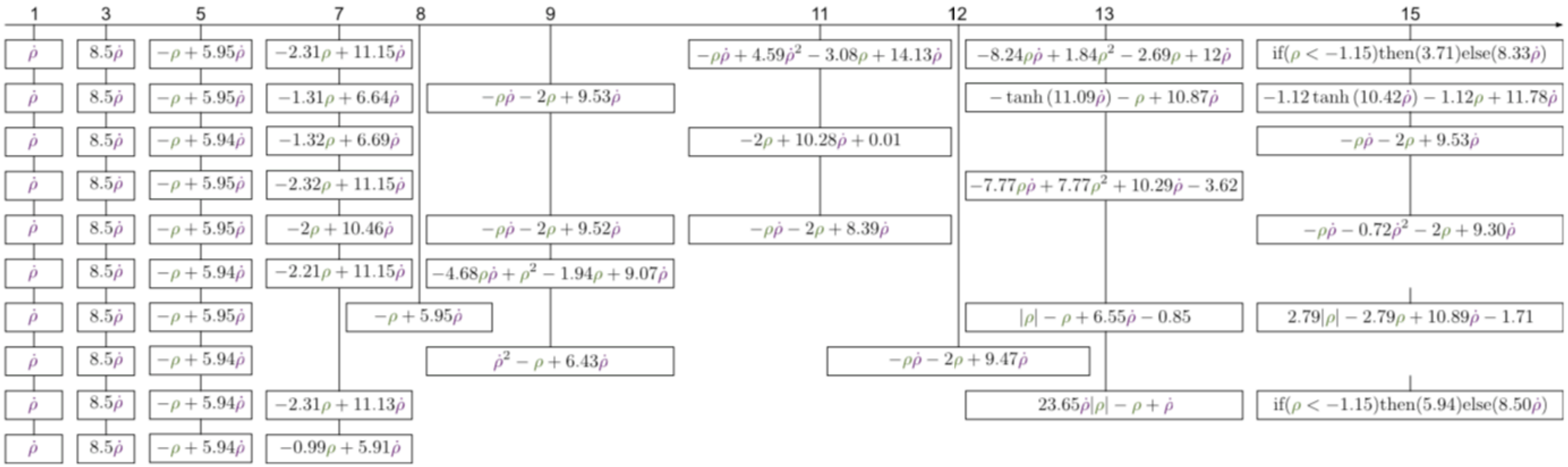
Hein, Daniel, Steffen Udluft, and Thomas A. Runkler. "Interpretable policies for reinforcement learning by genetic programming." Engineering Applications of Artificial Intelligence 76 (2018): 158-169.

GP with Model Complexity

- Results on learning Mount Car policies



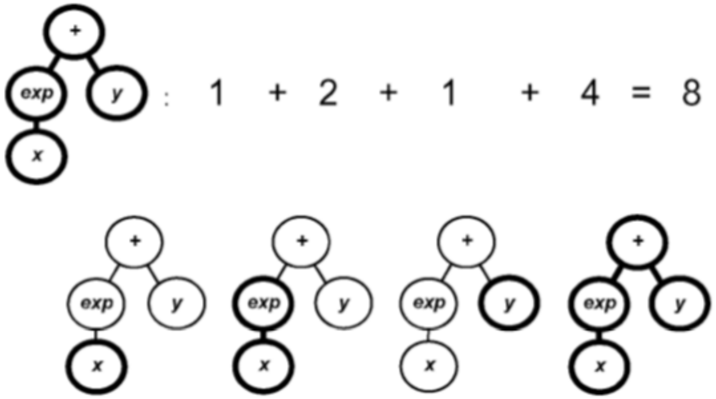




Hein, Daniel, Steffen Udluft, and Thomas A. Runkler. "Interpretable policies for reinforcement learning by genetic programming." Engineering Applications of Artificial Intelligence 76 (2018): 158-169.

GP with Model Complexity

- Expressional complexity: total number of nodes in all the subtrees
  - Prefer flatter trees rather than deeper trees
  - Fewer nested functions
- Order of Nonlinearity complexity

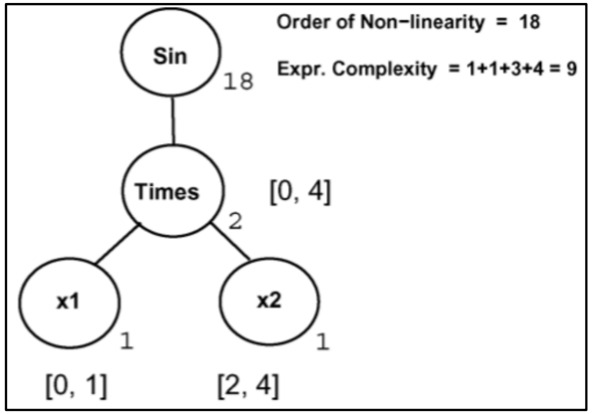


Node	Complexity
Constant	0
Variable	1
$f \circ g$	$Comp(g) * n_f$
$g_1 + g_2$ and $g_1 - g_2$	$\max(Comp(g_1), Comp(g_2))$
$g_1 * g_2$	$Comp(g_1) + Comp(g_2)$
g_1 / g_2	$Comp(g_1) + Comp(g_2) * n_{div}$

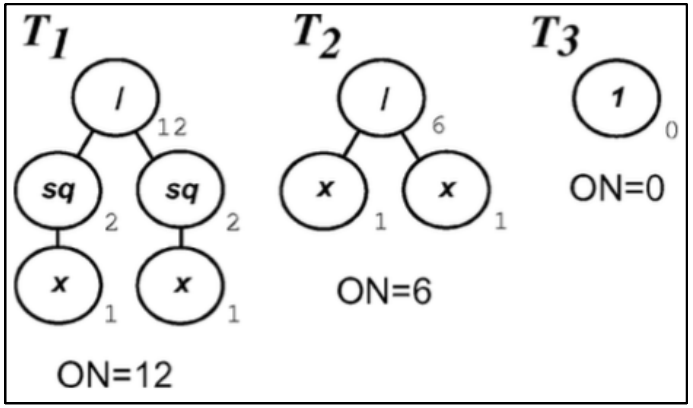
Vladislavleva, E. J., Smits, G. F., & Den Hertog, D. (2008). Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Transactions on Evolutionary Computation, 13(2), 333-349.

GP with Model Complexity

- Order of Nonlinearity complexity
  - n_f is the minimum degree of the Chebyshev polynomial approximation of $f(\cdot)$
  - n_{div} is the minimum degree of the Chebyshev polynomial approximation of $1/x$ in its range
- Chebyshev polynomial approximation
  - $\max_{x \in S \subseteq [a,b]} \left| f(x) - \sum_{i=0}^{n-1} c_i T_i(x;a,b) \right| \leq \epsilon$
  - $T_i(x;a,b) = T_i\left(\frac{2x-(b+a)}{b-a}\right)$
  - T_i is the Chebyshev polynomial



Node	Complexity
$f \circ g$	$Comp(g) * n_f$
g_1/g_2	$Comp(g_1) + Comp(g_2) * n_{div}$



Vladislavleva, E. J., Smits, G. F., & Den Hertog, D. (2008). Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Transactions on Evolutionary Computation, 13(2), 333-349.

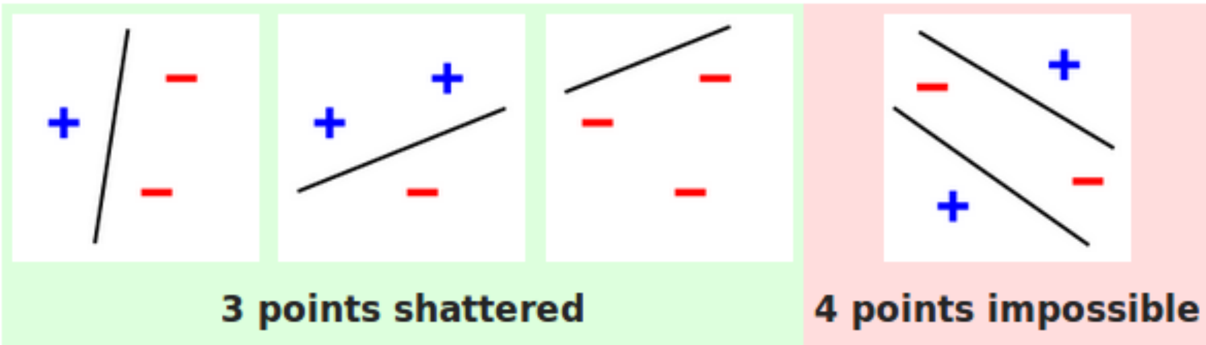
GP with Model Complexity

- Measure model complexity based on statistical learning theory
  - **VC dimension**: the capacity (complexity, expressive power, richness, or flexibility) of a space of functions that can be learned by a statistical classification algorithm.
  - How many points this family of functions can shatter?
- Structural risk minimization as fitness

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{1}{N}\left[h\left(\log\left(\frac{2N}{h}\right)+1\right) - \log\left(\frac{\eta}{4}\right)\right]}$$

Training error

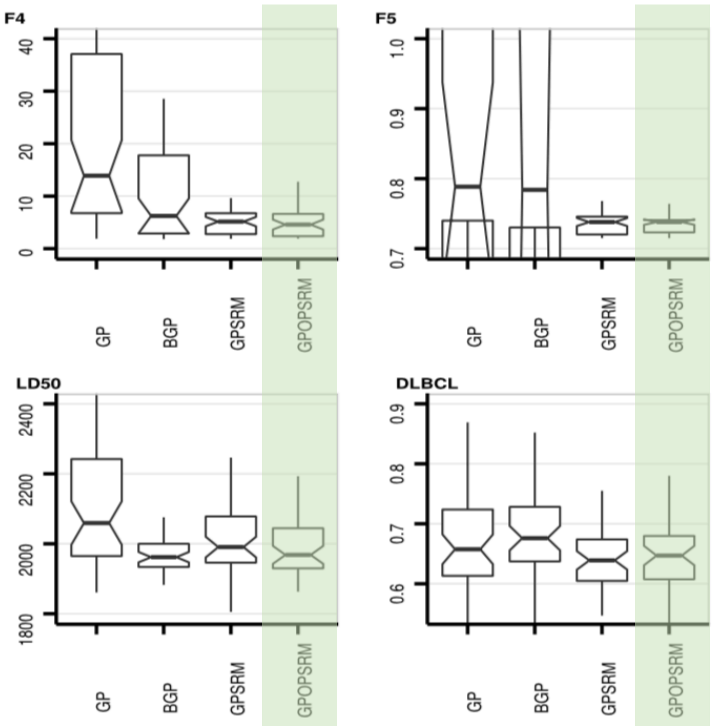
VC dimension (**empirically estimated**)



Chen, Q., Zhang, M., & Xue, B. (2018). Structural risk minimization-driven genetic programming for enhancing generalization in symbolic regression. IEEE Transactions on Evolutionary Computation, 23(4), 703-717.

GP with Model Complexity

- Among the best test error, more compact/interpretable model



	Method	Evolved Model	Simplified Model	Evolved Model	Simplified Model
f_6	GP	$(+(+(0-(\mathrm{sqrt}(*(0-(\mathrm{sqrt}(*(*-4.604x_2)x_1)))x_1)))(*(\mathrm{sqrt}(+(\%1.0(*(\mathrm{sqrt}(+-4.604(*-10.18x_1)))(*(*(\%1.0x_2)x_1)(+x_2x_1))))(*-10.182x_1)))(*(*(*(*0.343x_1)(\%1.0x_2))x_1)(0-(+(0-x_2)(*-4.604x_2))))))(*0.343(*(*(*0.343x_1)(\%1.0x_2))x_1)))$	$1.92x_1^2*\left(\frac{1}{\left(x_1+\frac{x_1^2}{x_2}\right)*(-10.18x_1+4.6)^{1/2}}-10.18x_1\right)^{1/2}+\frac{0.117x_1^2}{x_2}-(-x_1-(-4.6x_1x_2)^{1/2})^{1/2}$	$(+(+(0-x_2)(*(0-(*x_1x_1))(+(0-(*x_1x_1))x_1)))(\mathrm{sqrt}(+(0-(+(0-x_2)2.093))(+(+(0-(+(+(0-x_2)(+(0-x_2)2.093))2.093))(+(0-(+(0-x_2)(+(0-x_2)2.093)))(0-(0-x_2))))(\mathrm{sqrt}(0-(+(0-x_2)(+(0-x_2)2.093))))))))$	$x_1^4-x_1^3-x_2+(6x_2-8.372)^{1/2}+(2x_2-2.093)^{1/2}$
	BGP	$(+(+(*(0--0.332)(\mathrm{sqrt}(0--0.332)))-0.95)(+(*(+(\mathrm{sqrt}(0-x_2))(*-0.978x_1))(+(*x_1x_1)x_1))(+(+(+(+(*(*x_1x_1)(*x_1x_1))(*-0.978x_1))(\mathrm{sqrt}x_2))(0-x_2))(\mathrm{sqrt}x_2))))$	$x_1^4-0.978x_1^3-0.978x_1^2-0.978x_1+2\sqrt{x_2}-x_2+0.758$	$(+(*(*x_1x_1)(\mathrm{sqrt}(+(+(+(*(*x_1x_1)(*x_1x_1))(*(\%1.0x_1)(+(\%1.0x_1)x_2)))(*(\%1.0x_1)(0-x_1)))(*(\%1.0x_1)(+x_1x_2)))))(*(0-x_1)(+(\mathrm{sqrt}(+(*(*x_1x_1)(*x_1x_1))(*(\%1.0x_1)(+x_1x_2))))(\mathrm{sqrt}(\%1.0(+(+(0--0.456)(+(\%1.00.191)(+x_1x_2)))(+x_2(+(*x_1x_1)(+x_1x_2))))))))$	$x_1*(x_1^6+2x_1x_2+1)^{1/2}-x_1*(x_1^4+x_2/x_1+1)^{1/2}$
	GPSRM	$(+(0-(*(*x_1x_1)x_1))(+(\mathrm{sqrt}(+(*(0-(+x_2x_2))(\mathrm{sqrt}(+(0-(+(+x_2x_2)x_2))(*(0-(+x_2x_2))(*x_1x_1)))))(+x_2x_2)))(*(*x_1x_1)(*x_1x_1))))$	$x_1^4-x_1^3+\sqrt{2x_2\left(1-\sqrt{x_2(-2x_1^2-3)}\right)}$	$(+(+(0-(*(\mathrm{sqrt}x_1)x_2))(*(*x_1x_1)(0-x_1)))(+(*(*x_1x_1)(*x_1x_1))(+(\mathrm{sqrt}(*(*(\mathrm{sqrt}(*(\mathrm{sqrt}x_1)x_2))x_2)x_2))(+(*(*(+(0-(*(\mathrm{sqrt}x_1)x_2))(*(*x_1x_1)(0-x_1)))x_2)(+(0-x_1)x_1))(+(0-x_1)x_1)))))$	$x_1^4-x_1^3-x_2\sqrt{x_1}+\sqrt{(x_2)^2\sqrt{x_2\sqrt{x_1}}}$
	GPOPSRM	$(+(\mathrm{sqrt}(+(\mathrm{sqrt}(*(0-(+x_2(+x_2x_2)))(+(*(*(+x_2x_2)x_2)x_2)(*x_2(+x_2x_2)))))(\mathrm{sqrt}(*x_1x_1)))))(0-(\mathrm{sqrt}(0-(*(+x_2x_2)(+x_2x_2)))))(*(+(0-x_1)(*x_1x_1))(*x_1x_1)))$	$x_1^4-x_1^3+\sqrt{\sqrt{-3x_2(x_1+6(x_2)^2)}}$	$(+(*(*(\mathrm{sqrt}(*(*(0--0.192)(0--0.192))x_2)(*(0--0.192)(0--0.192))))x_2)x_2)(+(*x_1(*(*x_1x_1)x_1))(0-(*(*x_1x_1)x_1))))$	$x_1^4-x_1^3+0.036x_2^{5/2}$

Chen, Q., Zhang, M., & Xue, B. (2018). Structural risk minimization-driven genetic programming for enhancing generalization in symbolic regression. IEEE Transactions on Evolutionary Computation, 23(4), 703-717.

Dimensionally-Aware GP

- The combination should be dimensionally consistent
  - E.g. Time + Distance is meaningless
- Use grammar to keep dimensional consistency

An example grammar to construct features for a physics (Higgs) dataset

```
<start> ::= <E> | <A> | <F>
<E> ::= <E> + <E> | <E> - <E> | <E> * <F> | <E> / <F> |
        sqrt(<E2>) | <termE>
<A> ::= <A> + <A> | <A> - <A> | <A> * <A> | Acos(<F>) |
        Atan(<F>) | <termA>
<F> ::= <F> + <F> | <F> - <F> | <F> * <F> | <E> / <E> |
        <A> / <A> | <F> / <F> | cos(<A>) | sin(<A>) |
        tan(<A>) | <termF>
<E2> ::= <E2> + <E2> | <E2> - <E2> | <E2> * <F> |
         <E2> / <F> | <E> * <E> | <termE2>

E: energy;
E2: squared energy;
A: angle
F: float
termX: constant of type/dimension X
```

Evolved features

$$\cos\left(\phi^{lep} - \phi^{\tau}\right)$$

$$\cos\left(\theta^{lep} - \theta^{\tau}\right)$$

$$\cos\left(\phi^{missing} - \phi^{lep}\right)$$

$$p_T^{leading} \sum p_T^{jets} - \left(E_T^{missing} + p_T^{lep}\right)^2$$

$$m_{H^0}^2 + \left(p_T^{lep} + p_T^{\tau}\right)^2$$

Cherrier, N., Poli, J. P., Defurne, M., & Sabatié, F. (2019, June). Consistent feature construction with constrained genetic programming for experimental physics. In 2019 IEEE Congress on Evolutionary Computation (CEC) (pp. 1650-1658). IEEE.

