EVOLUTIONARY COMPUTING

EAs in Uncertain Environments

Sources of Uncertainty

- noise
- robustness
- fitness approximation
- time-varying fitness function

Noise

- fitness evaluations subject to noise, e.g. due to:
  - sensory measurement errors
  - randomized simulations
  - ...
- usually modelled as being distributed normally
- ideally EA must not be misled by noise
- in practice the expected fitness function is often approximated by averaged sum of a number of random samples

Robustness

- design variables subject to perturbations or changes after optimal have been found
  - e.g. due to manufacturing tolerances
- solution should still work
- a *robust* solution
- EA should work on an expected fitness based on a probability distribution of possible disturbances

Robustness and Noise

- very similar approach for EA but;
  - noise acts on the fitness function
    - cannot guarantee the same fitness value for the same individual in consecutive evaluations
  - robustness is due to perturbances in the design variables
    - fitness function is the same: solution changes after optimization

Fitness Approximation

- if fitness function too expensive to evaluate
- if an analytical fitness function is not available
  - ⇒ approximate fitness function generated from collected data or from simulations
- also known as meta-model
- approximate fitness function must be used together with original fitness function

Time Varying Fitness Function

- deterministic at any point in time but is dependent on time
  - the optimum also changes
- EA should be able to track the changing optimum
- information from preious environments should be re-used for speed up
- dynamic (changing) environments

Dynamic Environments - 1

- change in the environment through:
  - change in the objective function
  - change in the constraints
  - change in the problem instance
- usually causes optimum to change

Dynamic Environments - 2

- old solution must be adapted
- possible approaches:
  - treat as a new problem after change
    - change may not be detected immediately
    - new solution may not be too different from the old one ⇒ too time consuming to start from scratch
  - the optimization continuously adapts to the change

Dynamic Environments - 3

- EAs are good candidates to be used in optimizing in dynamic environments because:
  - EAs are based on natural evolution
  - in nature adaptation is continuous
- problem with standard EAs:
  - convergence
  - loss of diversity

Criteria for Categorization

- frequency of change
- severity of change
- predictability of change
- cycle length / cycle accuracy

Aspects to Consider when Designing an EA

- visibility of change
- necessity to change representation
- aspect of change
- EA influence on environment

Alternative Categorization

- constant (same change in every period)
- stationary (no change)
- periodic (returning to previous states)
- homogeneous (whole landscape moving coherently as opposed to different parts behaving differently)
- alternating (optimum jumps from one peak to another)

EA Approaches in Dynamic Environments

- restart after change
- generate diversity after a change
- maintain diversity throughout the run
- memory-based approaches
- multipopulation approaches

Restart

- population is re-initialized randomly after a change
- no information is transferred from the previous instance
- not recommended in most cases

Restart

- not very useful if the new solution is close to the old
  - some individuals may be transferred to new population to remedy this
  - the amount of info tranferred is important
    - too much may lead to convergence
    - too little slows down the search

Restart

- a knowledge base of individuals that perform well are kept, indexed with a measure of their environment
  - when change occurs, population is initialized using individuals that have performed well under similar conditions
  - it must be possible to measure environment similarities

Restart

- when problem representation changes due to change in environment (e.g. chromosome length) simple insertion of inidividual is not possible
  - individual must be adapted

Generate Diversity after Change

- adapting mutation rates
  - (triggered) hypermutation
  - variable local search

Adapting Mutation

- adapting mutation rate explicitly after change occurs
  - higher mutation rate helps converged population to spread out and search
- triggered hypermutation
  - whenever time averaged best performance of the GA worsens, mutation rate is increased drastically

©2007 Istanbul Technical University - Evolutionary Computation– Asst. Prof. Dr. A. Şima (Etaner) Uyar

Adapting Mutation

- Variable Local Search (VLS)
  - after change occurs, range of mutation is increased slowly
  - if population fitness does not improve, it is increased more
  - performs best with very small changes

©2007 Istanbul Technical University - Evolutionary Computation– Asst. Prof. Dr. A. Şima (Etaner) Uyar

Maintain Diversity Throughout

- random immigrants
- thermodynamical genetic algorithm (TDGA)
- sharing / crowding

