Asynchronously Evolving Solutions with Excessively Different Evaluation Time by Reference-based Evaluation

2014/07/15

Tomohiro Harada[1,2], Keiki Takadama[1]

[1]The University of Electro-Communications, Japan

[2]Research Fellow of the Japan Society for the Promotion of Science DC1, Japan

Genetic and Evolutionary Computation Conference (GECCO 2014)@Vancouver, BC, Canada

July 12-16, 2014

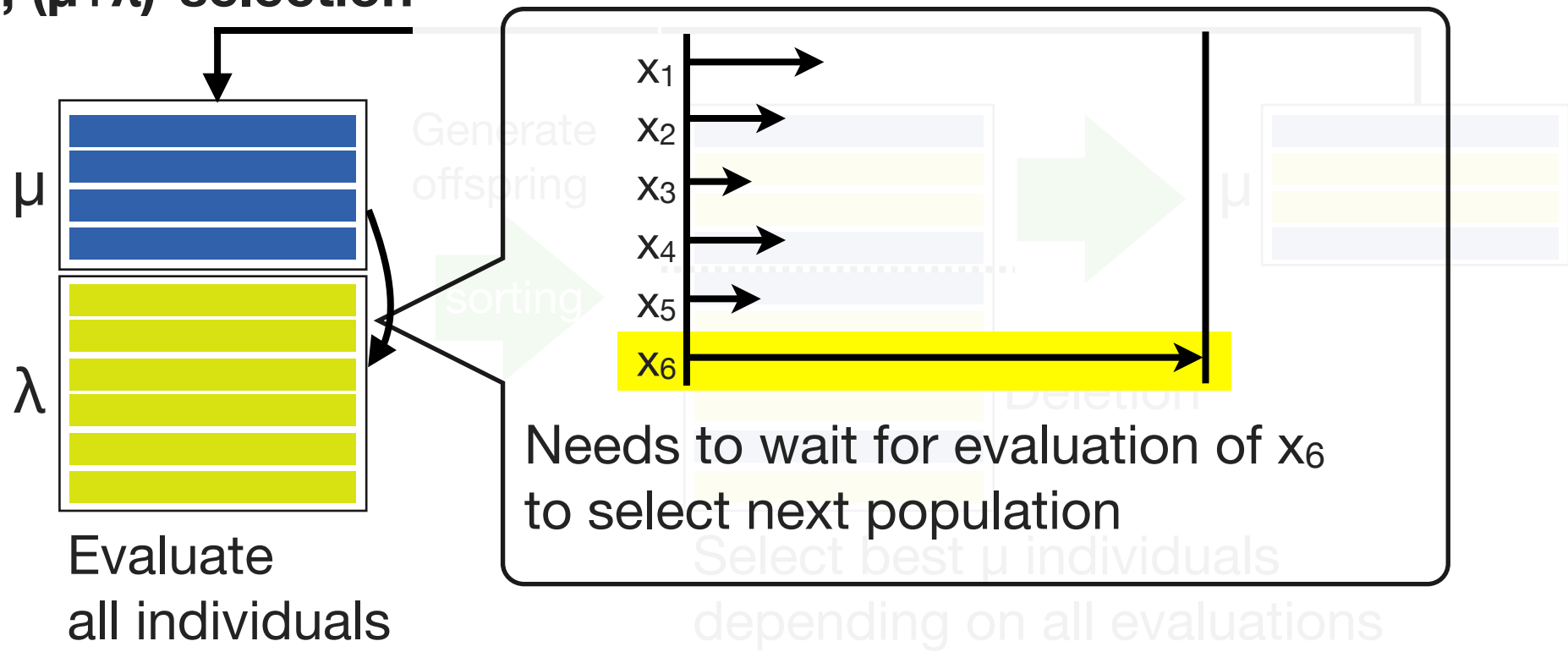
Synchronous EA and its problem

General EA=**Synchronous** EA

evolves individuals depending on evaluations of entire population

If evaluation time of individuals differs from each other
→sync. EA needs to wait for the slowest one

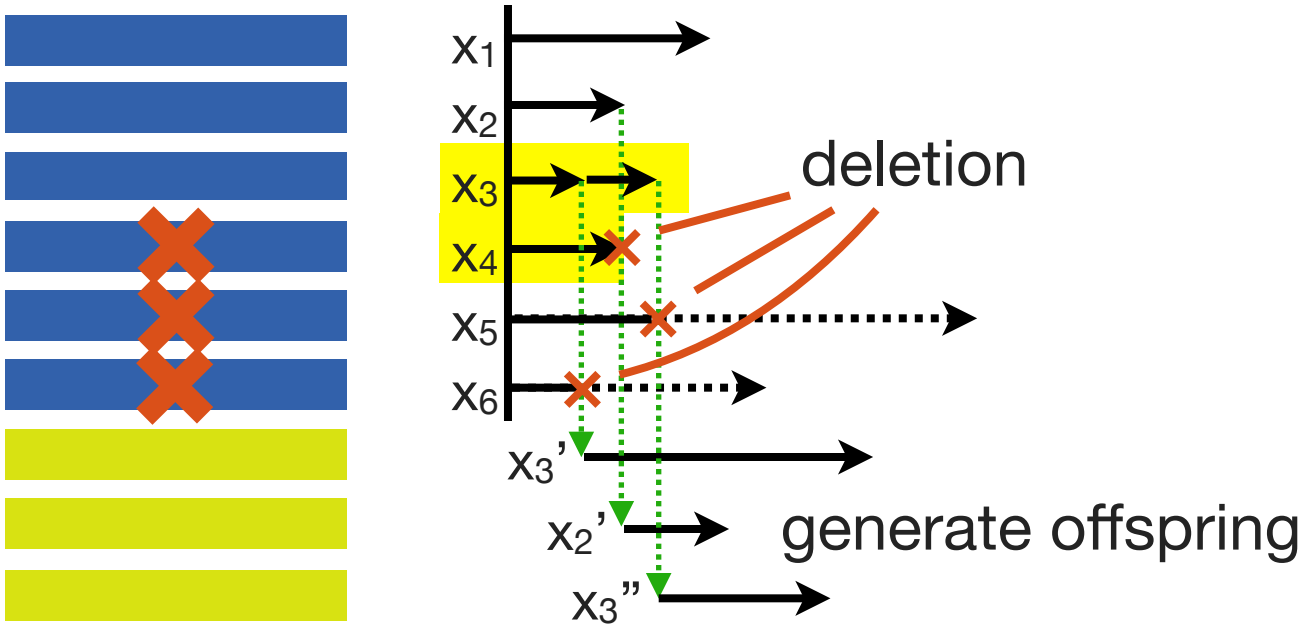
e.g., (μ+λ)-selection



Asynchronous EA

evolves individuals independently (**asynchronously**)

e.g., TAGP [Harada2013]

