Parallel Genetic Algorithms

By Trevor McCasland and Larry Hale
Summary of Genetic Algorithms

- Search algorithm based on the mechanics of natural selection
  - Initially starts with random guesses of solutions for a population
- Population evolves candidate solutions using genetic operators influenced by natural genetic variation
- Works well with little or no problem knowledge
  - Parameter tuning is difficult
- Inherently parallel
  - Requires a huge amount of computational resources
Summary of Genetic Operators

● Fitness Evaluation
  ○ Each individual is evaluated for its effectiveness

● Selection
  ○ Deciding which members of the population will be used to generate new solutions

● Crossover
  ○ Recombinations to generate new solutions

● Mutation
  ○ Random changes introduced to aid in finding a better solution
Genetic Algorithm Process

1. Initialize Population
2. Evaluate
3. Select
4. Mutate
5. Recombine
6. Next Generation
7. End?
   - Yes: Solution
   - No: Go back to Evaluate

Decision points:
- End? (Solution, Yes, No)

Flow:
- From Initialize Population to Evaluate
- From Evaluate to End?
- From End? to Solution or No
- From No to Select, then Mutate, Recombine, and Next Generation
- From Solution to End?
Classification of Parallel Genetic Algorithms

Classification relies on the computation/communication ratio

- Ratio is low for fine grain parallel GAs
- Fine grain has a single population and ideally a processing node for every individual

- Ratio is high for coarse grain parallel GAs
- Coarse grain has a population on each processing node
Master-Slave Parallelization

- Global single-population is stored in the master
- Computational load is distributed among slaves

![Diagram showing Master-Slave Parallelization]

- Master
- Genetic Operators
- Individuals
- Fitness
- Slaves
- Calculate Fitness
- Calculate Fitness
- Calculate Fitness
- Calculate Fitness
Master-Slave Synchronous vs Asynchronous

- Synchronous model
  - Only fitness calculations are done in slave processors
  - Stops and waits for all processors to finish
  - Selection, replacement, recombination, and mutation are all done in the master processor

- Asynchronous model
  - Selection, replacement, recombination, and mutation are all done in the slave processors
  - Slave processors do not wait for all processing to be done to continue on processing the next generation
Master-slave with clustering

- **Master:**
  - Generates clusters
  - Select optimal solution from final iteration of slave calculations
- **Slave:**
  - Generates initial population
  - Performs all other steps of genetic algorithm
  - Migrate at a migration interval
Fine-Grained Parallel GA

- Separated single-population
- Population is distributed over processors of a 2D mesh
- Neighbors share members of their population for parent selection
- Ideally would want one individual per processing unit

8-cell local neighborhood structure
Fine-Grained Parallel GA Basic Components

- Initialization
  - Initialization is the same as a simple GA but the individuals are evenly distributed across all processing units
- Selection
  - Each processing unit can select parents from its neighbors
- Replacement
  - Can replace individuals from other subpopulations or newly created individuals
- No change in crossover or mutation
Effects of Population Density in a Fine-Grained PGA

- The use of small isolated subpopulations in early stages of a run helps preserve genetic diversity across the whole population.
- Creating links to the isolated subpopulations by increasing population density can produce results similar to purely fine-grained PGA.
Percolation Model

- The 2-D mesh is initialized randomly with a certain probability to spawn a subpopulation in each cell
- Two new parameters called the *seeding interval* and the *seeding probability* are introduced

4-cell local neighborhood structure

After the *seeding interval* number of generations pass

With probability of $P_s$ to spawn a new subpopulation
Coarse-Grained Parallel GAs

- Multiple populations
- Each processor is assigned a simple GA and there is little communication between populations
- This also restricts the size of the selection and mating pools
Coarse-Grained Parallel GA Basic Components

- Initialization
  - Each island is created from the original population until the number of desired islands is satisfied

- Selection
  - Can select individuals from other populations

- Replacement
  - Can replace with individuals from other populations or newly created individuals

- No change in crossover or mutation
Migration in Coarse-Grained Parallel GAs

- Frequency of migration
  - Could occur periodically or based on a probability
  - Based on a minimum threshold of standard fitness deviation

- Amount of emigrants
  - Based on percentage of population or absolute value
  - Typically very small amounts of individuals are exchanged

- Selection and replacement
  - Random may be better because if everyone sends their best individual there is a higher risk for premature convergence
Communication Topologies

- **Static topologies**
  - Denser topologies mix the population better but has higher communication cost and can lead to premature convergence
  - Sparser topologies will grow independently and yield multiple solutions

- **Dynamic topologies**
  - Send individuals to populations that will have the most effect measured by genotypic difference between populations
Distributed Parallel GA Adaptive Migration Strategy

- Operates like a traditional coarse-grained parallel genetic algorithm but has two new operators for migration
- Migration Operator
  - Goal is to increase effectiveness of migration
  - Uses individual similarity for placement in migration buffer, greater similarity makes a greater chance of placement
- Accepted Operator
  - For every individual in the migration buffer
    - Only accept the individual if it does not exist in the population already and is better than the worst individual
Heterogeneity

- Heterogeneity provides a way of applying and tuning exploration/exploitation searches
- Co-operating Populations with Different Evolution Behavior (CoPDEB)
  - Populations have different probabilities of crossover and mutation
  - Populations have different exploration/exploitation properties
- Gradually Distributed Real-Coded GA
  - Creates two planes of search, one for exploration and the other for exploitation
  - Exchange of individuals between planes have different effects
CoPDEB

- Maintains populations of different evolution behavior by varying genetic operators.
- Exchange of information between populations allows them to co-operate, exploit the search space, and reintroduce previously lost genetic material.
Gradually Distributed Real-Coded GA

- This structure allows for a reliable spread search and an effective local tuning to be achieved simultaneously.
- Refinement degree increases clockwise or from $E_i$ to $e_i$.
- Expansion degree increases counter clockwise or from $e_i$ to $E_i$.
Parallel GA Hybrid Models

- Any logical combination of the master-slave, fine-grain, and coarse-grain models
Sources

- [http://neo.lcc.uma.es/Articles/albatroyaxx_2.pdf](http://neo.lcc.uma.es/Articles/albatroyaxx_2.pdf)
- [http://tracer.uc3m.es/tws/cEA/documents/cant98.pdf](http://tracer.uc3m.es/tws/cEA/documents/cant98.pdf)