Evolutionary Computation for Dynamic Optimization Problems

Shengxiang Yang
Centre for Computational Intelligence
School of Computer Science and Informatics
De Montfort University, United Kingdom
Email: syang@dmu.ac.uk
http://www.tech.dmu.ac.uk/~syang

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). Copyright is held by the owner(s).

ACM 978-1-4503-3488-4/15/07.
http://dx.doi.org/10.1145/2739482.2756589

Instructor/Presenter — Shengxiang Yang

Education and career history:
PhD, Northeastern University, China, 1999
Worked at King’s College London, University of Leicester, and Brunel University, 1999-2012
Joined De Montfort University (DMU) as Professor in Computational Intelligence (CI) in July 2012
Director of Centre for Computational Intelligence (CCI)

Research interests:
Evolutionary Computation (EC) and nature-inspired computation
Dynamic optimisation and multi-objective optimisation
Relevant real-world applications
Over 190 publications and over £1.2M funding as the PI
Chair of two IEEE CIS Task Forces
EC in Dynamic and Uncertain Environments
Intelligent Network Systems

Outline of the Tutorial

Part I: Fundamentals
- Introduction to evolutionary computation (EC)
- EC for dynamic optimization problems (DOPs): Concept and motivation
- Benchmark and test problems
- Performance measures

Part II: Approaches and Case studies
- EC enhancement approaches for DOPs
- Case studies

Part III: Issues, future work, and summary
- Relevant issues
- Future work
- Summary and references
What Is Evolutionary Computation (EC)?

- EC encapsulates a class of stochastic optimization algorithms, dubbed Evolutionary Algorithms (EAs)
- An EA is an optimisation algorithm that is
  - Generic: a black-box tool for many problems
  - Population-based: evolves a population of candidate solutions
  - Stochastic: uses probabilistic rules
  - Bio-inspired: uses principles inspired from biological evolution

EC Applications

- EAs are easy-to-use: No strict requirements to problems
- Widely used for optimisation and search problems
  - Financial and economical systems
  - Transportation and logistics systems
  - Industry engineering
  - Automatic programming, art and music design
  - ......

Design and Framework of an EA

Given a problem to solve, first consider two key things:
- Representation of solution into individual
- Evaluation or fitness function

Then, design the framework of an EA:
- Initialization of population
- Evolve the population
  - Selection of parents
  - Variation operators (recombination & mutation)
  - Selection of offspring into next generation
- Termination condition: a given number of generations

EC for Optimisation Problems

- Traditionally, research on EAs has focused on static problems
  - Aim to find the optimum quickly and precisely
- But, many real-world problems are dynamic optimization problems (DOPs), where changes occur over time
  - In transport networks, travel time between nodes may change
  - In logistics, customer demands may change
**What Are DOPs?**

- In general terms, "optimization problems that change over time" are called **dynamic problems/time-dependent problems**
  
  \[ F = f(\vec{x}, \vec{\phi}, t) \]

  \( \vec{x} \): decision variable(s); \( \vec{\phi} \): parameter(s); \( t \): time

- **DOPs**: special class of dynamic problems that are solved online by an algorithm as time goes by

---

**Why DOPs Challenge EC?**

- For DOPs, optima may move over time in the search space
  - Challenge: need to track the moving optima over time

- **DOPs challenge traditional EAs**
  - Once converged, hard to escape from an old optimum

---

**Why EC for DOPs?**

- Many real-world problems are DOPs
- EAs, once properly enhanced, are good choice
  - Inspired by natural/biological evolution, always in dynamic environments
  - Intrinsically, should be fine to deal with DOPs
- Many events on EC for DOPs recently

---

**Relevant Events**

- Books (Monograph or Edited):
  - Yang & Yao, 2013; Yang et al., 2007; Morrison, 2004; Weicker, 2003; Branke, 2002
- PhD Theses:
- Journal special issues:
  - Neri & Yang, 2010; Yang et al., 2006; Jin & Branke, 2006; Branke, 2005
- Workshops and conference special sessions:
  - EvoSTOC (2004–2015): part of Evo*
  - EvoDOP ('99, '01, '03, '05, '07, '09, '11): part of GECCO
Benchmark and Test DOPs