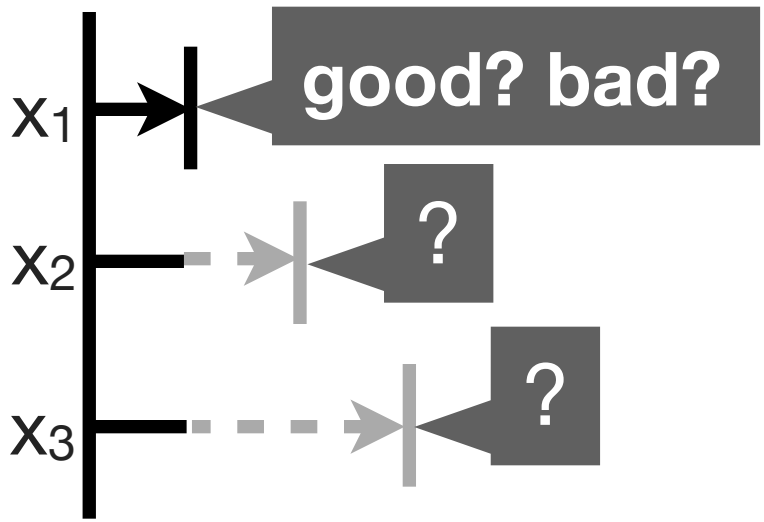


Advantage : Async. EA needs not to wait for other individuals
→can continue an evolution even in different evaluation time

Difficulties of Async. EA

1. How to preserve **good** individuals?

2. How to delete **bad** individuals?



Objective

To propose EA using **Asynchronous Reference-based Evaluation (ARE-EA)**

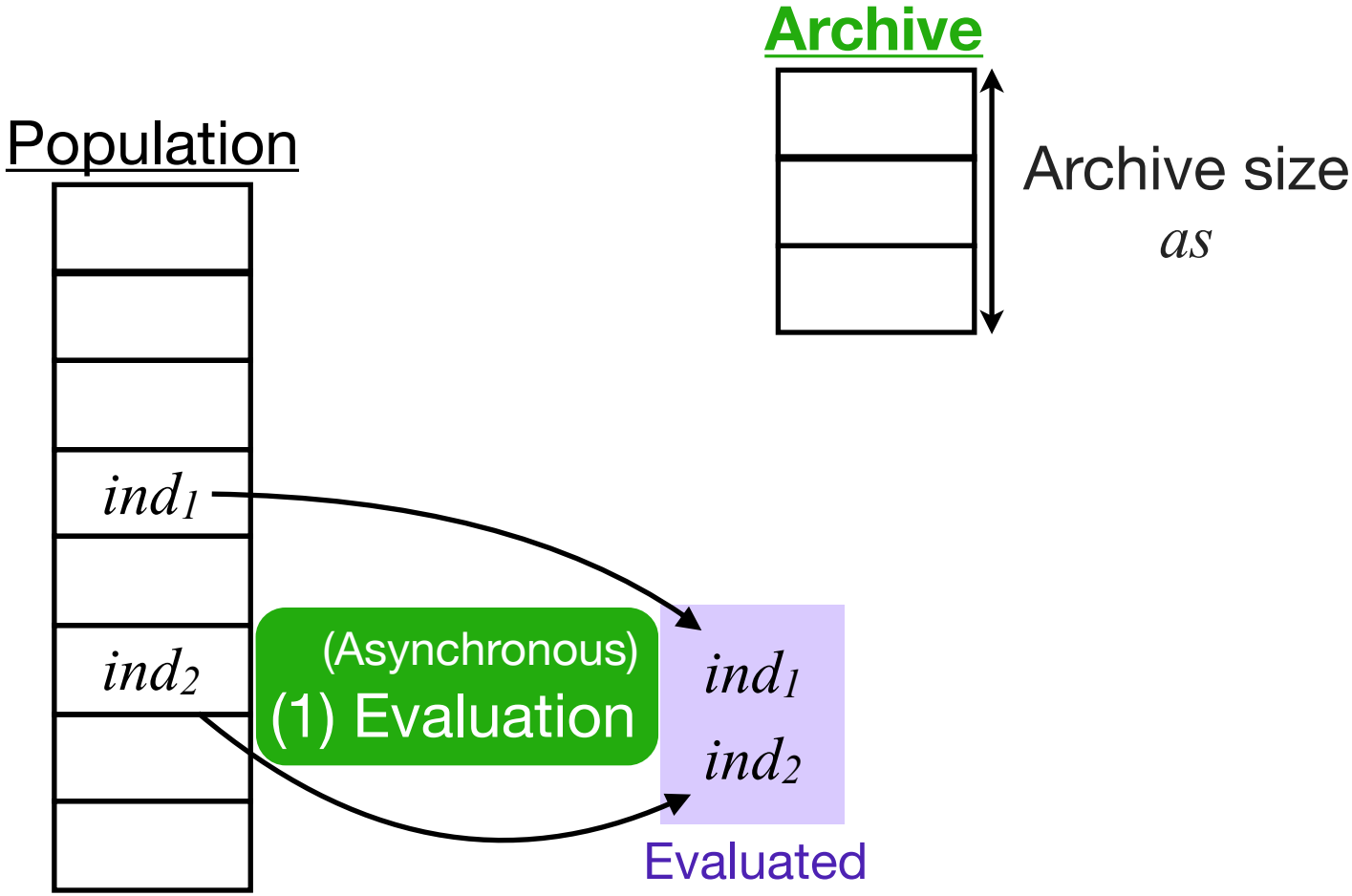
- **Archive** mechanism to preserve **good** individuals
- **Reference** individual to delete **bad** individuals

To investigate effectiveness of ARE-EA in situation where **evaluation times are excessively different**

1. Different computing speed (e.g., Difference of processing ability)
2. Evaluation failure (e.g., Infinite loop, Communication error)

