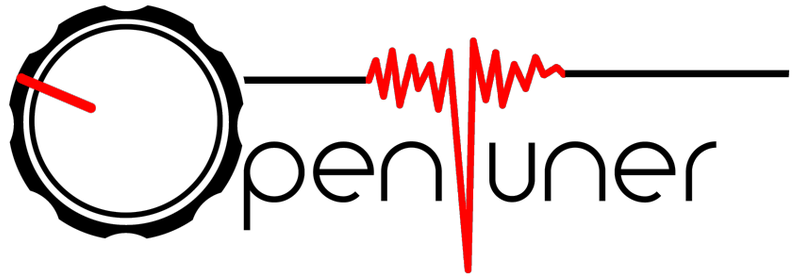


Search and Machine Learning

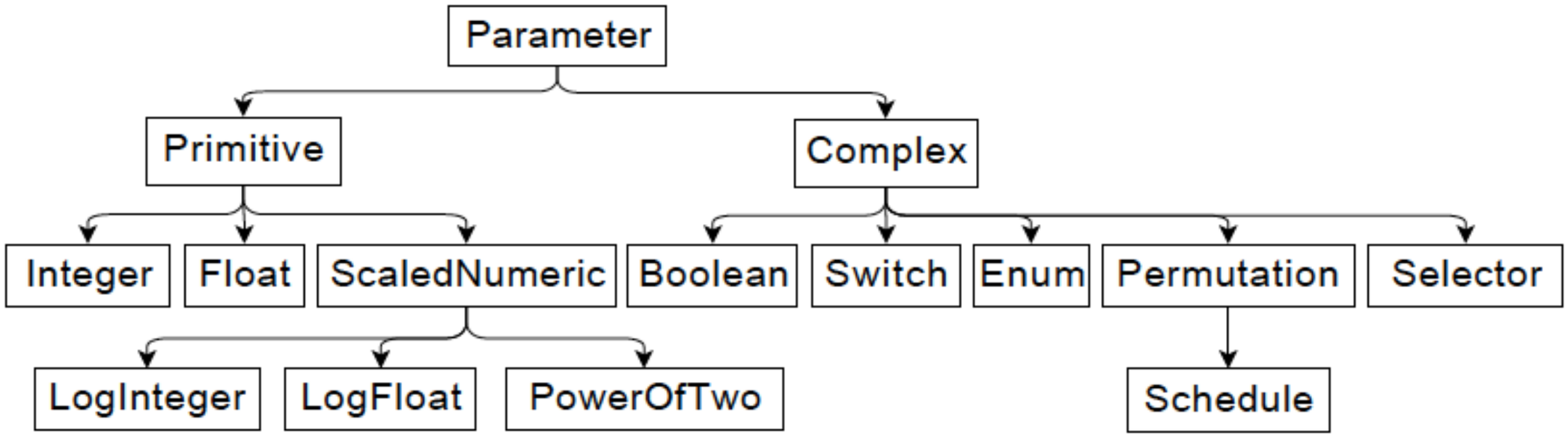
Kalyan Veeramachaneni, Jason Ansel, Shoaib Kamil,
Jeffrey Bosboom, Una-May O'Reilly,
Saman Amarasinghe

CGO Tutorial
February 8th, 2015

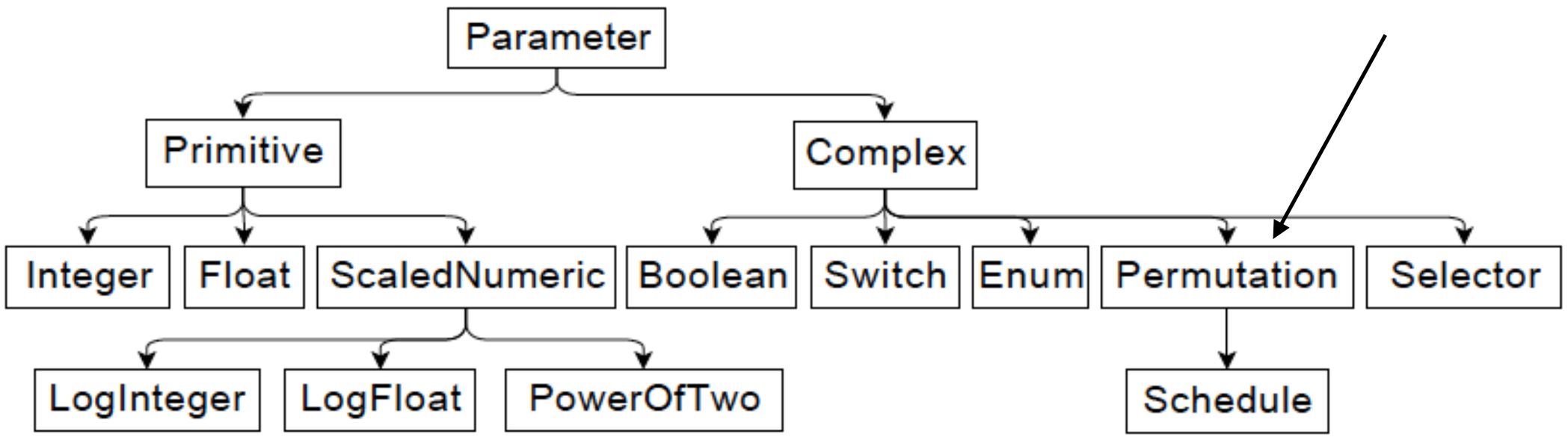
Overview

- Parameter types and tuning
- An example tuning problem - permutation
- A typical population based search process
  - Select-Create-Update process
- Options
  - Create —> Operators
  - Select - Update
- Composition of multiple search approaches
- Steps to take for a new problem
- Back to Mario example

