COMP422 — Week 10

Advanced Topics in Evolutionary Computation

Dr Bing Xue

School of Engineering and Computer Science

Victoria University of Wellington

Bing.Xue@ecs.vuw.ac.nz

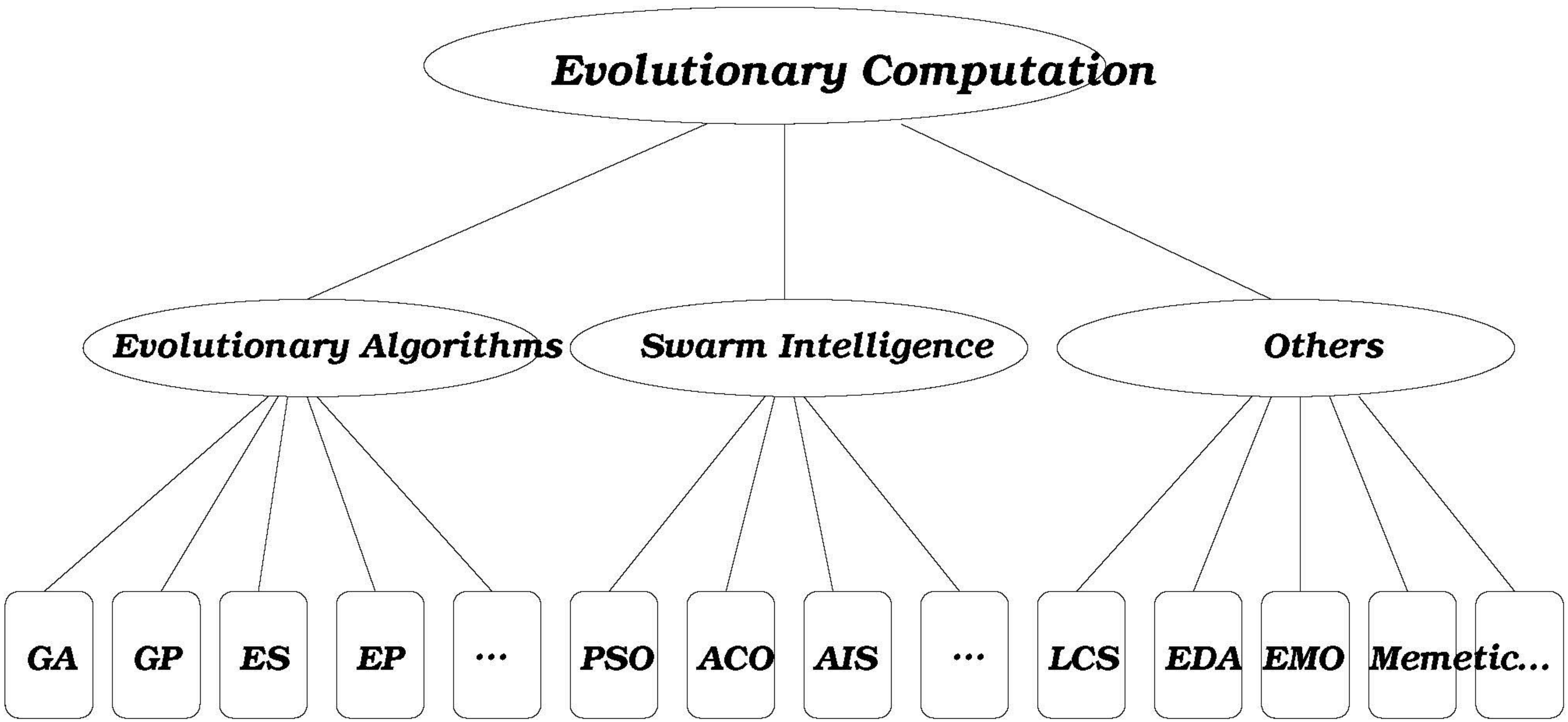
Outline

- Introduction: Evolutionary Computation
- Why Evolutionary Computation ?
- Strongly typed Genetic Programming
- Memetic Algorithms
- Differential Evolution
- Evolutionary Multi-objective Optimisation
  - Dr Yi Mei, next week

- Suggested Readings

Evolutionary Computation (EC)