©2007 Istanbul Technical University - Evolutionary Computation– Asst. Prof. Dr. A. Şima (Etaner) Uyar

Random Immigrants

- population is partly replaced by random new individuals in all generations
- equivalent to a mutation rate of 0.5 on some individuals
- preserves diversity in population

©2007 Istanbul Technical University - Evolutionary Computation– Asst. Prof. Dr. A. Şima (Etaner) Uyar

TDGA

- control diversity in population explicitly
  - through a measure named “free energy”
- for a minimization problem:

  F=<E>-TH

  where <E> is the average population fitness

  TH is the measure of diversity in population
- new population selected from parents and offspring one by one based on trying to minimize F
- T is a temperature parameter set to change importance of diversity over time

©2007 Istanbul Technical University - Evolutionary Computation– Asst. Prof. Dr. A. Şima (Etaner) Uyar

Introducing / Maintaining Diversity

- focusing on diversity slows down the optimization process
- results of tests performed show:
  - if the change is slow (low severity) triggered hypermutation performs better
  - in cases of higher severity changes, random immigrants perform better

Memory-based Approaches

- EA supplied with memory to recall useful information from past generations
- especially useful when optimum returns to previous locations
- two groups of approaches:
  - implicit memory approaches
  - explicit memory approaches
- better used with diversity-preserving methods

Implicit Memory Approaches

- implicit memory through redundant representations
  - diploidy/multiploidy
    - slow down convergence
    - increase diversity
    - dominance determination
- very good performance in oscillating environments

Explicit Memory Approaches

- explicit memory through introduction of memory to store good individuals
  - insertion of individuals from memory
    - percentage of individuals to be replaced is important
    - which individuals to replace is important
- best with periodically changing environments where old solutions are revisited

Multipopulation Approaches

- divide population into sub-populations
- allows tracking of peaks in search space
- different subpopulations maintain information about promising regions fo search space
- may be seen as self-adaptive memory

Multipopulation Approaches

- self-organizing scouts
- multinational GA
- shifting balance GA
- sentinels

Self Organizing Scouts

- when a peak is found
  - population splits
  - a small fraction called the "*scout population*" watches over the peak
  - rest of the population called the "*base population*" spreads out and continues search for new peaks
- over time, most promising regions of search space are covered while search continues

Self Organizing Scouts

- when a watched peak moves,
  - scout population may follow peak
  - scout population may request reinforcement
- population size is limited so,
  - individuals redistributed to sub-populations where they are most needed
  - unpromising regions may be abandoned
- successful results reported

Self Organizing Scouts

- design decisions
  - define a peak (which areas to be surveyed by scouts)
  - how to determine a peak is found so that population splits
  - how many individuals to keep at each peak
  - when to abandon a peak
  - what happens if two scout populations move towards same peak
  - how many peaks to survey simultaneously

Self Organizing Scouts

```
REPEAT
   Compute next generation of base and scout
  populations
   Adjust search space for scout populations
   IF (forking generation)
      Create new scout population
      Adjust number of individuals in base
          and scout populations
UNTIL termination criterion
```


Shifting Balance GA

- main aim is to increase exploratory power
- divide population into a *core* and a number of small *colony* populations
- core population exploits best optimum found
- colony populations forced to search in other parts of landscape (exploration)

Shifting Balance GA

- if a colony gets close to core population, it is driven away using a distance measure
- at intervals, colonies send emigrants to core to update its gene pool
- good performance only with small changes in the environment

Multinational GA

- grouping of individuals based on *hill-valley detection* procedure
  - for two points in the search space, the fitness of a number of random individuals on the line between the two points
  - valley is detected if a point is lower than both ends
  - detected valley define borders of subpopulation

Multinational GA

- requires many extra fitness evaluations to detect valleys
- mutation strength increases as the individual gets away from the best in its subpopulation
- good results reported on two peak environments
- shown to be better than sharing

Sentinels

- *sentinels* are population members distributed uniformly on search space
  - regular members for selection and crossover
  - they are never replaced
- when population converges around a peak and environment changes, other sentinels will get a chance to be selected for reproduction

Sentinels

- main aim is to have a uniform distribution of individuals on the search space
- *dispersion* automaticaly increases
- problem of good sentinel placement
  - successful methods exist in literature
- successful results reported