Parameter types



Parameter types




```
for c_x:
  for b_x:
    for b_y:
      for a_x:
        for a_y:
          compute_a()
        compute_b()
  for c_y:
    compute_c()
```

```
c_x
b_x
x_y
a_x
a_y
compute_a
compute_b
c_y
compute_c
```

Example:
Permute the placement
of these computations

A typical population based search process

Initialize | Evaluate | Select | Create | Evaluate | Update

1	2	3	4	5	6	7	8	9

9	6	4	7	3	2	5	1	8

1	6	2	5	3	9	8	7	4

⋮

2	1	5	3	9	7	4	6	8

8	5	7	1	2	4	9	3	6

⋮

6	2	5	3	7	4	1	8	9

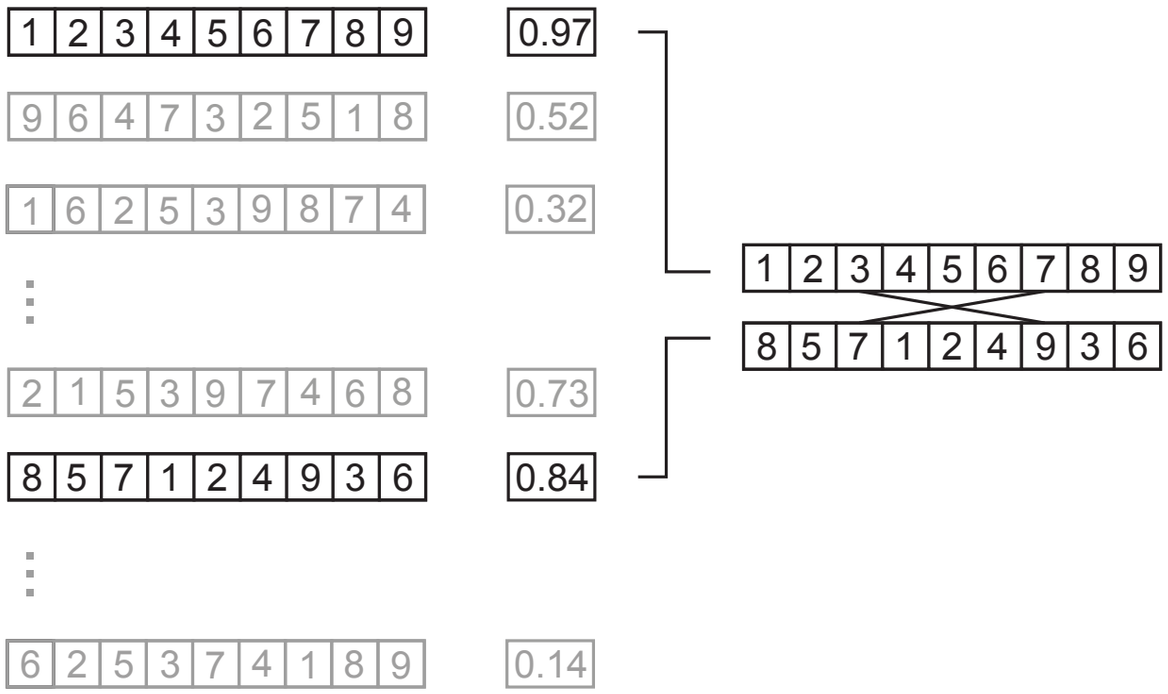
A typical population based search process

Initialize | **Evaluate** | Select | Create | Evaluate | Update

1	2	3	4	5	6	7	8	9	0.97
9	6	4	7	3	2	5	1	8	0.52
1	6	2	5	3	9	8	7	4	0.32
⋮									
2	1	5	3	9	7	4	6	8	0.73
8	5	7	1	2	4	9	3	6	0.84
⋮									
6	2	5	3	7	4	1	8	9	0.14

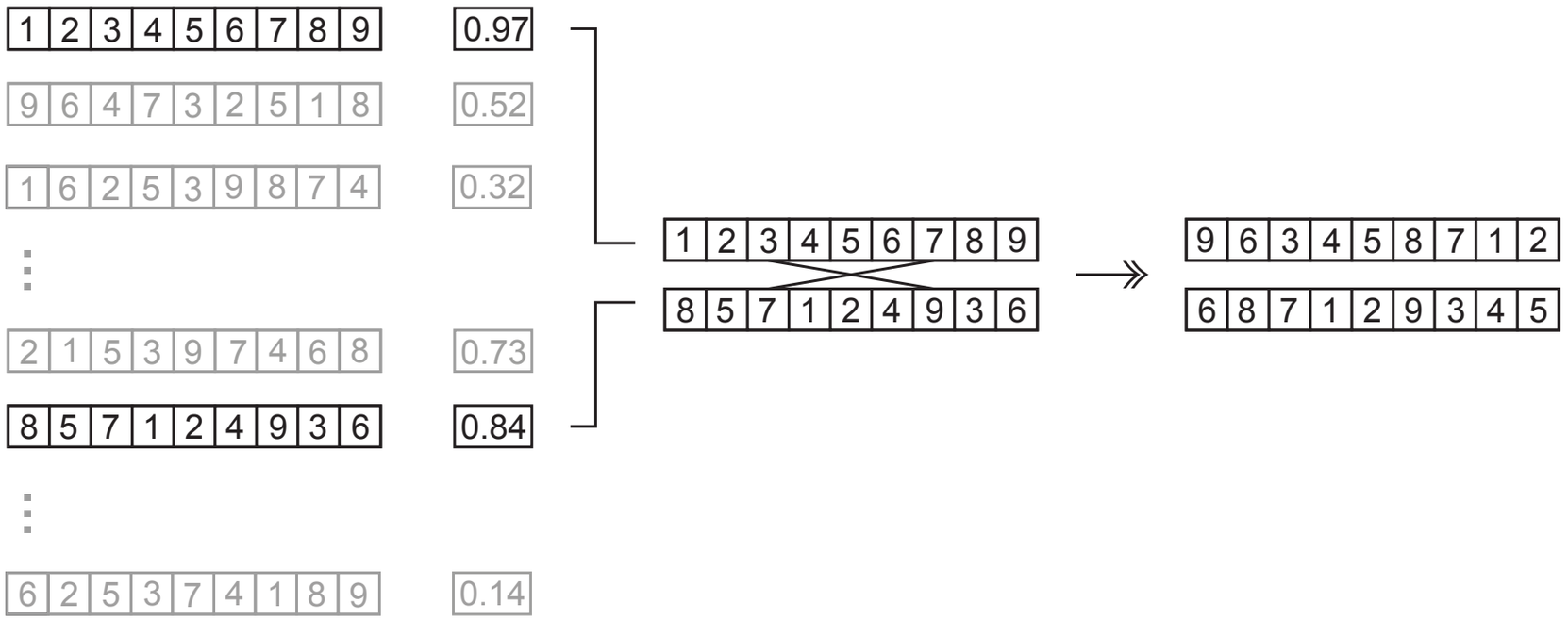
A typical population based search process