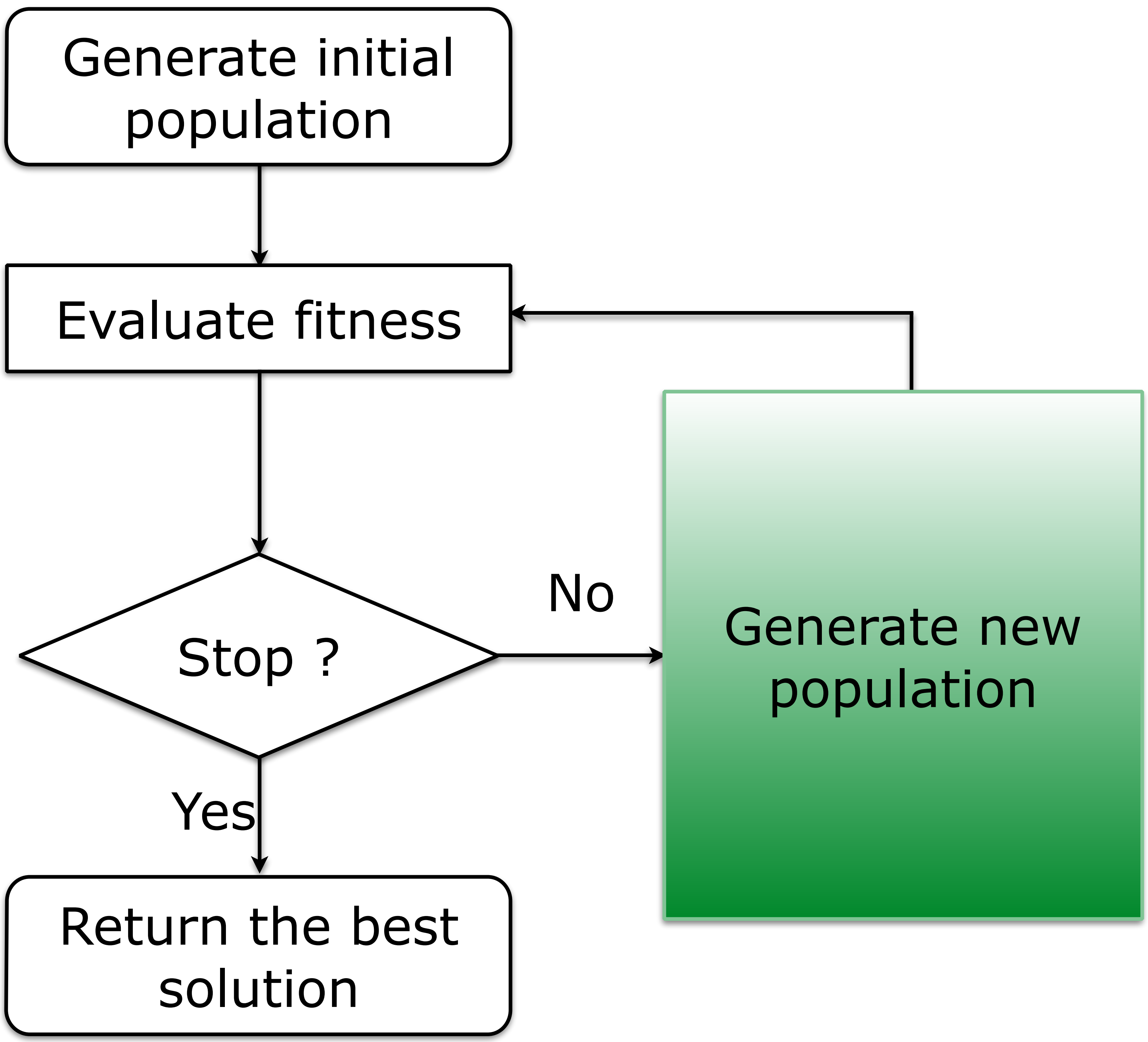
- A group of techniques inspired by the principles of biological evolution



Why Evolutionary Computation ?

- Don't need domain knowledge
- Don’t make any assumption
  - e.g. differentiable, linearity, separability, equality
  - EC can deal with non-differentiable, non-continuous, non-linear, noisy, flat, multi-dimensional, many local minima, constraints or stochasticity
- Easy to handle constraints
- EC can simultaneously build model structures and optimise parameters
- Population based search is particularly suitable for multi-objective optimisation

Flowchart of an EC method



Strongly Typed GP

- Evolved programs vs human-written programs
  - In standard GP systems, the inputs and outputs of the programs have the same type: float/double in general.
  - The type of terminals, the type of function arguments, and the return type of a function are the same — closure property.
  - Human written programs use different data types and different data structures.
  - Human written programs are more powerful.

- Can GP systems evolve computer programs with different data types, or automatically evolve different data structures?

Strongly Typed GP

- Montana[] introduced data types to GP and called the system Strongly Typed Genetic Programming – STGP.
- In STGP, each element in the primitive set is associated with a given data type, just as in the human-written programs.
  - The return type of a node can be any type which can be used as an argument type to its parent node.
  - In case the node is the root, the type returned is a solution of the problem.
  - Each function has the argument types associated with each element of its parameter list, constraining the types which subtrees of this function can take on.

Genetic Operators in STGP

- Crossover and mutation are restricted.
- Crossover:
  - In the first parent: starts by picking a random subtree
  - How to chose the crossover point for the second parent? – Randomly? restricted?
- Mutation:
  - How to select mutation point?
  - How to generate a replacement subtree? – Randomly? restricted?
- These constrains ensure type safety of the program.
- How about the reproduction?

Generic types in STGP

- Generic types are equivalence classes in a type system.
- Any member of one equivalence class may stand in for any other members of that class and the program will still be type safe.
- Example: VEC-2, VEC-3, ..., VEC-n can be replaced with VEC-k for a number of functions such as ADD, DOT-PRODUCT.

- Here VEC-k is a generic class with a parameterised size argument.

Properties of STGP

- STGP works by cutting down the search space:
  - Program generation process is restricted.
  - Closure in un-typed GP: any function is well-defined for all possible values that could be returned by other functions or terminals.
  - Closure in STGP is restricted to a particular data type: any function that needs to take an argument should consider the return type of the argument.
  - This immediately cuts down the number of branches by constraining the tree construction process.

