CSE/ECE 848
Introduction to Evolutionary Computation

Module 4, Lecture 21, Part 2
Parallel EC—Heterogeneous Islands

Erik D. Goodman, Executive Director
BEACON Center for the Study of Evolution in Action
Professor, ECE, ME, and CSE

Reviewing Gains from Homogeneous Parallel Search (Island GA)

Island-parallel GA, for example:

- Produces "superlinear" speedup relative to # of processors used (e.g., get a speedup EVEN IF all evaluations done on one machine) compared to single-population GA
- Each island can be run on a separate processor, if desired, with almost NO communications overhead
- Widely shown to be useful, if population sizes and number of islands are not chosen very poorly

But speedup is limited, and search can still get "stuck"

Multi-Level GAs

- Pioneering Work – DAGA2, MSU (based on GALOPPS)
- Island GA populations are on lower level, their parameters/operators/neighborhoods on chromosome of a single higher-level population that controls evolution of subpopulations

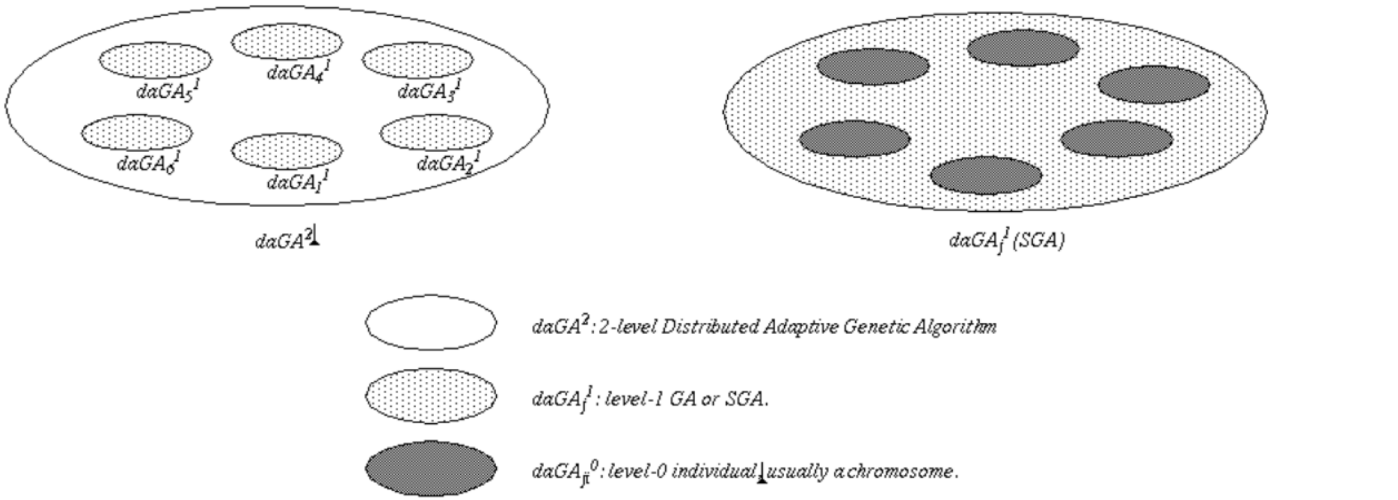


Figure 1: Demonstration of DAGA2 system.

Multi-Level GAs

DAGA2 (cont.)

- Can allow change in a subpop's parameters DURING a run, based on a progress measure
- Excellent performance – reproducible trajectories through operator space, for example

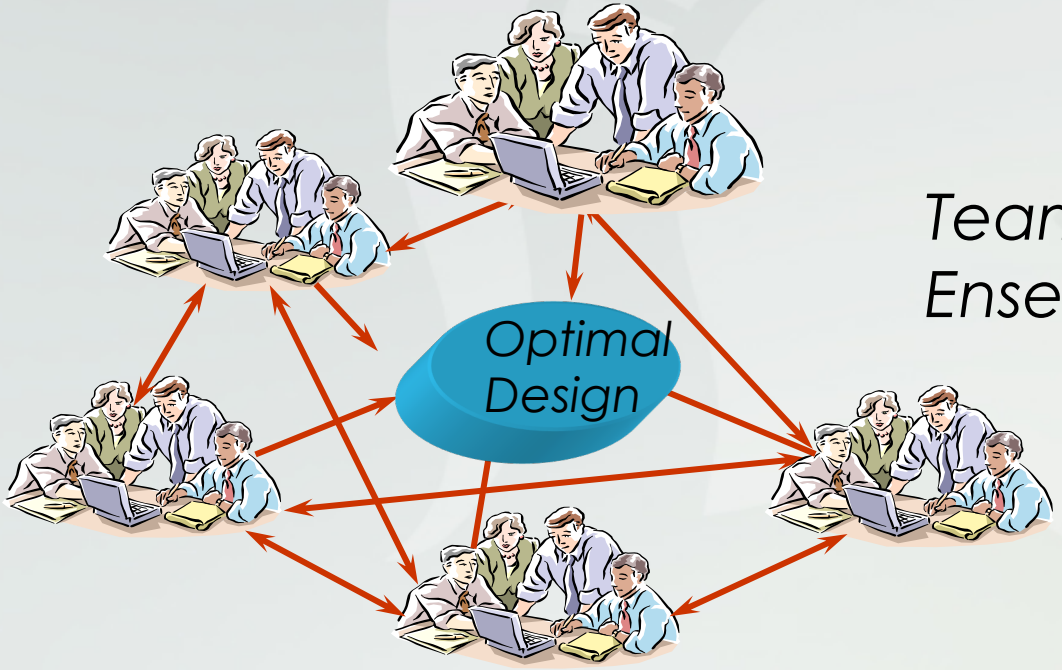
Refined Island Models – Heterogeneous/Hierarchical GAs

