

Parallel and Distributed Models in Evolutionary Computing

- ❑ Motivation
- ❑ Parallelization models
- ❑ Distributed models

Motivation

The evolutionary algorithms need a large amount of resources:

- ❑ Memory space (since they usually need large populations)
- ❑ Execution time (since the evolutionary process is usually long)

Costly operations:

- ❑ The evaluation of the population elements
- ❑ The application of operators

Solutions:

- ❑ Improving the convergence rate of the algorithm (by developing new operators)
- ❑ Increasing the efficiency of the implementation (parallel/distributed implementation)

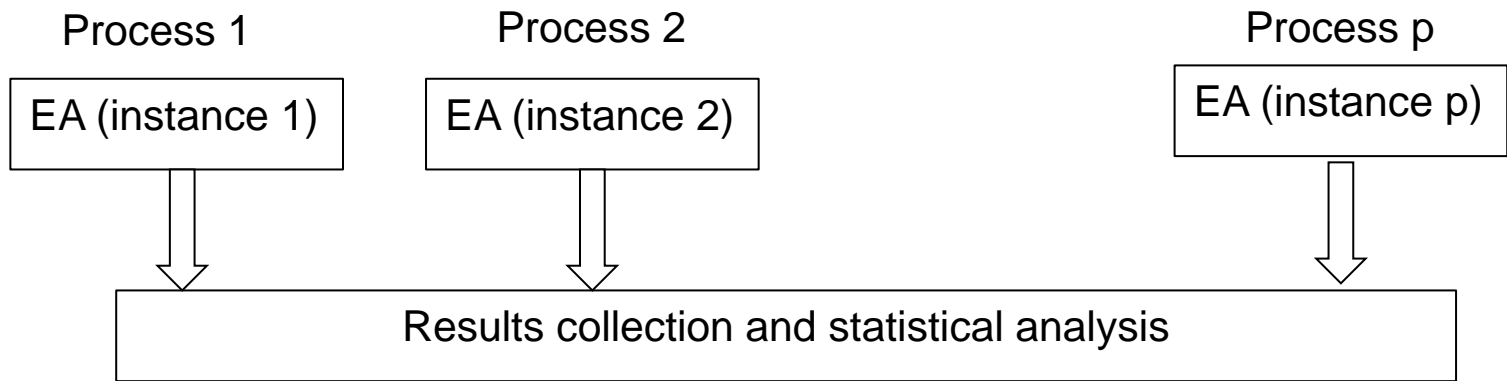
Parallel and distributed models

The parallelization can be implemented at different levels:

- ❑ Algorithm -> naive parallelization model
- ❑ Elements evaluation -> master-slave model
- ❑ Population -> island model
- ❑ Element -> cellular model