Properties of STGP

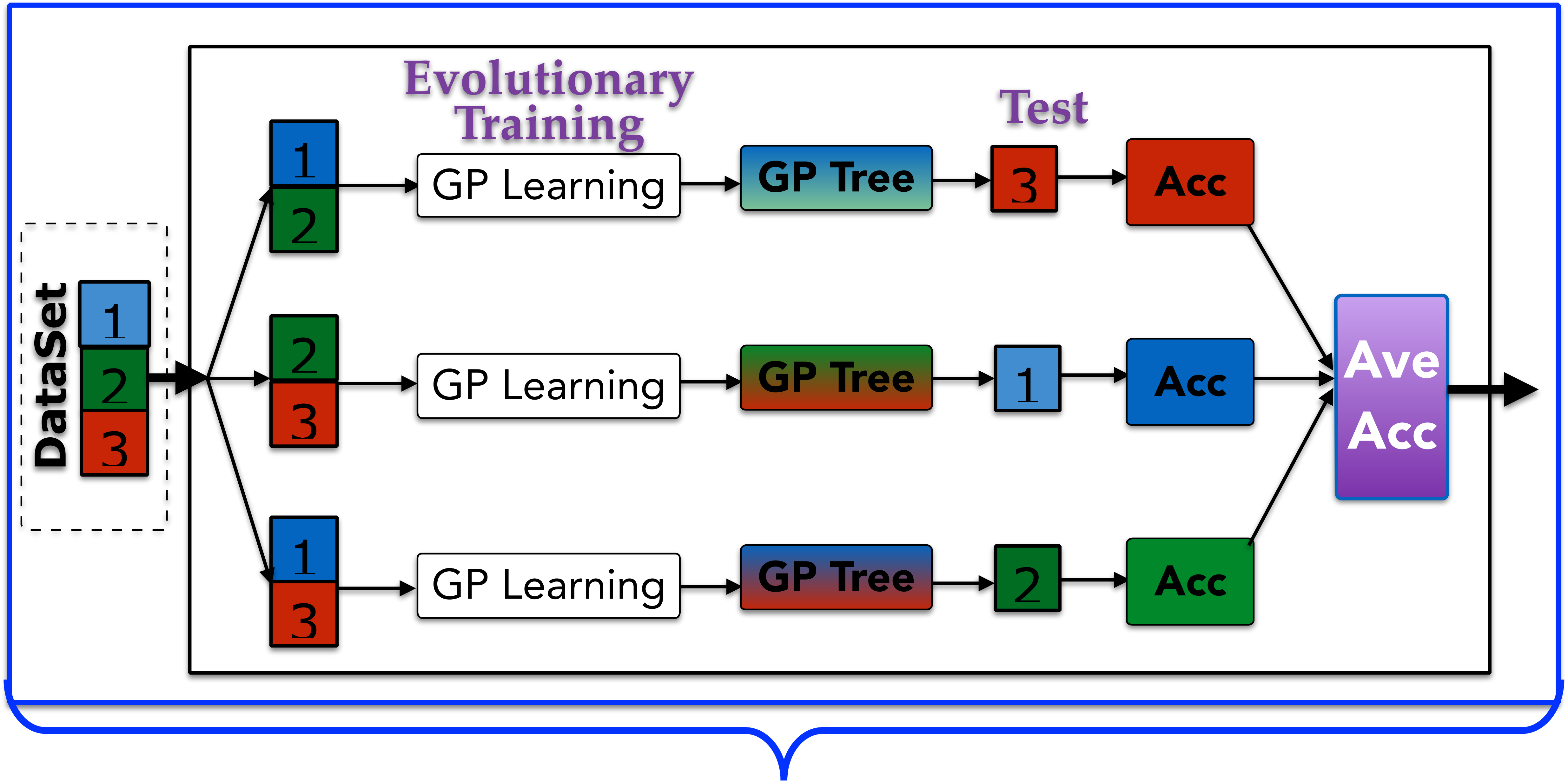
- Each primitive can do a lot more “meaningful” work in the space of a single node.
  - Such work usually needs a large subtree to implement in an un-typed GP system.
  - Solutions obtained in this way can be often much smaller than in un-typed GP.
- Program evolved by STGP is generally easier to interpret.
  - In general GP, it usually needs to get the value of the fitness or program output.
  - The programs evolved are some kinds of transformation, not an algorithm.
  - STGP can produce “meaningful” results.
  - In STGP, the program domain can match the problem domain very closely.
- STGP can evolve complex data structures.

Problems of STGP

- Terminals and functions have to be carefully designed to be consistently matched.
- It is very difficult to define good fitness (evaluation) functions, even for relatively simple problems.
- A STGP system is usually a domain dependent system/ method.
- The performance of such a system will be even worse than standard GP systems if primitive set and fitness were not properly defined.