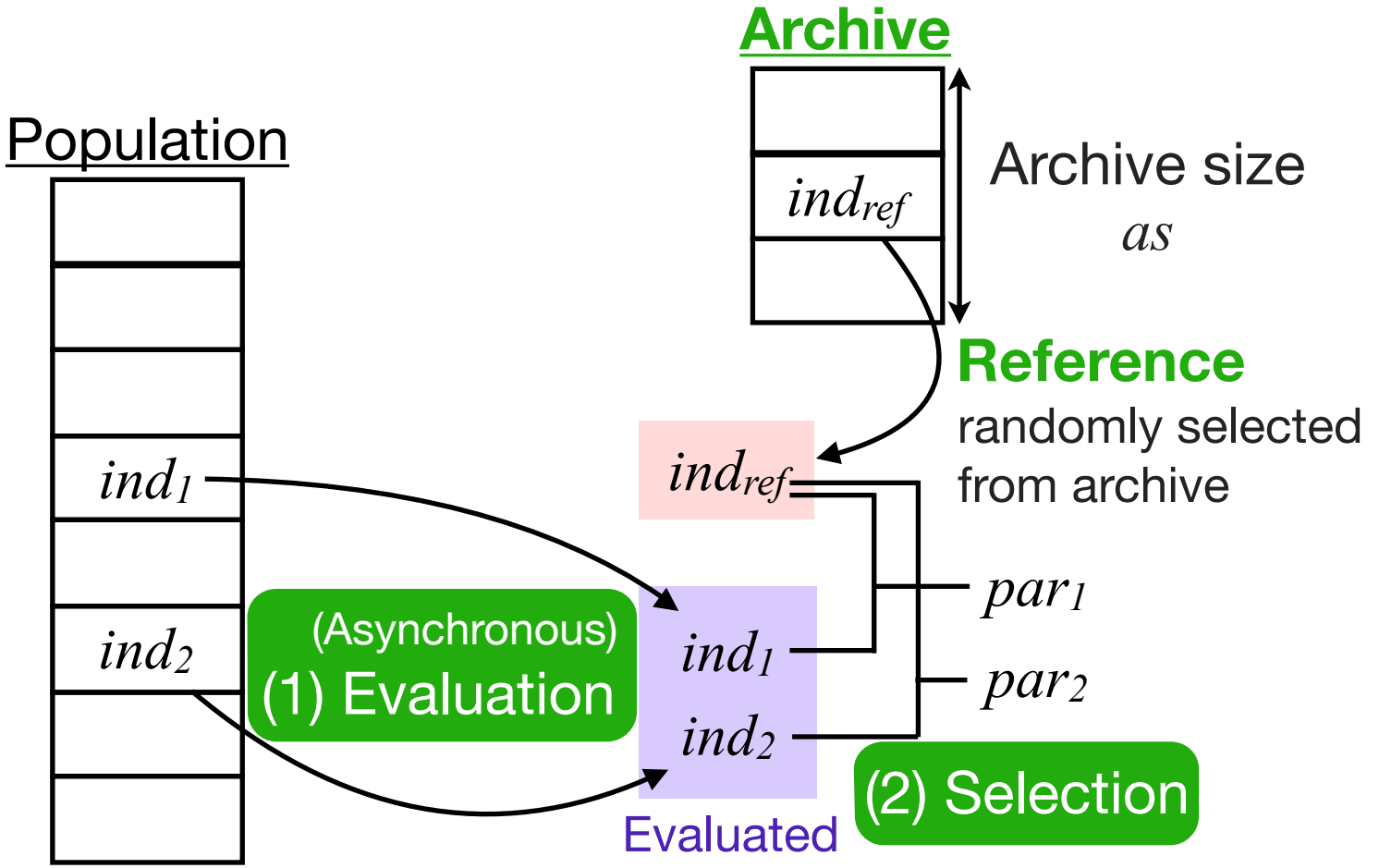
Proposed method
ARE-EA

(Asynchronous) (1) Evaluation | (2) Selection | (3) Mutation & Crossover
(4) Archiving | (5) Fitness deletion | (6) Reaper deletion



Proposed method
ARE-EA

(Asynchronous) (1) Evaluation | (2) Selection | (3) Mutation & Crossover
(4) Archiving | (5) Fitness deletion | (6) Reaper deletion