Naïve model

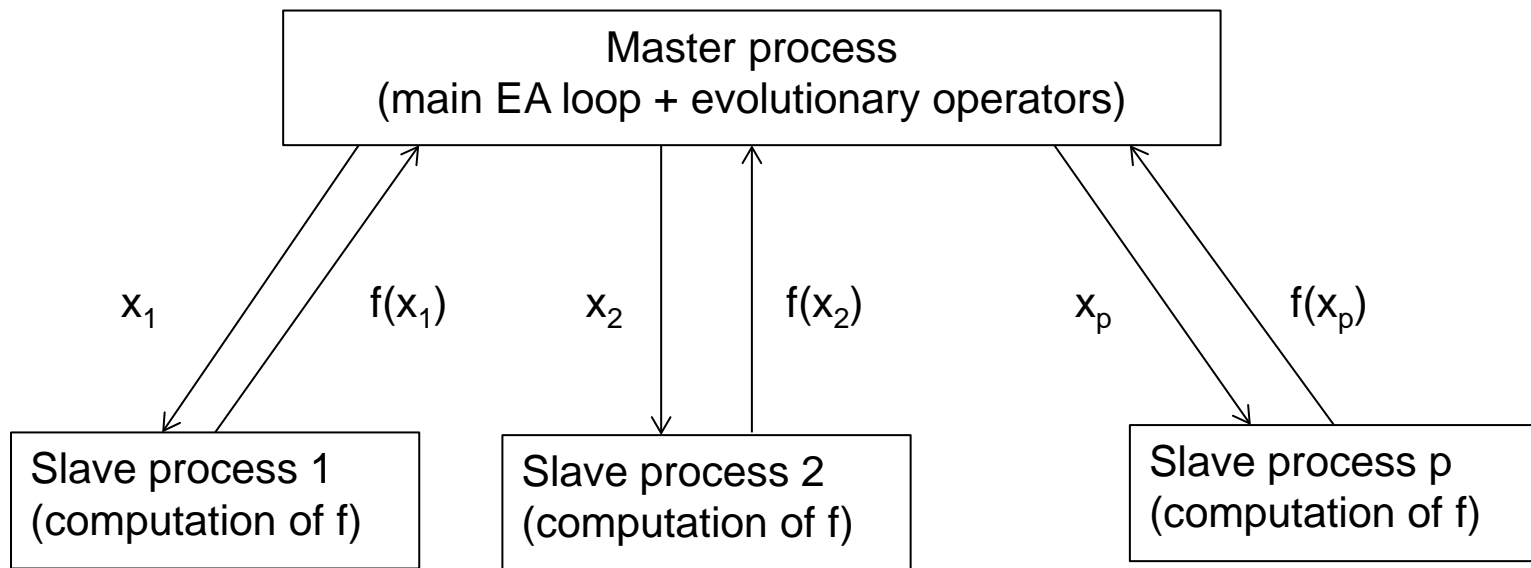
The algorithm is simultaneously executed on several processors which do not communicate



Is useful for statistical analysis or for parameter tuning

Master-slave model

The master process executes the EA and distributes the evaluation of the population elements to the slave processes



Master-slave model

Particularities:

- ❑ If the population size is larger than the number of available processors then the master process has to distribute the elements to processors.
- ❑ The evaluation time depends not only on the characteristics of the processor but also on the particularities of the element which should be evaluated (e.g. in genetic programming)
- ❑ In such a case there is necessary to synchronize the computations. In order to avoid frequent synchronization steps the generational (synchronous) strategy can be replaced with an **asynchronous (steady-state) strategy**

Master-slave model

Synchronous

Population initialization
Population evaluation
REPEAT
 Parents selection
 Generate a population of offspring
 Evaluate the offspring population
 Select the survivors
UNTIL <stopping condition>

Asynchronous

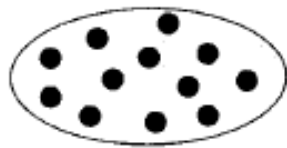
Population initialization
Population evaluation
REPEAT
 Parents selection
 Generate a new element
 Evaluate the new element
 Assimilate the new element in the population
UNTIL <stopping condition>

Master-slave model

- ❑ Is easy to implement
- ❑ Leads to a more efficient implementation only if the evaluation step is significantly more costly than the other operations involved in the EA.
- ❑ The behavior of the evolutionary algorithm (with respect to the convergence properties) is not changed
- ❑ It can be implemented both on systems with shared memory and on systems with distributed memory (including computer networks)

Structuring the population

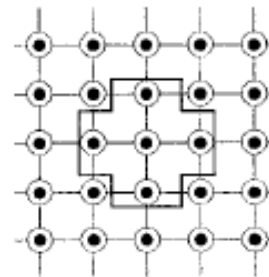
- ❑ The population can be unstructured (panmictic) or structured
- ❑ Structuring the population has an influence on the evolutionary process, one of its effects being the stimulation of the population diversity.
- ❑ There are different models:
 - ❑ Coarse-grain model (island model)
 - ❑ Fine-grain model (cellular model)