Dimensionally-Aware GP

- Each GP node has a dimensionality vector
  - Each dimension indicates the exponent of the corresponding unit of measurement
  - E.g., [0,0,1] means the dimension of mass
- The vector is changed/propagated by functions

Function	Operand Dimensionality	Result
Exponentiation:	[0,0,0]	[0,0,0]
Logarithm:	[0,0,0]	[0,0,0]
Square Root:	[x,y,z]	[x/2, y/2, z/2]
Addition:	[x,y,z], [x,y,z]	[x,y,z]
Subtraction:	[x,y,z], [x,y,z]	[x,y,z]
Multiplication:	[x,y,z], [u,v,w]	[x + u ,y + v, z + w]
Division:	[x,y,z], [u,v,w]	[x - u ,y – v, z - w]
Power:	[0,0,0], [0,0,0]	[0,0,0]
PowScalar (c):	[x,y,z]	[x*c, y*c, z*c]
If less than zero:	[0,0,0], [x,y,z], [x,y,z]	[x,y,z]

Keijzer, M., & Babovic, V. (1999, July). Dimensionally aware genetic programming. In Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2 (pp. 1069-1076).

Dimensionally-Aware GP

- Each GP node has a dimensionality vector
  - Each dimension indicates the exponent of the corresponding unit of measurement
  - E.g., [0,0,1] means the dimension of mass
- The vector is changed/propagated by functions

Function	Operand Dimensionality	Result
Exponentiation:	[0,0,0]	[0,0,0]
Logarithm:	[0,0,0]	[0,0,0]
Square Root:	[x,y,z]	[x/2, y/2, z/2]
Addition:	[x,y,z], [x,y,z]	[x,y,z]
Subtraction:	[x,y,z], [x,y,z]	[x,y,z]
Multiplication:	[x,y,z], [u,v,w]	[x + u ,y + v, z + w]
Division:	[x,y,z], [u,v,w]	[x - u ,y – v, z - w]
Power:	[0,0,0], [0,0,0]	[0,0,0]
PowScalar (c):	[x,y,z]	[x*c, y*c, z*c]
If less than zero:	[0,0,0], [x,y,z], [x,y,z]	[x,y,z]

Dimension transformation to resolve dimension violation

Function	Operand Dimensionality	Result
DimTransform c[x,y,z]:	[u,v,w]	[x + u ,y + v, z + w]

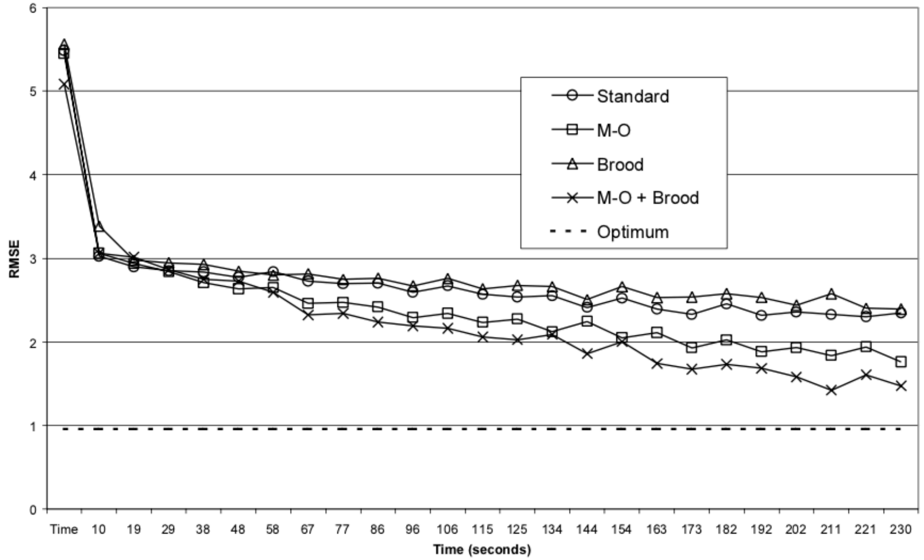
Keijzer, M., & Babovic, V. (1999, July). Dimensionally aware genetic programming. In Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2 (pp. 1069-1076).

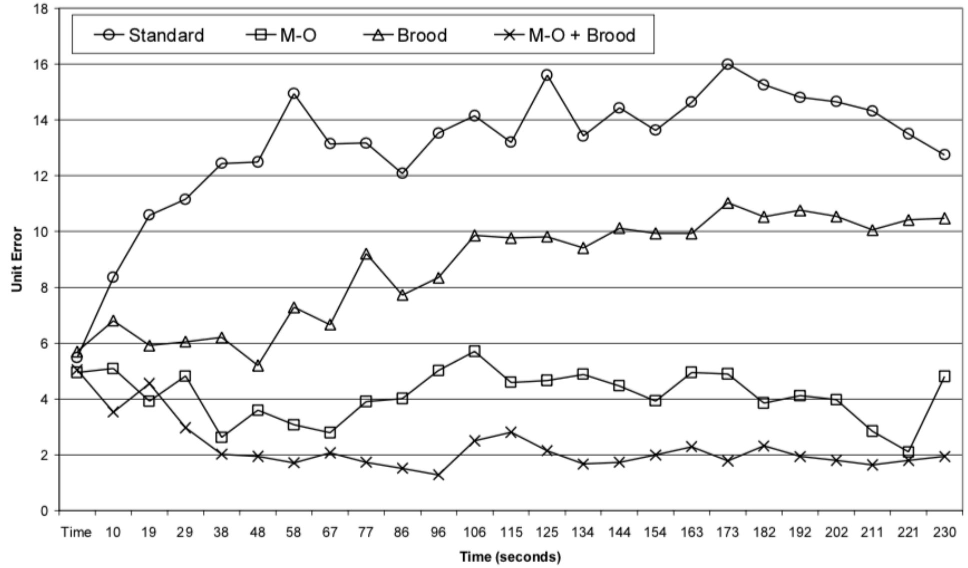
Dimensionally-Aware GP

- **CullingGP**: Dimensionality-Aware Breeding
  - Select two parents by tournament selection (the same as standard GP)
  - Generate many offspring (> 2)
  - Select the offspring with the best goodness-of-dimension

$$\textit{Goodness-of-Dimension} = \sum |x_i| + |y_i| + |z_i|$$

- Multi-objective (error vs goodness-of-dimension)
- Better test performance for noise data, better dimension violation





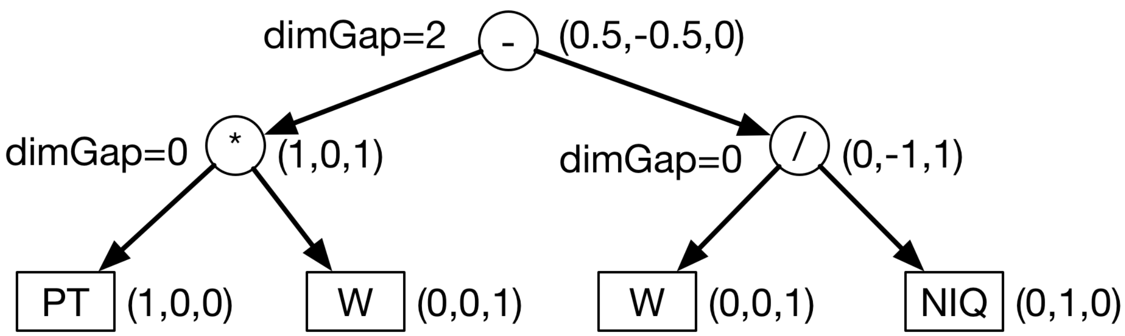
Keijzer, M., & Babovic, V. (1999, July). Dimensionally aware genetic programming. In Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2 (pp. 1069-1076).

Dimensionally-Aware GP for Scheduling Rules

- The job shop scheduling state features have three dimensions
  - **T**IME: processing time, due date, slack, …
  - **C**OUNT: number of remaining operations, number of jobs in the queue, …
  - **W**EIGHT: the weight (importance) of a job
- Dimensionality vector (T, C, W)
- Minimise Dimension gap:

$$\texttt{dimGap}(\texttt{node}) = \begin{cases} 0, & \text{if } \texttt{node} = \times \text{ or } / \\ \delta(\boldsymbol{\theta}(\texttt{c}_1), \boldsymbol{\theta}(\texttt{c}_2)), & \text{otherwise.} \end{cases}$$

$$\texttt{dimGap}(\texttt{tree}) = \sum_{\texttt{node} \in \texttt{tree}} \texttt{dimGap}(\texttt{node}),$$



Function(s)	Children Vector Values	Result
$+$, $-$, max and min	(T_1, C_1, W_1), (T_2, C_2, W_2)	$\left(\frac{T_1+T_2}{2}, \frac{C_1+C_2}{2}, \frac{W_1+W_2}{2}\right)$
$\times$	(T_1, C_1, W_1), (T_2, C_2, W_2)	$(T_1 + T_2, C_1 + C_2, W_1 + W_2)$
$/$	(T_1, C_1, W_1), (T_2, C_2, W_2)	$(T_1 - T_2, C_1 - C_2, W_1 - W_2)$

Mei, Y., Nguyen, S., & Zhang, M. (2017, November). Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling. In Asia-Pacific Conference on Simulated Evolution and Learning (pp. 435-447). Springer, Cham.

Dimensionally-Aware GP for Scheduling Rules

- Constrained DAGP
  - $fit(x) = obj(x) + \alpha(t) * dimGap(x)$
- The penalty is adaptive based on balance between dimGap and obj in the population
  - $\alpha(0) = -\frac{cov(dimGap(pop_0), obj(pop_0))}{var(dimGap(pop_0))}$
  - $\alpha(t+1) = \alpha(t) - \eta * \left(\frac{cov(dimGap(pop_t), obj(pop_t))}{var(dimGap(pop_t))} + \alpha(t)\right)$
  - $\eta = 0.01$ is the learning rate
- Dimensions of the JSS terminals/state features

Notation	Description	Dimension
WIQ	Work In Queue	TIME
MWT	Machine Waiting Time	TIME
PT	Processing Time	TIME
NPT	Next Processing Time	TIME
OWT	Operation Waiting Time	TIME
NWT	Next Machine Waiting Time	TIME
WKR	Work Remaining	TIME
WINQ	Work In Next Queue.	TIME
rFDD	Relative FDD	TIME
rDD	Relative DD	TIME
TIS	Time In System	TIME
SL	Slack	TIME
NIQ	Number of operations In Queue	COUNT
NOR	Number of Operations Remaining	COUNT
NINQ	Number of operations In Next Queue	COUNT
W	Weight	WEIGHT

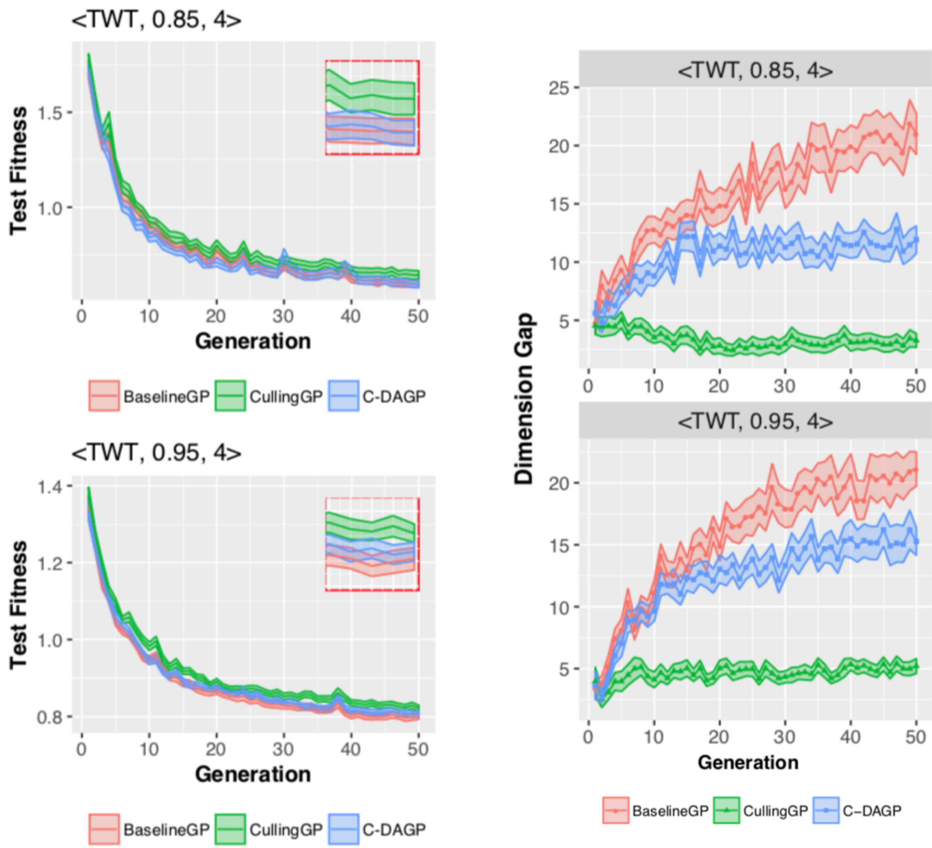
Mei, Y., Nguyen, S., & Zhang, M. (2017, November). Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling. In Asia-Pacific Conference on Simulated Evolution and Learning (pp. 435-447). Springer, Cham.

Dimensionally-Aware GP for Scheduling Rules

- **Similar performance** with the baseline GP (much better than CullingGP)
- Much smaller dimension violation than the baselineGP (larger than CullingGP)

An evolved rule for minimising TWT

```
rule = B1/B2
B1  = max((SL+PT)*max(min(SL,
      WINQ),PT)/WKR,PT)
B2  = W*WKR/(max((SL+PT),WKR)
      *max(W,PT))
```



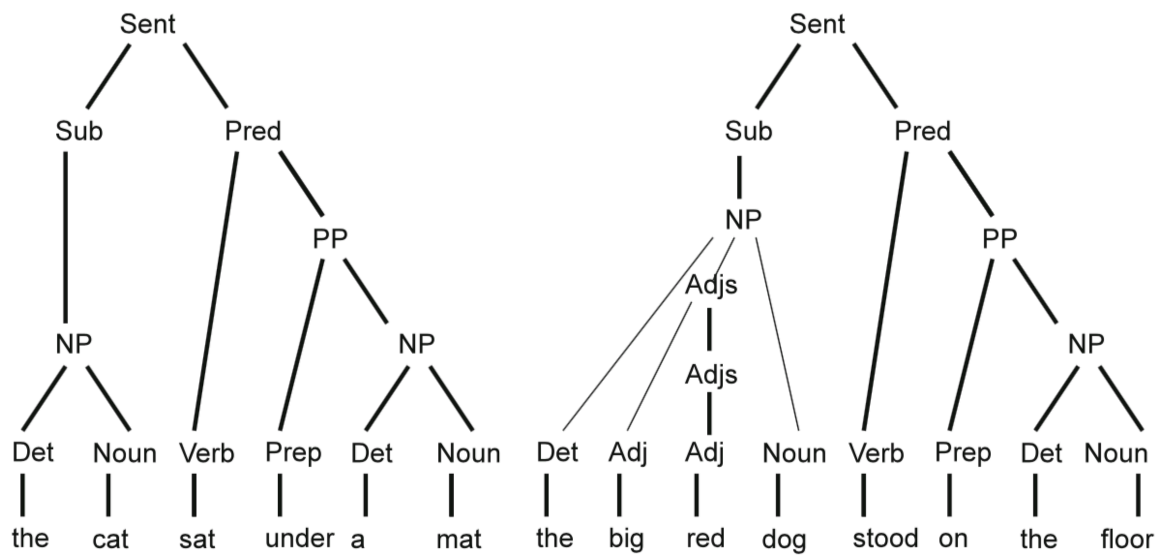
Mei, Y., Nguyen, S., & Zhang, M. (2017, November). Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling. In Asia-Pacific Conference on Simulated Evolution and Learning (pp. 435-447). Springer, Cham.

Grammar Guided GP

- Define the meaningful combinations (can include dimensionality consistency) in the GP tree

Table 1 English grammar fragment

Sent → Sub Pred	PP → Prep NP	Prep → ''on''|''under''
Sub → NP	Adjs → Adj Adjs	Noun → ''cat''|''dog''
Pred → Verb PP	Adjs → Adj	|''floor'' |''mat''
NP → Det Noun	Verb → ''sat''|''stood''	Adj → ''big''|''small''
NP → Det Adjs Noun	Det → ''a''|''the''	|''red'' |''black''



McKay, R. I., Hoai, N. X., Whigham, P. A., Shan, Y., & O'neill, M. (2010). Grammar-based genetic programming: a survey. Genetic Programming and Evolvable Machines, 11(3), 365-396.

Grammar Guided GP

- Grammar for rational polynomials

Exp → Poly/Poly	Trm → Coef * Prod	Coef → ‘‘x0’’|‘‘x1’’
Poly → Trm	Prod → Var	Var → ‘‘v0’’|‘‘v1’’
Poly → Trm + Poly	Prod → Var * Prod	

- Grammar for ideal gas law

Exp → Trm	Trm → Trm Mul Trm	Add → ‘‘+’’|‘‘ − ’’
Exp → Trm Add Trm	Trm → Var	Mul → ‘‘×’’|‘‘/’’
	Trm → Const	Var → ‘‘T’’|‘‘V’’

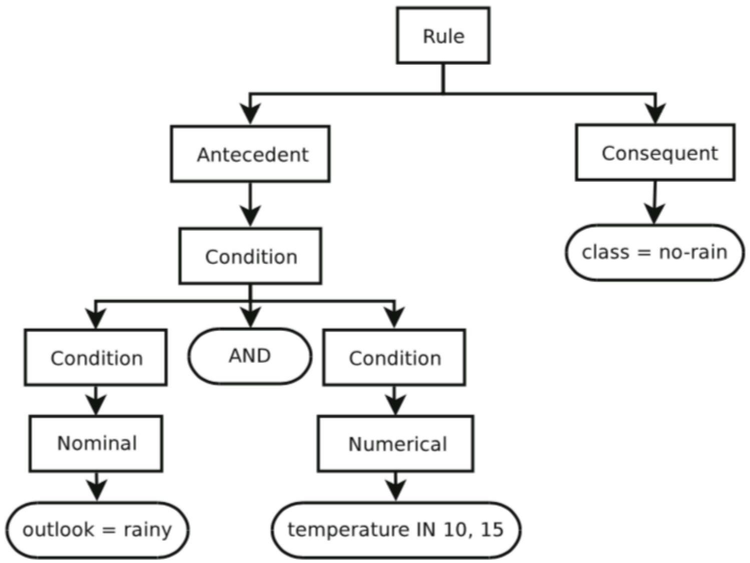
- Crossover and mutation respect the grammar
  - Swap subtrees with the same type

McKay, R. I., Hoai, N. X., Whigham, P. A., Shan, Y., & O’neill, M. (2010). Grammar-based genetic programming: a survey. Genetic Programming and Evolvable Machines, 11(3), 365-396.

