Evolutionary Dynamic Optimisation
Problems and Challenges

Philipp Rohlfshagen
prohlf@essex.ac.uk
www.philipprohlfshagen.net

University of Essex

February 24, 2011

---

This work was supported by EPRSC grant no EP/E058884/1.
Outline

1. Evolutionary Dynamic Optimisation: An Overview
   - Evolutionary Computation and Optimisation
   - Dynamic Optimisation Problems
   - Performance Measures
   - Review of Techniques

2. Evolutionary Dynamic Optimisation: An Assessment
   - Assessment of the Field
   - Solution Concepts
   - Conclusions
Assume a mapping $f : X \rightarrow \mathbb{R}$

- Search space $X$ (set of potential solutions)
- $f(x) \in \mathbb{R}$ is a measure of quality of $x \in X$
- Goal: find $x^*$ such that $\forall x \in X$, $f(x^*) \geq f(x)$, $\geq \in \{\geq, \leq\}$

Functions encountered are often very complex

- NP-hard
- Black-Box

Often content with satisficing (not optimising)

- Time taken to find $x$ such that $f(x) \geq c$
- The value $f(x)$ given bounded resources (memory, time)

\[
\text{performance} \equiv \max\{f'_1, \ldots, f'_\tau\}
\]
Evolutionary Computation

- Use Evolutionary Algorithms to tackle $f(x)$
  - Class of nature-inspired heuristics (usually black box)
  - Genetic Algorithms, Genetic Programming, Evolution Strategies, ...

- Overall framework:
  - Initial population: random and diverse
  - Explore: mutation and crossover
  - Exploit: selection
  - Convergence over time: similarity of individuals
Dynamic Optimisation Problems

- Many real-world problems change over time
  - Machines fail unexpectedly, dynamic traffic patterns, etc.
  - Need to respond in real-time (e.g., re-schedule)
  - New technologies allow us to capture more data (e.g., GPS)

- Assume a mapping $f : X \times \mathbb{N} \rightarrow \mathbb{R}$
  - Add the component time $t \in \mathbb{N}$

- Time-variant series of instances $I(\cdot)$ of the same problem $\Pi$

  \[ I(\pi = 0) \rightarrow I(\pi = 1) \rightarrow \ldots \rightarrow I(\pi = m) \]

- Changes may affect:
  - Parameters/coefficients, constraints, domain, dimensionality, ...
Trajectory-Based Performance Measures

- How to evaluate algorithms in the dynamic domain?
  - Need to take time into account
  - Single solution no longer appropriate: tracking

All evaluations matter: \[ \text{ONLINE} = \frac{1}{t} \sum_{i=1}^{t} f_i \]

Assume model: \[ \text{OFFLINE} = \frac{1}{t} \sum_{i=1}^{t} \max\{f'_1, f'_2, \ldots, f'_i\} \]

Assume known changes: \[ \text{M-OFFLINE} = \frac{1}{t} \sum_{i=1}^{t} \max\{f'_{\lfloor t/\tau \rfloor_1}, \ldots, f'_{\lfloor t/\tau \rfloor_i}\} \]

Don’t assume known changes: \[ \text{COLLECTIVE} = \frac{1}{G} \sum_{i=1}^{G} (\text{BoG}_i) \]
How to evaluate algorithms in the dynamic domain?

- Need to take time into account
- Single solution no longer appropriate: tracking

All evaluations matter

\[
\text{ONLINE} = \frac{1}{t} \sum_{i=1}^{t} f_i
\]

Assume model

\[
\text{OFFLINE} = \frac{1}{t} \sum_{i=1}^{t} \max\{f'_1, f'_2, \ldots, f'_i\}
\]

Assume known changes

\[
\text{M_OFFLINE} = \frac{1}{t} \sum_{i=1}^{t} \max\{f'_{\lfloor t/\tau \rfloor_1}, \ldots, f'_{\lfloor t/\tau \rfloor_i}\}
\]

Don’t assume known changes

\[
\text{COLLECTIVE} = \frac{1}{G} \sum_{i=1}^{G} (\text{BoG}_i)
\]
Application of EAs to DOPs: Challenges

- Simply apply standard EA to DOP: no extensions
  - Loss of diversity (convergence) is a problem
  - Even convergence to the global optimum may be problematic

- Restart algorithm following change
  - Destroys all domain-specific information
  - Time taken to re-optimise may be too long
  - Want to improve on random restarts

- Almost all techniques fall in between these extremes
  - Exploit domain specific information (population, memory; next)
  - Allow for (re-)adaptation by maintaining diversity
Techniques for DOPs