Model panmictic



Island model

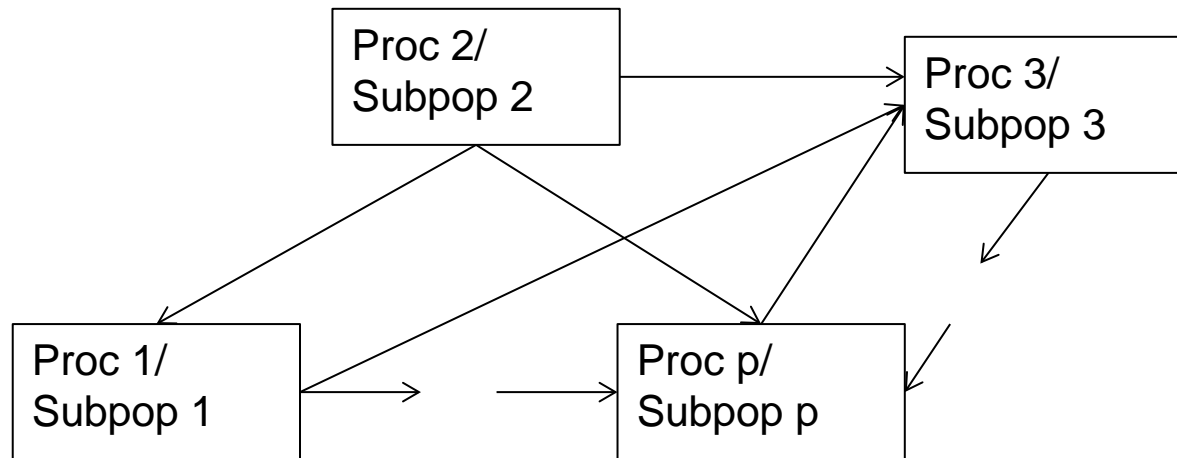


Cellular model

Alba, Tomassini; Parallelism and EAs, 2002

Island model

- ❑ Consists of dividing the population in subpopulations (“islands” or “demes”) on which there are executed identical or different EAs and which communicate between them by a so-called migration process.



- ❑ A processor can deal with one or several subpopulations
- ❑ In each subpopulation the evolutionary operators are applied for a given number of iterations then a migration process is initiated.

Island model

The communication processed between subpopulations is characterized by:

- ❑ Communication topology
- ❑ Communication strategy
- ❑ Parameters controlling the communication

These elements have an important influence on the behaviour of the algorithm and on its efficiency.

Island model

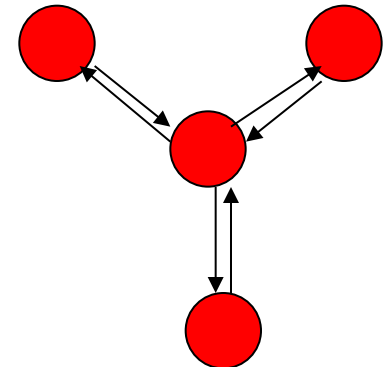
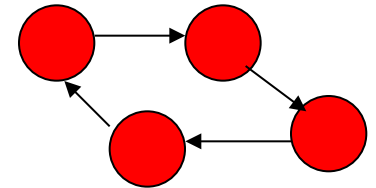
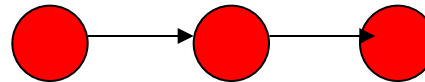
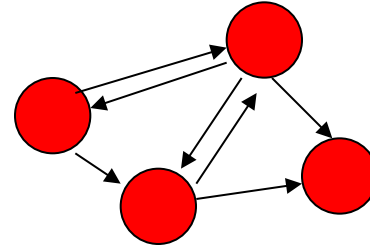
Communication topology

Random

Ring

Linear

Star



Island model

- ❑ Communication strategy
 - ❑ Migration: an element from the source subpopulation is exchanged with an element from the destination subpopulation
 - ❑ Pollination: a copy of an element from the source subpopulation is transferred in the target subpopulation