Grammar Guided GP for Association Rule Mining

- **Items** $\mathcal{I} = \{i_1, i_2, \dots, i_n\}$, **transactions** $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$, $t_j \subseteq \mathcal{I}$ is a subset of $\mathcal{I}$
- **Association rule** $X \rightarrow Y, X \subset \mathcal{I}, Y \subset \mathcal{I}, X \cap Y = \emptyset$
- If the **antecedent** $X \subset t_j$, then highly likely that the **consequent** $Y \subset t_j$ as well
- $Support(X)$: number of transactions containing X
- $Support(X \rightarrow Y)$: number of transactions containing both X and Y
- $Condifence(X \rightarrow Y) = \frac{support(X \rightarrow Y)}{support(X)}$
- $fit(R = X \rightarrow Y) = \frac{support(R)}{support(X)} * \frac{support(R)}{support(Y)}$

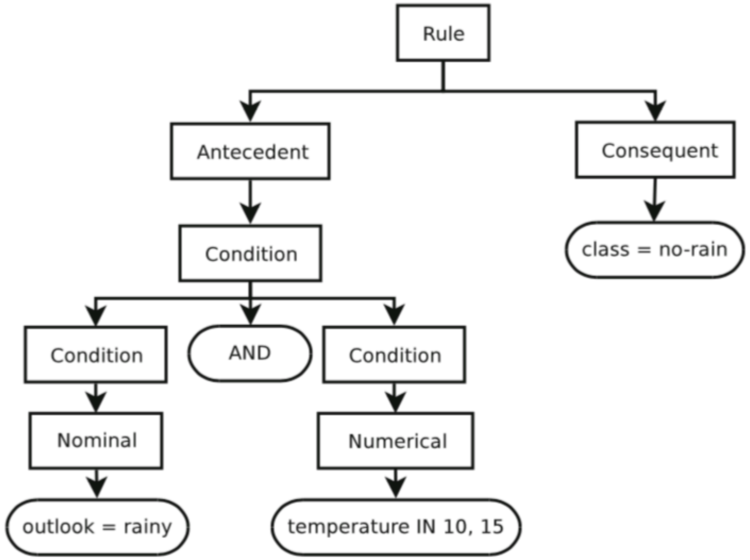
```
<Rule> ::= <Antecedent>, <Consequent>
<Antecedent> ::= <Condition> (<Condition> AND <Condition>)*
<Consequent> ::= class=value
<Condition> ::= <Numerical> | <Nominal>
<Numerical> ::= name IN Min_value, Max_value
<Nominal> ::= name=value
```



Padillo, F., Luna, J. M., & Ventura, S. (2019). A grammar-guided genetic programming algorithm for associative classification in big data. Cognitive Computation, 11(3), 331-346.

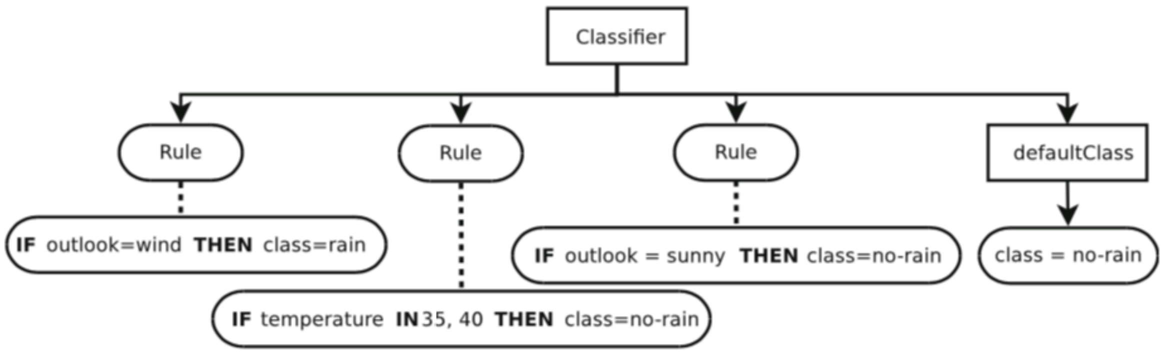
Grammar Guided GP for Association Rule Mining

- Step 1: Rule extraction
- Step 2: Rule selection




```
<Rule> ::= <Antecedent>, <Consequent>
<Antecedent> ::= <Condition> (<Condition> AND <Condition>)*
<Consequent> ::= class=value
<Condition> ::= <Numerical> | <Nominal>
<Numerical> ::= name IN Min_value, Max_value
<Nominal> ::= name=value
```

```
<Classifier> ::= <Rules>, <DefaultClass>
<Rules> ::= rule (rule)*
<DefaultClass> ::= class=value
```



Padillo, F., Luna, J. M., & Ventura, S. (2019). A grammar-guided genetic programming algorithm for associative classification in big data. Cognitive Computation, 11(3), 331-346.

Grammar Guided GP for Association Rule Mining

- Can get better effectiveness and complexity
- For rules set $C = \{R_1, \dots, R_n\}$
- $complexity = n \sum_{i=1}^{n} attributes(R_i)$

Effectiveness

Algorithm	Ranking
(a) Ranking for accuracy measure	
DAC	4.350
MRAC	4.150
MRAC+	2.700
DFAC-FFP	2.150
G3P-ACBD	*1.650*
(b) Ranking for kappa measure	
DAC	4.600
MRAC	4.100
MRAC+	2.600
DFAC-FFP	1.900
G3P-ACBD	*1.800*

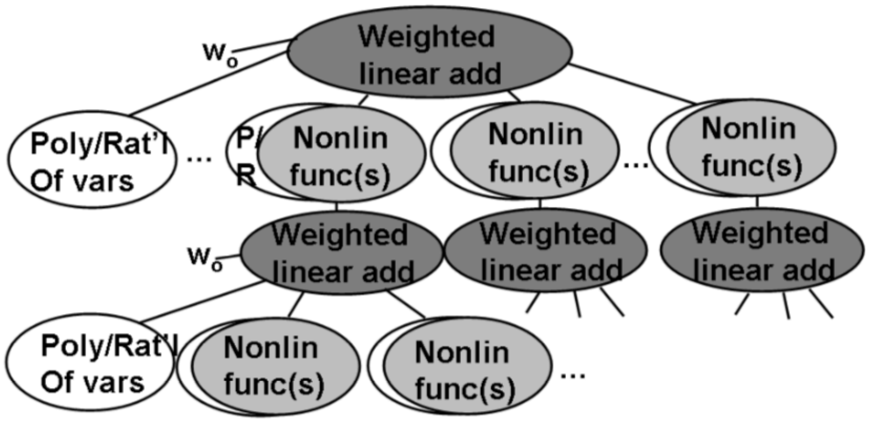
Complexity

Algorithm	Ranking
MRAC+	2.800
DFAC-FFP	2.200
G3P-ACBD	*1.000*

Padillo, F., Luna, J. M., & Ventura, S. (2019). A grammar-guided genetic programming algorithm for associative classification in big data. Cognitive Computation, 11(3), 331-346.

Canonical Form Function Expressions In Evolution (CAFFEINE)

- Special layer-based representation
  - Linear layer: polynomial/rational of the variables + non-linear components
  - Non-linear layer: a non-linear function of the linear layer
  - Example: $-10.3 + 3.1 * x_6 + 1.87 * x_1 * \log(-1.95 + 10.3 * (x_2 * x_7)/x_5)$
- Use grammar to implement



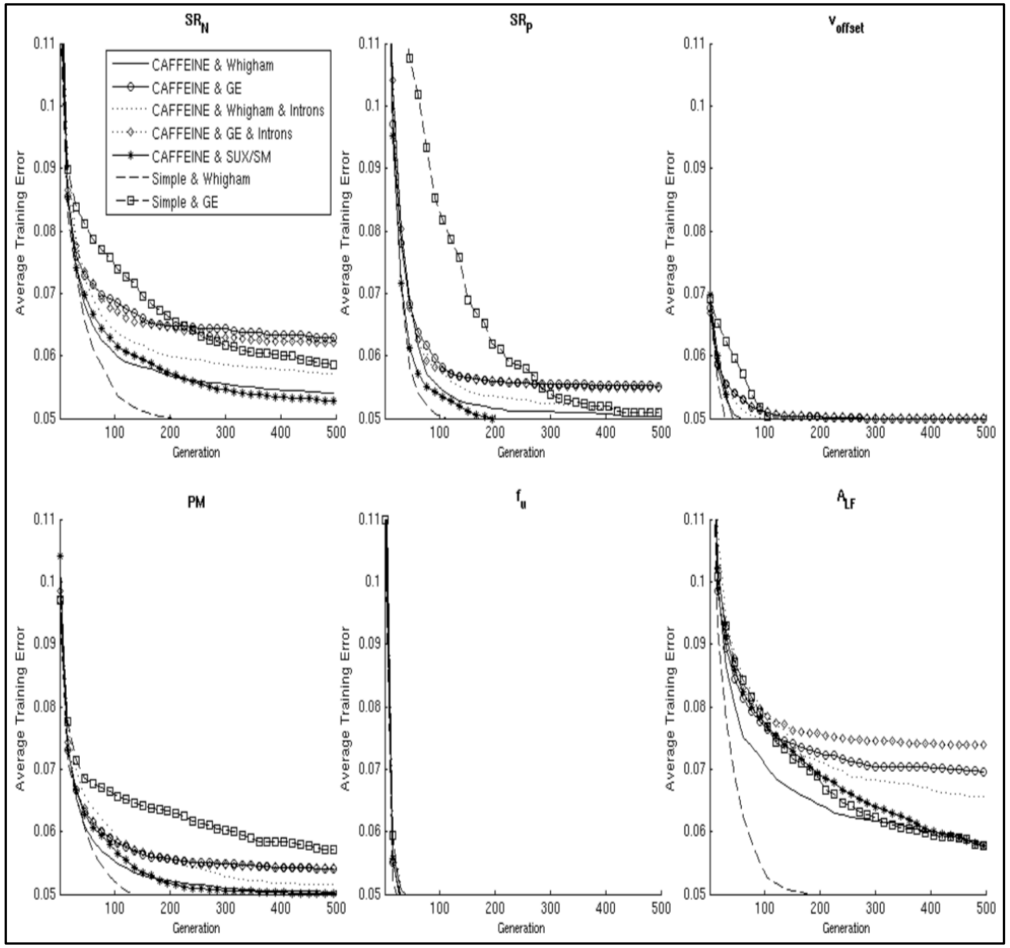

```
REPVC ::= VC | REPVC * REPOP | REPOP
REPOP ::= REPOP * REPOP | OP_1ARG(W + REPADD) |
          OP_2ARG(2ARGS)
2ARGS ::= W + REPADD, MAYBEW | MAYBEW, W+REPADD
<OP_2ARG> ::= DIVIDE | POW | MAX | …
<OP_1ARG> ::= INV | LOG10 | …
<VAR> ::= X1 | X2 | … | Xn | W
```

VC is a vector representing the polynomial/rational, e.g. [1,0,-2,1]=$x_1 * x_4/x_3^2$

McConaghy, T., & Gielen, G. (2006, July). Canonical form functions as a simple means for genetic programming to evolve human-interpretable functions. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (pp. 855-862)

Canonical Form Function Expressions In Evolution (CAFFEINE)

- Performs relatively well, and can get simple models



Perf.	Symbolic Model
A_{LF}	-10.3 + 7.08e-5 / id1 +1.87 * ln(-1.95e+9 + 1.00e+10 / (vsg1*vsg3) +1.42e+9*(vds2*vsd5) / (vsg1*vgs2*vsg5*id2))
f_u	10^(5.68 - 0.03 * vsg1 / vds2 – 55.43 * id1+ 5.63e-6 / id1)
PM	90.5 + 190.6 * id1 / vsg1 + 22.2 * id2 / vds2
v_{offset}	- 2.00e-3
SR_p	2.36e+7 + 1.95e+4 * id2 / id1 - 104.69 / id2 + 2.15e+9 * id2 + 4.63e+8 * id1
SR_n	- 5.72e+7 - 2.50e+11 * (id1*id2) / vgs2 + 5.53e+6 * vds2 / vgs2 + 109.72 / id1

McConaghy, T., & Gielen, G. (2006, July). Canonical form functions as a simple means for genetic programming to evolve human-interpretable functions. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (pp. 855-862)

GP with Simplification

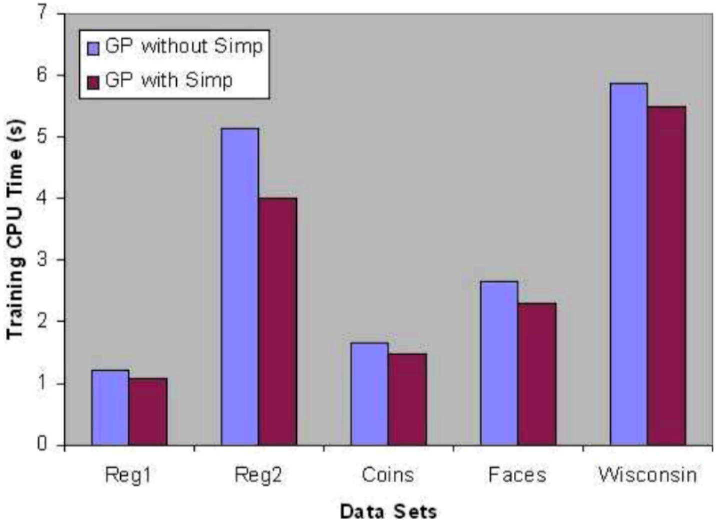
- GP tree tends to have many redundant branches (introns)
- Use algebraic simplification to simplify the trees during GP process

No. Precondition	Effective Result	No. Precondition	Effective Result
(1) if<0(A, b, c)	$\rightarrow$ b if A $<$ 0, else c	(2) if<0(a, b, b)	$\rightarrow$ b
(3) A + B	$\rightarrow$ C, C = A + B	(4) A - B	$\rightarrow$ C, C = A - B
(5) A $\times$ A	$\rightarrow$ C, C = A $\times$ B	(6) A $\div$ B	$\rightarrow$ C, C = A $\div$ B
(7) A + (B + c)	$\rightarrow$ C + c, C = A + B	(8) A + (B - c)	$\rightarrow$ C - c, C = A + B
(9) A - (B + c)	$\rightarrow$ C - c, C = A - B	(10) A - (B - c)	$\rightarrow$ C + c, C = A - B
(11) A $\times$ (B $\times$ c)	$\rightarrow$ C $\times$ c, C = A $\times$ B	(12) A $\times$ (B $\div$ c)	$\rightarrow$ C $\div$ c, C = A $\times$ B
(13) A $\div$ (B $\div$ c)	$\rightarrow$ C $\times$ c, C = A $\div$ B	(14) A + (b + C)	$\rightarrow$ B + b, B = A + C
(15) A + (b - C)	$\rightarrow$ B + b, B = A - C	(16) A - (b + C)	$\rightarrow$ B - b, B = A - C
(17) A - (b - C)	$\rightarrow$ B - b, B = A + C	(18) A $\times$ (b $\times$ C)	$\rightarrow$ B $\times$ b, B = A $\times$ C
(19) A $\times$ (b $\div$ C)	$\rightarrow$ C $\times$ b, B = A $\div$ C	(20) A $\div$ (b $\div$ C)	$\rightarrow$ B $\div$ b, B = A $\times$ C
(21) a $\div$ 1	$\rightarrow$ a	(22) a $\div$ a	$\rightarrow$ 1
(23) 0 $\div$ a	$\rightarrow$ 0	(24) 0 $\times$ a = a $\times$ 0	$\rightarrow$ 0
(25) a $\times$ 1 = 1 $\times$ a	$\rightarrow$ a	(26) a + 0 = 0 + a	$\rightarrow$ a
(27) a - 0	$\rightarrow$ a	(28) a - a	$\rightarrow$ 0
(29) a $\times$ $\frac{1}{b}$ = $\frac{1}{b}$ $\times$ a	$\rightarrow$ $\frac{a}{b}$	(30) a $\times$ $\frac{b}{a}$ = $\frac{b}{a}$ $\times$ a	$\rightarrow$ b
(31) a $\div$ 0	$\rightarrow$ 0	(32) A $\div$ 0	$\rightarrow$ 0

Zhang, M., & Wong, P. (2008). Explicitly simplifying evolved genetic programs during evolution. International Journal of Computational Intelligence and Applications, 7(02), 201-232.