Initialize Evaluate **Select** Create Evaluate Update



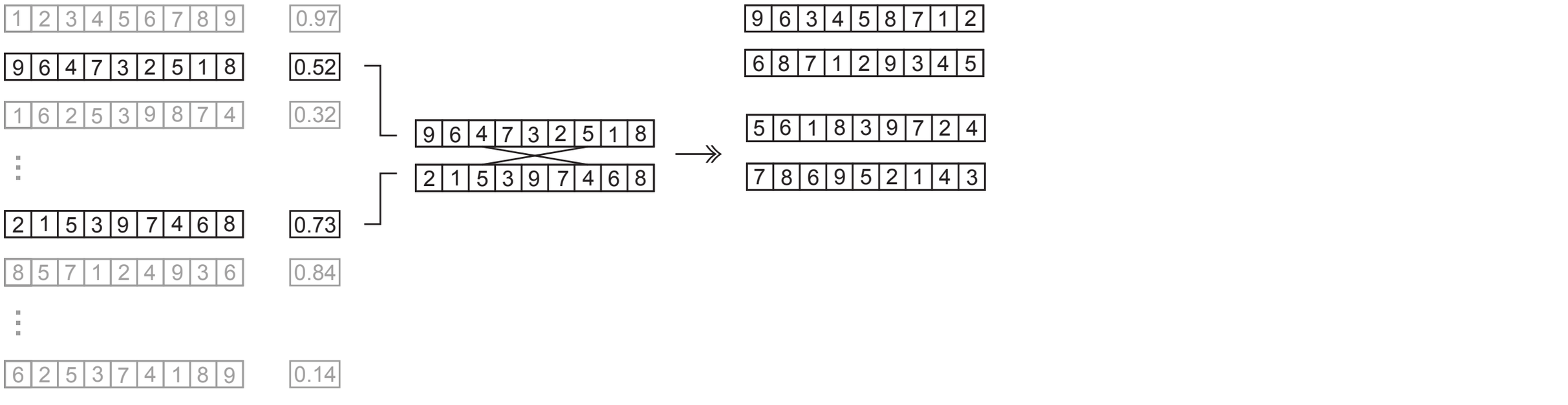
A typical population based search process

Initialize Evaluate Select **Create** Evaluate Update



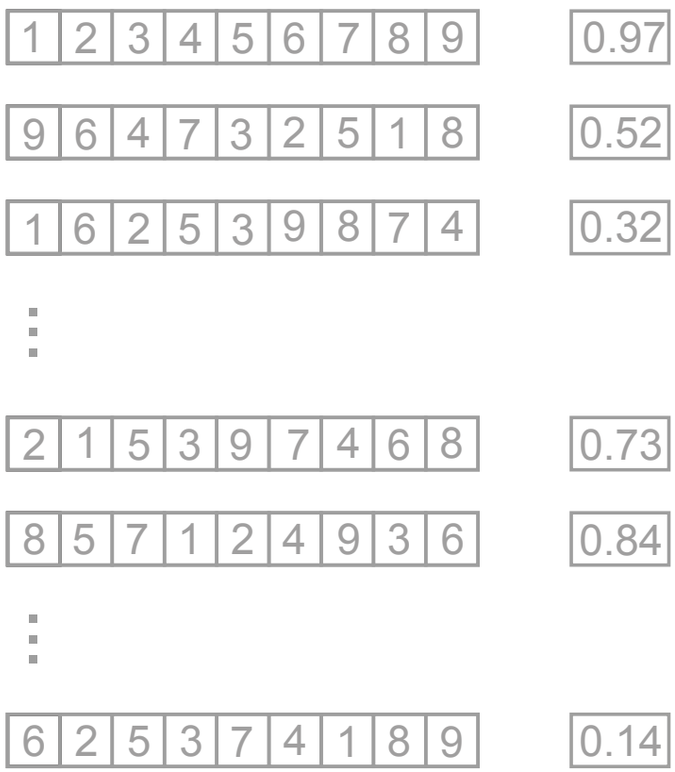
A typical population based search process