- Basic idea: change base static problem(s) to create DOPs
- Real space:
  - Switch between different functions
  - Move/reshape peaks in the fitness landscape
- Binary space:
  - Switch between ≥ 2 states of a problem: knapsack
  - Use binary masks: XOR DOP generator (Yang & Yao’05)
- Combinatorial space:
  - Change decision variables: item weights/profits in knapsack problems
  - Add/delete decision variables: new jobs in scheduling, nodes added/deleted in network routing problems

Moving Peaks Benchmark (MPB) Problem

- Proposed by Branke (1999)
- The MPB problem in the D-dimensional space:
  \[ F(\vec{x}, t) = \max_{i=1,\ldots,p} \left\{ H_i(t) \right\} \]
  \[ = W_i(t) \sum_{j=1}^{D} (x_j(t) - X_j)^2 \]

- The dynamics:
  \[ H_i(t) = H_i(t-1) + \text{height}_i \cdot \sigma \]
  \[ W_i(t) = W_i(t-1) + \text{width}_i \cdot \sigma \]
  \[ \vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|} ((1 - \lambda)\vec{r} + \lambda \vec{v}_i(t-1)) \]
  \[ \vec{X}_i(t) = \vec{X}_i(t-1) + \vec{v}_i(t) \]

- \( \sigma \sim N(0, 1) \); \( \lambda \): correlated parameter
- \( \vec{v}_i(t) \): shift vector, which combines random vector \( \vec{r} \) and \( \vec{v}_i(t-1) \) and is normalized to the shift length \( s \)

The DF1 Generator

- Proposed by Morrison & De Jong (1999)
- The base landscape in the D-dimensional real space:
  \[ f(\vec{x}) = \max_{i=1,\ldots,p} \left[ H_i - R_i \cdot \sqrt{\sum_{j=1}^{D} (x_j - X_j)^2} \right] \]

- \( \vec{x} = (x_1, \ldots, x_D) \): a point in the landscape; \( p \): number of peaks
- \( H_i, R_i, X_i = (X_1, \ldots, X_D) \): height, slope, center of peak \( i \)
- The dynamics is controlled by a logistics function:
  \[ \Delta t = A \cdot \Delta t_{i-1} \cdot (1 - \Delta t_{i-1}) \]
  \[ A \in [1.0, 4.0] \]: a constant; \( \Delta t \): step size of changing a parameter

Dynamic Knapsack Problems (DKPs)

- Static knapsack problem:
  - Given \( n \) items, each with a weight and a profit, and a knapsack with a fixed capacity, select items to fill up the knapsack to maximize the profit while satisfying the knapsack capacity constraint
- The DKP:
  - Constructed by changing weights and profits of items, and/or knapsack capacity over time as:
    \[ \text{Max } f(\vec{x}(t), l) = \sum_{i=1}^{n} p_i(t) \cdot x_i(t), \text{ s. t. } \sum_{i=1}^{n} w_i(t) \cdot x_i(t) \leq C(t) \]
    \[ \vec{x}(t) \in \{0, 1\}^n \]: a solution at time \( t \)
    \[ x_i(t) \in \{0, 1\} \]: indicates whether item \( i \) is included or not
    \[ p_i(t) \text{ and } w_i(t) \]: profit and weight of item \( i \) at \( t \)
    \[ C(t) \]: knapsack capacity at \( t \)
### The XOR DOP Generator

- **The XOR DOP generator** can create DOPs from any binary $f(\vec{x})$ by an XOR operator "⊕" (Yang, 2003; Yang & Yao, 2005).
- Suppose the environment changes every $\tau$ generations.
- For each environmental period $k = \lfloor t/\tau \rfloor$, do:
  - Create a template $T_k$ with $\rho \times I$ ones.
  - Create a mask $\vec{M}(k)$ incrementally:
    $$\vec{M}(0) = \vec{0} \quad \text{(the initial state)}$$
    $$\vec{M}(k+1) = \vec{M}(k) \oplus \vec{T}(k)$$
  - Evaluate an individual:
    $$f(\vec{x}, t) = f(\vec{x} \oplus \vec{M}(k))$$
  - $\tau$ and $\rho$ controls the speed and severity of change respectively.

