CSE/ECE 848 Introduction to Evolutionary Computation

Module 4 - Lecture 20 - Part 3

Dynamic Problems in Evolutionary Computation

Wolfgang Banzhaf, CSE
John R. Koza Chair in Genetic Programming

Ubiquity of DOPs

- Many real world problems are
  - noisy, accompanied by uncertainty
  - changing over time
- Changes could be related to
  - the objective/fitness function
  - the particular problem instance / representation
  - varying constraints
  - environmental parameters

Example of a DOP

Job scheduling problem:

- Arrival of new tasks
- Breakdown of machines
- Changes in economic or financial conditions
- Variance in available resources

Types of a DOP

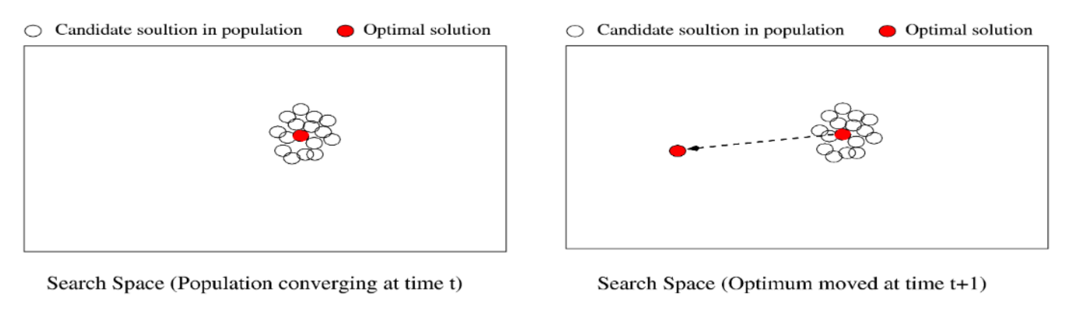
- The optimum moves linearly/nonlinearly in design space
- The optimum oscillates periodically between a number of locations
- The optimum jumps randomly

What does matter?

- Speed of change
- Severity of change
- Predictability of change
  - e.g. Exactness of periodicity, location, etc
- Detectability of change

DOP Challenges

- If optimum moves over time, one has to chase it!



From: S. Yang, EC for DOP, 2018

- But: Once converged, EC algorithms have a hard time to escape
- Or: EC algorithms might be confused because optimum moves faster than population

Methods for DOPs

- Maintaining diversity of the population to prevent it from converging
- Using explicit memory
- Using implicit memory
- Self-adaptation and learning EC methods

Diversity Maintenance

- Add randomly generated individuals (Grefenstette 1992)
- Fitness sharing (Anderson 1991)
- Aging of individuals (Ghosh et al 1998)
- Adaptive chaotic mutations (Nanayakkara et al 1999)
- Set lower bound on mutation rates (in ES) to prevent it from converging (Jin et al 2004)

Explicit and Implicit Memory

- Explicit memory
  - Store best solution in history (create an archive) and add them to the population if necessary (Mori et al 1998)
  - Store promising genetic material in a “gene library” for reuse (Tekol and Can 2003)
- Implicit memory
  - Multiple populations (Oppacher and Wineberg 1999)
  - Redundant coding (Smith 1987)

Self-adaptation and Learning

Self-adaptation

- Hyper-mutation and selection
  - Increase mutation rate when time-averaged best performance worsens (Cobb and Grefenstette 1993)
  - Raise selection pressure temporarily (Yang and Tinos 2008)
- Increase/decrease mutation rate with frequency of improvements (could become destabilized) (Weicker and Weicker 2000)

Life-time learning

- Lamarckian learning processes (modulation of learning parameters) (Sasaki and Tokyo 1998)
- Population-based incremental learning algorithm (PBIL, an EDA-type algorithm with modulation of parameters) (Yang and Richter 2009)