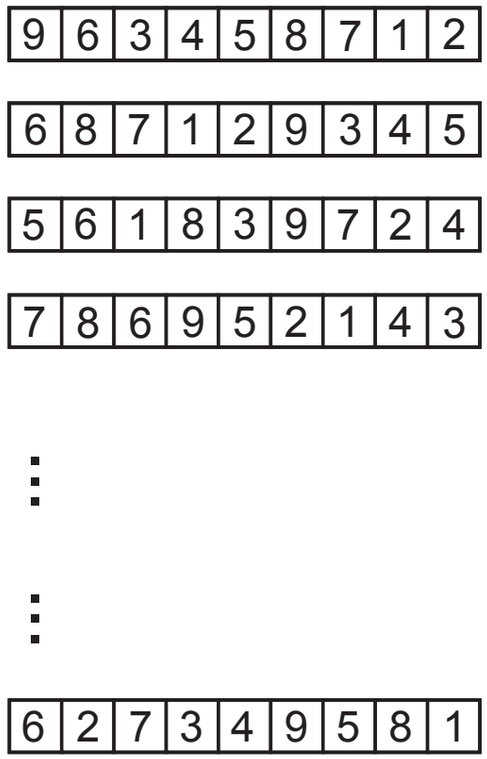
Initialize Evaluate Select Create Evaluate Update



A typical population based search process

Initialize → Evaluate → Select → Create → Evaluate → Update





A typical population based search process

Initialize Evaluate Select Create **Evaluate** Update

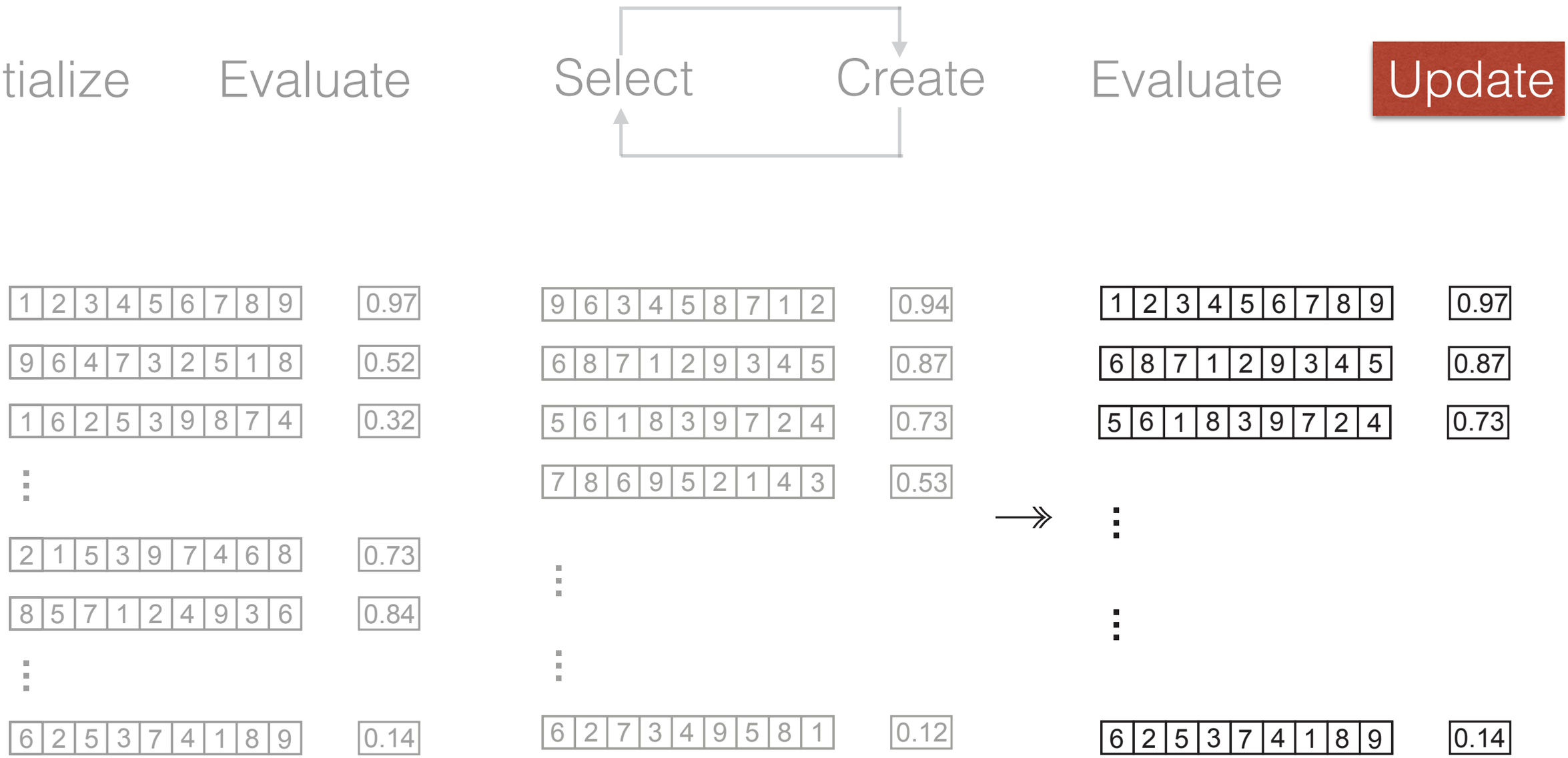
1 2 3 4 5 6 7 8 9	0.97
9 6 4 7 3 2 5 1 8	0.52
1 6 2 5 3 9 8 7 4	0.32
⋮	
2 1 5 3 9 7 4 6 8	0.73
8 5 7 1 2 4 9 3 6	0.84
⋮	
6 2 5 3 7 4 1 8 9	0.14