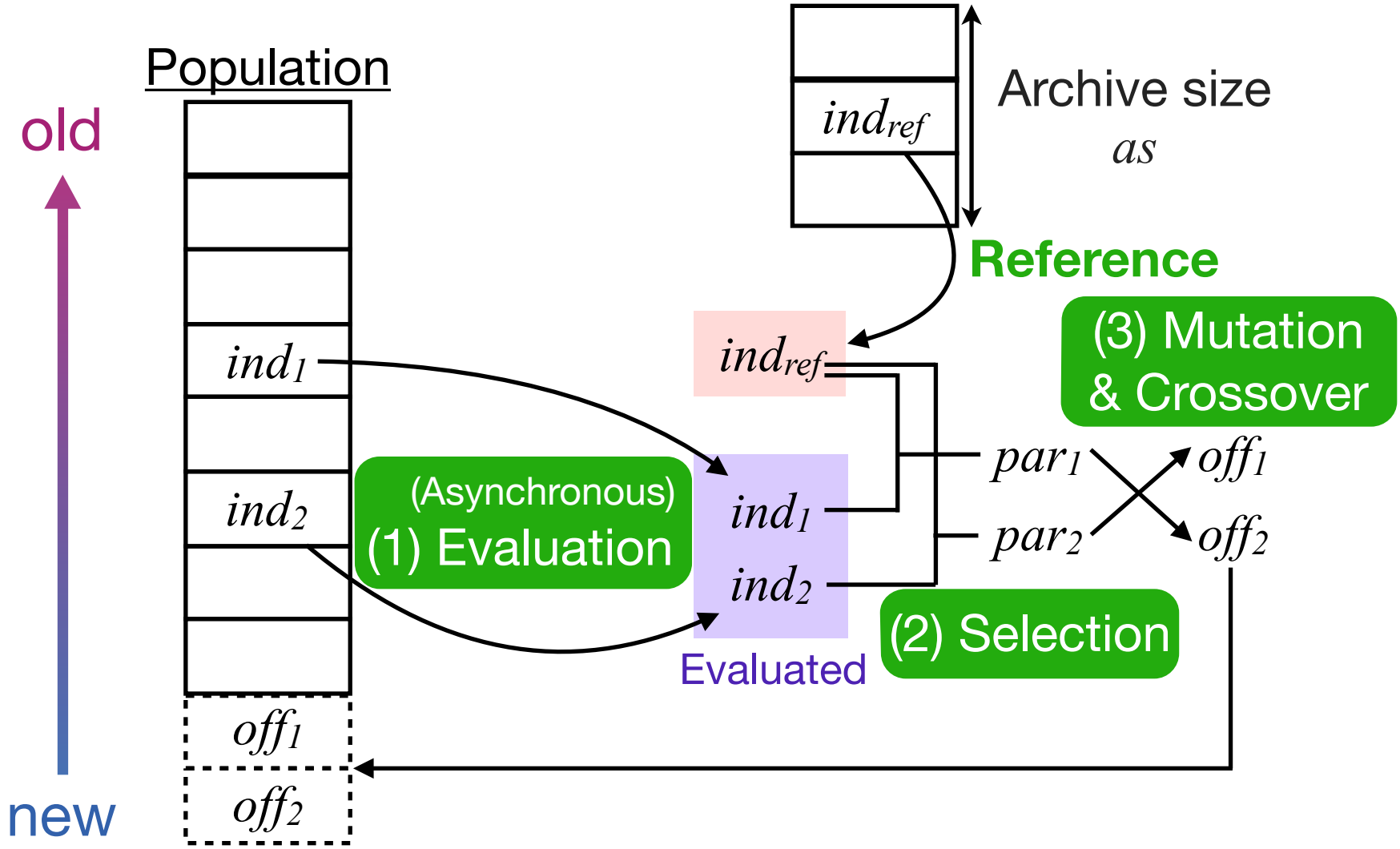


Good individuals (ind_{ref}) are utilized for selection

Proposed method
ARE-EA

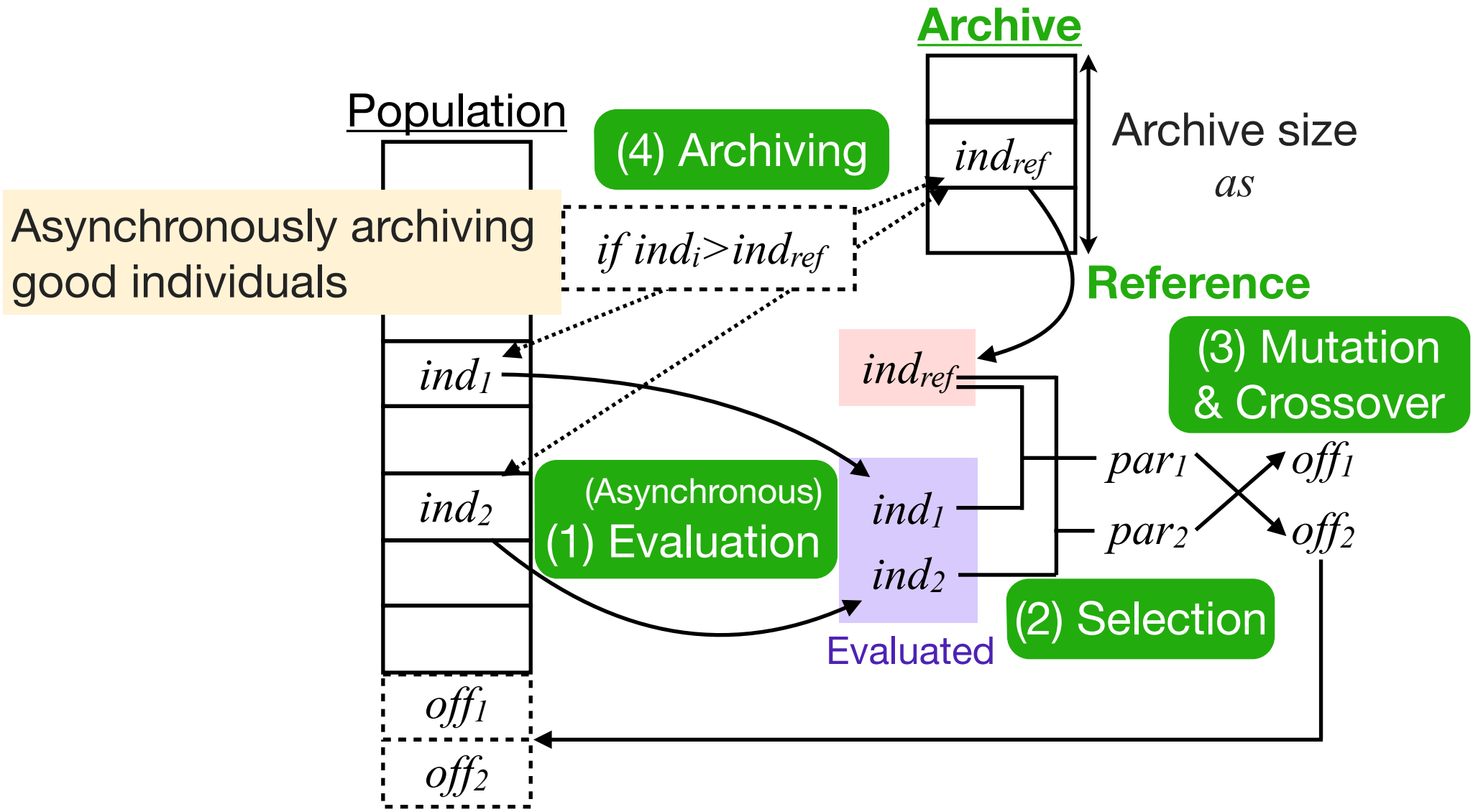
(Asynchronous) (1) Evaluation | (2) Selection | (3) Mutation & Crossover

(4) Archiving | (5) Fitness deletion | (6) Reaper deletion



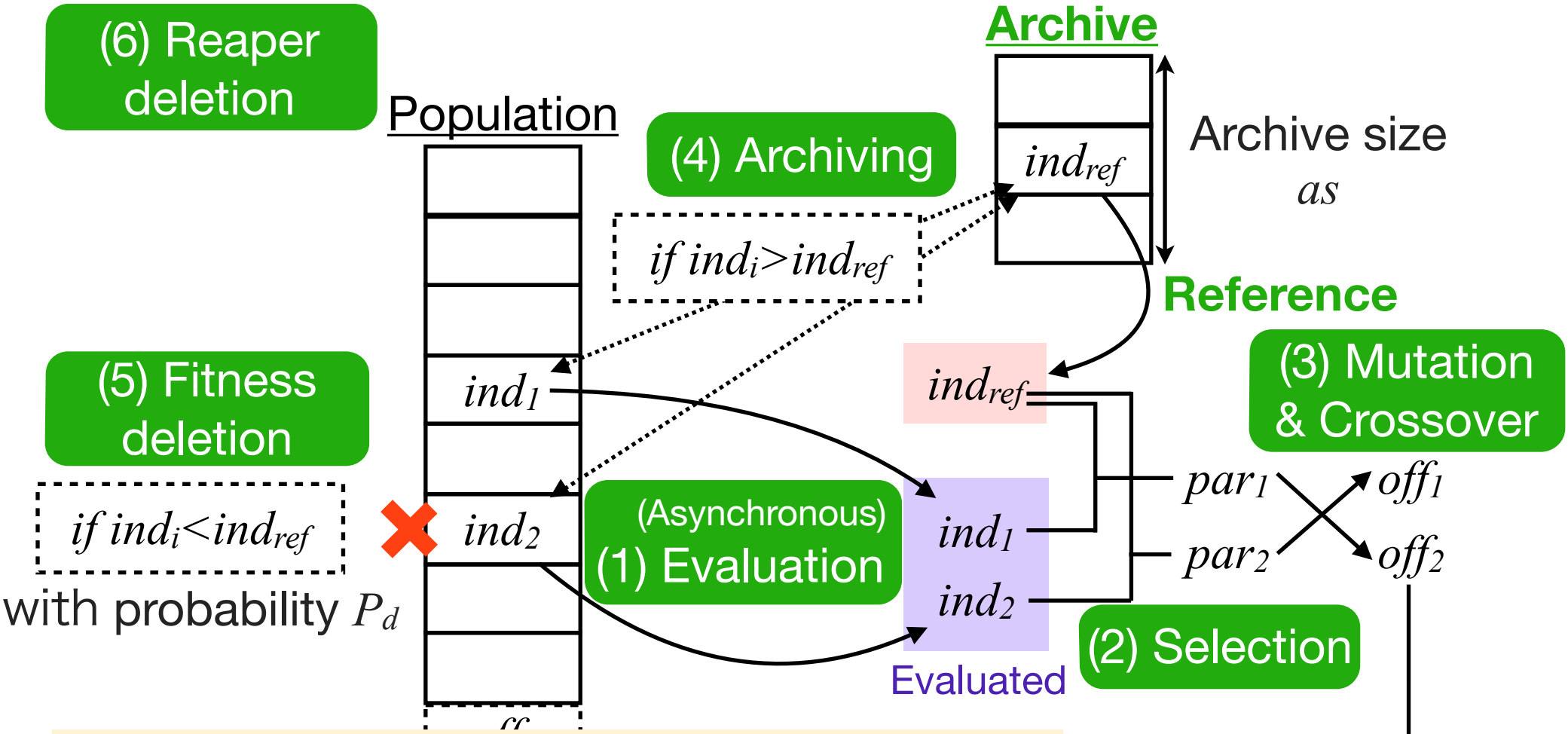
Proposed method
ARE-EA

(Asynchronous) (1) Evaluation | (2) Selection | (3) Mutation & Crossover
(4) Archiving | (5) Fitness deletion | (6) Reaper deletion