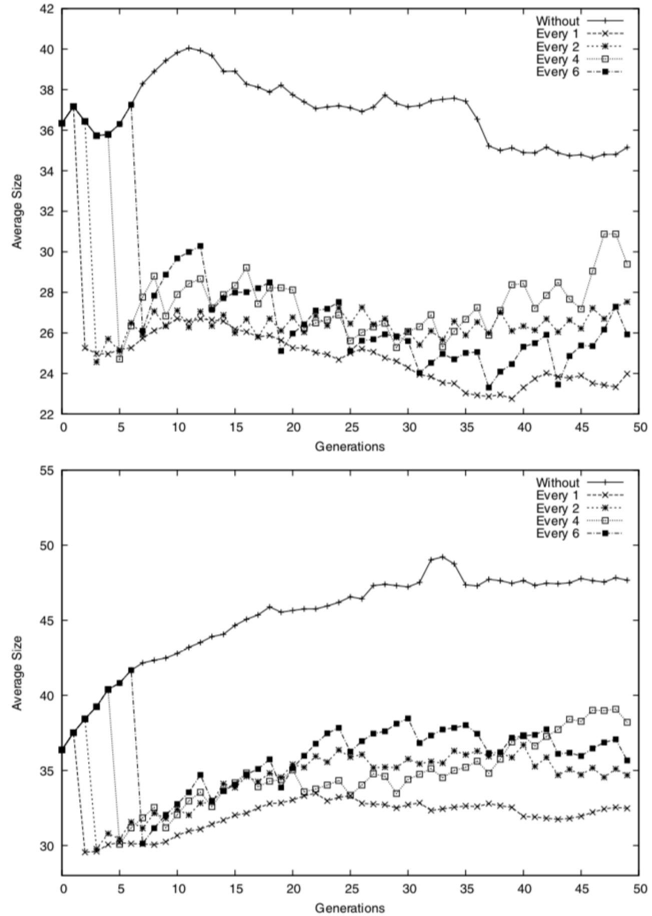
GP with Simplification

- With proper simplification frequency, can achieve faster training time and smaller size without worsening accuracy
- Evolved rules easier to interpret



(1) if $(((F_5 + F_0 - 0.48) \times 0.383 + F_5) \times 0.0059) < 0$
then class = benign else class = malignant

(2) if $(3F_5 + F_1 + F_2 + F_7 + F_0 - F_0 \times F_3 - 0.84) < 0$
then class = benign else class = malignant



Zhang, M., & Wong, P. (2008). Explicitly simplifying evolved genetic programs during evolution. International Journal of Computational Intelligence and Applications, 7(02), 201-232.

GP with Simplification

- More simplification based on logic operators

Precondition(s)	Simplification
Min(A, A+B) and B is always non-negative	A
Min(A, A+B) and B is always negative	A+B
Min(A, A-B) and B is always non-positive	A-B
Min(A, A+B) and B is always non-positive	A

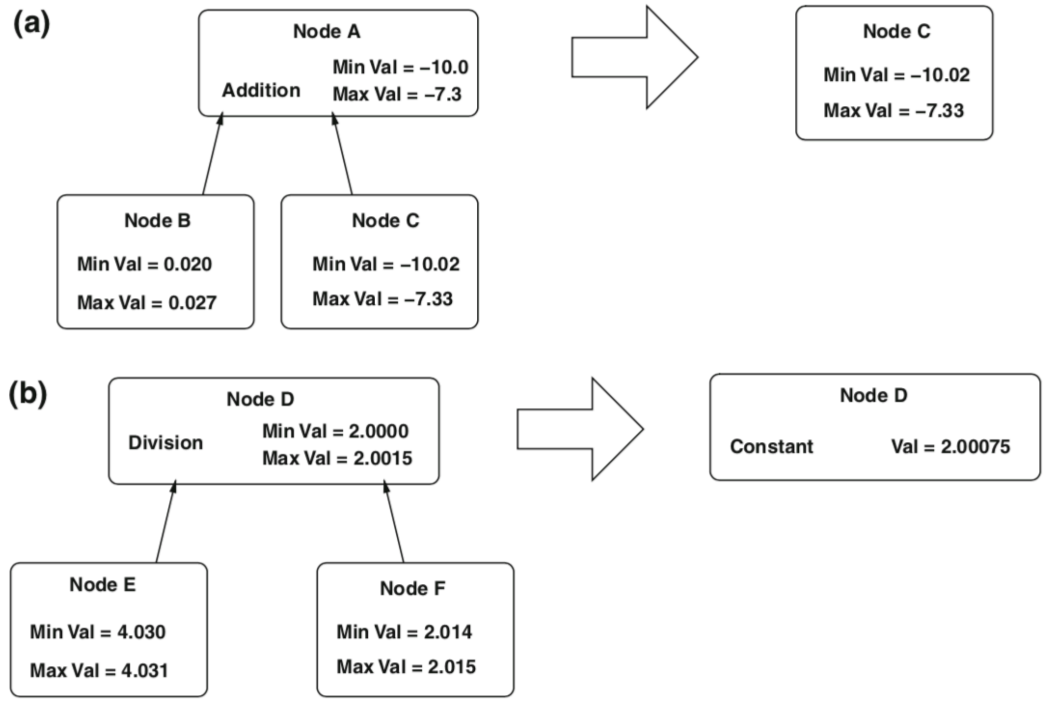
- Get the sign of the features using domain knowledge
  - E.g., the job shop state attributes for scheduling rule learning
- Propagate the sign in the tree

$root(B)$	$s1$	$s2$	output	$root(B)$	$s1$	$s2$	output
$+^*$	>0	≥ 0	>0	$\max^*$	>0	#	>0
$+^*$	≥ 0	≥ 0	≥ 0	$\max^*$	≥ 0	#	≥ 0
$+^*$	$=0$	$=0$	$=0$	$\max^*$	<0	<0	<0
$+^*$	<0	≤ 0	<0	$\max^*$	≤ 0	≤ 0	≤ 0
$+^*$	≤ 0	≤ 0	≤ 0	$\max^*$	$=0$	$=0$	$=0$
$-$	>0	≤ 0	>0	$\min^*$	<0	#	<0
$-$	≥ 0	≤ 0	≥ 0	$\min^*$	≤ 0	#	≤ 0
$-$	$=0$	$=0$	$=0$	$\min^*$	>0	>0	>0
$-$	<0	≥ 0	<0	$\min^*$	≥ 0	≥ 0	≥ 0
$-$	≤ 0	≥ 0	≤ 0	$\min^*$	$=0$	$=0$	$=0$
$\times^*$	>0	>0	>0	$\div$	>0	>0	>0
$\times^*$	≥ 0	≥ 0	≥ 0	$\div$	≥ 0	>0	≥ 0
$\times^*$	$=0$	$=0$	$=0$	$\div$	#	$=0$	>0 (=1)
$\times^*$	<0	<0	>0	$\div$	<0	<0	>0
$\times^*$	≤ 0	≤ 0	≥ 0	$\div$	≤ 0	<0	≥ 0
$\times^*$	>0	<0	<0	$\div$	>0	<0	<0
$\times^*$	≥ 0	≤ 0	≤ 0	$\div$	<0	>0	<0
				$\div$	≤ 0	>0	≤ 0
				$\div$	≥ 0	<0	≤ 0

Panda, S., & Mei, Y. (2021). Genetic Programming with Algebraic Simplification for Dynamic Job Shop Scheduling. IEEE CEC.

GP with Simplification

- **Numerical simplification**
  - Empirically, how much a child contributes to its parent's output
  - Check the value range of the nodes
  - If the range of a child is much smaller than the parent's min absolute value, simplify the parent to the other child
  - If the range of a node is much smaller than its own min absolute value, simplify it to a constant
- Can show comparable classification performance and reduce the program size dramatically (~40%)



Kinzett, D., Johnston, M., & Zhang, M. (2009). Numerical simplification for bloat control and analysis of building blocks in genetic programming. Evolutionary Intelligence, 2(4), 151-168.

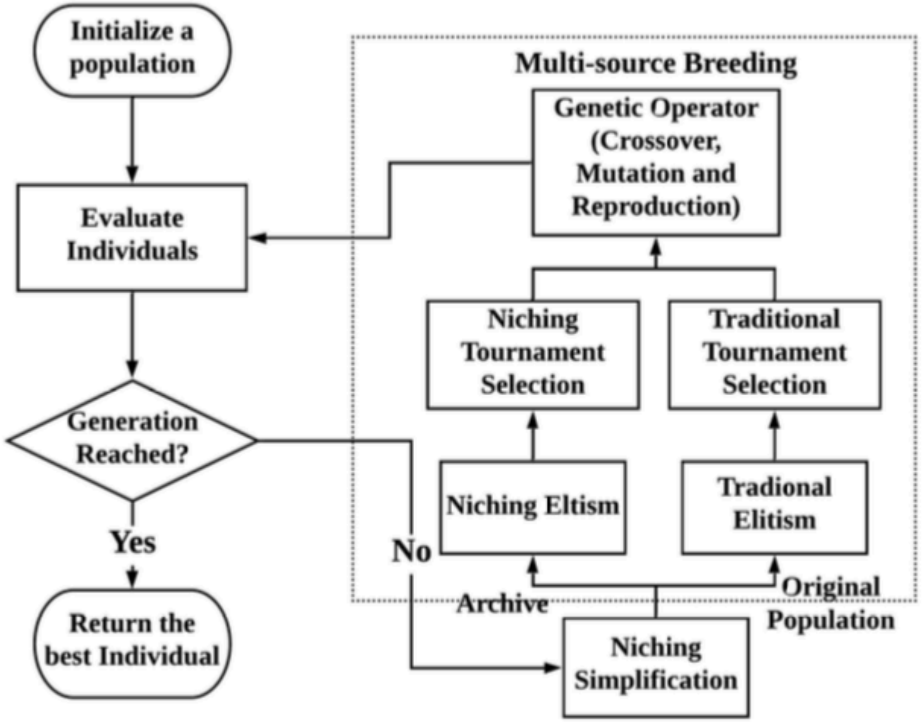
GP with Phenotypic Simplification

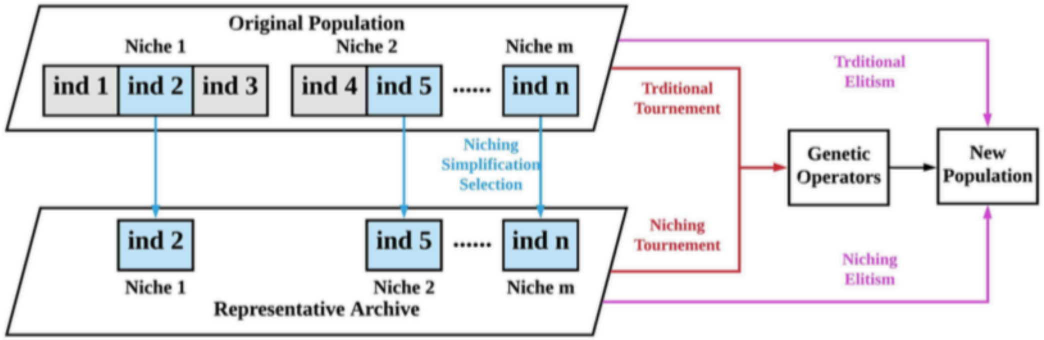
- Define phenotypic behaviour of GP trees
  - E.g., predicted values in regression, priority values in decision making
- Group/Cluster the GP trees based on phenotypic behaviour
- Simplification: replace a GP tree with a smaller/simpler tree with the same phenotypic behaviour
- **Niching GP**
  - Niching in the GP population based on phenotypic behaviour
  - External archive: the smallest GP individual in each niche
  - Multi-source breeding: select the parents from the original population and archive

Wang, S., Mei Y., Zhang, M. & Yao X. (2021). Genetic Programming with Niching for Uncertain Capacitated Arc Routing Problem. IEEE Transactions on Evolutionary Computation.

GP with Phenotypic Simplification

- Niching GP

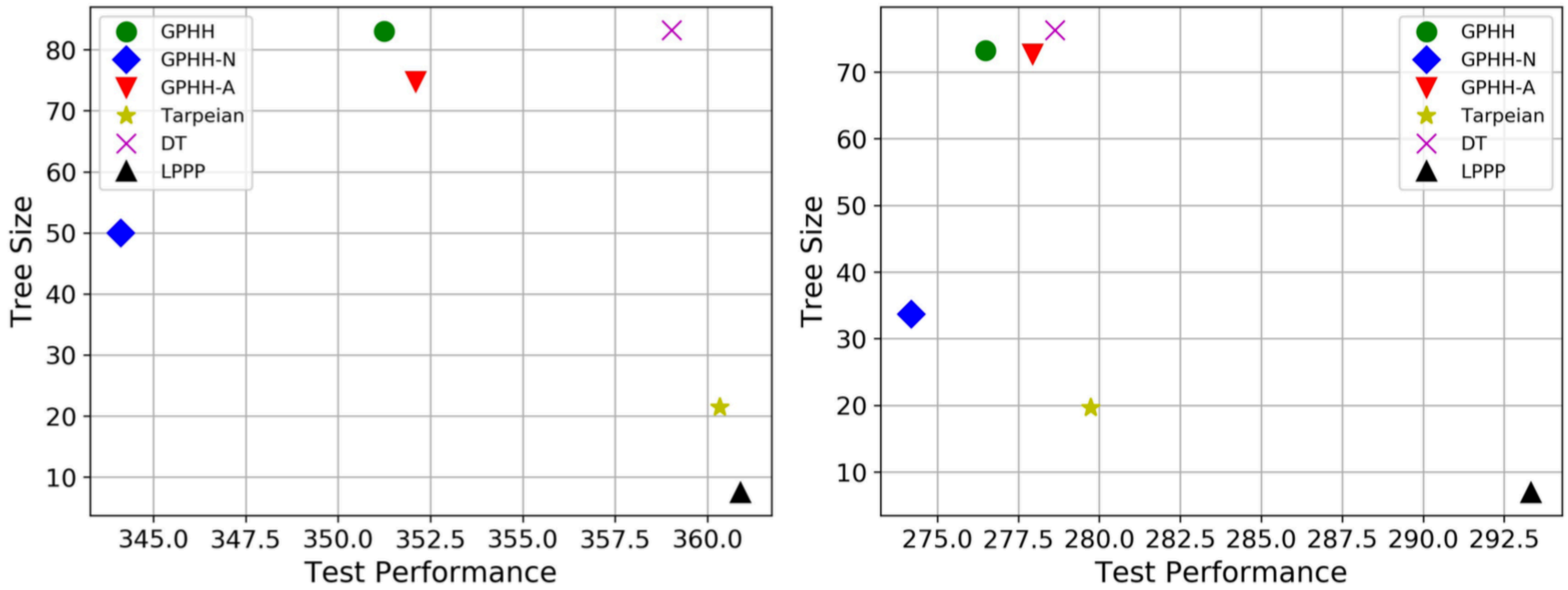




Wang, S., Mei Y., Zhang, M. & Yao X. (2021). Genetic Programming with Niching for Uncertain Capacitated Arc Routing Problem. IEEE Transactions on Evolutionary Computation.

GP with Phenotypic Simplification

- Results on Evolving routing policy for UCARP
  - **Better balance** between test performance and tree size

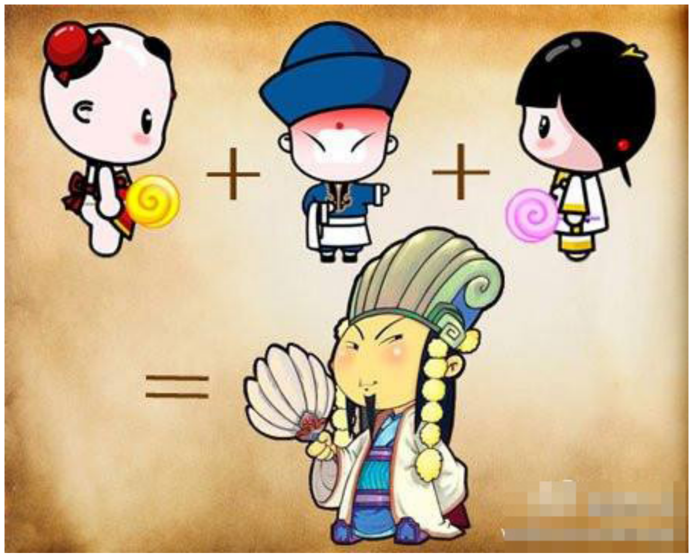


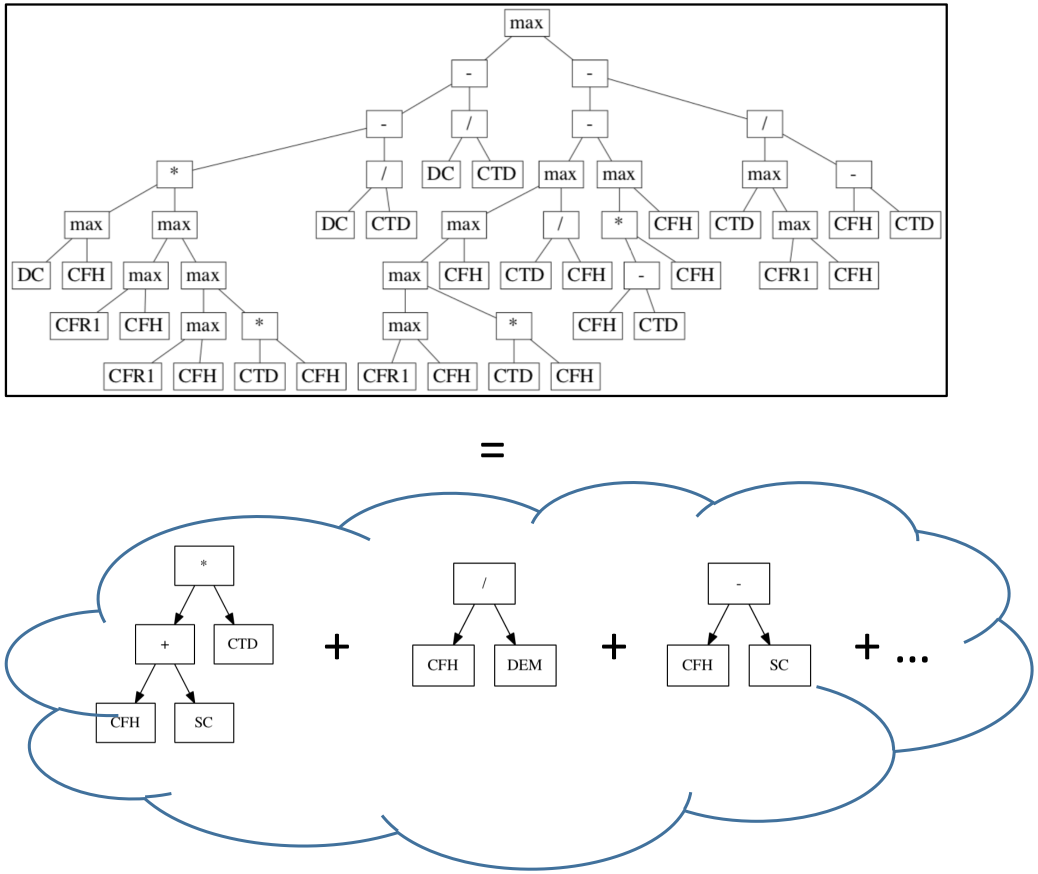
(a) Ugdb1 (b) Uval2B

Wang, S., Mei Y., Zhang, M. & Yao X. (2021). Genetic Programming with Niching for Uncertain Capacitated Arc Routing Problem. IEEE Transactions on Evolutionary Computation.