- Diversity-preserving Techniques
  - Increase level of population diversity, constantly or reactively
  - Hyper-mutations [5], random immigrants [10]

- Memory
  - Implicit, explicit, direct, indirect
  - diploidy [9], archives [19]

- Representations
  - Additional complexity to deal with dynamics
  - Folding GA [8], duality [18]
Techniques for DOPs

- Search Operators and Memetic Algorithms
  - Constant, adaptive or reactive operators
  - Dissortative mating [6], adaptive mutation [2], hyper-selection [20]
  - Memetic algorithm with adaptive hill climbing [17]

- Speciation and Multi-populations
  - Distribute focus across the search space, implicitly or explicitly
  - Niching [4], forking GA [15], self-organising scouts [3]

- Anticipation and Prediction
  - Try to predict future states of the problem, time-linkage
  - Future population [16], linear/non-linear regression [13, 14]
Many new algorithms have been suggested in recent years
- Mostly variations of classical evolutionary algorithms
- Baseline performances often given by classical algorithms
- Limited number of (weak?) benchmark problems
- Arbitrary dynamic versions of classical problems (e.g., TSP)

How has the field progressed in the last 10 years?

Popovici and Wiegand (2005 CEC tutorial)
- Understand your problem:
  - what do you want to achieve?
- Understand your algorithm:
  - to what types of solutions is it drawn?
- Match the two:
  - use the right tool for the right job.
Assessment of the Field

Many new algorithms have been suggested in recent years
- Mostly variations of classical evolutionary algorithms
- Baseline performances often given by classical algorithms
- Limited number of (weak?) benchmark problems
- Arbitrary dynamic versions of classical problems (e.g., TSP)

How has the field progressed in the last 10 years?

Popovici and Wiegand (2005 CEC tutorial)
- Understand your problem:
  what do you want to achieve?
- Understand your algorithm:
  to what types of solutions is it drawn?
- Match the two:
  use the right tool for the right job.
**Assessment of the Field**

- **Many** new algorithms have been suggested in recent years
  - Mostly variations of classical evolutionary algorithms
  - Baseline performances often given by classical algorithms
  - Limited number of (weak?) benchmark problems
  - Arbitrary dynamic versions of classical problems (e.g., TSP)

**How has the field progressed in the last 10 years?**

- Popovici and Wiegand (2005 CEC tutorial)
  - Understand your problem:
    - what do you want to achieve?
  - Understand your algorithm:
    - to what types of solutions is it drawn?
  - Match the two:
    - use the right tool for the right job.
Understand Your Problem

- Assumptions (often expressed only implicitly)
  1. Successive global optima are correlated (in genotype space)
  2. Changes are unknown (and may not be detected easily)
  3. Constant frequency of change (coincides with population update)

1. Distances between successive global optima
   - Analyse SSP using FDC [12, 11]
   - Distances depend on instance: *Fitness-based* assumption
   - Theoretical result: large magnitude of change is easier

2. Change detection in the combinatorial domain
   - Looked at SSP, QAP, TSP, KP, NK, SAT
   - Change by smallest possible degree
   - Almost all points affected: change can be detected easily

3. Frequency of change
   - Choice of intervals criticised by numerous practitioners
What Do You Want to Achieve?

- Given a function $f(x, t)$ that somehow changes over time
  - Maximise a trajectory of f-values (multiple f-values over time)
  - Most common metric: collective mean fitness

- Collective mean fitness is algorithm dependent
  - Measures f-value every $N$ time steps (i.e., every generation)
  - Difficult to reduce to smaller time-scale
  - Adjust population sizes of algorithms to be compared

- Implementation Schedule $\mathcal{I}$
  - Part of the problem’s specifications
  - A set of time points $t \in \{0, \ldots, t_{\text{end}}\}$ specifying solutions to be implemented

- Online versus offline: pseudo-online
  - All metrics are offline: best-so-far (pseudo-online)
  - Assumes accurate, constantly updated model
Use *Solution Concepts* to identify solutions
- Popularised by Ficici in the field of co-evolution [7]
- All algorithms *implement* some solution concept

Divergence between actual and desired outcome in co-evolution
- Lack of properly formulated solution concept
- Primary search effort: gradient following
- Secondary search effort: gradient creation
- Practitioners often use same solution concept for both

Similar phenomenon in evolutionary dynamic optimisation
- Top-down instead of bottom-up (dynamics not properly acknowledged)
- Assumptions are carried over from the stationary domain

Algorithm \(\rightarrow\) problem \(\rightarrow\) performance measure \(\rightarrow\) solutions
- Problem \(\rightarrow\) solutions \(\rightarrow\) performance measure \(\rightarrow\) algorithm
Are algorithms drawn to the right solutions?