Proposed method
ARE-EA

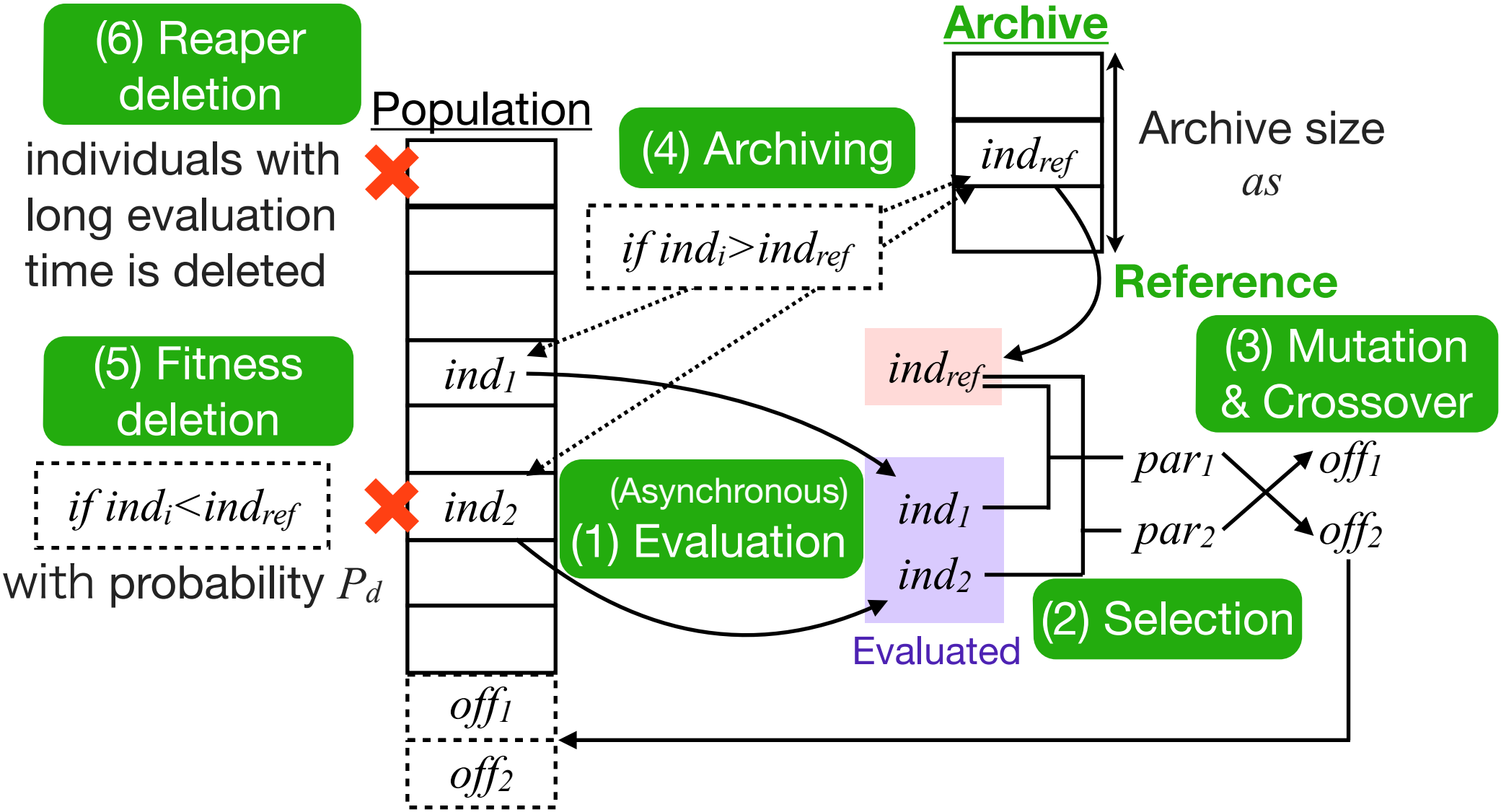
(Asynchronous) (1) Evaluation | (2) Selection | (3) Mutation & Crossover
(4) Archiving | (5) Fitness deletion | (6) Reaper deletion



Deletion based on comparison with reference
→delete bad individuals

Proposed method
ARE-EA

(Asynchronous) (1) Evaluation | (2) Selection | (3) Mutation & Crossover
(4) Archiving | (5) Fitness deletion | (6) Reaper deletion



Experiment

Comparison

ARE-GP vs. **(μ+λ)-GP**

(asynchronous) (synchronous)

in situation where evaluation times are excessively different

Testbed problem

Employing Linear GP (LGP) testbeds

	Symbolic regression
1	$f(x)=x^4+x^3+x^2+x$
2	$f(x)=x^6-2x^4+x^2$
3	$f(x)=sin(x^2)\times cos(x)-1$
4	$f(x)=ln(x+1)+ln(x^2+1)$

[J. McDermott, et al., 2012]

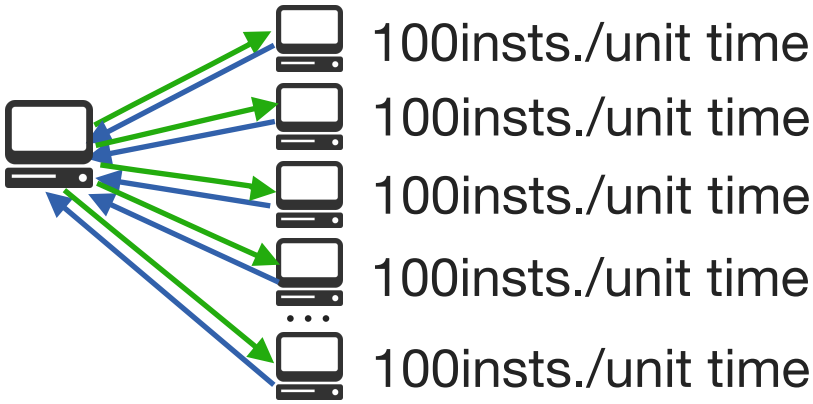
Instruction set
+, -, ×, /, *sin, cos, exp, ln*
$x_0 \ldots x_7$, *constant*={1, 2, …, 9}

Settings

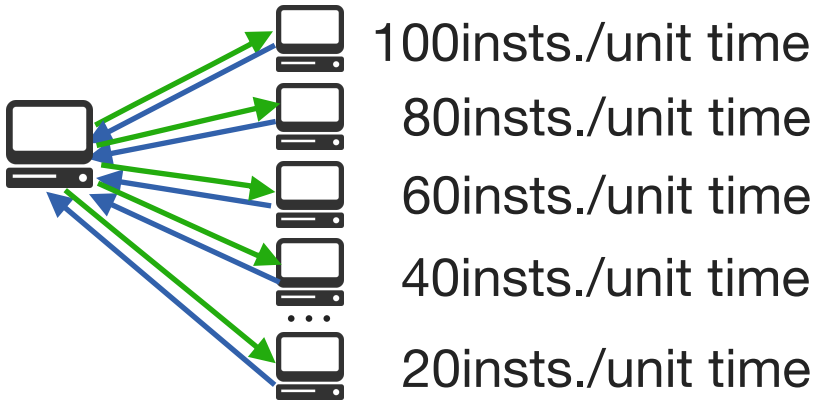
Different evaluation time situations

(1) Different computing speed (e.g., Difference of processing ability)

Same

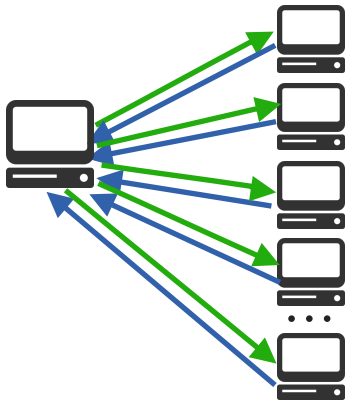


Different

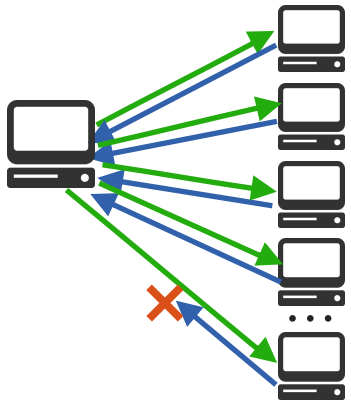


(2) Evaluation failure (e.g., Infinite loop, Communication error)

No failure



Failure



Cases

Failure \ Speed	Same	Different
No failure	Case1	Case2
Failure	Case3	Case4

*(μ+λ)-GP uses ideal limitation time to cut off evaluations in Cases3&4
*ARE-GP : archive size as=5, deletion probability P_d=0.5