→

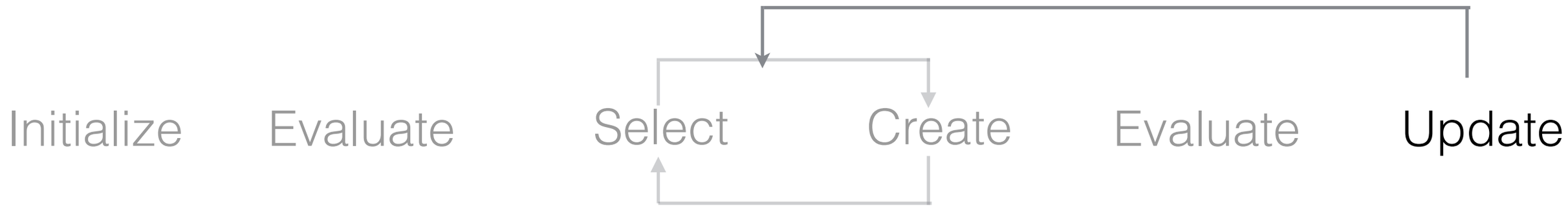
9 6 3 4 5 8 7 1 2	0.94
6 8 7 1 2 9 3 4 5	0.87
5 6 1 8 3 9 7 2 4	0.73
7 8 6 9 5 2 1 4 3	0.53
⋮	
⋮	
6 2 7 3 4 9 5 8 1	0.12

A typical population based search process

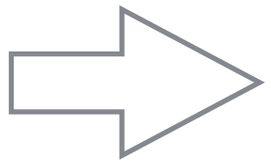
Initialize Evaluate Select Create Evaluate **Update**



A typical population based search process



1	2	3	4	5	6	7	8	9	0.97
6	8	7	1	2	9	3	4	5	0.87
5	6	1	8	3	9	7	2	4	0.73
⋮									
⋮									
6	2	5	3	7	4	1	8	9	0.14



Continue

Create —> Requires operators

- Operators take two or more parameters and create two or more new parameters
- Let's take the permutation example:

```
1 2 3 4 5 6 7 8 9
8 5 7 1 2 4 9 3 6
```

← Input

```
1 2 | 3 4 5 | 6 7 8 9
8 5 | 7 1 2 | 4 9 3 6
```

← cross over points

Create —> Requires operators

- Operators take two or more parameters and create two or more new parameters
- Let's take the permutation example:

1 2 3 4 5 6 7 8 9
8 5 7 1 2 4 9 3 6 ← Input

1 2 | 3 4 5 | 6 7 8 9
8 5 | 7 1 2 | 4 9 3 6 ← cross over points

_ _ |3 4 5| _ _ _ _ ← Copy over a portion of good material from #1

Create —> Requires operators

- Operators take two or more parameters and create two or more new parameters
- Let's take the permutation example:

1 2 3 4 5 6 7 8 9
8 5 7 1 2 4 9 3 6 ← Input

1 2 | 3 4 5 | 6 7 8 9
8 5 | 7 1 2 | 4 9 3 6 ← cross over points

_ _ |3 4 5| _ _ _ _ ← Copy over a portion of good material from #1

Create —> Requires operators

- Operators take two or more parameters and create two or more new parameters
- Let's take the permutation example:

1 2 3 4 5 6 7 8 9
8 5 7 1 2 4 9 3 6 ← Input

1 2 | 3 4 5 | 6 7 8 9
8 5 | 7 1 2 | 4 9 3 6 ← cross over points

9

_ _ |3 4 5| _ _ _ _ ← Copy over a portion of good material from #1

Create —> Requires operators

- Operators take two or more parameters and create two or more new parameters
- Let's take the permutation example:

1 2 3 4 5 6 7 8 9
8 5 7 1 2 4 9 3 6

← Input

1 2 | 3 4 5 | 6 7 8 9
8 5 | 7 1 2 | 4 9 3 6

← cross over points

9 6

_ _ |3 4 5| _ _ _ _

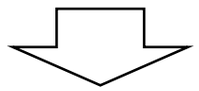
← Copy over a portion of good material from #1

Create —> Requires operators

- Operators take two or more parameters and create two or more new parameters
- Let's take the permutation example:

1 2 3 4 5 6 7 8 9
8 5 7 1 2 4 9 3 6 ← Input

1 2 | 3 4 5 | 6 7 8 9
8 5 | 7 1 2 | 4 9 3 6 ← cross over points



9 6 8

_ _ |3 4 5| _ _ _ _ ← Copy over a portion of good material from #1

Create —> Requires operators

- Operators take two or more parameters and create two or more new parameters
- Let's take the permutation example:

1 2 3 4 5 6 7 8 9
8 5 7 1 2 4 9 3 6 ← Input

1 2 | 3 4 5 | 6 7 8 9
8 5 | 7 1 2 | 4 9 3 6 ← cross over points

⇩

9 6 8 7 1 2

_ _ |3 4 5| _ _ _ _ ← Copy over a portion of good material from #1

Create —> Requires operators

- Operators take two or more parameters and create two or more new parameters
- Let's take the permutation example:

1 2 3 4 5 6 7 8 9
8 5 7 1 2 4 9 3 6 ← Input

1 2 | 3 4 5 | 6 7 8 9
8 5 | 7 1 2 | 4 9 3 6 ← cross over points

⇩

9 6 8 7 1 2

⇩

9 6 | 3 4 5 | 8 7 1 2 ← Copy over a portion of good material from #1

Create:
Several options exist for operators

- For permutation parameter there are several operators
  - Partially mapped crossover
  - Partition crossover
  - Ordered crossover
  - Edge crossover
  - Cycle crossover