K-fold Cross Validation with GP

- Example: 3-fold cross validation in GP for classification



Repeat this process *N (N>=30)* times with *N* different random seeds for GP

Same as any other stochastic method

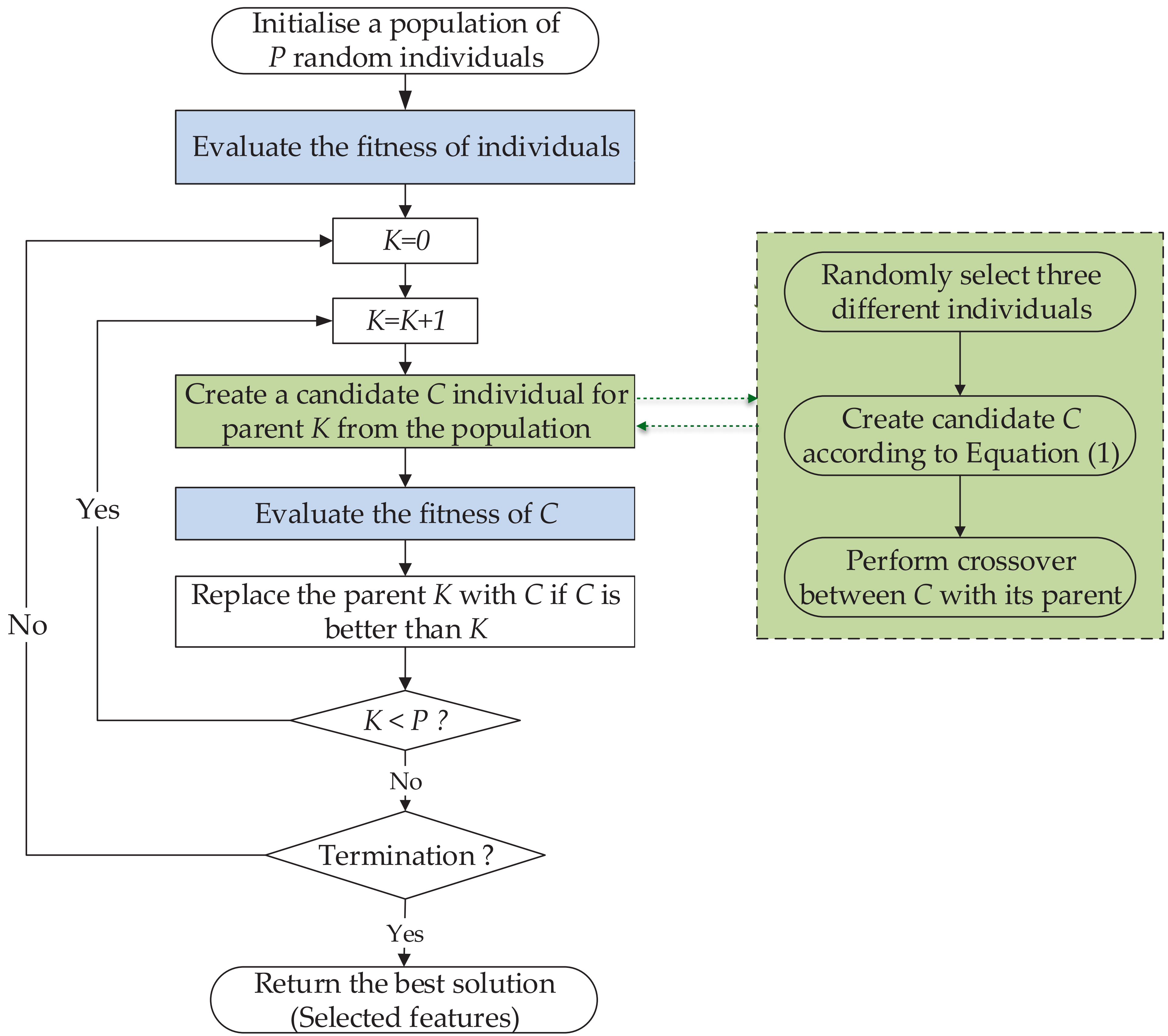
Differential Evolution (DE)

- Differential evolution (DE) is an EC technique, first developed in 1997
- DE employs the mutation operation to produce a mutant candidate solution
- There are different types of mutation strategies
- **DE/rand/1** as an example:

$$C_{id} = \begin{cases} x_{id}^{i,r1} + F * (x_{d}^{i,r2} - x_{id}^{i,r3}), \text{ if } rand() < CR \\ x_{id}, \ otherwise \end{cases}$$

- *F* in (0,1), *CR* is the crossover rate

Differential Evolution (DE)



Five most frequently used DE mutation schemes

"DE/rand/1": $\vec{V}_i(t) = \vec{X}_{r_1^i}(t) + F \cdot (\vec{X}_{r_2^i}(t) - \vec{X}_{r_3^i}(t)).$