Evaluation Criterion 20 trials

Average *fitness* according to the **same elapsed unit time**

$$fitness = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i^*)^2$$

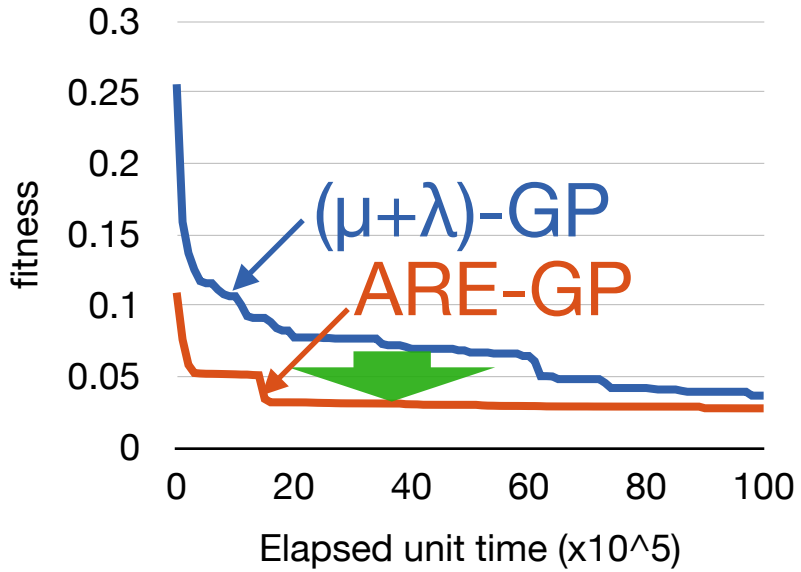
m : # of test data
$\hat{y}_i$: output y_i^* : target value

Failure \ Speed	Same	Different
No failure	Case1	Case2
Failure	Case3	Case4

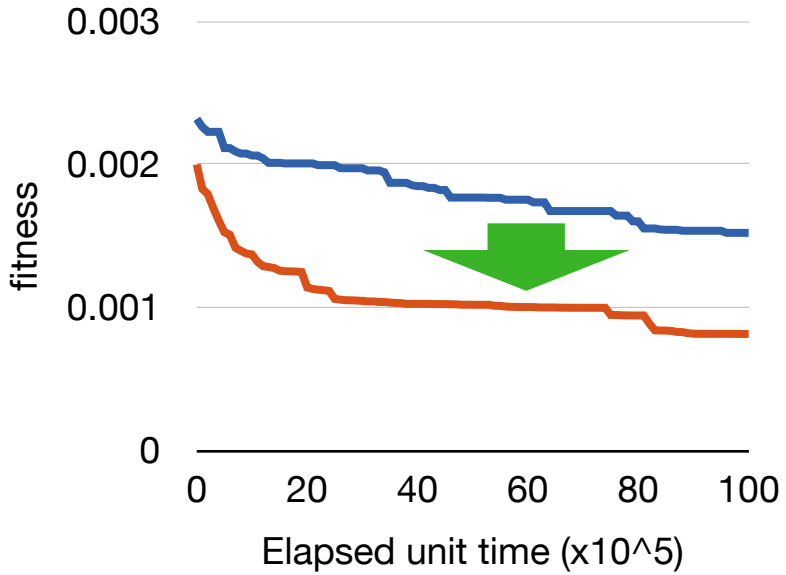
Result : Case1

Speed / Failure	Same	Different
No failure	Case1	Case2
Failure	Case3	Case4

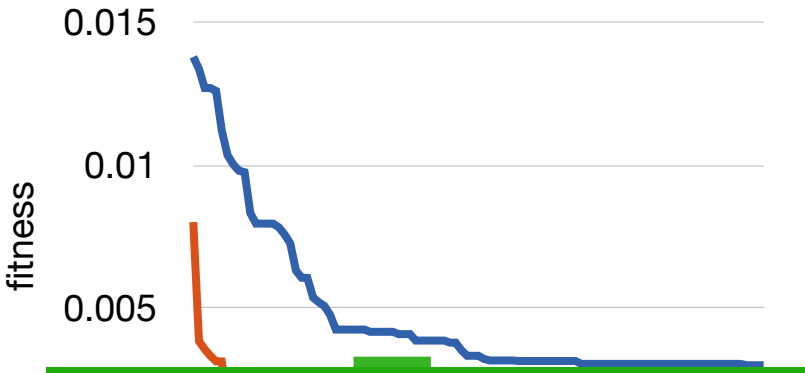
(1)$f(x)=x^4+x^3+x^2+x$



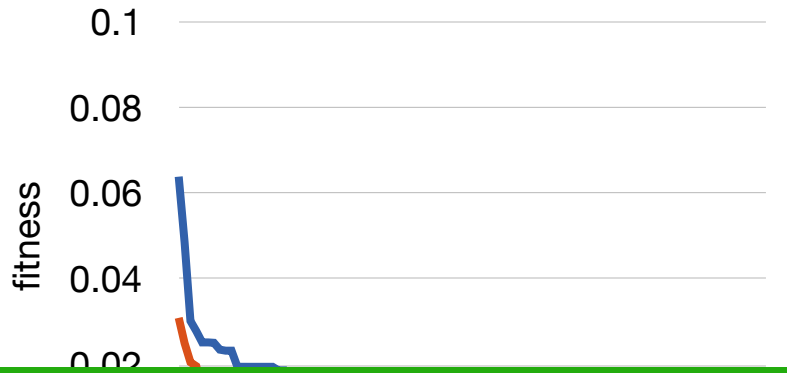
(2)$f(x)=x^6-2x^4+x^2$



(3)$f(x)=sin(x^2)\times cos(x)-1$



(4)$f(x)=ln(x+1)+ln(x^2+1)$



ARE-GP>(μ+λ)-GP
→ARE-GP evolves individuals without waisting time

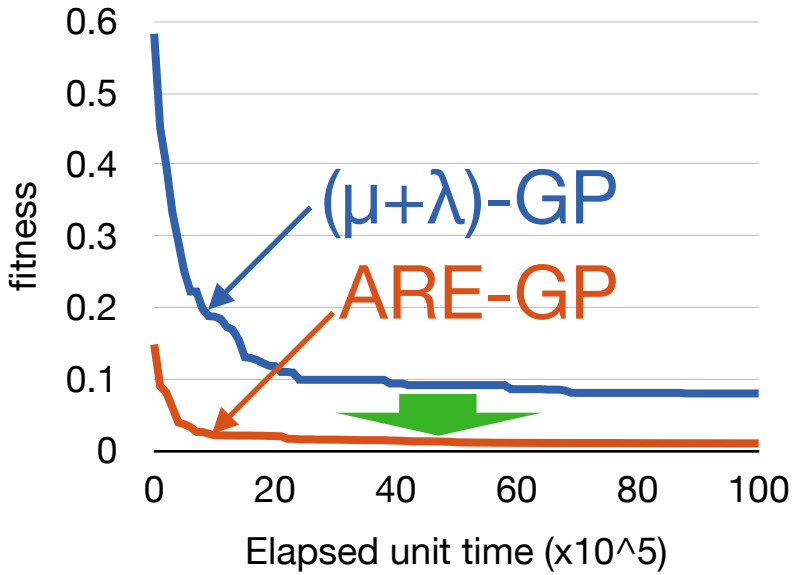
Failure \ Speed	Same	Different
No failure	Case1	Case2
Failure	Case3	Case4