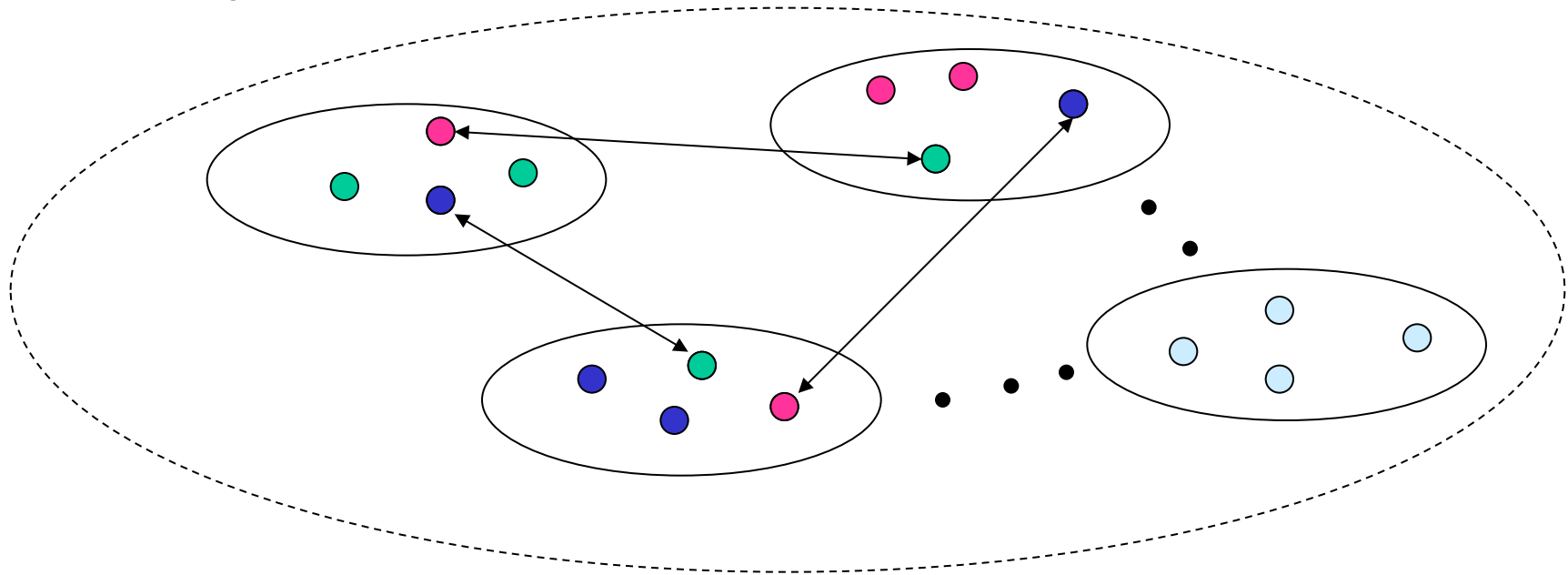
- ❑ Selection of the element in the source subpopulation
 - ❑ Random
 - ❑ Elitist (one of the best elements)

- ❑ Selection of the element in the destination subpopulation
 - ❑ Random
 - ❑ Elitist (one of the best elements in the case of migration; one of the worst elements in the case of pollination)

Island model

□ Example:

- Elements exchange
- The global distribution of the elements remains unchanged; only the distribution of elements in the subpopulations is changed

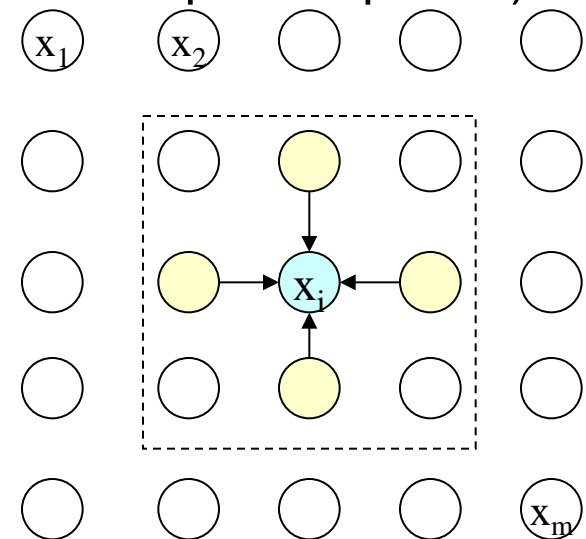


Island model

- ❑ Specific parameters:
 - ❑ Migration frequency:
 - ❑ Based on the number of generation
 - ❑ Based on the subpopulations properties
 - ❑ Migration probability:
 - ❑ A high value means a lot of communication between subpopulations