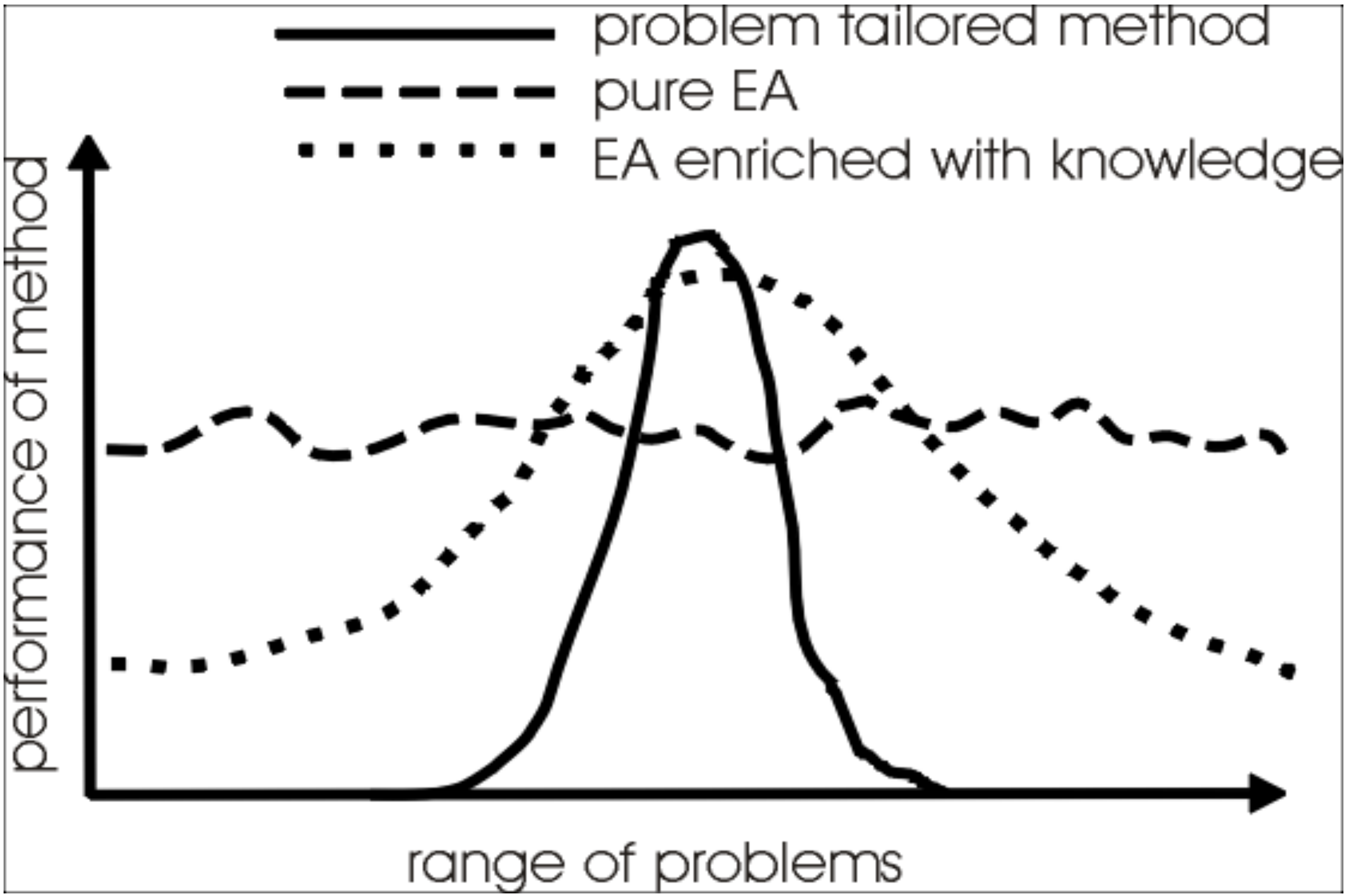
"DE/best/1": $\vec{V}_i(t) = \vec{X}_{best}(t) + F.(\vec{X}_{r_1^i}(t) - \vec{X}_{r_2^i}(t)).$

"DE/target-to-best/1": $\vec{V}_i(t) = \vec{X}_i(t) + F.(\vec{X}_{best}(t) - \vec{X}_i(t)) + F.(\vec{X}_{r_1^i}(t) - \vec{X}_{r_2^i}(t)),$

"DE/best/2": $\vec{V}_i(t) = \vec{X}_{best}(t) + F.(\vec{X}_{r_1^i}(t) - \vec{X}_{r_2^i}(t)) + F.(\vec{X}_{r_3^i}(t) - \vec{X}_{r_4^i}(t)).$

"DE/rand/2": $\vec{V}_i(t) = \vec{X}_{r_1^i}(t) + F_1.(\vec{X}_{r_2^i}(t) - \vec{X}_{r_3^i}(t)) + F_2.(\vec{X}_{r_4^i}(t) - \vec{X}_{r_5^i}(t)).$

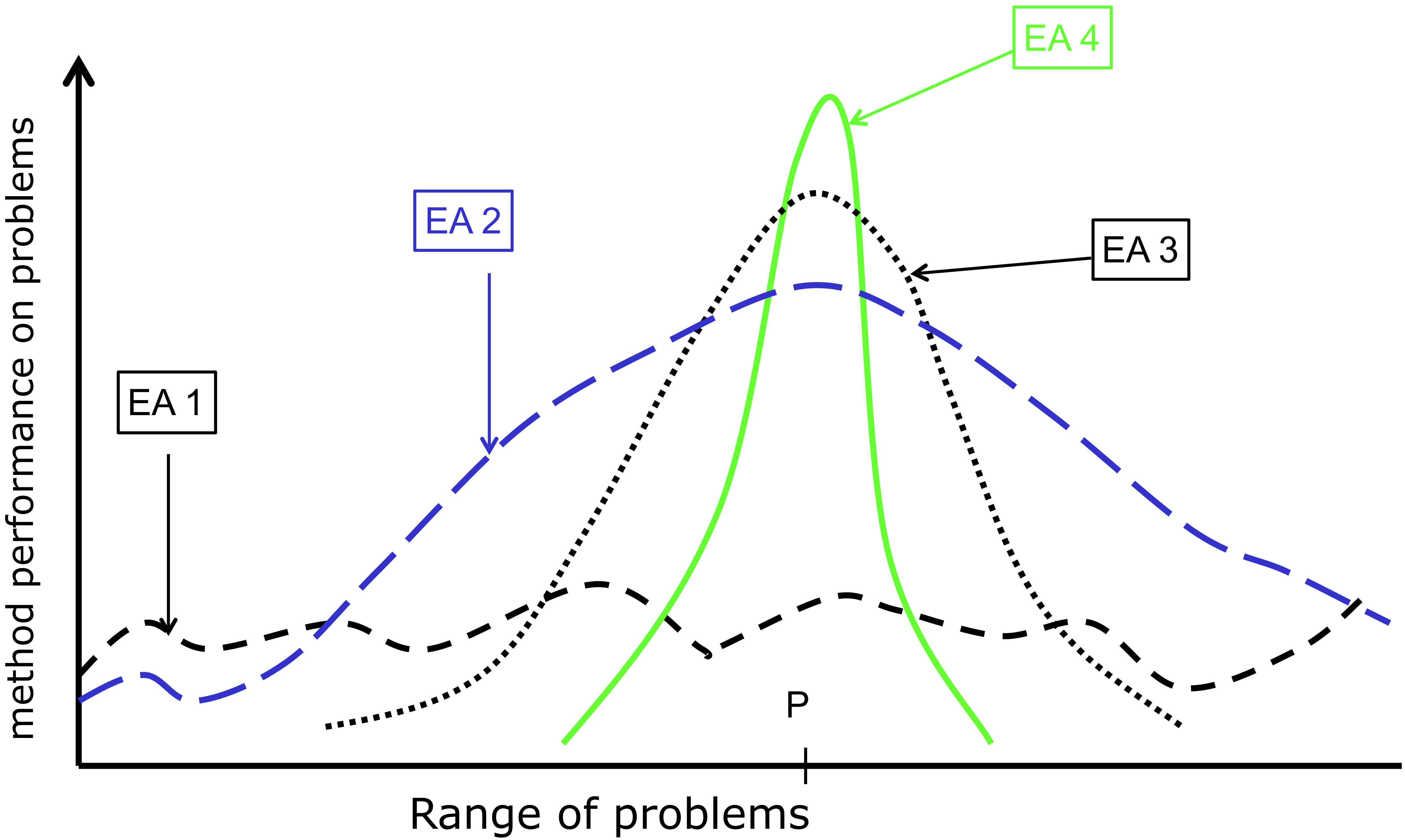
Evolutionary Algorithms and domain knowledge