- For many problems, useful to use different representations/levels of refinement/types of models, allow them to exchange "nuggets"
- GALOPPS was first package to support this
- Injection Island architecture arose from this, now used in HEEDS, etc.
- Hierarchical Fair Competition was a later development (Jianjun Hu), breaking populations by fitness bands

Multiple, Heterogeneous Search Agents

Why not use many DIFFERENT search agents at the same time, working with

- DIFFERENT TOOLS
- DIFFERENT views of the problem?



Team: Intellectual Diversity
Ensemble: Better Answers

Approaches to Heterogeneous Agents

Agents might differ according to their:

- Physical/spatial domain
- Temporal extent of simulation
- Number of design variables
- Resolution of design variables
- Stochasticity of variables
- Performance measures
- Loading cases
- Constraint enforcement
- Analysis models
- Search methods
- …

Examples of Population-to-Population Differences in a Heterogeneous GA

- Different GA parameters (pop size, crossover type/rate, mutation type/rate, etc.)
  - Multi-level or all one level, without a master pop
- Different Representations, for example:
  - Hierarchy – one-way migration from least refined representation to most refined
  - Different models in different subpopulations
  - Different objectives/constraints in different subpops (sometimes used in Evolutionary Multiobjective Optimization ("EMO"))
- What is NEEDED if use different reps?
  - A way to translate a migrant from old to new rep

RECAPPING: Multiple Agents vs. Multiple Processors: Two DIFFERENT Ideas

Multiple Agents (coarse-grain parallel GA):

- Conceptually, work in parallel
- May be doing:
  - Same tasks
  - Different tasks
- May be implemented:
  - On one processor (switching contexts among agents)
  - On multiple processors (loosely coupled may be adequate)

Multiple Processors:

- May be used by a SINGLE search agent, "farming out" evaluations to a set of CPU's
- May be used by MULTIPLE, INTERACTING agents, working on DIFFERENT versions of the problem, different search methods, SIMULTANEOUSLY, each agent using one or many processors

MESSAGE: hardware need not drive the architecture of the search!

- But, well used, multiple processors can often speed search dramatically, with appropriate structuring
- For this purpose, steady-state GA (EC) may work better than generational approaches—less need to wait for other processes to finish, easier to keep many cores busy, not waiting

Why Heterogeneous Agents?

Why not just run several independent optimizations, using different methods, different problem representations, etc.?

- Good idea
- Better idea: run them simultaneously, letting each inform the others DURING SEARCH about promising design features it has found
  - Agents with simpler views make fast progress—they don't need to find the best solutions, **just not migrate the bad ones,** so slower agents don't have to look there!
  - They help tell agents with more complex models where to look ➔ far less wasted effort
  - This communication helps even in early stages of search

What Is Needed to Use Heterogeneous Agents?

- **Process integration**, so analyses can be run automatically, results may be extracted and passed to other agents
- **A powerful framework** to enable communication among agents during a run
- A translator for mapping one rep to another
- Even better if each agent has **hybrid method** capabilities within itself! No reason not to take BOTH gains!

Mapping Heterogeneous Agents to Computers

- During setup or for fast-to-analyze problems, can run all agents on one processor
- For speed, can run each agent on a different machine
- Each agent can still "farm out" evaluations to other machines using cluster/grid technologies, for even more speed, if desired and if enough software licenses for fitness evaluation are available
- Just need a way to specify how to translate a design from one representation to another when it moves from agent to agent
- We'll see an example in the next lecture…

Another Type of Parallel GA: Fine-Grain Parallel GAs

- Individuals distributed on cells in a tessellation, one or few per cell (often, toroidal checkerboard, as often done in Avida)
- Mating typically among near neighbors, in some defined neighborhood
- Offspring typically placed near parents
- Can help to maintain spatial "niches," thereby delaying premature convergence
- Interesting to view as a cellular automaton
- Not very commonly used today on real-world GA problems