Parallel Evolutionary Algorithms.
Coevolution.

Petr Pošík

Motivation

EAs 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 a 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 occasionally. (**Island model.**)
4. Run a single GA with selection that takes only a few individuals into account. (**Spatially embedded model.**)
5. Run a 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 *increasing 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 the 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 only with the neighbors nearby. 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 in the same 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
  - concentrates 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 :-(

Learning outcomes

After this lecture, a student shall be able to

- explain the main motivation for parallelization;
- explain the difference between parallel *implementation* and parallel *model* of EA;
- describe the features of individual combinations of sequential/parallel implementation and global/parallel model;
- know which parts of individual EAs (fitness evaluation, selection, replacement, mutation, crossover, model building, etc.) can be implemented in parallel easilly, and explain why;
- explain the principle and features of the master-slave parallelization;
- implement island model and explain its features, describe the characteristics of migration operator (type, topology, degree of connectivity, trigger, selection/replacement strategy, count);
- describe spatially embedded model, heterogeneous model, injection model, and their use cases.

What is “coevolution”?

Coevolution in EAs:

- The fitness of individuals in a population
  - is not given by the characteristics of the individual (only), but
  - is *affected by the presence of other individuals in the population*.
- It is closer to the biological evolution than ordinary EAs are.

Coevolution can help in

- dealing with increasing difficulty of the problem,
- providing diversity in the system,
- producing not just high-quality, but also robust solutions,
- solving complex or high-dimensional problems by breaking them into nearly decomposable parts.

Types of coevolution

By relation type:

- cooperative (synergic, compositional)
- competitive (antagonistic, test-based)

By the entities playing role in the relation:

- 1-population
  - intra-population
  - individuals from the same population cooperate or compete
- N-population
  - inter-population
  - individuals from distinct populations cooperate or compete

1-population competitve coevolution

Example: The goal is to evolve a game playing strategy

- successful against diverse opponents!!!

How would you proceed in an ordinary EA?

Problem: fitness evaluation

- by playing several games against human player? Against conventional program?
  - Problem: No learning gradient! Needle in a haystack. All randomly generated players will almost surely loose against any advanced player.
- by playing several games against internet players?
  - A bit better. . . but beware (Blondie24)

Solution: Intra-population competitive coevolution

- by playing several games against other strategies in the population.
- All individuals of the same type.
- In the beginning, all are probably quite bad, but some of them are a bit better.
- The fitness (the number of games won) may not rise as expected since your opponents improve with you.

2-population competitive coevolution

Example: The goal is to evolve a sorting algorithm

- able to sort any sequence of numbers
- correctly and quickly.

How would you proceed in an ordinary EA?

Problem: fitness evaluation

- Test all possible input sequences? Slow, intractable.
- Test only a fixed set of sequences? Which ones?

Solution: Inter-population competitive coevolution

- 2 populations, 2 species:
  - sorting algorithms
  - test cases (sequences to sort)
- Fitness evaluation:
  - Algorithm: by its ability to sort. How many sequences is it able to sort correctly? How quickly?
  - Test case: by its difficulty for the current sorting algorithms. How many algorithms did not sort it?
- Predator-prey relationship

N-population cooperative coevolution

Example: The goal is to evolve a team consisting of

- a goalie, back, midfielder, and forward
- so that they form a good team together.

How would you proceed in an ordinary EA?

Fitness evaluation:

- by simulating a number of games between teams

Problem: Evolution

- Represent all 4 strategies in 1 genome, evolve them all in 1 population.
- Theoretically possible, but the space is too large.
- May result in a team of players which wouldn't perform well if substituted to another team.

Solution: N-population cooperative coevolution

- 4 separate populations
- Evolve players which would play well with any other team members

Cooperation:

- symbiotic relationship
- good performance of the team $\Rightarrow$ high contribution to fitness of all members

1-population cooperative coevolution

Example: **Niching** methods for

- diversity preservation
- maintaining several stable subpopulations in diverse parts of the search space

Examples of niching methods:

- fitness sharing
- crowding

Principle:

- better individuals similar to others already in population are thrown away in favour of worse, but diverse individuals
- the selection process is affected by the presence of other individual in the neighborhood

Fitness in coevolution

Some important classifications of fitness

- by its time-dependence:
  - **static**: does not change with time
  - **dynamic**: changes with time
- by the stochastic element:
  - **deterministic**: generates the same ordering of a set of individuals
  - **stochastic**: can generate different orderings of the same set of individuals
- by the role of other individuals in evaluation:
  - **absolute**: measured independently of other individuals
  - **relative**: measured with respect to individuals in the current population
- by its role in the EA:
  - **internal**: optimization criterion used by selection
  - **external**: used to measure the progress of the algorithm

Ideally, external fitness

- should be **static**, **deterministic** and **absolute**
- can easilly be used as internal fitness

External fitness in coevolution:

- impossible (hard) to define
- often, it is **relative**, but measured with a carefully chosen, large enough set of other individuals (**static**) sufficiently many times (almost **deterministic**)

Internal fitness in coevolution:

- **relative**: affected by other individuals
- **dynamic**: affected by evolving individuals (needs re-evaluation)
- **stochastic**: usually evaluated against a smaller number of individuals

“Fitness” in sport

Football league:

- all teams play against all others
- points awarded for win, draw, and loss
- teams sorted by the earned points

Tennis players:

- tournaments divided to various levels, with different point amounts
- points awarded to players by their final standings in tournament

Golf players:

- tournaments have different prize money to distribute to tournament winners
- highly paid tournaments attract more players and are harder to win
- players sorted by the won prize money

Chess Elo ratings:

- each player is assigned a level, based on historic results
- matches between players of different levels
- the player’s level increases (decreases) if she recently won more (less) matches than expected

None of these systems is static:

- Is Pete Sampras better than Roger Federer?
- Is Arnold Palmer better than Tiger Woods?
- ...