Select-update

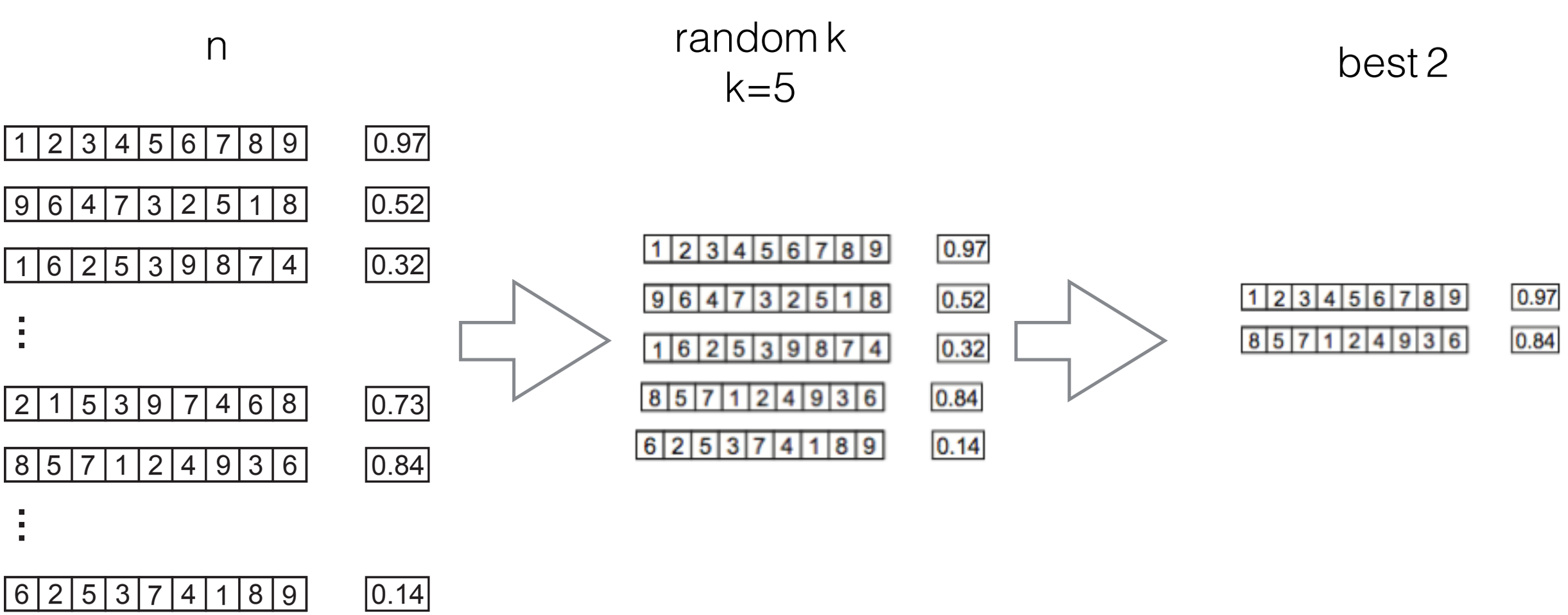
- Select —> chooses the solutions among the population from which new solutions will be created
  - biasing the search towards to better solutions
- Update—> updates the entire population towards better search spaces
  - has stronger influence on convergence

Particle Swarm Optimization (PSO)
Select-update

- State based—> for each member of the population, a history is maintained
- Select —> Individual based
  - for every individual select its previous best
  - select the best solution seen so far
- Update—> only update individuals history if it finds a better solution in the search space

Genetic Algorithms
Select-update

- Select —> Tournament selection
  - select randomly k from n
  - among these k select the top 2
  - allows enough mixing



Genetic Algorithms
Select-update

- Update—> multiple ways that allow us to control exploration and exploitation
  - strong elitism
    - combine both old and new and select the top n
  - weak elitism
    - do it on a per individual basis, select if the new one it created is better than itself.

Composition of multiple approaches

GA - Genetic algorithms
PSO- Particle swarm optimization
DE- Differential evolution

