Parallel Genetic Algorithms

Petr Pošík

Motivation

GAs applied on complex tasks need long run times to solve the problem:

- What is usually the most time-consuming task when solving real-world problems?
  - Fitness evaluation!!!
    - In complex tasks solved by GAs, chromosome is long, often genotype-phenotype mapping must be applied, ...
    - In GP, when evolving classifiers, functions, or programs, the fitness must be assessed by measuring the success when applying the classifier, function, or program on a set of training task instances
  - In EDAs, model building is very time consuming!

⇒ PARALLELIZE!!!

- Which of the above can be parallelized easilly???

Agenda

How can we parallelize?

1. Run several independent GAs in parallel.
2. Run single GA, but distribute the time consuming things to parallel machines. (**Master-slave model.**)
3. Run several *almost independent* GAs in parallel; exchange a few individuals from time to time. (**Island model.**)
4. Run single GA with selection that takes only a few individuals into account. (**Spatially embedded model.**)
5. Run hybrid parallel GA. (**Hierarchical model.**)
6. Other, less standard possibilities. (**Injection model, heterogenous PGA.**)

But first:

- The difference between parallel model and parallel implementation.

Parallel Implementation vs. Parallel Model

Sequential implementation:

- The algorithm is able to run on a single machine in a single process, often in a single thread only.

Parallel implementation:

- The algorithm is able to take advantage of multiple CPU cores or multiple machines.

The effect of parallelization:

- Reduction in the solution time by *adding a computational power*.
- The speed-up should be proportional to the number of parallel machines.

Global model:

- The population is not divided in any way, the selection operator can consider all individuals.

Parallel model:

- The population is somehow divided into subpopulations, which limits mainly the selection operator.

The effect of parallelization:

- Changes the algorithm behavior substantially.

Possible combinations:

- Sequential implementation of the global model (usual case, simple GA)
- Parallel implementation of the global model (master-slave, brute-force speed-up)
- Sequential implementation of a parallel model (modified behavior)
- Parallel implementation of a parallel model (modified behavior, brute-force speed-up)

Parallelization of the Global Model

Master-slave model

Master

- runs the evolutionary algorithm, and
- controls the slaves, distributes the work.

Slaves

- take batches of individuals from the master,
- evaluate them, and
- send their fitness back to master.

Other possibilities:

- Sometimes we can parallelize also initialization, mutation, and (with a bit of care) crossover.
- The hardest parts to parallelize are selection and replacement.
- When does the parallelization actually pay off???

Master-slave implementation does not change the behavior of the global model.

- Hints on implementation (locking, synchronizing) can be found in [Luk09, chap. 5].

[Luk09] Sean Luke. *Essentials of Metaheuristics*. 2009. available at http://cs.gmu.edu/~sean/book/metaheuristics/.

Island Model

Also called *coarse-grained PGA* or *multi-deme GA*:

- By far the most often used model of PGA.
- Population divided into several subpopulations (demes).
- Demes evolve independently. *Almost*.

Migration:

- Occassionally, the demes exchange some individuals.

The profit from island model:

- Demes are smaller:
  - converge faster,
  - can converge to different local optima, but
  - can converge prematurelly.
- Thanks to migration, new, *potentially good* (not random), genetic material can be obtained from other demes.

DEMO: Island model of PGA applied on TSP
`http://labe.felk.cvut.cz/~posik/pga`

Migration

Migration topology: Where should we take the migrants from and where should we put them?

- static: given in advance, does not change during evolution
- dynamic: the sources and targets are chosen right before particular migration event
  - can take the similarity of demes into account when choosing sources and targets
- degree of connectivity (DOC), δ:
  - the number of demes used as sources of migrants for another deme in one particular migration event
  - topologies with the same DOC exhibit similar behavior
  - in a comparison of fully-conected topology, 4D hypercube, 4×4 toroidal net, and one-way and two-way rings, densely connected topologies were able to find the global optimum with lower number of evaluation

Migration trigger: When should we run the migration?

- static schedule: migrate every n^{th} generation, at predefined time instants
- feedback trigger: migrate when it is needed, when the deme diversity drops below certain level
  - initiated by a source deme or by a target deme
  - diversity $\rightarrow$ convergence; population convergence vs. convergence in time

Migration (cont.)

Migration type: Can the migration events occur individually or in batches?

- batch: all migration events occur in the same time, all demes send emigrants to their targets and take the immigrants from their sources
- individual: a migration event (migrants move from one deme to another) can occur any time, independently of other events

Migration selection and replacement strategy:
Which individuals should be selected as emigrants? Which individuals should be replaced by imigrants?

- Best, worst
- Best, random
- Random, worst

Migration count: How many individuals should we migrate?

- often chosen from the interval

$$n_{\text{mig}} \in \langle 1, \frac{\text{deme size}}{\delta + 1} \rangle$$

Other possibilities, issues:

- sometimes, migration is described as *synchronous* or *asynchronous*, not used here; the meaning is not clear: synchronous with time vs. synchronous with other mig. event
- increase the fitness of migrants so that they can influence the target deme at least for 1 generation
- term *epoch* in the context of PGAs describes the part of evolution betweem 2 migration events

Other Parallel Models

Spatially Embedded Model

Also called *fine-grained PGA*:

- Population has a structure (1D grid, 2D toroidal grid, 3D cube, etc.)
- Each individual has a position in this structure.
- Individuals are allowed to breed with only nearby neighbors. Replace individual in certain slot with children bred from neighbors of this slot.
- The best individuals do not spread in the population so fast. Diversity promotion.
- Easy parallelization via multithreading.
- Very efficient model for *vector processors*, often found on GPUs:
  - many identical operations can be performed in parallel at one time

Model Combinations

Hierarchical model:

- various combinations of the above mentioned models, e.g.
- island model where each deme uses master-slave fitness evaluation,
- island model where each deme uses spatilly embedded model, etc.

Heterogenous model:

- Each deme uses a different optimizer
  - Different parameter settings
  - Different operators of selection, crossover, mutation and/or replacement
  - Completely different optimization algorithm (local search, differential evolution, ...)
  - Can each deme use a *different fitness function*???

Injection Model

Heterogenous island model where

- each deme uses a different fitness function!!!
- Usable when many quality criteria must be assessed; each deme
  - concetrates on one criterion and
  - submits partial solutions to other demes to be reworked using another criterion.
- Each deme preserves solutions of high quality when only its particular criterion is applied.

Summary

- Parallelization can increase the speed the EA:
  - parallel implementations
  - parallel models
- Parallel models change the behavior of the EA:
  - they can reduce the danger of premature convergence and speed-up the algorithm in the same time.
- There are many possibilities on parallelization:
  - the optimal decision depends on the (parallel) computer architecture and on the task being solved
  - all possibilities introduce their own set of tunable parameters :-(