- Global optimum: $x_t^* \forall t \in I$ (from collective mean fitness)
- Have to content with satisficing (solutions that are “good enough”)

Maximising the collective mean fitness means:

- Find high quality solutions quickly (cumulative reward; convergence)
- Sample high quality solutions following a change (diversity)
- Ability to improve on current solution quality (diversity)
- Area under the curve [1]

$$\text{max}\{f'_1, \ldots, f'_\tau\} \neq \frac{1}{G} \sum_{i=1}^{G} (BoG_i)$$
To What Types of Solutions is the Algorithm Drawn?

$T \quad T+1$

$x_T^* \in P \quad x_T^{*+1} \in P$

need to take future into account

$x_T^* \in P \quad x_T^{*+1} \in P$

require rapid convergence

not considered

additional diversity

asumptions

starting point depends on problem

not considered

change
Solution? A 2-Objective Solution Concept

- Optimisation in the dynamic domain entails 2 objectives
  - Optimise the presence, optimise the future
  - Even first objective differs from the stationary case
  - Existing algorithms use same solution concept for both search efforts
  - Implicit acknowledgment of dynamics confluences these search efforts

- Example: random immigrants
  - Introduces $\gamma N$ random solutions into population every generation
  - Maintains diversity but significantly hinders progress
  - “Random guesses” of what points may be useful in the future
  - $\rightarrow$ not drawn to desired solutions

- Can we use the same solution concept for both objectives?
  - Depends on the problem: objectives need to be aligned
  - Search tactics could differ substantially
  - Might need to sample different parts of the search space
Solution? A 2-Objective Solution Concept

- A clear separation of objectives has many advantages
  - Minimise interference if objectives are orthogonal (example next)
  - Can carry out search efforts in serial, parallel
  - Mutual exchange of information between search efforts

- Simple function to illustrate:

\[
twoMax(x) = \max\{|x|_0, |x|_1\} + \prod_{i=1}^{n} x_i
\]
Conclusions

- Need to better understand the problem: bottom-up approach
  - A different notion of optimality
  - Need to make explicit assumptions about the problem and its dynamics
  - Identify solutions of interest and analyse how they may be obtained

- Design algorithms accordingly: implemented solution concepts
  - Fully exploit the problem’s dynamics
  - No need to address both search efforts with the same mechanism
  - Might need to explore different parts of the search space
  - Need more advanced algorithms (time series prediction, data mining, ...)

- Estimate future problem instances
  - Reduce the black-box uncertainty over time (learning)
  - Aim for expected monotonic increase in f-values over time
  - Current algorithms “stabilise” very quickly
E. Alba and B. Sarasola.
Abc, a new performance tool for algorithms solving dynamic optimisation problems.

A. Boumaza.
Learning environment dynamics from self-adaptation: a preliminary investigation.

A multi-population approach to dynamic optimization problems.

W. Cedeno and V. R. Vemuri.
On the use of niching for dynamic landscapes.

H. G. Cobb.
An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependant nonstationary environments.
C. M. Fernandes and A. C. Rosa.
InTech, 2008.

S. G. Ficici.
*Solution Concepts in Coevolutionary Algorithms*.

A. Gaspar and P. Collard.
From GAs to artificial immune systems: Improving adaptation in time dependent optimization.

D. E. Goldberg and R. E. Smith.
Nonstationary function optimization using genetic algorithms with dominance and diploidy.
J. J. Grefenstette.
Genetic algorithms for changing environments.

T. Jones and S. Forrest.
Fitness distance correlation as a measure of problem difficulty for genetic algorithms.

P. Rohlfshagen and X. Yao.
Attributes of dynamic combinatorial optimisation.

A. Simoes and E. Costa.
Improving prediction in evolutionary algorithms for dynamic environments.
A. Simoes and E. Costa.
Prediction in evolutionary algorithms for dynamic environments using markov chains and nonlinear regression.

S. Tsutsui, Y. Fujimoto, and A. Ghosh.
Forking genetic algorithms: Gas with search space division schemes.

A “futurist” approach to dynamic environments.

A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems.
S. Yang.


S. Yang.

Memory-based immigrants for genetic algorithms in dynamic environments.

S. Yang and R. Tinos.

Hyper-selection in dynamic environment.