Michalewicz's view on EAs in context

A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing Hybridisation with other techniques: Memetic Algorithms

Michalewicz's Interpretation



Evolutionary Algorithms and Domain Knowledge

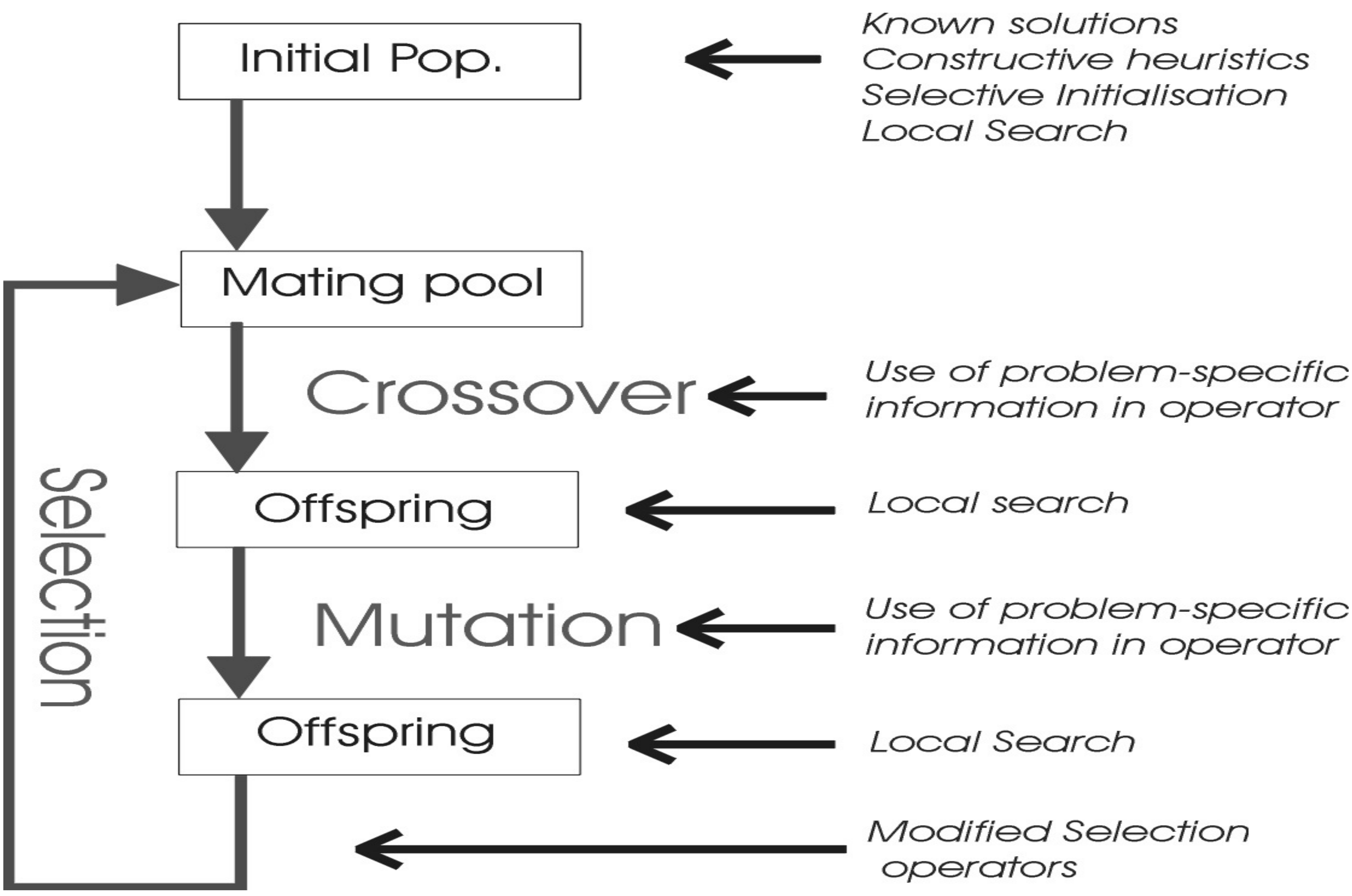
- Fashionable after the 90's:

  to add problem specific information into the EAs by means of specialised crossover, mutation, representations, and local search

- Result: The performance curve deforms and
  - makes EAs better in some problems,
  - worst on other problems
  - amount of problem specific is varied.