Result : Case4

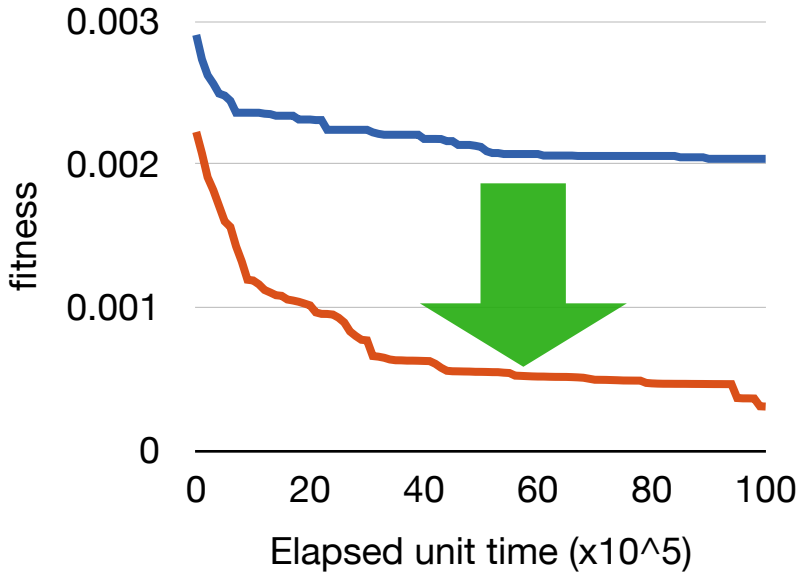
Failure \ Speed	Same	Different
No failure	Case1	Case2
Failure	Case3	Case4

(Ideal limitation time is used)

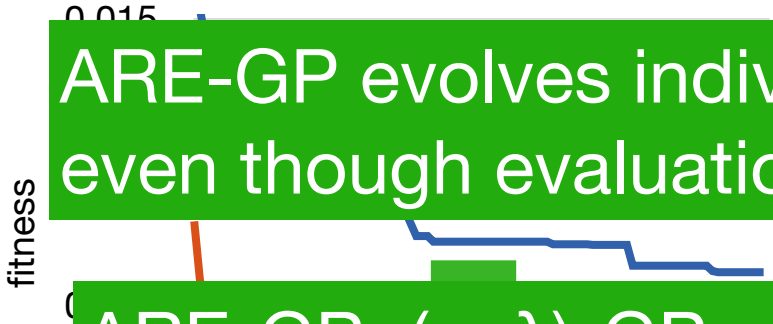
(1)$f(x)=x^4+x^3+x^2+x$



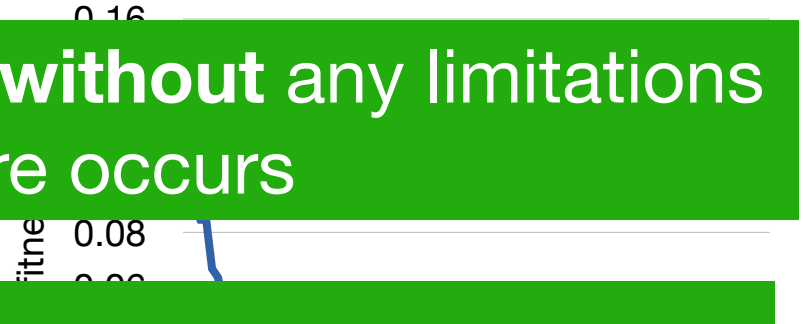
(2)$f(x)=x^6-2x^4+x^2$



(3)$f(x)=sin(x^2)\times cos(x)-1$



(4)$f(x)=ln(x+1)+ln(x^2+1)$



ARE-GP evolves individuals **without** any limitations even though evaluation failure occurs

ARE-GP>(μ+λ)-GP
ARE-GP efficiently evolves individuals in situation where evaluation times are excessively different

Result

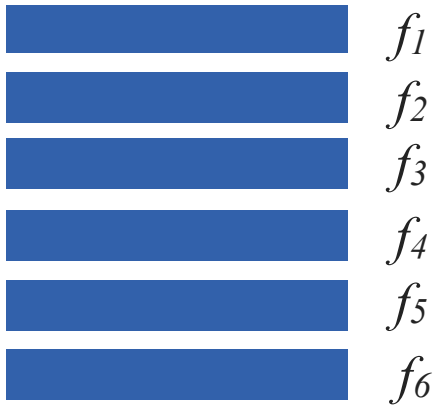
ARE-GP > **(μ+λ)-GP**
in all cases and in all testbeds

ARE-GP

only uses **a few** evaluations

(μ+λ)-GP

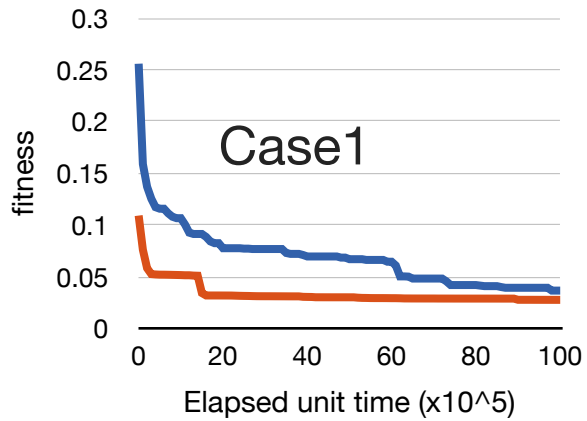
can use **all** evaluations

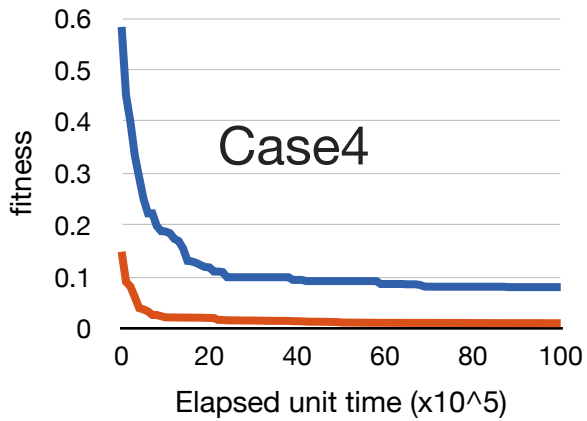


Comparison of Case1&Case4

Ratio of fitness between Case1 and Case4

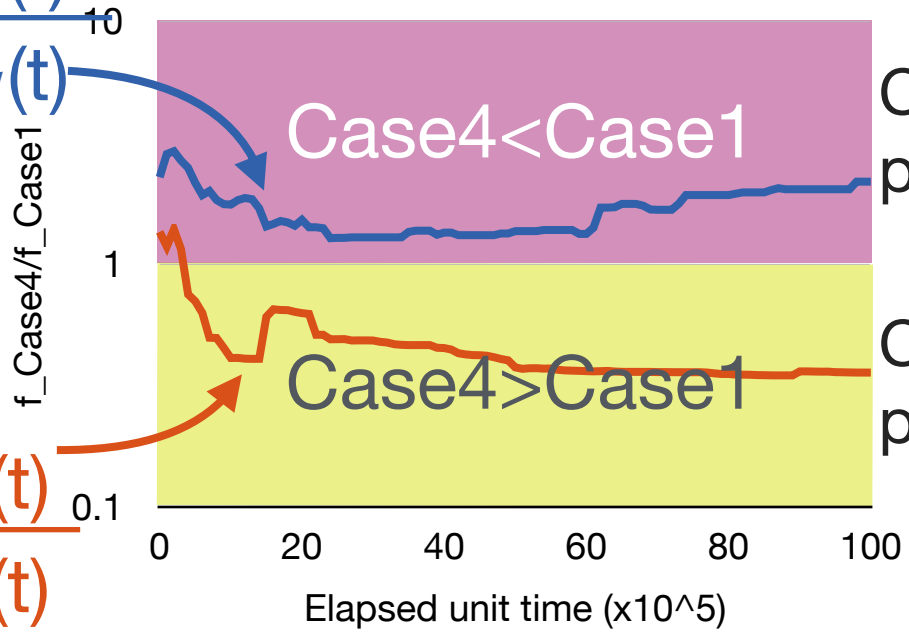
(1) $f(x)=x^4+x^3+x^2+x$





$$\frac{(\mu+\lambda)\text{-GP.}f_{case4}(t)}{(\mu+\lambda)\text{-GP.}f_{case1}(t)}$$