Cellular model

- ❑ The elements are placed in the nodes of a grid (characterized by a given topology)
- ❑ Only the neighbours are involved in the selection and crossover process
- ❑ In a parallel implementation each element is assigned to a processor (appropriate for implementations on supercomputers)



<http://neo.lcc.uma.es/cEA-web/index.htm>

Cellular model

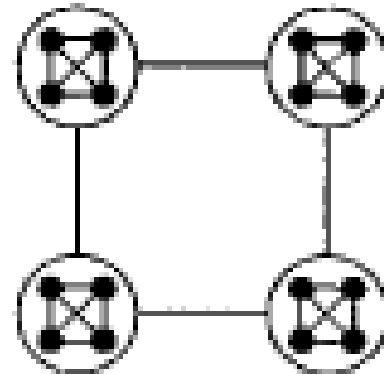
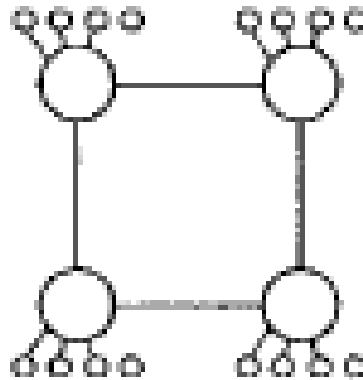
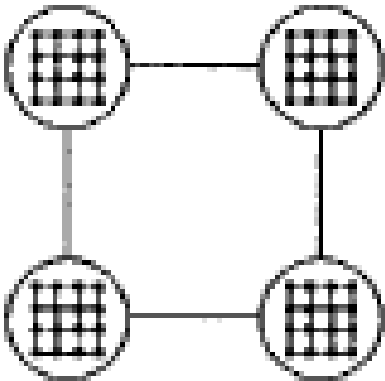
- ❑ Can be used also in the case of sequential implementations since it induces a different dynamics.
- ❑ Somehow similar to cellular automata
- ❑ There are two variants:
 - ❑ **Synchronous**: all offspring are computed in parallel and the replacement is done simultaneously
 - ❑ **Asynchronous**: the new elements replace their parents as soon as they are generated (asynchronously)

Cellular model

- ❑ Asynchronous variants:
 - ❑ Random selection of elements involved in the reproduction process
 - ❑ The cells in the grid are scanned systematically (e.g. row by row)
 - ❑ The elements are processed in the order given by a random permutation
- ❑ The asynchronous variant is usually quicker than the synchronous one

Hybrid variants

- ❑ The master/slave, island and cellular models can be combined in one of the following variants:
 - ❑ Island+cellular
 - ❑ Island+MasterSlave
 - ❑ Island+island