- Hybridised Evolutionary Computation
  - EA vs. EC

Where to Hybridise ?



Memetic Algorithms (MAs)

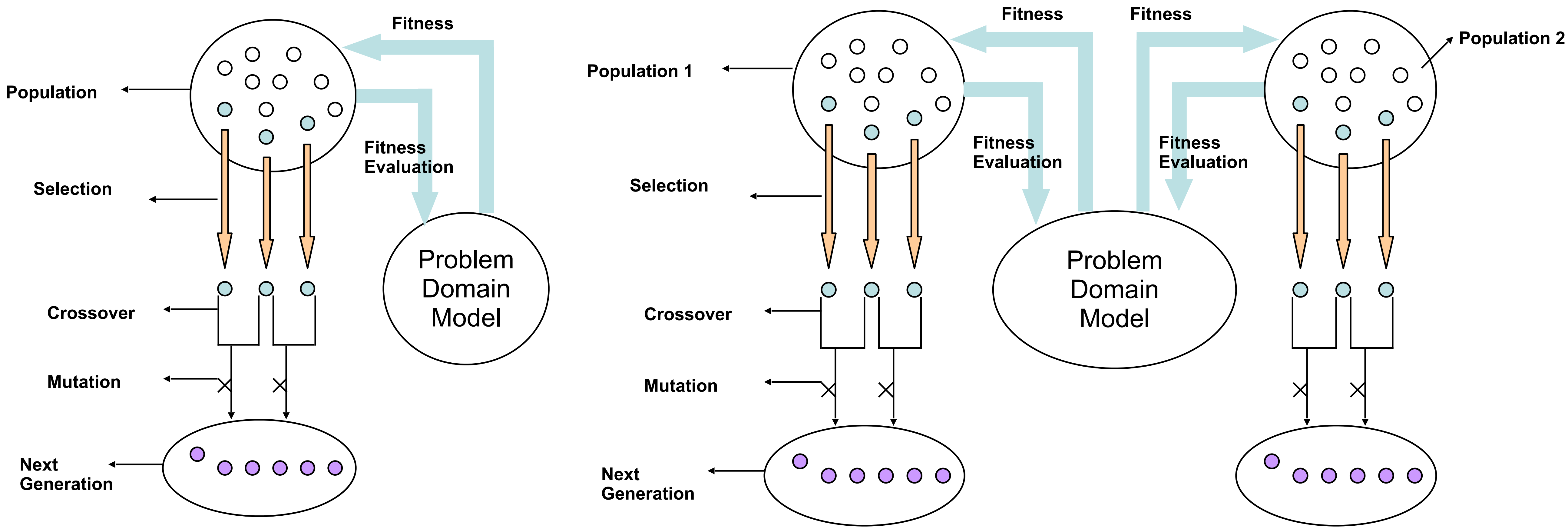
- Memetic Algorithms:
  - MAs are carefully orchestrated interplay between (stochastic) global search and (stochastic) local search algorithms
  - **GA + local search**
  - More general now: EC + local search
- Adding Domain Knowledge to EAs

Why Memetic Algorithms (MAs) ?

- Complex problems can be partially decomposable:
  - different subproblems be better solved by different methods
  - Subproblem specific information can be placed into variation operators or into local searchers
  - In some cases there are exact/approximate methods for subproblems
- EC is good at exploring the search space but find it difficult to zoom-in good solutions
- Problems have constraints associated to solutions and heuristics/ local search are used to repair solutions found by EC
- If heuristic/local search strategies in MAs are "first class citizens" then one can raise the level of generality of the algorithm without sacrificing performance by letting the MA choose which local search to use.
- ***hyper-heuristic***

Coevolution

- Cooperative coevolution
- Competitive coevolution



Other Important Topics

- Evolutionary Multi-objective optimisation
- Evolutionary Many-objective optimisation
- Learning classifier systems
- Large-scale optimisation
- Dynamic optimisation
- Constraint optimisation
- Artificial immune systems
- hybrid EC with other methods

- EC for DM and ML
- Real-world applications

IEEE CIS Evolutionary Computation Pioneer award

- 2016: Marco Dorigo
- 2015: Thomas Bäck
- 2014: George Burgin
- 2013: Xin Yao
- 2012: Russell C. Eberhart and James Kennedy
- 2012: J.David Schaffer
- 2011: Larry J. Eshelman
- 2010: John Greffenstette
- 2009: David E. Goldberg
- 2008: David B. Fogel
- 2005: Kenneth De Jong
- 2004: Richard Friedberg
- 2003: John H. Holland
- 2002: Ingo Rechenberg
- 2002: Hans-Paul Schwefel
- 2001: Michael Conrad
- 2000: George Box
- 1999: Alex S. Fraser
- 1998: Lawrence J. Fogel