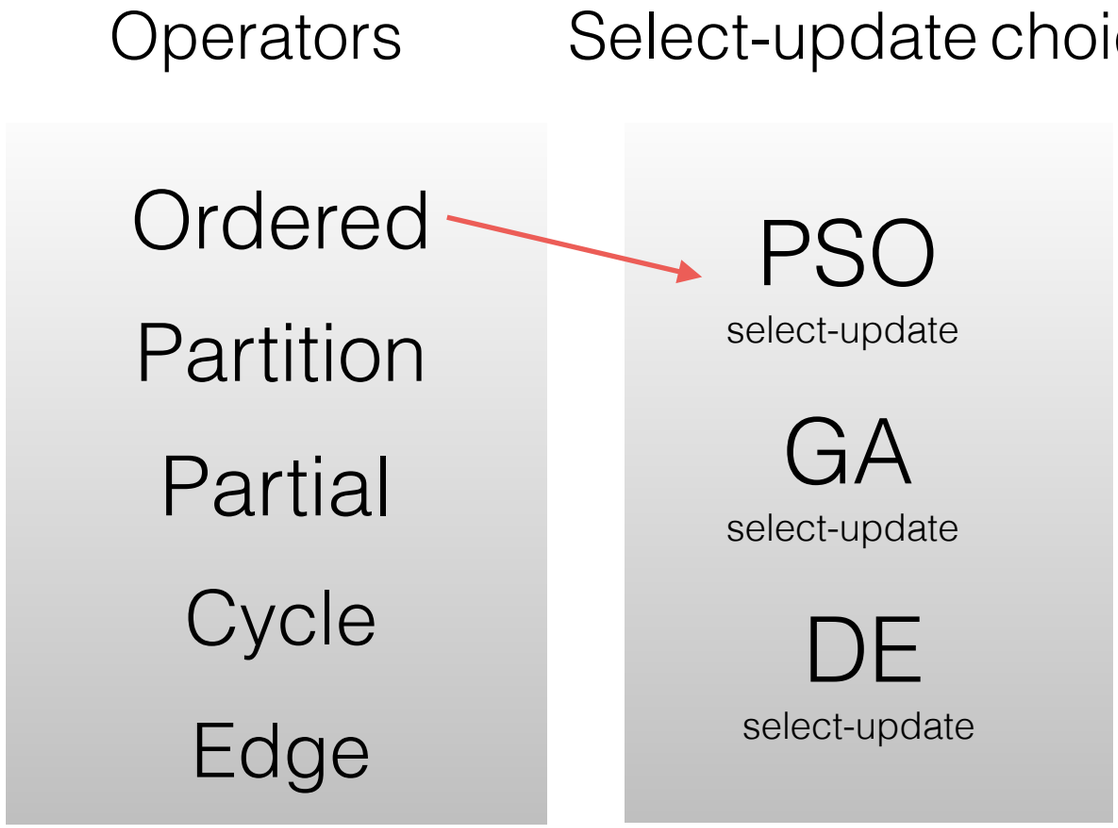
For permutation

Operators	Select-update choices
Ordered Partition Partial Cycle Edge	PSO select-update GA select-update DE select-update

Composition of multiple approaches

GA - Genetic algorithms
PSO- Particle swarm optimization
DE- Differential evolution

For permutation



Ordered—PSO

Composition of multiple approaches

GA - Genetic algorithms
PSO- Particle swarm optimization
DE- Differential evolution

For permutation

Operators

Ordered

Partition

Partial

Cycle

Edge

Select-update choices

PSO
select-update

GA
select-update

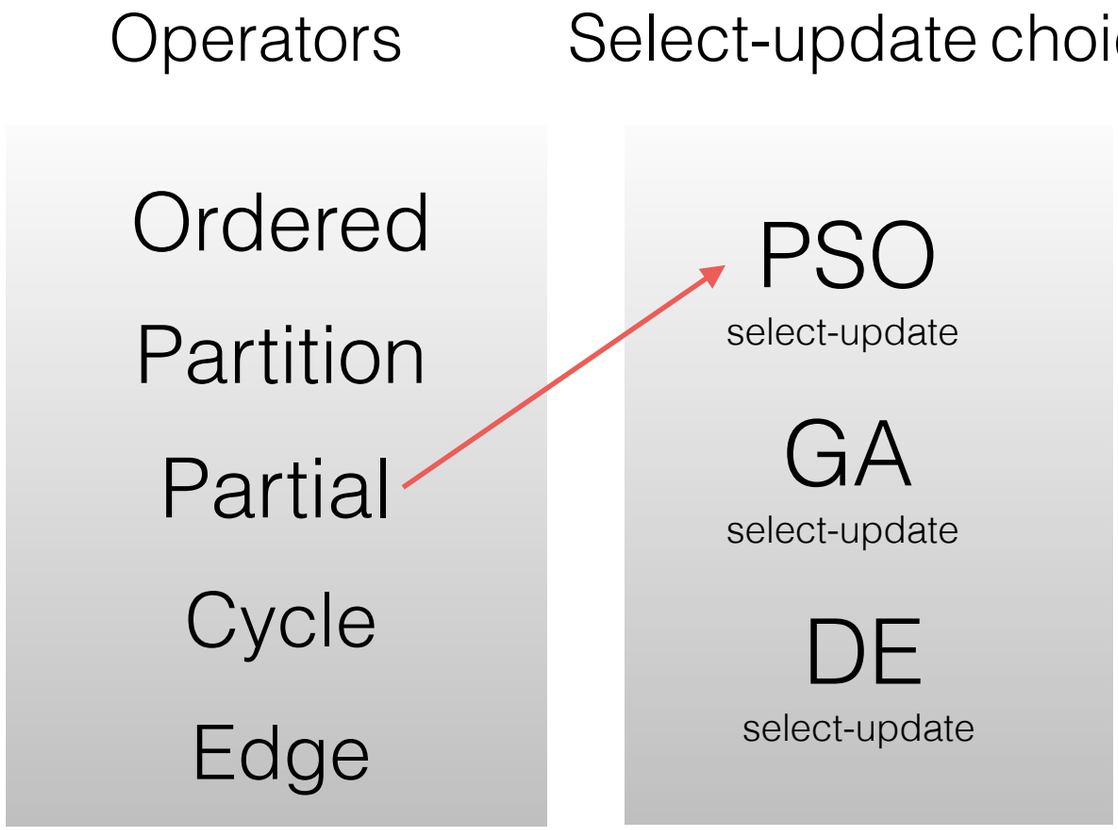
DE
select-update

Ordered—PSO
Ordered—GA

Composition of multiple approaches

GA - Genetic algorithms
PSO- Particle swarm optimization
DE- Differential evolution