Ensemble GP

- “Two heads are better than one”
- A group of simple GP rules can make the same/better decisions than a single complex GP rule
  - Simple and reasonably good rules
  - Mutually complementary





Ensemble GP

- How to evolve the simple, reasonably good, and complementary GP rules?
  - Ensemble learning methods: bagging, boosting
  - Cooperative co-evolution: the context vector is a group of rules
- **Bagging GP**
  - Divide GP into multiple cycles, each cycle evolves one rule
  - Used different training subset in each cycle
  - Limitation: GP is slow to get each rule
- **Boosting GP**
  - Learn each rule sequentially, using the same training set
  - Adjust the weight of each training sample
  - $fit(x) = \sum_{S \in train} w(S) * tc(x, S)$
  - Limitation: GP is slow to get each rule, cannot use large training set, poor generalization
- **CCGP**
  - Co-evolve each rule in a sub-population, use context vector to evaluate fitness
  - Limitation: hard to consider complementary

Ensemble GP

- Empirical comparison for evolving routing policies (output the priority of each customer)
  - 5 rules (5 x 200 for CCGP), 5 training samples
  - Depth = 4 (simple enough)
  - **Aggregation**: sum up the outputs of the rules
- BaggingGP and Boosting GP are very poor
- CCGP is good
  - No worse performance
  - Smaller size per tree

Instance	SimpleGP		BaggingGP		BoostingGP		CCGP	
	tc	size	tc	size	tc	size	tc	size
ugdb1	367.8	11.9	397.1	12.2	399.4	12.9	364.9	5.2
ugdb2	386.1	11.9	424.9	12.8	433.0	12.2	372.3	6.7
ugdb8	485.0	12.8	548.7	12.2	568.9	13.1	467.7	6.9
ugdb23	255.5	12.4	266.4	12.7	268.8	12.7	256.0	5.6
uval9A	348.0	13.1	374.0	12.2	375.9	12.9	341.3	7.6
uval9D	501.1	13.5	561.4	11.8	542.2	12.1	490.4	7.4
uval10A	444.4	11.8	475.2	11.8	471.9	12.1	445.2	6.9
uval10D	641.7	13.1	693.9	12.9	685.2	12.6	649.1	6.7

Wang, S., Mei, Y., Park, J., & Zhang, M. (2019, December). Evolving ensembles of routing policies using genetic programming for uncertain capacitated arc routing problem. In 2019 IEEE SSCI (pp. 1628-1635). IEEE.

Ensemble GP

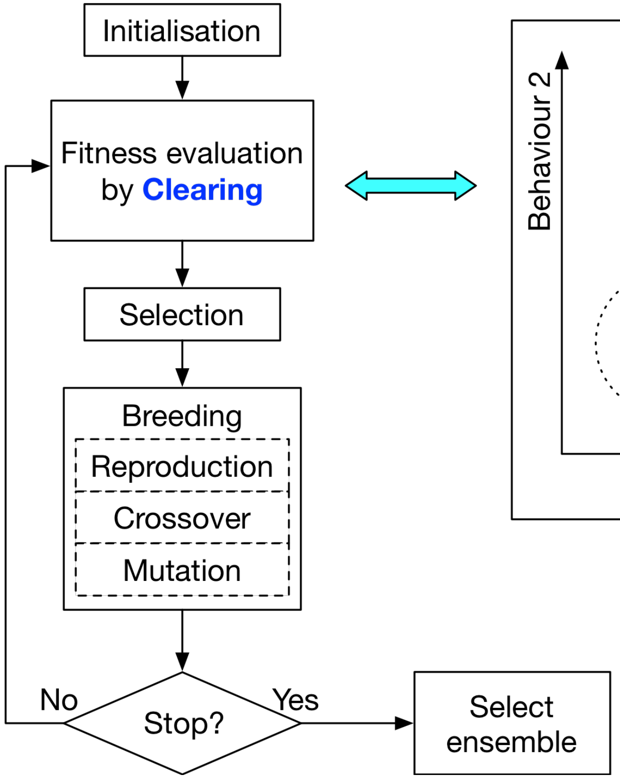
- However, CCGP cannot guarantee that the rules are complementary
  - One rule can dominate the decision of the ensemble
  - Diversity may not be enough
- Consider niching to maintain diversity
- **DivNichGP**
  - A single population with different niches
  - Different niches tend to complement each other
  - No need to pre-define the number of rules (depends on number of niches)

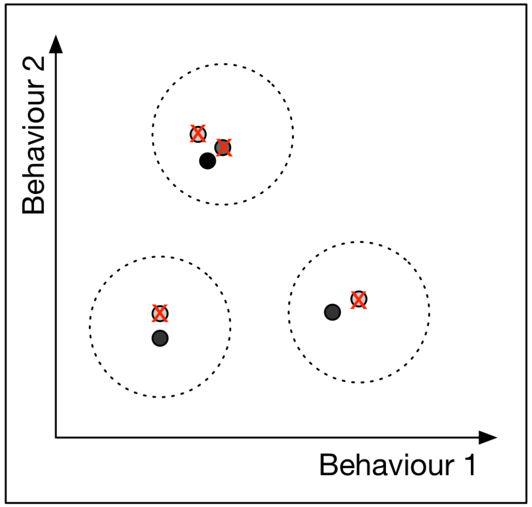
Pool size	92	90	88	86	84	82	80	78	76	74	72	70	68	66	64	62	60	58	56
Rank in rp1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rank in rp2	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Rank in rp3	7	24	30	61	0	77	41	31	0	31	30	0	0	41	39	52	0	13	14
Rank in rp4	7	16	34	22	0	56	39	27	0	17	24	0	0	40	53	53	0	18	14
Rank in rp5	9	17	42	42	1	77	53	54	66	67	66	1	2	5	7	6	7	0	9

Wang, S., Mei, Y., & Zhang, M. (2019, July). Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem. In Proceedings of GECCO (pp. 1093-1101).

Ensemble GP

- Use clearing to construct niches
  - In each niche, only k best individuals are retained, others are set to very bad fitness
  - Use phenotypic distance
- **Ensemble selection**
  - Sort final individuals by fitness
  - Include them one by one into the ensemble
  - Stop if the ensemble cannot improve





Wang, S., Mei, Y., & Zhang, M. (2019, July). Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem. In Proceedings of GECCO (pp. 1093-1101).

Ensemble GP

- Test on evolving UCARP routing policies
  - SimpleGP has max depth of 8, CCGP and DivNichGP have max depth of 4
  - DivNichGP has much better performance and size

Instance	SimpleGP		CCGP		DivNichGP	
	tc	size	tc	size	tc	size
ugdb1	354.8	75.7	364.9	5.2	348.4	21.3
ugdb2	377.1	68.3	372.3	6.7	364.8	21.4
ugdb8	499.8	67.5	467.7	6.9	467.4	23.4
ugdb23	252.1	68.3	256.0	5.6	250.9	19.9
uval9A	340.3	65.3	341.3	7.6	333.4	21.5
uval9D	478.3	68.3	490.4	7.4	504.5	25.1
uval10A	440.6	58.1	445.2	6.9	439.4	21.6
uval10D	630.2	65.7	649.1	6.7	636.8	24.8

Wang, S., Mei, Y., & Zhang, M. (2019, July). Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem. In Proceedings of GECCO (pp. 1093-1101).

Ensemble GP

- Ensemble size varies for different instances
- DivNichGP has better complementary (none of the rules dominates the decisions)
- Rules easier to interpret

Instance	Ensemble size
ugdb1	5.7
ugdb2	9.2
ugdb8	1.6
ugdb23	7.2
uval9A	8.9
uval9D	1.2
uval10A	6.0
uval10D	1.8

A good ensemble for ugdb1

DMS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
rp_1	2	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0
rp_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
rp_3	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
rp_4	1	0	0	1	0	3	0	0	0	3	1	0	1	0	0	0	0	0	2	0
rp_5	7	6	3	1	2	3	2	2	2	0	1	0	1	0	0	1	0	0	0	1
rp_6	1	1	0	1	0	1	0	0	1	0	0	0	1	0	4	0	3	0	0	1
rp_7	1	1	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	1

$$rp_1 = \max\{\mathrm{DEM1}, \mathrm{CFH}\} * (\mathrm{SC} + \mathrm{CTD}) * \left(\frac{\mathrm{CFH}}{\mathrm{DC}} - \min\{\mathrm{RQ}, 0.32\}\right),$$

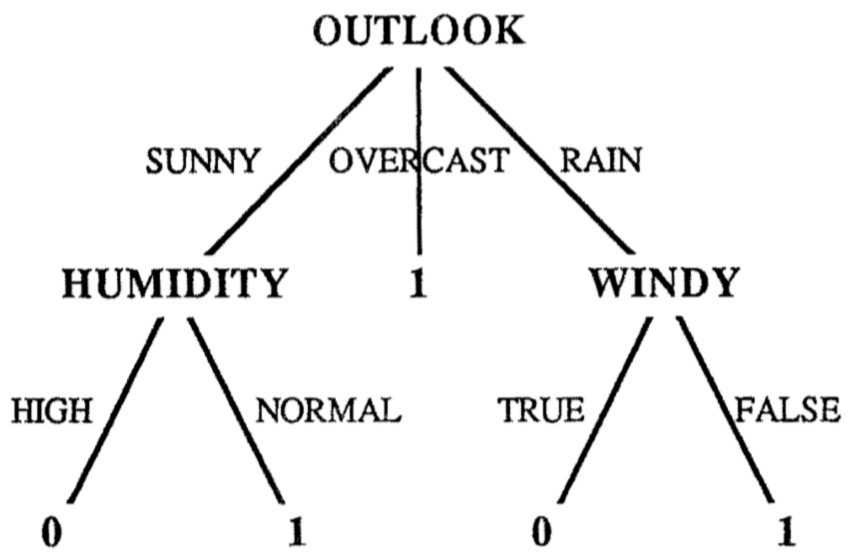
$$rp_2 = \min\{\mathrm{CFH}, \mathrm{DC}\} * \mathrm{CTD} - \frac{\mathrm{DC}}{\min\{\mathrm{CFH}, \mathrm{DC}\}},$$

$$rp_3 = \max\{\mathrm{FRT}, \mathrm{CFH}\} * \mathrm{DEM} * \mathrm{CTD} * \left(\frac{\mathrm{CFH}}{\mathrm{DC}} - \min\{\mathrm{FUT}, 0.32\}\right).$$

Wang, S., Mei, Y., & Zhang, M. (2019, July). Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem. In Proceedings of GECCO (pp. 1093-1101).

GP for Decision Tree Induction

- (Shallow/Small) decision tree is a class of easy-to-interpret models
- Conventional DT learning algorithms (ID3, C4.5, …) are mainly greedy local search
- Use GP to automatically induce decision trees by more global search
- Each attribute/feature is a function
  - #children = #possible values of the feature
  - Nominal features (discretize the features)
- **Terminal set**: class names
- **Function set**: feature tests



Koza, John R. "Concept formation and decision tree induction using the genetic programming paradigm." International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 1990.

GP for Decision Tree Induction

- Grammar-based GP (BNF) grammar

```
<Tree> ::= “if-then-else” <Cond><Tree><Tree> | Class
<Cond> ::= <Cond> “And” <Cond> | <Cond> “Or” <Cond> |
           “Not” <Cond> | Variable <RelationOperation> Threshold
<RelationOperation> ::= “>” | “<“ | “=“
```

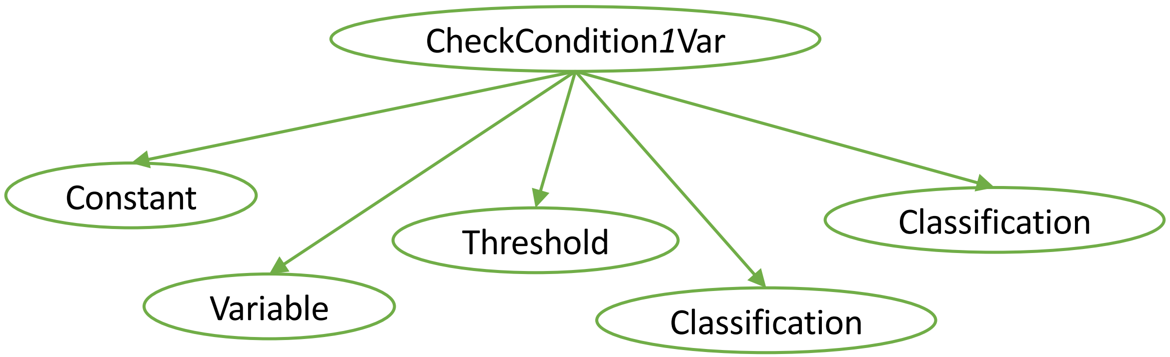
- Variable is a feature
- Threshold is a real number

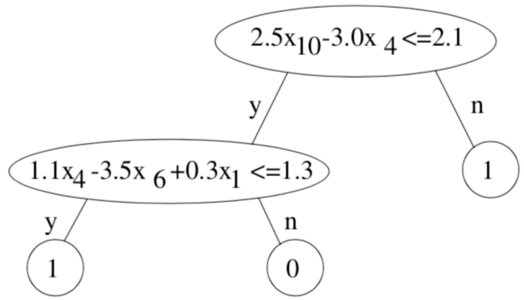
Li, J., Li, X., & Yao, X. (2005, September). Cost-sensitive classification with genetic programming. In 2005 IEEE congress on evolutionary computation (Vol. 3, pp. 2114-2121). IEEE.

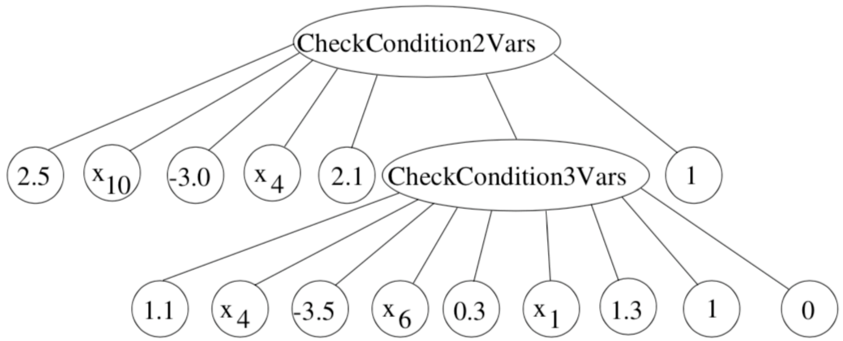
GP for Decision Tree Induction

- Use strongly typed GP to generate valid DTs (All continuous features)
  - Three types
    - **Variable**: Int[0, #features-1], access the value of a feature, ONLY terminals have this type.
    - **Constant**: Double[-10,10], coefficient/weight, ONLY terminals have this type.
    - **Classification**: Int[0, #classes-1], predict the class, BOTH terminals and functions have this type.

Terminal set	Variable, Constant, Classification
Function set	CheckCondition1Var, CheckCondition2Vars, CheckCondition3Vars





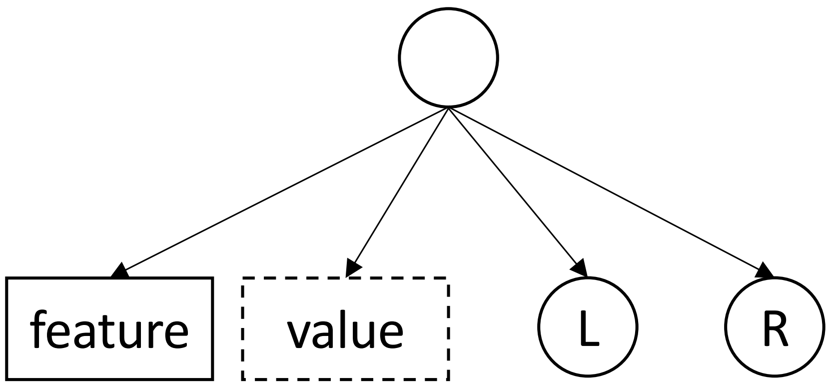


Bot, M. C., & Langdon, W. B. (2000, April). Application of genetic programming to induction of linear classification trees. In European Conference on Genetic Programming (pp. 247-258). Springer, Berlin, Heidelberg.

