Ministry of Science and Higher Education of the Russian

Federation National Research University ITMO

Evolutionary computing

Spring

2024

Laboratory work No. 6

DISTRIBUTED EVOLUTIONARY ALGORITHMS

1- Analysis of the work:

in this work, we implement the two common parallelization strategies for evolutionary algorithms, those strategies are:

The Master-Slave Algorithm: which employs a centralized control structure where a master thread distributes tasks to multiple slave threads.

The Island Model Algorithm: which divides the population into several sub populations (islands) that evolve independently and exchange information periodically.

2- Goals:

In this experiment, we aim to investigate the impact of parallelization on the performance of evolutionary algorithms. Specifically, we will assess the scalability of each algorithm by varying the number of threads or islands and measuring the execution time and solution quality for different problem sizes.

3- Code implementation:

A) crossover and mutation:

In the work from lab 2, only the initialization and crossover implementation were used for this lab. The mutation used in lab 2 was good (uniform mutation), but a better implementation was used, which is Gaussian mutation that gave much better results.

B) MasterSlaveAlg:

- EvolutionEngine: In the MasterSlaveAlg, the evolutionary engine is configured to use the master-slave strategy (Steady). where the evolutionary algorithm (SteadyStateEvolutionEngine) is responsible for distributing tasks across threads based on the configured threading mode.

```
AbstractEvolutionEngine<double[]> algorithm = new SteadyStateEvolutionEngine<double[]>(
        factory, pipeline, evaluator, selection, populationSize, forceSingleCandidateUpdate: false, random);
```

- Setting Single-Threaded or Multi-Threaded Mode: we added a boolean to allow mode switching for the algorithm to run in either single-threaded or multi-threaded mode.

```
algorithm.setSingleThreaded(isSingleThread);
```

- Monitoring and Reporting: we added an observer to monitor the population's progress and report relevant information.

```
algorithm.addEvolutionObserver(new EvolutionObserver() {
```

C) IslandAlg:

- IslandEvolution : this part is used to set up an island model for evolutionary computation. Where the migration strategy chosen is the RingMigration.

```
RingMigration migration = new RingMigration();
IslandEvolution<double[]> island_model = new IslandEvolution<>( islandCount: 10,
        migration, factory, pipeline, evaluator, selection, random);
```

- Evaluation Observer:

Evolution observer is attached to the island model to monitor the evolution process. Which will receive notifications about population updates and island-specific population updates during the evolutionary process.

```
island_model.addEvolutionObserver(new IslandEvolutionObserver() {
```

4- Conducting Experiments:

	Complexity = 0		parameters population - dimension - generation	**Complexity = 1**		parameters population - dimension - generation
	Lead time	**Result**		**Lead time**	**Result**	
ST (Single Thread)	26ms	0	Pop 2	31ms	4.97	Pop 4
MS (Master-Slave)	41ms	0	Dim 2	39ms	5.87	Dim 3
Islands	258ms	0	Gen 10	604ms	9.997	Gen 20
	Complexity = 2			**Complexity = 3**		
	Lead time	**Result**		**Lead time**	**Result**	
ST (Single Thread)	56ms	5.64	Pop 10	68ms	8.31	Pop 20
MS (Master-Slave)	61ms	7.70	Dim 4	89ms	8.48	Dim 5
Islands	1309ms	9.999	Gen 50	2648ms	9.9994	Gen 100
	Complexity = 4			**Complexity = 5**		
	Lead time	**Result**		**Lead time**	**Result**	
ST (Single Thread)	47ms	7.84	Pop 200	196ms	7.44	Pop 2000
MS (Master-Slave)	63ms	7.78	Dim 5	169ms	7.72	Dim 10
Islands	1317ms	9.9998	Gen 10	18995ms	9.9999	Gen 10

Now after conducting experiments, we can answer the following questions:

1. Which algorithm model is better under what conditions?

- In the context of having good results, the Island algorithm is giving much better results, which can be due to its inherent diversity and exploration capabilities. By maintaining multiple populations on different islands and allowing periodic migration of individuals between them, the Island algorithm promotes genetic diversity and facilitates exploration of the solution space more effectively. This increased exploration can help avoid local optima and discover better solutions, leading to superior performance compared to the master-slave approach, which typically operates with a single population and may struggle to explore the solution space thoroughly.

- in the context of the fast performance, the master-slave algorithm is much faster than the island algorithm even in single-threaded mode maybe it's due to differences in their computational overhead and resource utilization. Because In the master-slave algorithm, the master thread coordinates the evolution process and distributes tasks to slave threads for parallel execution. This parallelism allows for efficient utilization of available computational resources, enabling faster execution. While the island algorithm involves more complex communication and synchronization between different island populations. But it offers benefits in terms of exploration and solution quality, this additional overhead can result in longer execution times compared to the master-slave approach, particularly when the problem size or complexity increases. And therefore, even in single-threaded mode, the master-slave algorithm may outperform the island algorithm in terms of execution time due to its more efficient resource utilization and lower computational overhead.

2. How will increasing the dimension of the problem affect the algorithms?

- when having low number of generations and population size, the island will be a better choice, since it explores the search space more efficiently, at the same time, it won't take very long time.
- while when having bigger number of generations and population size, the master slave will be better because it will be able to go through the search space more with these high values of parameters and with much lower time than the Island algorithm.

3. What is the effect of increasing population size?

- increasing the population size will result in having better performance for the master slave algorithm, that's because as we saw in the table, the island algorithm time is increasing much faster than the master slave. So the island algorithm may face worse performance in scenarios where inter-island communication overhead is significant, while the master-slave algorithm may encounter challenges with scalability or diminishing returns in highly parallelized environments. But in general, the specific performance characteristics of each algorithm depend on the problem domain, algorithm parameters, and hardware infrastructure, so the results of this experiment may not be good to generalize.

4. Is there a limit to the number of islands?

- The number of islands can be limited by hardware resources such as available memory and processing power. Additionally, beyond a certain point, increasing the number of islands might not lead to significant improvements in result quality and might only increase execution times due to communication overhead between islands. So we should choose the island number depending on many scenarios (the value of other parameters, and the resources that we have).