The same holds for fitness assessment in coevolution!

Problems with fitness assessment: 1-pop. competitive coevolution

Cycles, etc.

- What if A beats B, B beats C, but C beats A?
- What if A beats B, but B beats far more individuals than A?
- The quality assessment depends on what we really want:
  - A player that beats the most other players?
  - A player that beats the most other "good" players?
  - A player that wins by the most total points on average?
- Often, additional matches are executed.
- But, do you want to spend your fitness budget
  - on evaluating current individuals more precisely, or
  - on searching further?

2 competitive populations (illustration)

Lotka-Volterra model (Predator-prey population dynamics):

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = -\gamma y + \delta xy$$

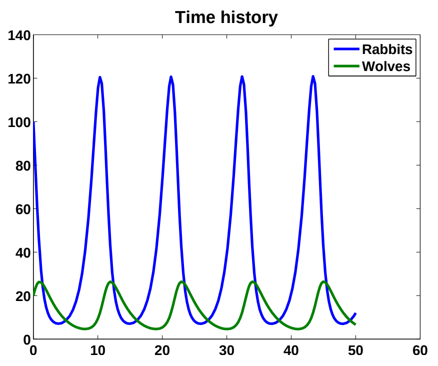
where x is the number of prey (rabbits) and y is the number of predators (wolves).

Assumptions:

1. The prey population has always food enough.
2. The predators eat only the prey.
3. The rate of change of population is proportional to its size.
4. The environment is static.

Meaning:

- The change of the prey population (dx/dt) is composed of
  - increase due to the newly born individuals (proportional to the population size, αx) and
  - decrese caused by the predation (which is proportional to the rate of predator-prey meetings, βxy).
- The change of the predator population (dy/dt) is composed of
  - decrease due to natural death (proportional to the population size, γy) and
  - increase alowed by the food suply (proportional to the rate of predator-prey meetings, δxy).



Problems with fitness assessment: 2-pop. competitive coevolution

Arms races

- one population learns a trick and forces the second population to learn a new trick to beat the first one...
- one population may evolve faster than the other:
  - all individuals from that population beat all the individuals from the other
  - no selection gradient in either population ⇒ uniform random selection
  - external fitness in both populations drops until the gradient re-emerges
- not exactly what was shown by Lotka-Volterra, but similar
- Solution:
  - detect such situation (but how?)
  - delay the evolution of the better population until the worse one catches up

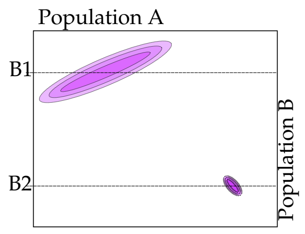
Problems with fitness assessment: N-pop. cooperative coevolution

Hijacking (in team of goalie, back, midfield, and forward):

- a really good forward takes over one population, any team will play well thanks to him
- members of all other populations have almost the same fitness ⇒ uniform random selection
- Solution: apply some form of *credit assignment*

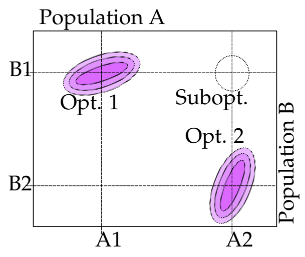
Relative overgeneralization

- when evaluated by average score, worse (but more robust) individual B1 will have higher score than better (but volatile) B2
- use maximum score (more tests needed)
- but again, the choice depends on what we want — a player able to get the highest score, or a player that would form a good team with the most other team members?



Miscoordination

- when the team components are not independent
- Pop. A evolved A2 (but not A1), pop. B evolved B1 (but not B2)
- Neither A2 nor B1 survives



Summary: Coevolution

Summary

Coevolution

- can be cooperative or competitive (or both)
- can take place in 1 population or in more populations
- fitness is not fixed during evolution
- introduces new unexpected dynamics to the system (new issues to be solved)

Appropriate when

- no explicit fitness function can be formed
- there are too many fitness cases
- the problem is modularizable (divide and conquer)

Learning outcomes

After this lecture, a student shall be able to

- define "coevolution", explain what makes it different from ordinary evolution in the context of optimization algorithms;
- explain differences between *cooperative* and *competitive* coevolution;
- explain differences between *intra-* and *inter-*population coevolution;
- define features of fitness measure (static/dynamic, deterministic/stochastic, absolute/relative, internal/external);
- describe the features of an ideal external (and internal) fitness, and describe the features of internal fitness in coevolution;
- exemplify individual types of cooperative/competitive intra-/inter-population coevolution; and
- exemplify various types of issues that can be observed in them (how to order individuals based on inconsistent matches, loss of diversity when one population evolves faster than the other, hijacking, relative overgeneralization, miscoordination).