Evolutionary computation in dynamic persistent environments

David Keil

October 4, 2004

Framingham State College
Computer Science Club talk series

Joint work with Dina Goldin,
University of Connecticut

The function-optimization problem

- Let $f : \mathbf{N}^k \rightarrow \mathbf{R}$ for some k (the *arity* of f)
- *Problem*: Find some $x \in \mathbf{N}^k$ s.t. $f(x)$ is maximal
- *Example*: Suppose x is the set of proportions of ingredients in a fuel mixture, $f(x)$ is fuel efficiency under this mixture
- *Optimizing* $f(x)$ means finding the most efficient mixture
- For an *algorithm* to optimize a function we must have $f : X \rightarrow Y$ with X, Y finite

Fitness and function optimization

- *Example*: Suppose f is viewed as a *fitness* function and x is the set of attributes of individuals of a population
- Then finding x s.t. $f(x)$ is maximal is finding the fittest possible individual of the species, i.e., those with the best attributes to assure survival
- Evolution by natural selection tends to optimize fitness, over many generations

Example: Checkers (Samuel, 1950s)

- Let fitness function $f : \mathbf{N}^k \rightarrow \mathbf{R}$ be an evaluator of checkers board positions from a black or red point of view
- Let $x \in \mathbf{N}^k$ be a k-tuple of *weights* for each of k different criteria for evaluating a checkers position
- *Example*: let x_1 be relative importance of number of kings, x_1 be relative importance of number of opponent checkers threatened, etc.
- Then writing a good checkers-playing program reduces to finding a good set of relative weights $x_{1, \ldots,} x_k$ for these criteria

The evolutionary algorithm

$t \leftarrow 0$, *Initialize* (P_0), $y \leftarrow$ *Evaluate* (P_0)
While not *terminate* (y, t) do
    $t \leftarrow t + 1$
    $P_t \leftarrow Select(P_{t-1}, y)$
    $P_t \leftarrow Alter(P_t)$
    $y \leftarrow Evaluate\ (P_t)$

- t is time; P_t is a population at time t
- *Evaluate* is a fitness function to be optimized, applied to individuals in the population
- *Select* is a method of choosing some members of P to survive to the next generation
- *Alter* is a method of changing P in a random way, e.g., by genetic mutation or crossover

No Free Lunch

- *Theorem*: For any function-optimization *algorithm*, for any environment (fitness function) in which the algorithm performs well…
- …there is some environment in which it performs equally poorly
- Precisely, no function-optimization algorithm performs better in the general case than random choice
- *Result:* Function optimization usually requires knowledge of the domain, i.e., heuristics; can only achieve approximation to optimality

Dynamic environments

- The function-optimization problem defined above is *static*, assumes *f* will be the same later as now
- Optimization in the real world, e.g., fitness in natural environments, such as climate, is often *dynamic*
- A separate, but related, category of environment is *stochastic* (imperfectly predictable)

Interactive computation

- *Definition:* An *interactive computation* is an ongoing exchange of data among computing agents, such that the output of each may causally influence its later inputs
- *Definition:* A *computing agent with persistent state* (CAPS) is an agent that accepts inputs, emits outputs, and has a *state* or *memory* whose value may evolve from one I/O step to the next
- *Discussion:* It can be shown that computing agents with persistent state are capable of a wider range of behaviors than ones without persistent state

Dynamic persistent environments

- *Definition:* A *dynamic persistent environment* (DPE) is a CAPS, *E*, that may interact with some other CAPS, *M*, with *E* changing state due to *M*'s actions in a way perceptible to *M*. We call *E* a *dynamic persistent environment with respect to M*.
- *Discussion:* If some inputs received from *E* by *M* have *reward value*, then *E* generates a *fitness function* w.r.t. *M* at each interaction step.
- This function maps *M*'s output to the value of *M*'s immediate reward
- Note that this function varies with the state of *E*, i.e., evolves over time

Evolution in DPEs

$t \leftarrow 0$, *Initialize* (P_0, E_0), $y \leftarrow$ *Evaluate* (P_0, E_0)
While not *terminate* (y, t) do
$\quad t \leftarrow t + 1$
$\quad E_t \leftarrow \textit{Alter-environment}(E_{t-1}, P_{t-1})$
$\quad P_t \leftarrow \textit{Select}(P_{t-1}, y)$
$\quad P_t \leftarrow \textit{Alter-population}(P_t)$
$\quad y \leftarrow \textit{Evaluate}\ (P_t, E_t)$

- The evolutionary algorithm (Slide 5) must model the environment explicitly if the environment is dynamic and persistent
- Note that the fitness function *Evaluate* is relative to the evolving state of the environment here

Resolving the No Free Lunch paradox

- Whereas no general-purpose function optimizing algorithm exists, the process of evolution has provably yielded adaptive and even intelligent life
- The key to resolving the paradox is that natural evolution does not optimize (static) functions, but operates in *dynamic persistent environments*

Overcoming limitations of algorithmic problem solving

- *Internal* and *indirect* interaction can support general-purpose adaptive behavior capable of attaining fitness in arbitrary dynamic persistent environments
- *Evolutionary* techniques appear to be the main way forward, provided the evolution occurs online, as the evolved objects interact with their environment

Some interaction patterns in natural dynamic settings

1. **Termites gathering chips**
   Protocol: Move at random, pick up chip when encountered, put down when another found
2. **Ants foraging for food**
   Ants leave chemical trail, prefer existing trails, blaze shorter and shorter trails to and from food
3. **Slime mold dividing and aggregating**
   These amoeba may aggregate by emitting chemical, migrating toward its greatest concentration

Self-organization and emergent behavior

- *Definition:* **Self-organization** is the interaction of a set of processes or structures at a lower level of a system to yield global structures or behavior at a higher level
- *Example:* Chemical reactions
- *Contrast to:* Centralized, algorithmic behavior
- System behavior that is not the sum of component behaviors is called *emergent*

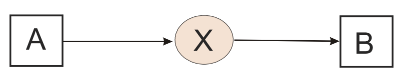
Stigmergy

- *Definition:* A variety of self-organization in which mobile agents interact via their shared environment
- *Contrast to:* direct interaction; centralized interaction
- *Examples:*
  - termites gathering chips,
  - ants foraging,
  - slime mold aggregation

Indirect interaction

Interaction via persistent, observable state changes, in which the destination of output is any agent that observes these state changes

- Agents *A* and *B* (right) may interact with each other *indirectly* via shared variable *x*



- Features:
  - anonymity (recipient ID not used in access)
  - time delay (state changes persist)
  - space decoupling (agents *A*, *B* need not meet)

Indirect interaction and multiagent systems

- In a MAS characterized by locality of interaction and mobility of agents, it is only possible for agents to influence overall system behavior remotely, i.e., indirectly
- Richness of multiagent interaction:
  - due partly to ability of each agent to interact with multiple others
  - hence indirectly with *all* others (otherwise system partitions)

References

D. Goldin and D. Keil. Modeling indirect interaction in open computational systems. *Proceedings*, *TAPOCS 2003*.

Z. Michalewicz. *Genetic algorithms + data structures = evolution programs*, 3rd Ed. Springer, 1996.

D. Wolper and W. Macready. No free lunch theorems for search. *Santa Fe Institute technical report SFI-TR-95-02-010*, 1995.