Suitable Benchmark Problems

- should be possible to vary many of the environmental variables
  - peak heights
  - peak shapes
  - peak locations
- should provide benchmarking for binary and real valued encodings

Suitable Benchmark Problems

- should be possible to vary change dynamics
  - change frequency
  - change severity
    - slow drifting motion
    - oscillation
    - abrupt (catastrophic) changes

Suitable Benchmark Problems

- should be simple to implement
  - computational efficiency
- should be simple to analyze
- should allow conjectures to real world problems

Commonly used Benchmark Problems

- moving peaks benchmark
- DF1
- XORing generator
- dynamic knapsack problem
- dynamic MKP
- dynamic bit-matching problem

Measuring Performance

- goal is to track the progression of optima as closely as possible
- should not compare individuals evaluated with different fitness functions
- not meaningful to use
  - best-so-far curves
  - error plots (if optima known)
  - regular offline / online performance measures

Requirements for Good Performance Measures

- intuitive meaning
- straightforward methods for statistical significance testing of comparative results
- measurement of performance over a sufficiently large exposure to landscape dynamics

Performance Measures

- measures that require the optima to be known
  - difference between optimum and best individual just before each change
  - modified offline performance
  - average Euclidean distance to optimum point in each generation
- the first two also require the detection of change)

Performance Measures

- measures not giving info on overall landscape
  - average best-of-generation values at each generation over several runs of same problem
  - (*BestOfGeneration – WorstWithinTimeWindow*) compared to

  (*BestWithinTimeWindow – WorstWithinTimeWindow*)

Online Performance

- average of all evaluations over entire run
- every evaluation requires testing the real world

x is the online performance, $\mathbf{e_t}$ is the *t*-th evaluation and **T** is the number of evaluations considered:

$$x = \frac{1}{T}\sum_{t=1}^{T} e_t$$

Offline Performance

- average of the best values found so far
- optimization is done in a simulated environment and only best solutions are transferred to real world
  - for non-stationary environments, offline performance should only consider individuals evaluated since the last change
  - requires that the changes are known/detected

Offline Performance for Dynamic Environments

$$x' = \frac{1}{T}\sum_{t=1}^{T} e_t' \quad where$$
$$e_t' = \max\{e_\tau, e_{\tau+1}, \ldots, e_t\}$$

x': offline performance for dynamic environments
T: the number of evaluations to be considered

- τ is the last time step (τ <t) at which a change occurred

Offline Error

- same as offline performance but calculates error
- works only if optima are known

Collective Mean Fitness

$$F_C = \frac{\sum_{g=1}^{G}\left(\sum_{m=1}^{M}\frac{F_{BG}}{M}\right)}{G}$$

M: number of runs
G: number of generations

- possible to give also collective mean error if optimal fitness values known
- approximately 20 environment changes sufficient for small range of fitness values)

Measure for Tracking Ability

$$A_T = \frac{\sum_{g=1}^{G}\left(\frac{\sum_{m=1}^{M}\sqrt{\sum_{i=1}^{N}(x_{i,BG} - x_{i,BP})^2}}{M}\right)}{G}$$

M: number of runs
G: number of generations
N: number of dimensions
$x_{i,BG}$: position of optimum
$x_{i,BP}$: position of point

- requires position of optima to be known

Detecting Changes

- change may explicitly be made known to the system
- change has to be detected by the system

Detecting Changes

- indicators for change:
  - deterioration of population performance
  - deterioration of the time averaged best performance
  - re-evaluation of several individuals (assume at least one of the evaluations should change)
  - explicitly maintain a model of environment and when response predicted by model differs from actual response, assume change has occured

Detecting Changes

- false change detection is detrimental to EA performance
- better to have approaches which do not require the detection of changes

References (*for Part 1*)

- Jürgen Branke, *Evolutionary Optimization in Dynamic Environments*, Kluwer, 2002.
- Karsten Weicker, *Evolutionary Algorithms and Dynamic Optimization Problems*, der Andere Verlag, 2003.
- Ronald W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*, Springer, 2004.
- Yaochu Jin, Jürgen Branke, “Evolutionary Optimization in Uncertain Environments – A Survey”, IEEE Transactions on Evolutionary Computation, Vol. 9, No. 3, IEEE Press, 2005.