GP for Decision Tree Induction

- **Strongly typed** GP, both continuous and nominal features (binary classification)
  - **Terminals** include
    - **Features**: integer, index of the feature
    - **Values**: real [0,1), threshold for a numerical feature, or the index of the nominal feature value
    - **Class**: binary, leaf nodes of the tree
  - **Function** node: integer x real x binary x binary -> binary
    - If feature is **numeric** with the range $[l, u]$, then $out = \begin{cases} L, if\ x_{feature} \leq (u-l)*value + l, \\ R, otherwise \end{cases}$
    - If feature is **nominal** taking from $\{V_1, \ldots, V_k\}$, then $out = \begin{cases} L, if\ x_{feature} = V_{\lfloor k*value \rfloor}, \\ R, otherwise \end{cases}$

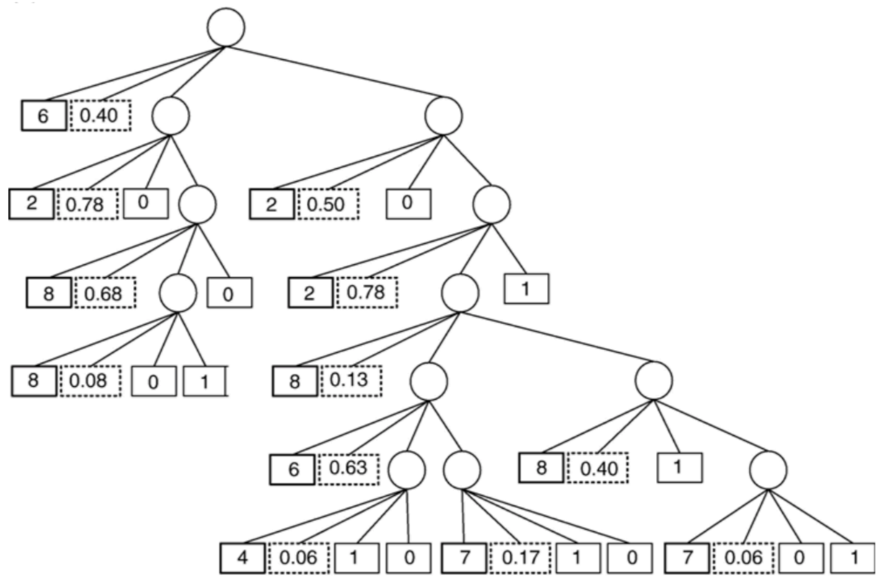
Type	Terminal	Function
Integer	Feature	
Real	Value	
Binary	Class	Node

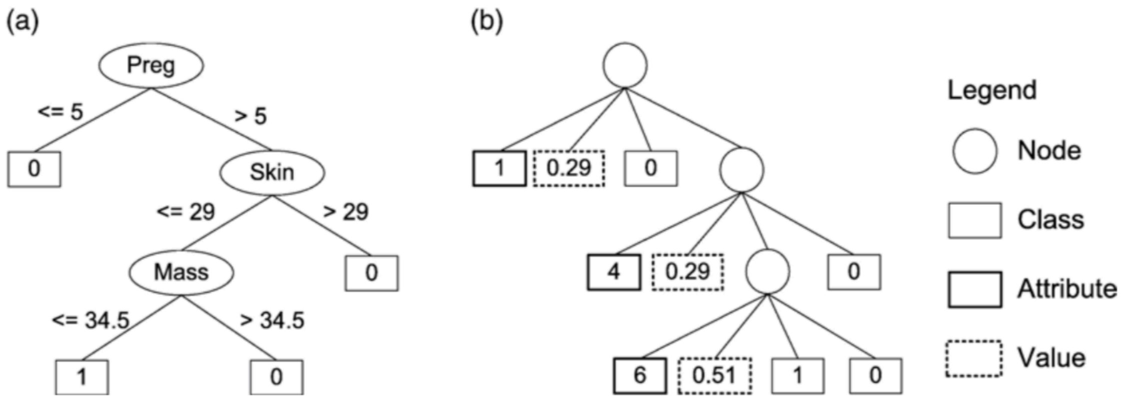


Zhao, Huimin. "A multi-objective genetic programming approach to developing Pareto optimal decision trees." Decision Support Systems 43.3 (2007): 809-826.

GP for Decision Tree Induction

- Example
  - Preg is feature 1
  - Skin is feature 4
  - Mass is feature 6

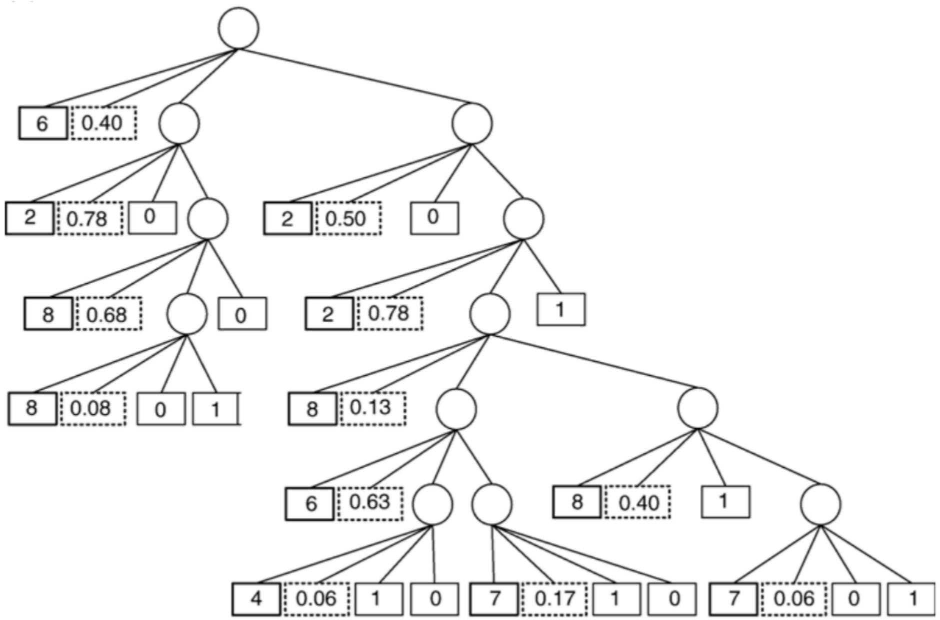


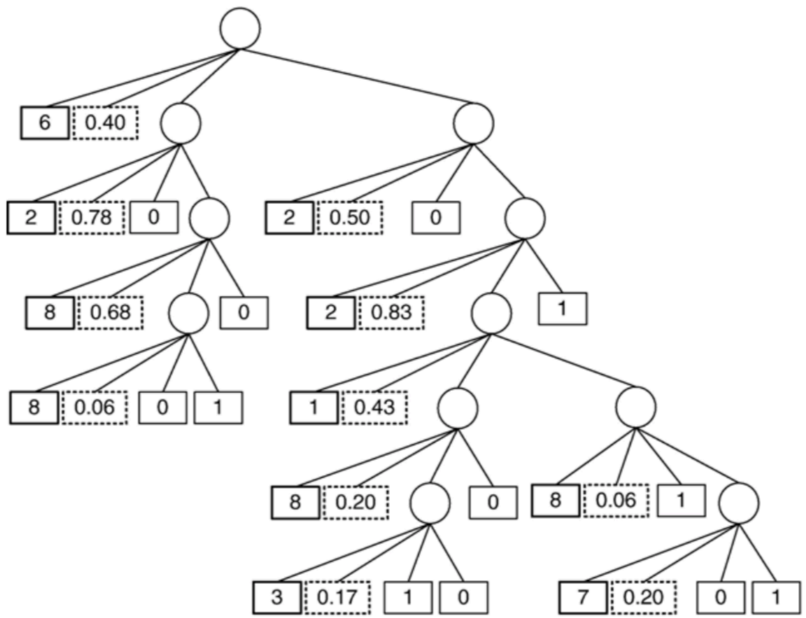


Zhao, Huimin. "A multi-objective genetic programming approach to developing Pareto optimal decision trees." Decision Support Systems 43.3 (2007): 809-826.

GP for Decision Tree Induction

- **GP crossover and mutation**
  - Standard tree-based crossover: swap two random sub-trees of the parents
  - Standard tree-based mutation: randomly select a sub-tree of the parent, and replace with a newly generated sub-tree

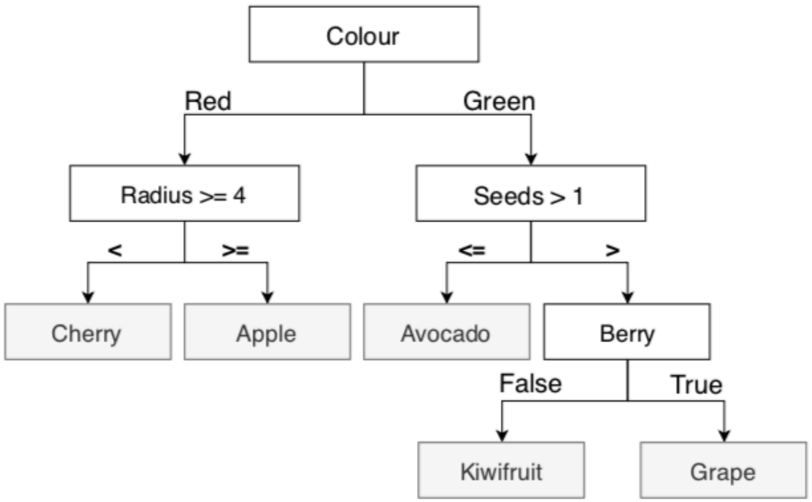


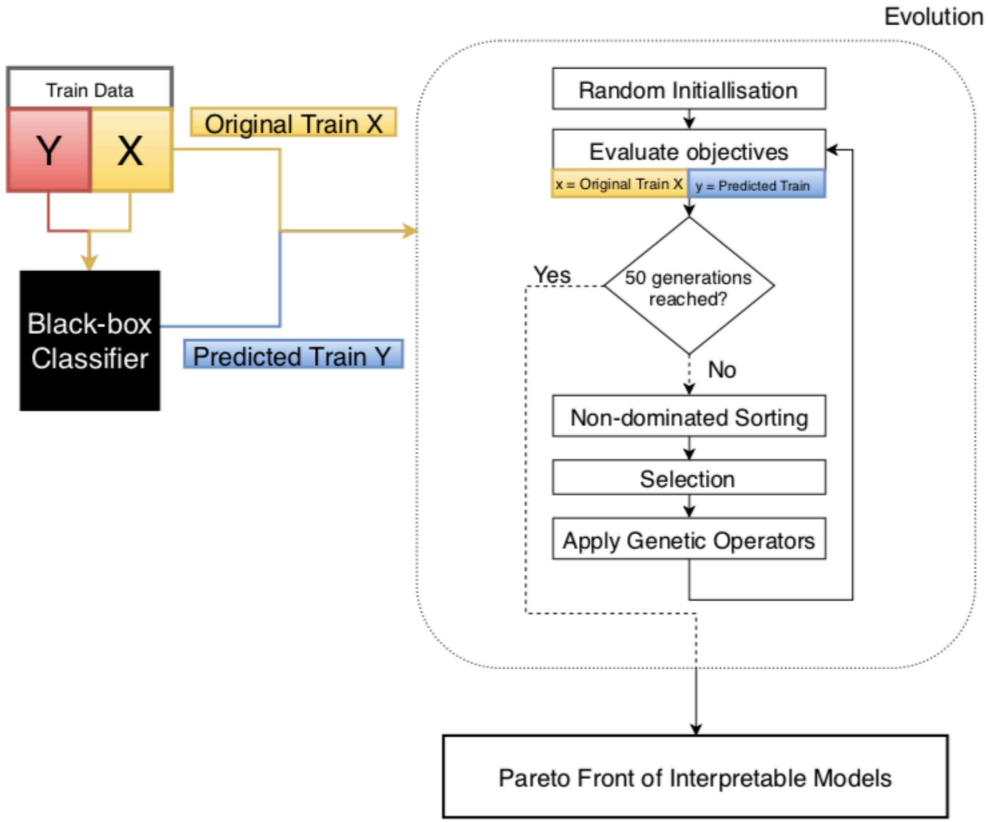


Zhao, Huimin. "A multi-objective genetic programming approach to developing Pareto optimal decision trees." Decision Support Systems 43.3 (2007): 809-826.

GP for Post-hoc Global Interpretability

- Use GP to evolve a decision tree to approximate a black-box ML model
- Multi-objective GP
  - **F1**: reconstruction ability (max F1-score)
  - **F2**: interpretability (#split points)
- Strongly-typed GP

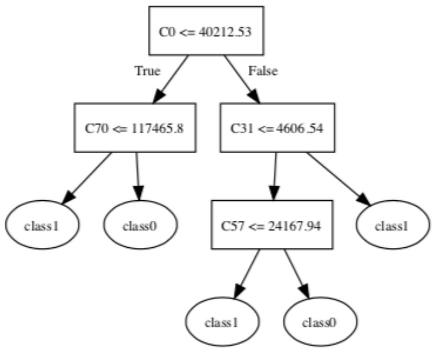




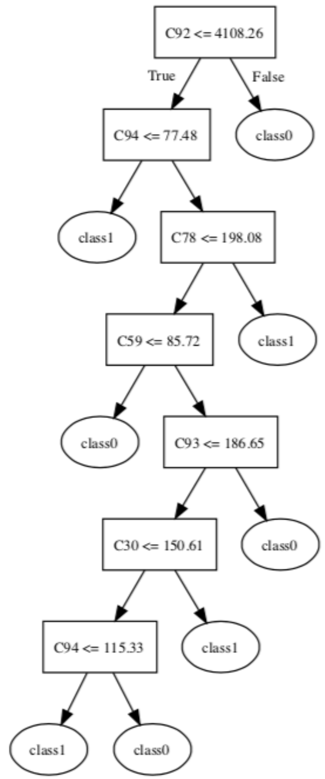
Evans, B. P., Xue, B., & Zhang, M. (2019, July). What's inside the black-box? a genetic programming method for interpreting complex machine learning models. In Proceedings of GECCO (pp. 1012-1020).

GP for Post-hoc Global Interpretability

- Better trade-off between accuracy and interpretability
- Has potential to further improve reconstruction ability (f1-socre)



(a) Proposed ($f_1 = 0.757$)



(b) Simplified Decision Tree ($f_1 = 0.759$)

(c) Decision Tree ($f_1 = 0.790$)

IF (X43 = 3) AND (X78 = 3) THEN: Class1
ELSE IF (X62 = 0) THEN: Class1
ELSE DEFAULT: Class1

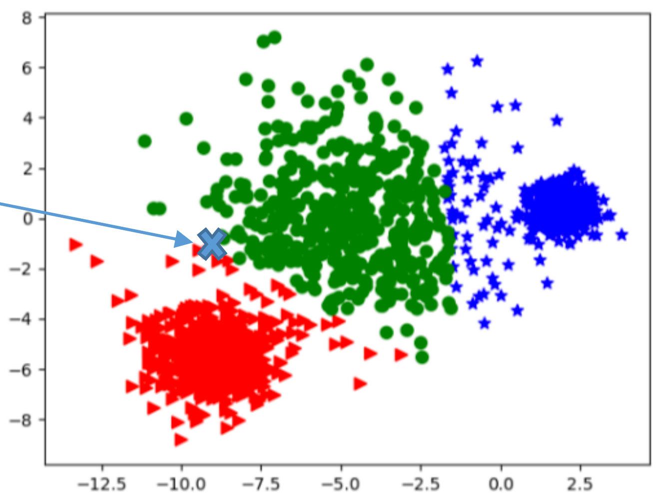
(d) Bayesian Rule List ($f_1 = 0.704$)

Evans, B. P., Xue, B., & Zhang, M. (2019, July). What's inside the black-box? a genetic programming method for interpreting complex machine learning models. In Proceedings of GECCO (pp. 1012-1020).

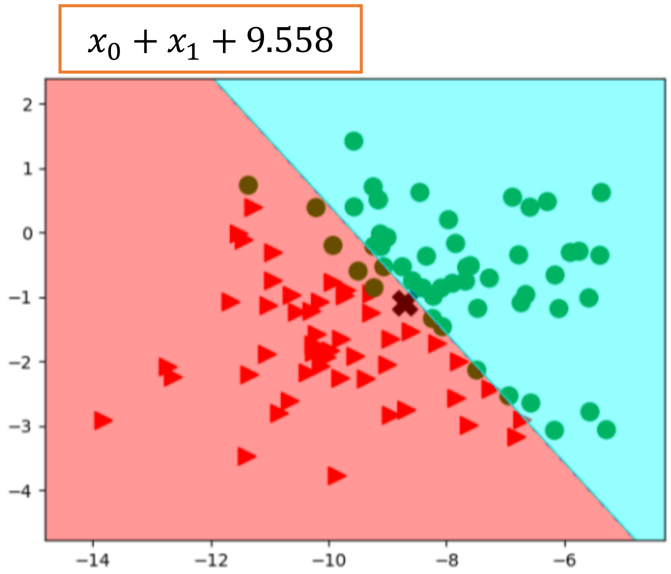
GP for Post-hoc Local Interpretability

- Given an input of a complex pre-trained ML model: $\boldsymbol{x} \in R^n$
  1) Generate m sample points around the input $\boldsymbol{x}$ from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{x}, I_n \times \sigma)$, called noise set η
  2) Find an explainer model which is easy to interpret, and can mimic the behavior of the original complex model
  3) Use GP to evolve the explainer model, to minimize the RMSE between the GP model and the complex pre-trained ML model

Input to be explained



Dataset with 1500 samples, 2 features, 3 classes



Noise set with 100 samples around the input, the prediction of the pre-trained complex model and the GP explainer model

Ferreira et al. (2020) Applying Genetic Programming to Improve Interpretability in Machine Learning Models. arXiv

GP for Post-hoc Local Interpretability

- Explain the pre-trained Random Forest, DNN, SVM/R models
- Compared with Lime and Decision Tree explainers
- Better overall error on the tested classification and regression problems

Explainer	Average Error	Std Dev
Lime	7.577	36.913
DT	0.083	0.329
GPX	**0.065**	**0.508**

- Example: Boston regression dataset
  - One input point: GPX model is $\frac{x_{ptratio}^2 x_{nox}}{28.390}$
  - Another input point: GPX model is $\frac{x_{indus}}{x_{lstat}} + x_{ptratio}$
  - Pupil-teacher ratio by town is important in both cases
  - Different regions could have different criteria

Ferreira et al. (2020) Applying Genetic Programming to Improve Interpretability in Machine Learning Models. arXiv

GP with Visualisation

- **Manifold learning**: learn a mapping from high-dimensional data to much lower-dimensional (e.g. 2 or 3) data that can be visualised
  - PCA (linear), MDS (non-linear), t-SNE (non-linear)
- The state-of-the-art manifold learning methods are not interpretable
  - No mapping back to the original features, transformation is opaque
- **GP-MaL**: a multi-tree GP, each tree representing one transformed dimension
  - Terminal set: the scaled input features, random constant
  - Function set:

Category	Arithmetic			Non-Linear		Conditional		
Function	$+$	$\times$	$5+$	Sigmoid	ReLU	Max	Min	If
No. of Inputs	2	2	5	1	1	2	2	3
No. of Outputs	1	1	1	1	1	1	1	1

  - Fitness function:
    - Preserving neighbourhood in the low-dimensional space

$$fitness = \frac{1}{n^2} \sum_{I \in X} similarity(N_I, N_I')$$

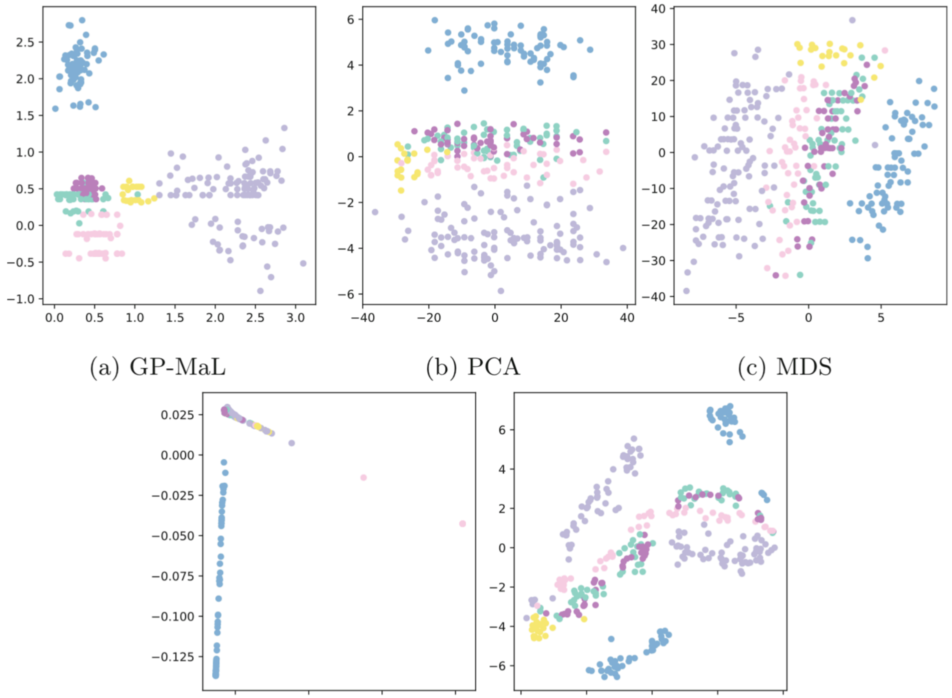
$$similarity(N, N') = \sum_{a \in N} agreement(|pos(a, N) - pos(a, N')|)$$

Lensen, A., Xue, B., & Zhang, M. (2019, April). Can genetic programming do manifold learning too?. In EuroGP (pp. 114-130). Springer, Cham.

GP with Visualisation

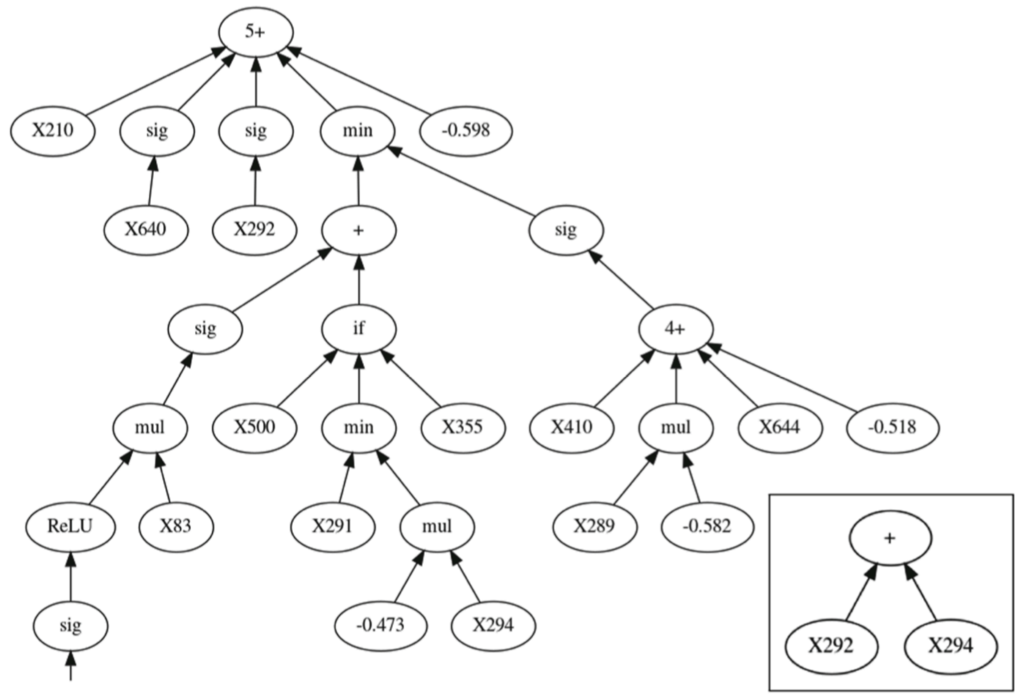
- Can achieve better data separation (higher accuracy) on some datasets
- Can potentially interpret the trees (they are symbolic)

Dermatology dataset



(a) GP-MaL (b) PCA (c) MDS

(d) LLE (e) tSNE



(a) (b)

Lensen, A., Xue, B., & Zhang, M. (2019, April). Can genetic programming do manifold learning too?. In EuroGP (pp. 114-130). Springer, Cham.

GP with Visualisation

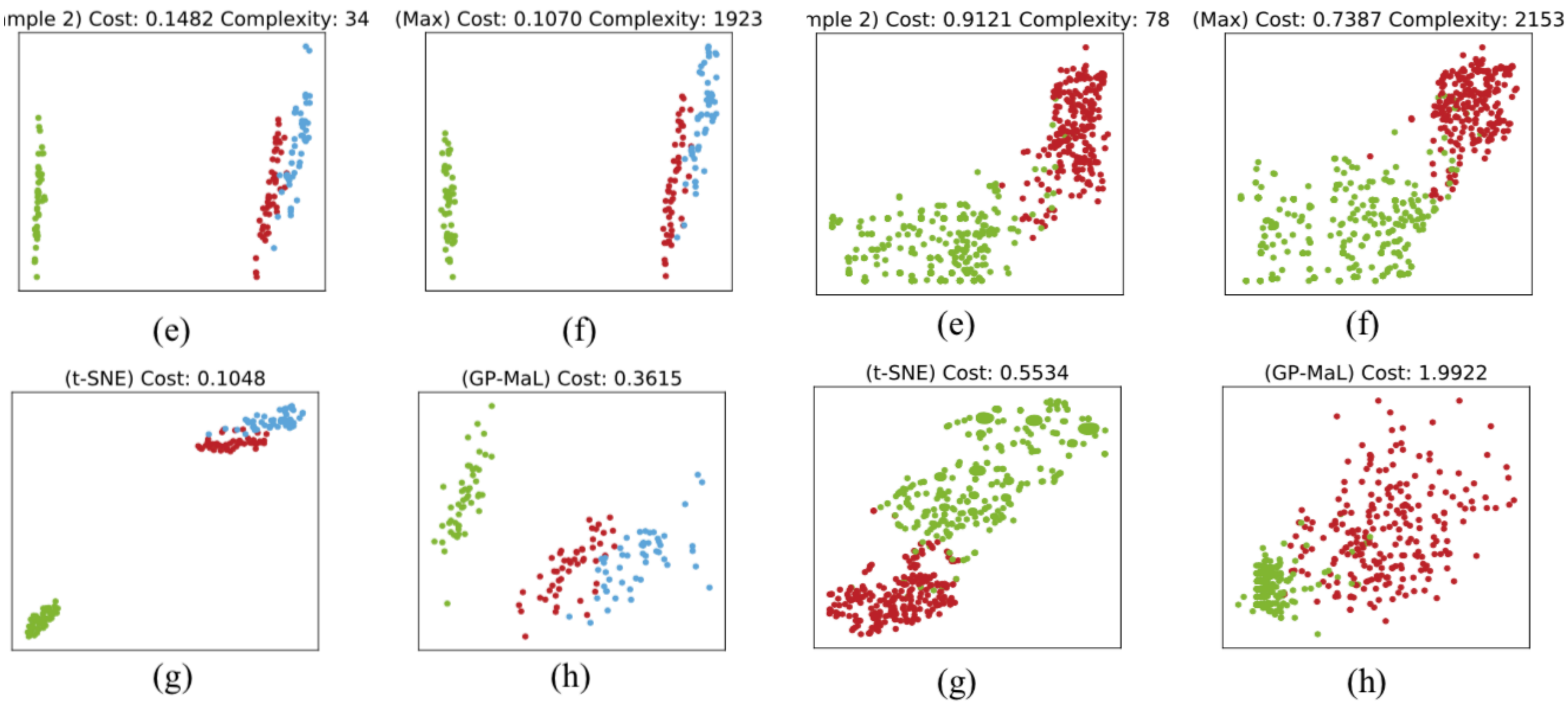
- T-distributed stochastic neighbour embedding (t-SNE) is a powerful manifold learning / dimensionality reduction method
- However, how t-SNE creates the visualisation from original features is opaque
- **GP-tSNE**: Multi-objective Multi-tree GP (MODA/D)
  - Terminal set: the features, mean of each feature and its 3 nearest neighbours, constants
  - Function set: similar as GP-MaL
  - Fitness function:
    - **F1**: t-SNE based: $KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$
    - $p_{j|i} = \frac{\exp\left(-\frac{\left\|x_i - x_j\right\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq l} \exp\left(-\frac{\left\|x_k - x_l\right\|^2}{2\sigma_i^2}\right)}, \quad p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}, \quad q_{ij} = \frac{\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_{k \neq l}\left(1 + \left\|y_k - y_l\right\|^2\right)^{-1}}$
    - **F2**: model complexity, count the number of nodes in the tree

- x's are in high-dimensional space
- y's are in low-dimensional space

Lensen et al. (2020) Genetic Programming for Evolving a Front of Interpretable Models for Data Visualization, IEEE Transactions on Cybernetics

GP with Visualisation

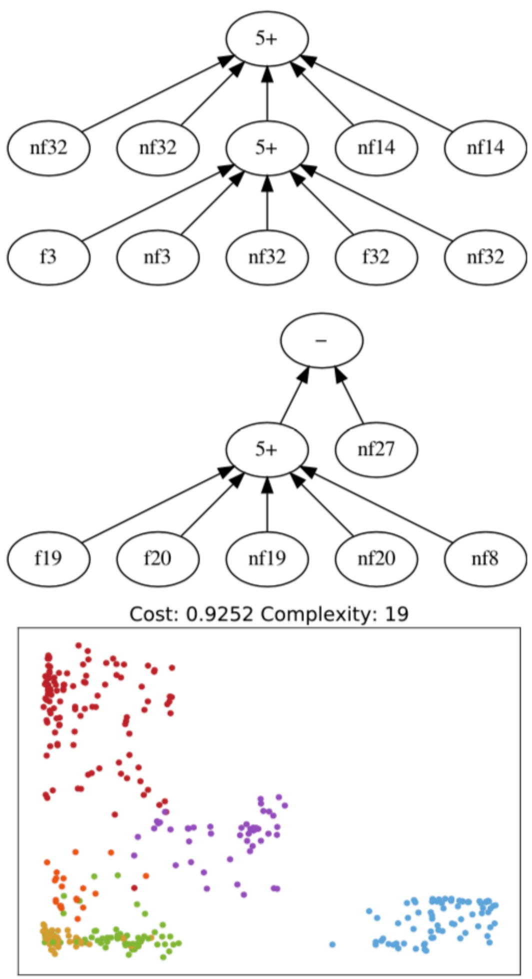
- Better results than GP-MaL, although may not be as good as t-SNE

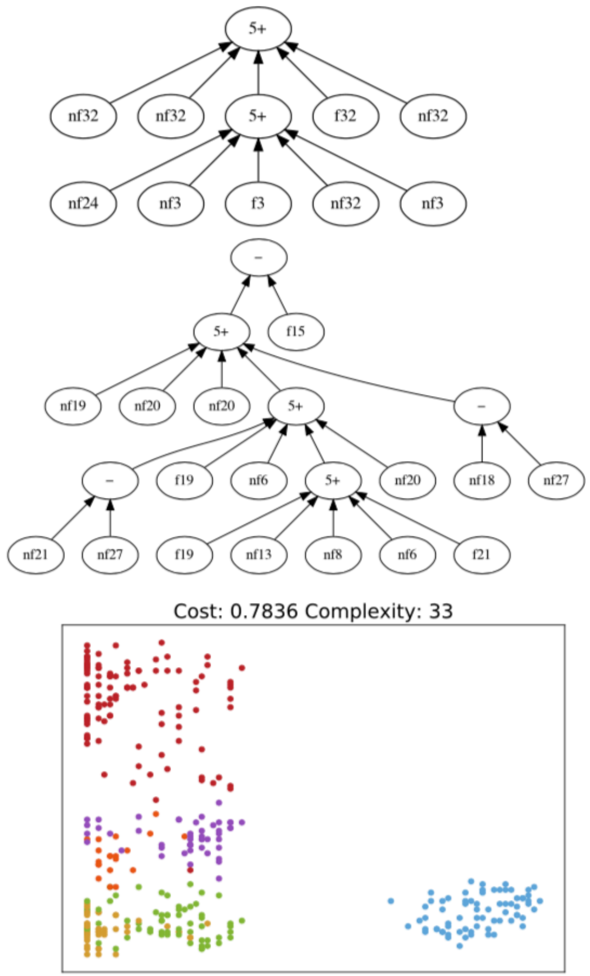


Lensen et al. (2020) Genetic Programming for Evolving a Front of Interpretable Models for Data Visualization, IEEE Transactions on Cybernetics

GP with Visualisation

- Good balance between cost and interpretability

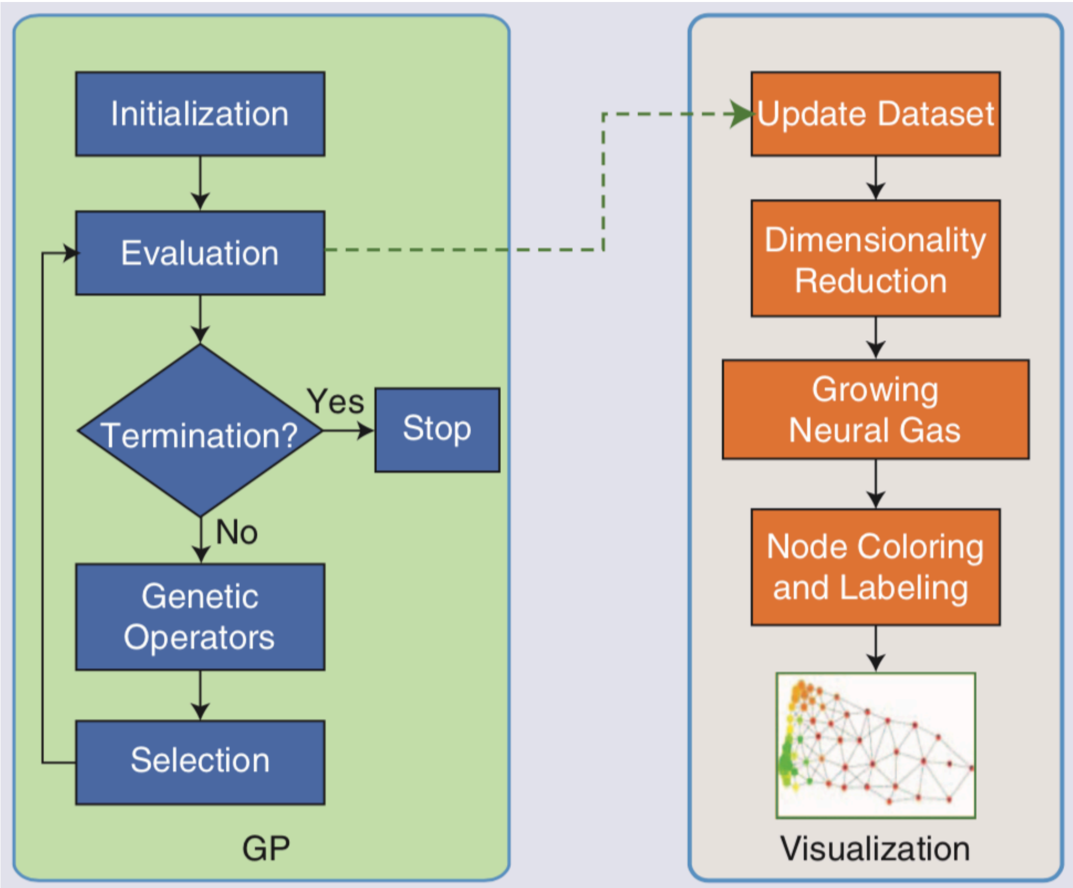


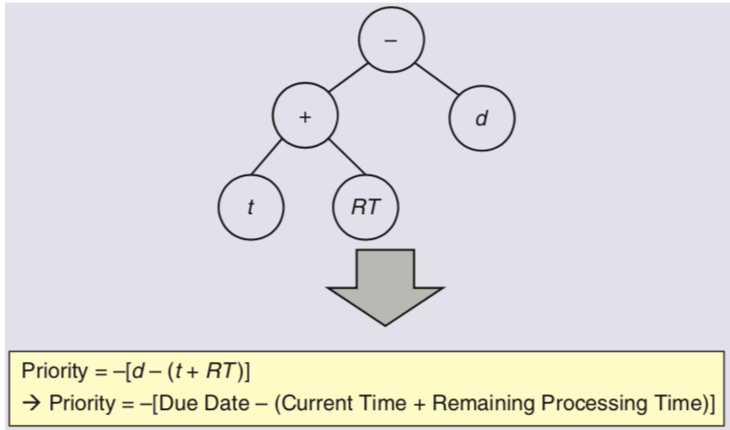


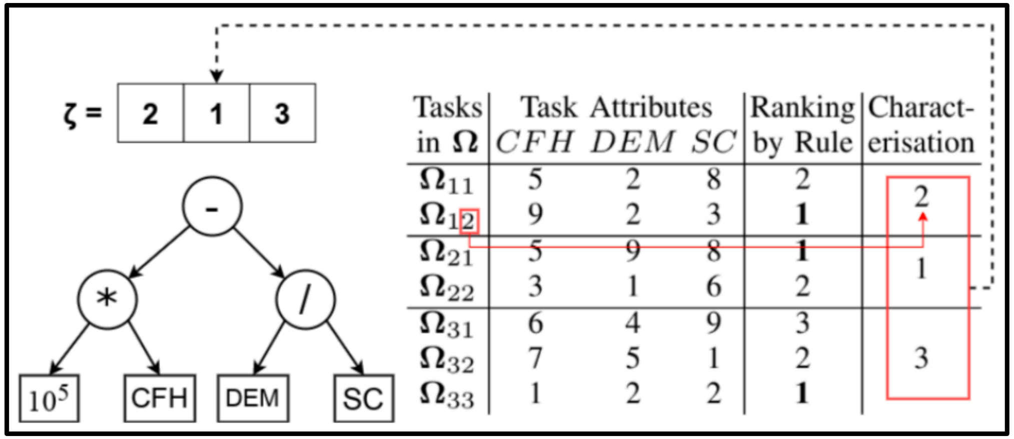
Lensen et al. (2020) Genetic Programming for Evolving a Front of Interpretable Models for Data Visualization, IEEE Transactions on Cybernetics