IEEE Transactions on Evolutionary Computation Outstanding Paper Award (data until 2015)

- Urvesh Bhowan, Mark Johnston, Mengjie Zhang and Xin Yao, "Evolving Diverse Ensembles Using Genetic Programming for Classification with Unbalanced Data," Vol. 17, No. 3, 2013, pp. 368-386.
- Oliver Schütze, Xavier Esquivel, Adriana Lara, and Carlos A. Coello Coello, "Using the Averaged Hausdorff Distance as a Performance Measure in Evolutionary Multi-Objective Optimization," Vol. 16, No. 4, August 2012, pp. 504-522.
- Adriana Lara, Gustavo Sanchez, Carlos A. Coello Coello and Oliver Schütze, "HCS: A New Local Search Strategy for Memetic Multiobjective Evolutionary Algorithms", Vol. 14, No. 1, February 2010, pp. 112-132.
- A. K. Qin, V. L. Huang and P.N. Suganthan, "Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization", Vol. 13, No. 2, April 2009, pp. 398-417.
- Q. H. Nguyen, Y. S. Ong and M. H. Lim, "A Probabilistic Memetic Framework" Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization", Vol. 13, No. 3, June 2009, pp. 604-623.
- S. Y. Chong, P. Tino and X. Yao, "Measuring generalization performance in coevolutionary learning," Vol. 12, No. 4, August 2008, pp. 479-505.
- Qingfu Zhang and Hui Li, "MOEA/D: a multi-objective evolutionary algorithm based on decomposition," Vol. 11, No.6, December 2007, pp. 712-731.

IEEE Transactions on Evolutionary Computation Outstanding Paper Award (data until 2015)

- Joshua Knowles, "ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization", Vol. 10, No. 1, February 2006, pp. 50-66.
- C. Blum and M. Dorigo, Search Bias, "Ant-Colony Optimization: On the role of Competition-balanced Systems", Vol. 9, No. 2, April 2005, pp. 159-174.
- A. Pruegel-Bennett, "Symmetry breaking in population-based optimization," Vol. 8, No. 1, February 2004, pp. 63-79.
- J. Knowles and D. Corne, "Properties of an adaptive archiving algorithm for storing nondominated vectors," Vol. 7, No. 2, April 2003, pp. 100-116.
- M. Clerc and J. Kennedy, "The particle swarm explosion: stability and convergence in a multi-dimensional complex space," Vol. 6, No. 1, February 2002, pp. 58-73.
- A. Sierra, J. A. Macias and F Corbacho, "Evolution of functional link networks," Vol. 5, No. 1, February 2001, pp. 54-65.
- C. Dimopoulous and A. M. S. Zalzala, "Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons," Vol. 4, No. 2, July 2000, pp. 93-113.
- A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," Vol. 3, No. 2, July 1999, pp. 124-141.

IEEE CIS Task Forces

- TF1: Theoretical Foundations of Bio-inspired Computation, Chair: Pietro Oliveto, Vice Chairs: Per Kristian Lehre and Frank Neumann
- TF2: Differential Evolution, Chair: Janez Brest, Vice Chairs: Ponnuthurai N Suganthan and Ferrante Neri
- TF3: Swarm Intelligence, Chair: Yuhui Shi, Vice Chairs: Marco Dorigo and Xiaodong Li
- TF4: Artificial Immune Systems, Chair: Mario Pavone, Vice Chair: Carlos A. Coello Coello
- TF5: Cultural Algorithms, Chair: Robert Reynolds, Vice Chairs: Yuhui Shi and Gary Yen
- TF6: Artificial Life and Complex Adaptive Systems, Chair: Chrystopher Nehaniv, Vice Chairs: Terry Bossomaier and Hiroki Sayama
- TF7: Evolutionary Algorithms Based on Probabilistic Models, Chair: John McCall, Vice Chairs: Qingfu Zhang and Aimin Zhou
- TF8: Evolutionary Computation in Dynamic and Uncertain Environments (ECiDUE), Chair: Shengxiang Yang, Vice Chairs: David Pelta and Changhe Li

IEEE CIS Task Forces

- TF9: Evolutionary Computer Vision and Image Processing, Chair: Sergio Damas, Vice Chairs: Mario Köppen and Mengjie Zhang
- TF10: Evolutionary Scheduling and Combinatorial Optimization, Chair: Su Nguyen, Vice Chairs: Rong Qu and Mark Johnston
- TF11: Evolvable Hardware, Chair: Andy Tyrrell, Vice Chairs: Martin Trefzer and Lukas Sekanina
- TF12: Large Scale Global Optimization, Chair: Daniel Molina, Vice Chair: Ponnuthurai N Suganthan
- TF13: Nature-Inspired Constrained Optimization, Chair: Efren Mezura-Montes, Vice Chair: Helio Barbosa
- TF14: Multi-Objective Evolutionary Algorithms, Chair: Sanaz Mostaghim, Vice Chairs: Katya Rodriguez-Vazquez and Andrew Lewis
- TF15: Evolutionary Computation for Feature Selection and Construction, Chair: Bing Xue, Vice Chairs: Yaochu Jin and Mengjie Zhang
- TF16: Data-Driven Evolutionary Optimization of Expensive Problems, Chair: Chaoli Sun, Vice Chairs: Jonathan Fieldsend and Yew-Soon Ong