For permutation

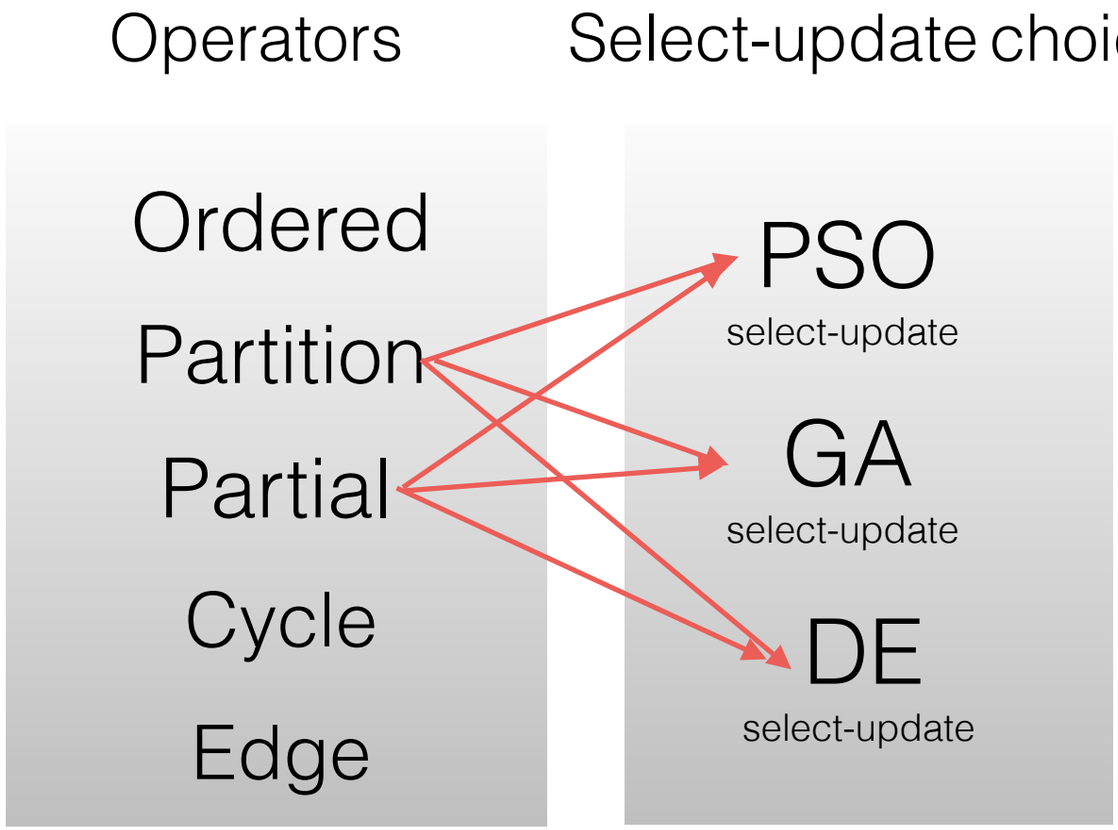


Ordered—PSO
Ordered—GA
Partial—PSO

Composition of multiple approaches

GA - Genetic algorithms
PSO- Particle swarm optimization
DE- Differential evolution

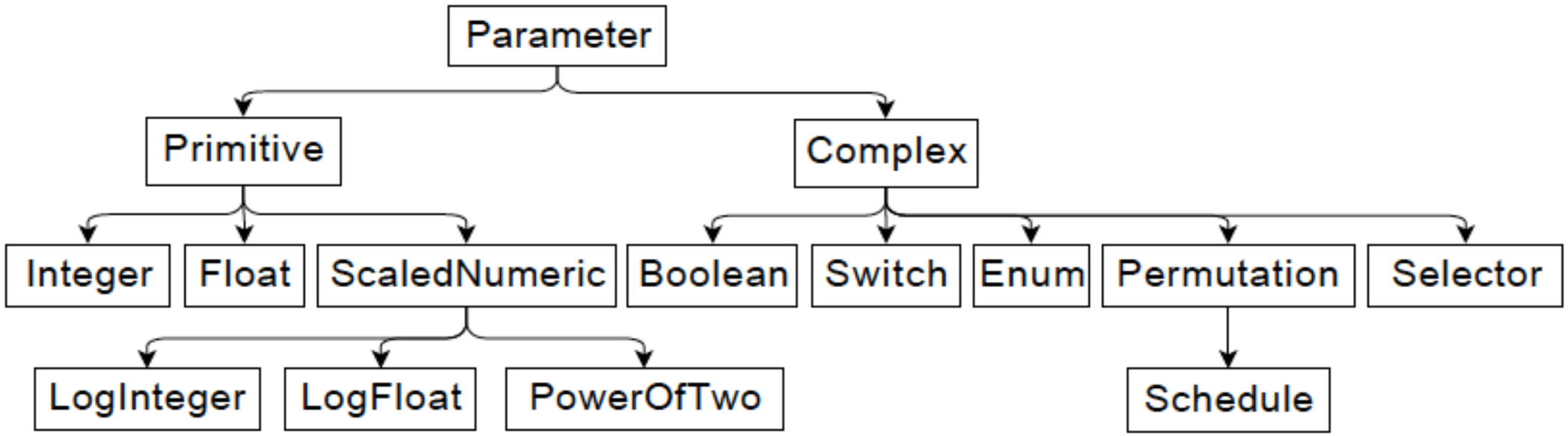
For permutation



Ordered—PSO
Ordered—GA
Partial—PSO

Partition—PSO
Partition—GA
Cycle—PSO

Parameter types



For each of these parameters we have operators, combined with techniques

Steps when trying for a new problem ?

- Design a representation
  - either uses an existing parameter
  - add new parameter
    - add operators that work on this new parameter
  - Choose the select-update/technique

Back to Mario

- Naive representation
  - 5 choices (left, right, jump, duck, run)
  - 12,000 frames
  - encode a bit string 60,000 bits long
  - first 5 are for decision making for the first frame and second 5 are for second frame and so on
  - each bit represents whether or not a choice is made at the frame

Back to Mario

- Duration representation
- 1000 - EnumerationParameters for direction of movement (biased 3:1 to move to right)
  - Enumerated parameter options
    - L = left, R = right, B = run, N = none
    - Actual definition of options :
    - ["R", "L", "RB", "LB", "N", "LR", "LRB", "R2", "RB2", "R3", "RB3"]

- 1000 - IntegerParameters for duration of each direction
  - Range:1-60 frames

- 1000 - IntegerParameters for which frames to jump
  - Range 1-24000 frames

- 1000 - IntegerParameters for duration of each jump
  - Range: 1-32 frames

- Better because the number of dimensions of search is 4000
- Decoupled jump