$$\frac{\text{ARE-GP.}f_{case4}(t)}{\text{ARE-GP.}f_{case1}(t)}$$

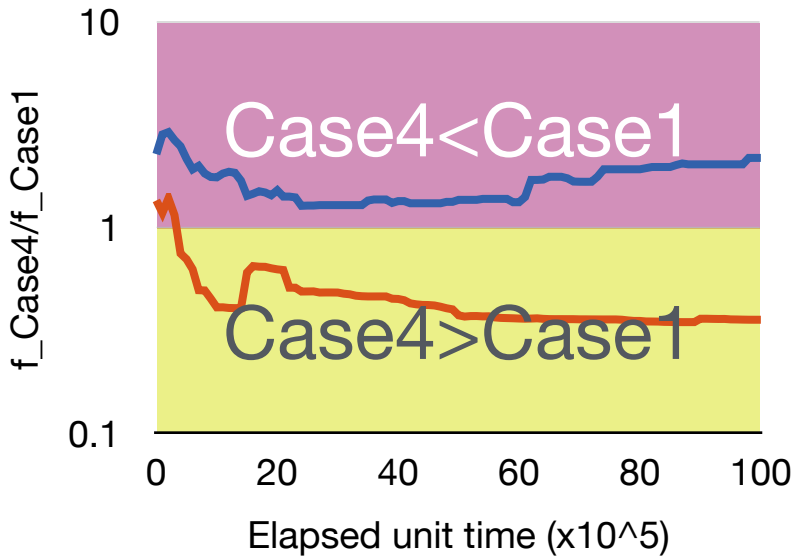


Case4 decreases performance than Case1

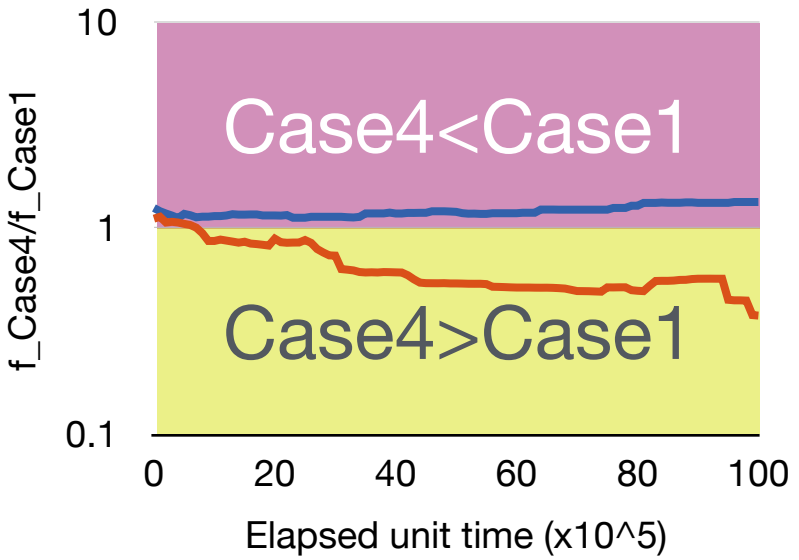
Case4 increases performance than Case1

Comparison of Case1&Case4

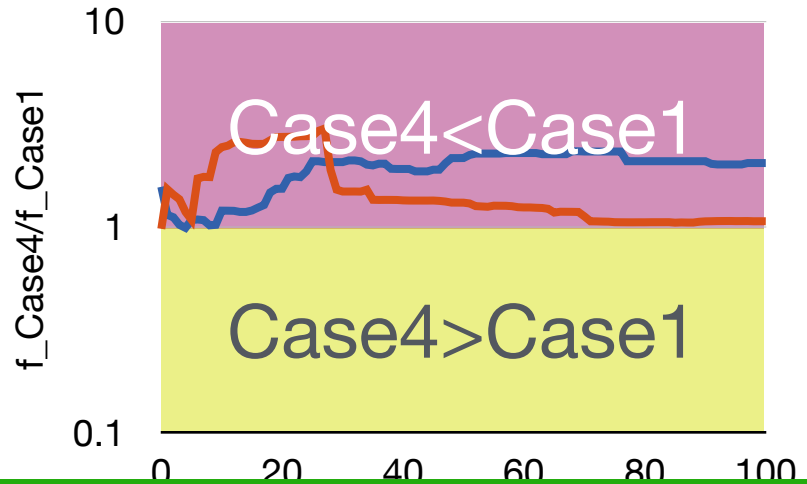
(1)$f(x)=x^4+x^3+x^2+x$



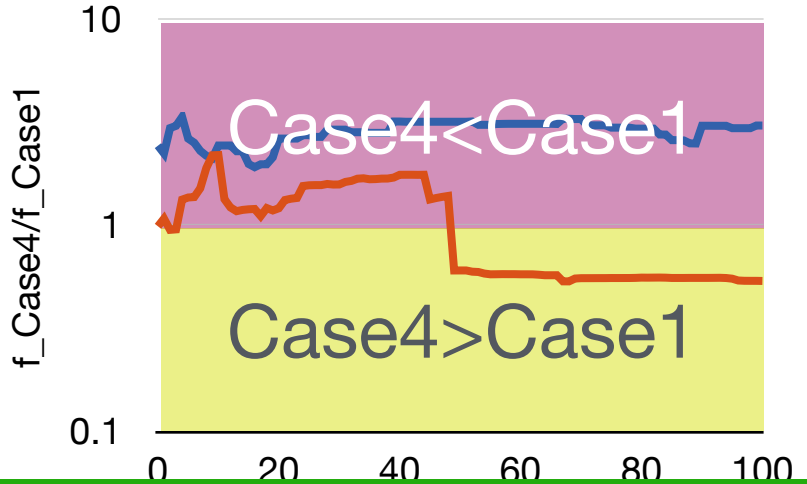
(2)$f(x)=x^6-2x^4+x^2$



(3)$f(x)=sin(x^2)\times cos(x)-1$



(4)$f(x)=ln(x+1)+ln(x^2+1)$



ARE-GP has possibility to **improve** search performance in different evaluation time

Conclusion

Objective

- Proposing EA using **Asynchronous Reference-based Evaluation (ARE-EA)**
- Investigating effectiveness of ARE-EA in excessively different evaluation times
  - different computing speed
  - evaluation failure

Implications

- ARE-GP>(μ+λ)-GP
  - in excessively different evaluation time
- ARE-GP improves performance in different evaluation time

Future works

- Validation in parallel computing environment
- Adaptation of parameters P_d and *as*