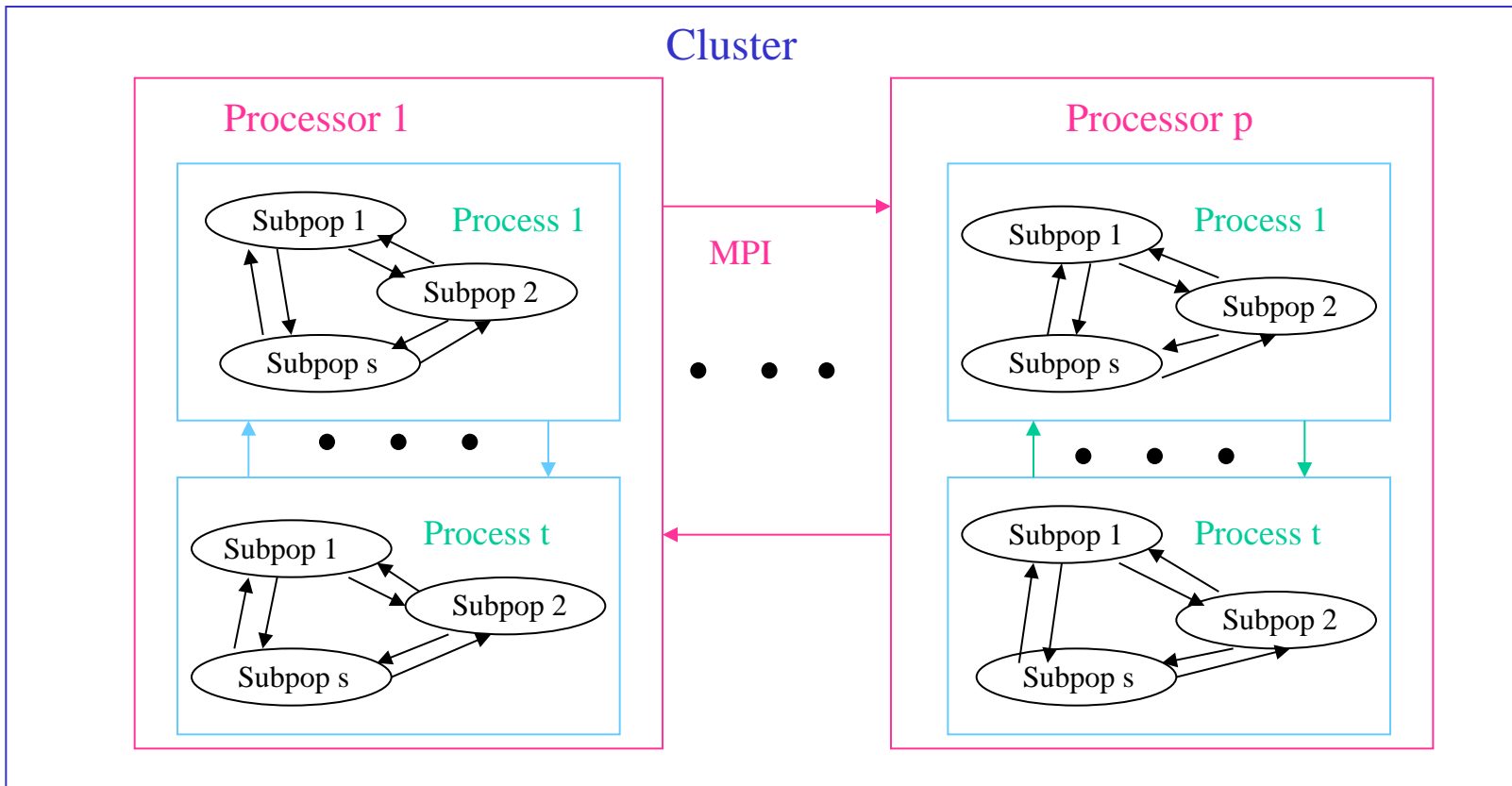


Implementation

- ❑ The appropriate computing environment depends on the model granularity and on the communication
 - ❑ **Master-slave model**: appropriate for cluster architectures
 - ❑ **Island model**: both for cluster and distributed architectures
 - ❑ **Cellular model**: multi-processors
- ❑ Software: tools PVM, MPI, OpenMP etc.

Implementation

- Example (for an island model implemented in a cluster environment)



Implementation

- ❑ **Current trend:** implementation on GPUs and hybrid CPU+GPU
- ❑ There are reported results corresponding to all models:
 - ❑ **Master-slave** (e.g. the EA is executed on CPU while elements evaluation is executed on GPU)
 - ❑ **Fine-grained (cellular)** – the whole EA is executed on GPU (rmk: there are also implementations which use CPU for generating random values); there are reported results of Cellular EAs using up to 10000 elements in the population
 - ❑ **Coarse-grained (island model)** – the population initialization and distribution in subpopulations is done on CPU, while the EA on each subpopulation is executed on GPU (the migration is realized by shuffling the subpopulations on GPU VRAM)
 - ❑ **Hierarchical models**

[Biblio: Arenas et al; GPU Parallel Computation in Bioinspired Algorithms. A review.]