Visualising GP Process

- It is important to understand how the population evolves during the GP process
- Visualise the phenotypic representation using Growing Neural Gas Network



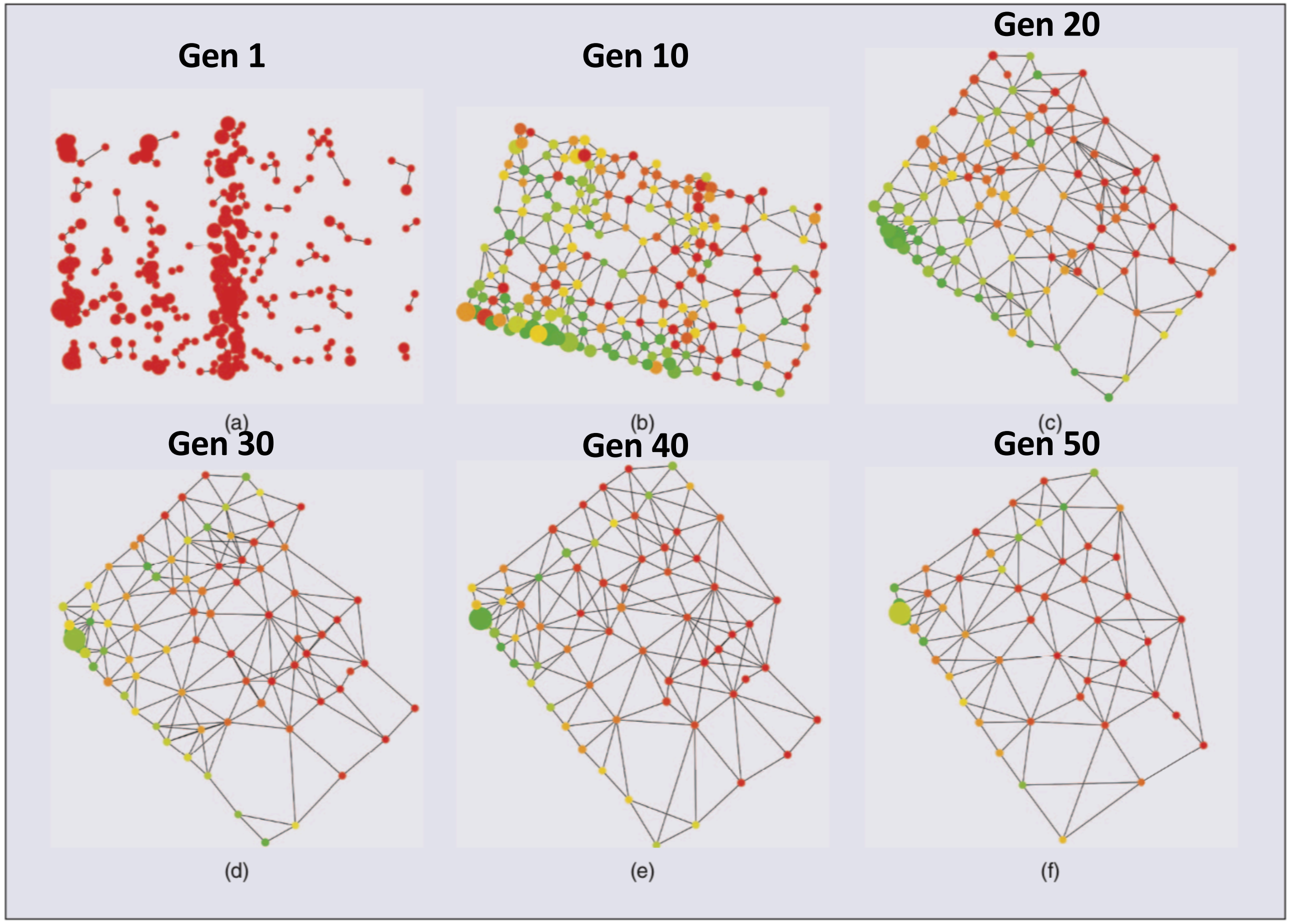




Nguyen, S., Zhang, M., Alahakoon, D., & Tan, K. C. (2018). Visualizing the evolution of computer programs for genetic programming [research frontier]. IEEE Computational Intelligence Magazine, 13(4), 77-94.

Visualising GP Process

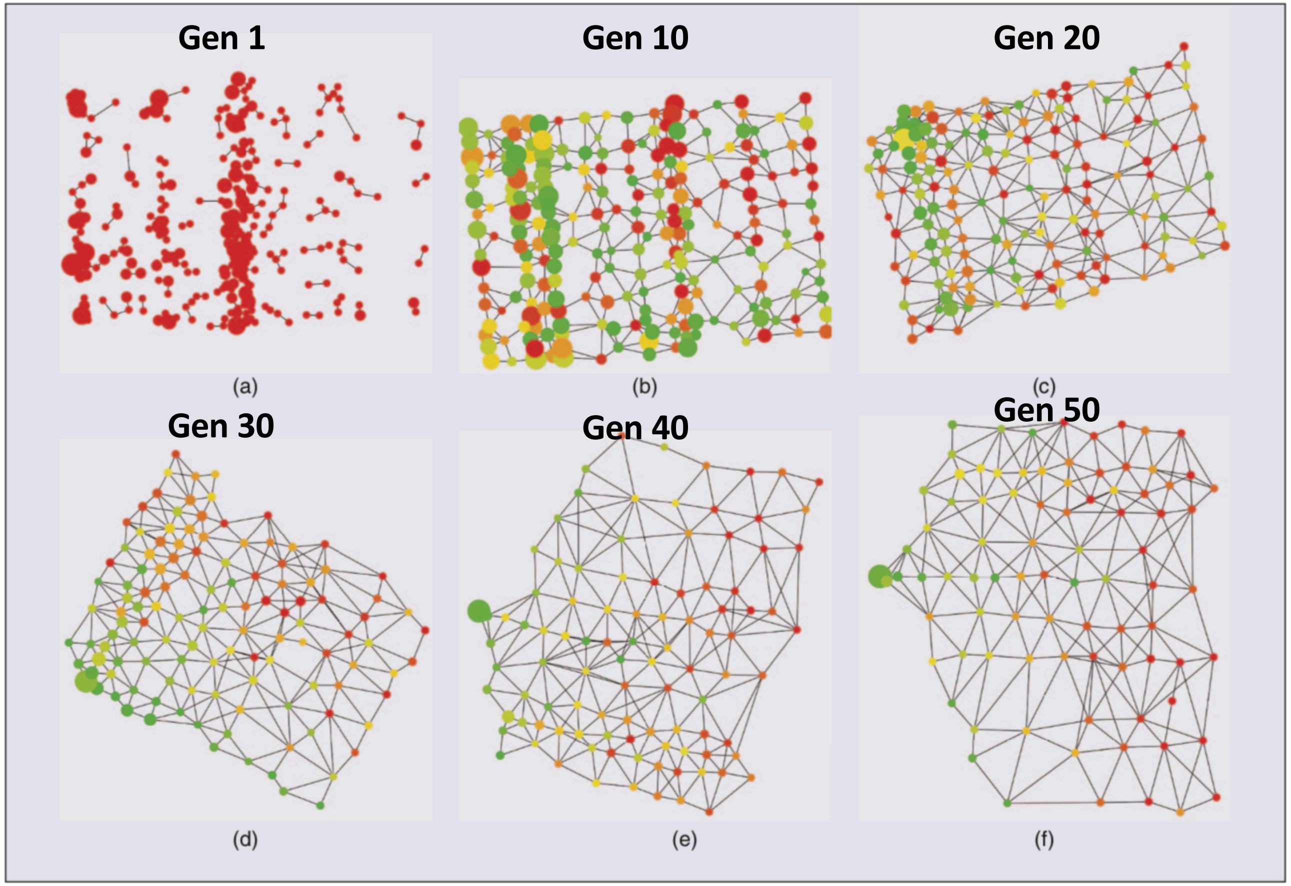
- Tree-based GP (80% crossover, 15% mutation)
  - Starts to show trend at gen 10
  - Quick converge since gen 20



Nguyen, S., Zhang, M., Alahakoon, D., & Tan, K. C. (2018). Visualizing the evolution of computer programs for genetic programming [research frontier]. IEEE Computational Intelligence Magazine, 13(4), 77-94.

Visualising GP Process

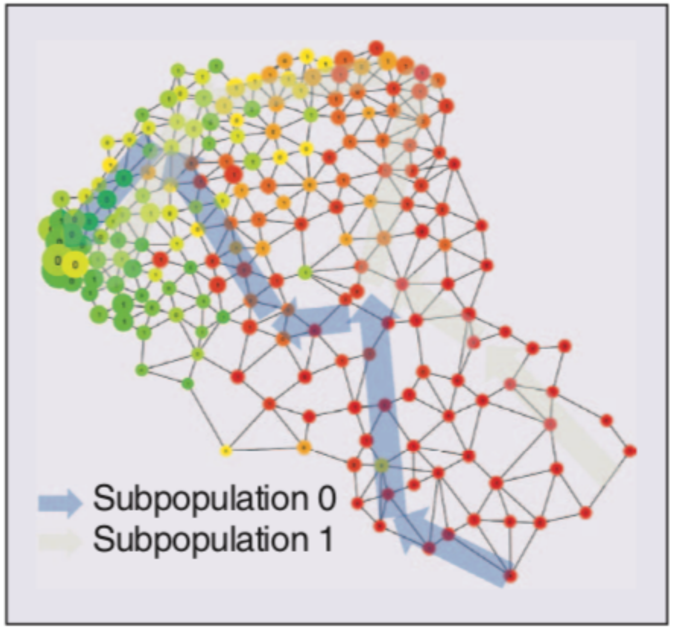
- Tree-based GP (15% crossover, 80% mutation)
  - Slower convergence
  - Still exploring at gen 30
  - Finally converge
- Higher mutation
- Higher exploration



Nguyen, S., Zhang, M., Alahakoon, D., & Tan, K. C. (2018). Visualizing the evolution of computer programs for genetic programming [research frontier]. IEEE Computational Intelligence Magazine, 13(4), 77-94.

Visualising GP Process

- Can also see how the population moves
- Can observe phenotypic and fitness diversity



TGP (80% crossover) **TGP (15% crossover)**

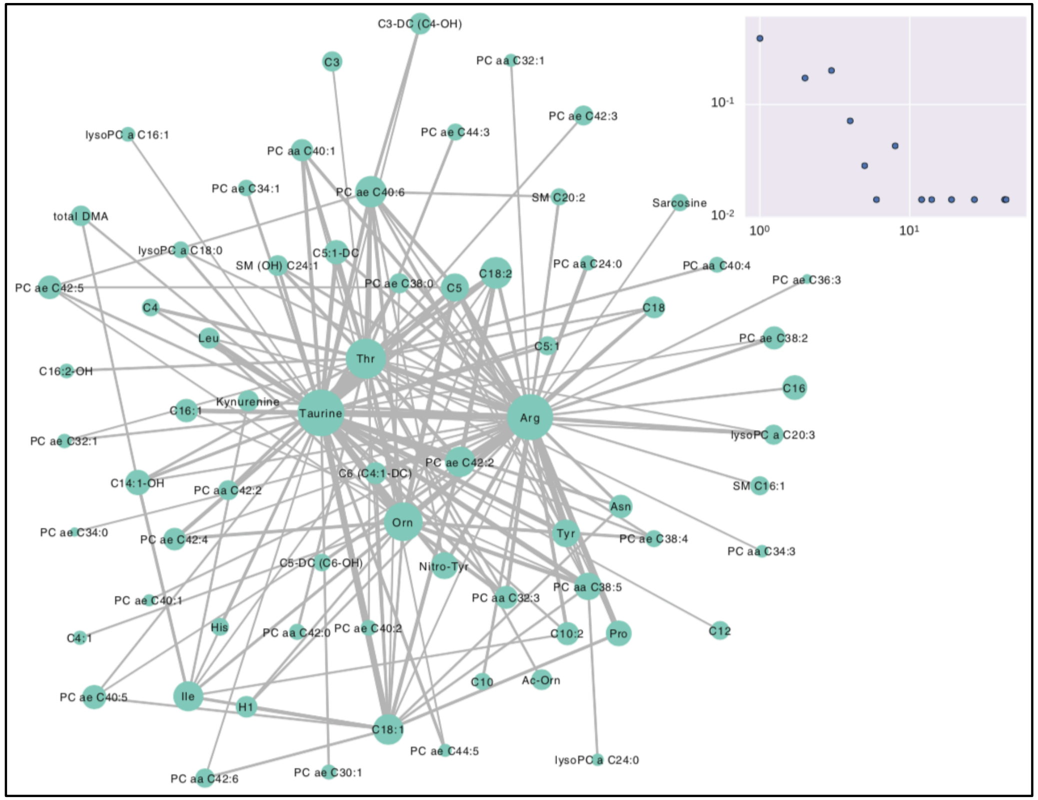
Darker nodes have better fitness

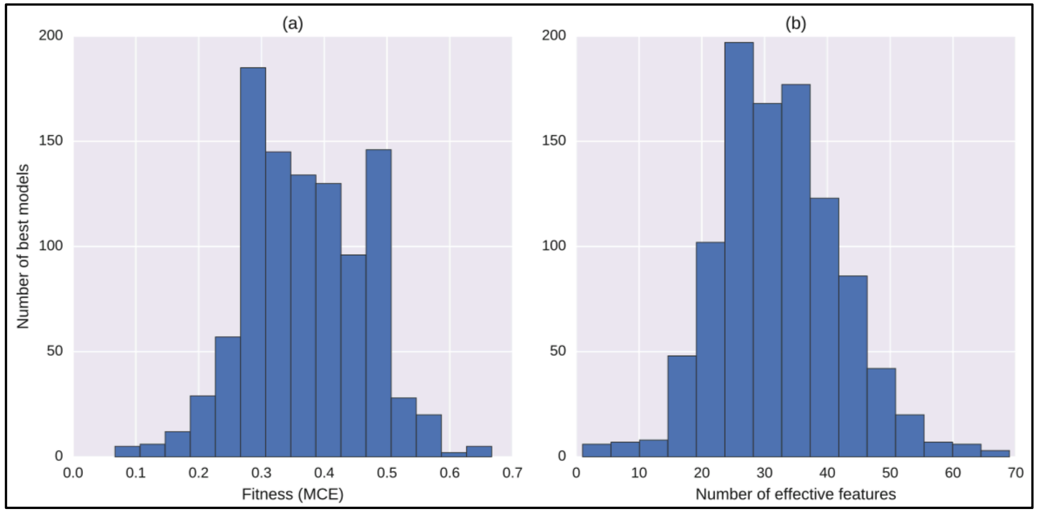
Nguyen, S., Zhang, M., Alahakoon, D., & Tan, K. C. (2018). Visualizing the evolution of computer programs for genetic programming [research frontier]. IEEE Computational Intelligence Magazine, 13(4), 77-94.

Linear GP

- Linear program, very similar to the real program, so easier to understand
- Test on bioinformatics: Metabolomics Data for Osteoarthritis
  - 167 features in total

```
I1:  if r[1]> r[3]
I2:     then r[0] = r[2] + 0.5
I3:  r[4] = r[2] / r[0]
I4:  if r[0] > 4
I5:     then if r[3] < 10
I6:             then r[5] = r[3] - r[4]
I7: r[4] = r[4] * r[1]
I8: r[0] = r[5] + r[4]
```





Hu, Ting. "Can Genetic Programming Perform Explainable Machine Learning for Bioinformatics?." Genetic Programming Theory and Practice XVII. Springer, Cham, 2020. 63-77.

Outline

- Introduction to XAI
- Introduction to GP
- Better Interpretability Through GP
- **Challenges and Future Directions**

Challenges and Future Directions

- **Measures** of Interpretability
  - Why (How much) is A more interpretable/explainable than B? – subjective
  - Questionnaire/Interview?
- **Forms** of interpretability, e.g., contrastive: why event P happened *instead of* another event Q?
- **Cross-disciplinary**: understand interpretability
  - Cognitive science
  - Social science
  - Psychology
  - …
- **Tradeoff** between Interpretability and Accuracy

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. Artificial intelligence, 267, 1-38.

Conclusions

- Interpretable AI Techniques
  - Intrinsic interpretability
  - Post-hoc explanation
- Global Interpretation: interpret the whole model
- Local Interpretation: interpret for a specific instance
- GP has a great potential for XAI
  - Symbolic + computational
  - Flexible representation
  - Multi-objective