### Constructing Cyclic Dynamic Environments

- Can extend the XOR DOP generator to create cyclic environments:
  - Construct $K$ templates $\vec{T}(0), \ldots, \vec{T}(K-1)$.
  - Form a partition of the search space.
  - Each contains $\rho \times I = I/K$ ones.
  - Create $2K$ masks $\vec{M}(i)$ as base states:
    $$\vec{M}(0) = \vec{0} \quad \text{(the initial state)}$$
    $$\vec{M}(i+1) = \vec{M}(i) \oplus \vec{T}(iK), i = 0, \ldots, 2K-1$$
  - Cycle among $\vec{M}(i)$’s every $\tau$ generations.

### Constructing Cyclic Environments with Noise

We can also construct cyclic environments with noise:

- Each time before a base state is entered, it is bitwise changed with a small probability $\rho$. (Shengxiang Yang, De Montfort University)

### Dynamic Traveling Salesman Problems

- **Stationary traveling salesman problem (TSP):**
  - Given a set of cities, find the shortest route that visits each city once and only once.
- **Dynamic TSP (DTSP):**
  - May involve dynamic cost (distance) matrix $D(t)$.
  - $d_{ij}(t)$: cost from city $i$ to $j$; $n$: the number of cities.
  - The aim is to find a minimum-cost route containing all cities at time $t$.
  - DTSP can be defined as $f(x, t)$:
    $$f(x, t) = \min\left\{ \sum_{i=1}^{n} d_{x_i,x_{i+1}}(t) \right\}$$
    where $x_i \in 1, \ldots, n$. If $i \neq j$, $x_i \neq x_j$, and $x_{n+1} = x_1$.
Dynamic Permutation Benchmark Generator

- The dynamic benchmark generator for permutation-encoded problems (DBGP) can create a DOP from any stationary TSP/VRP by swapping objects:
  - Generate a random vector \( \vec{r}(T) \) that contains all objects every \( f \) iterations
  - Generate another randomly re-order vector \( \vec{r}'(T) \) that contains only the first \( m \times n \) objects of \( \vec{r}(T) \)
  - Modify the encoding of the problem instance with \( m \times n \) pairwise swaps


Effect on Algorithms

- Similar with the XOR DOP generator, DBGP shifts the population of an alg. to new location in the fitness landscape
- The individual with the same encoding as before a change will have a different cost after the change

<table>
<thead>
<tr>
<th>Evolutionary Algorithms</th>
<th>Population of Individuals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = (0, 0.5, 1.5, 2) = 18</td>
</tr>
<tr>
<td></td>
<td>2 = (0, 0.5, 1.5, 1) = 18</td>
</tr>
<tr>
<td></td>
<td>3 = (0, 1.5, 1.5, 0) = 9</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>s = (0, 0.5, 1.5, 2) = 25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Heuristic Information Matrix</th>
<th>Population of Individuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

- Can extend for cyclic and cyclic with noise environments

GDBG: Dynamic Change Types

- Change types:
  - Small step: \( \Delta \phi = \alpha \cdot \|\phi\| \cdot \text{rand()} \)
  - Large step: \( \Delta \phi = \|\phi\| \cdot (\alpha + (1 - \alpha)\text{rand()} \)
  - Random: \( \Delta \phi = \|\phi\| \cdot \text{rand()} \)
  - Chaotic: \( \phi(t+1) = A \cdot \phi(t) \cdot (1 - \phi(t)/\|\phi\|) \)
  - Recurrent: \( \phi(t+1) = \phi(t(P)P) \)
  - Recurrent with nosy: \( \phi(t+1) = \phi(t(P)P) + \alpha \cdot \|\phi\| \cdot \text{rand()} \)

- More details:

Generalized DOP Benchmark Generator (GDBG)

- Proposed by Li & Yang (2008), GDBG uses the model below:
  - In GDBG, DOPs are defined as:
    - \( F = f(x, \phi(t), t) \)
    - \( \phi \): system control parameter
    - Dynamism results from tuning \( \phi \) of the current environment
      - \( \phi(t+1) = \phi(t) \oplus \Delta \phi \)
      - \( \Delta \phi \): deviation from the current control parameter(s)
    - The new environment at \( t+1 \) is as follows:
      - \( f(x, \phi(t), t) \rightarrow f(x, \phi(t) \oplus \Delta \phi, t) \)
DOPs: Classification

Classification criteria:
- Time-linkage: Does the future behaviour of the problem depend on the current solution?
- Predictability: Are changes predictable?
- Visibility: Are changes visible or detectable?
- Cyclicity: Are changes cyclic/recurrent in the search space?
- Factors that change: objective, domain/number of variables, constraints, and/or other parameters

DOPs: Common Characteristics

Common characteristics of DOPs in the literature:
- Most DOPs are non time-linkage problems
- For most DOPs, changes are assumed to be detectable
- In most cases, the objective function is changed
- Many DOPs have unpredictable changes
- Most DOPs have cyclic/recurrent changes

Performance Measures

- For EC for stationary problems, 2 key performance measures
  - Convergence speed
  - Success rate of reaching optimality
- For EC for DOPs, over 20 measures (Nguyen et al., 2012)
  - Optimality-based performance measures
    - Collective mean fitness or mean best-of-generation
    - Accuracy
    - Adaptation
    - Offline error and offline performance
    - Mean distance to optimum at each generation
  - Behaviour-based performance measures
    - Reactivity
    - Stability
    - Robustness
    - Satisficability
    - Diversity measures

Performance Measures: Examples

Collective mean fitness (mean best-of-generation):

\[ F_{BOG} = \frac{1}{G} \times \sum_{j=1}^{N} \left( \frac{1}{N} \times \sum_{i=1}^{G} F_{BOG_i} \right) \]

- \( G \) and \( N \): number of generations and runs, resp.
- \( F_{BOG_i} \): best-of-generation fitness of generation \( i \) of run \( j \)

Adaptation performance (Mori et al., 1997)

\[ Ada = \frac{1}{T} \sum_{t=1}^{T} \left( t_{best}(t) / t_{opt}(t) \right) \]

Accuracy (Trojanowski and Michalewicz, 1999)

\[ Acc = \frac{1}{K} \sum_{i=1}^{K} \left( t_{best}(i) - t_{opt}(i) \right) \]

- \( t_{best}(i) \): best fitness for environment \( i \) (best before change)
Part II: Approaches and Case studies

- EC enhancement approaches for DOPs
- Case studies

EC for DOPs: First Thinking

- Recap: traditional EAs are not good for DOPs
- Goal: to track the changing optimum
- How about restarting an EA after a change?
  - Natural and easy choice
  - But, not good choice because:
    - It may be inefficient, wasting computational resources
    - It may lead to very different solutions before and after a change.
    - For real-world problems, we may expect solutions to remain similar
- Extra approaches are needed to enhance EAs for DOPs

Memory Approaches

- Cyclic DOPs: change cyclically among a fixed set of states

- Memory works by storing and reusing useful information

- Two classes regarding how to store information
  - Implicit memory: uses redundant representations
    - Multiploidy and dominance (Ng & Wong, 1995; Lewis et al., 1998)
    - Dualism mechanisms (Yang, 2003; Yang & Yao, 2005)
  - Explicit memory: uses extra space to store information

EC for DOPs: General Approaches

- Many approaches developed to enhance EAs for DOPs
- Typical approaches:
  - Memory: store and reuse useful information
  - Diversity: handle convergence directly
  - Multi-population: co-operate sub-populations
  - Adaptive: adapt generators and parameters
  - Prediction: predict changes and take actions in advance
- They have been applied to different EAs for DOPs
Implicit Memory: Diploid Genetic Algorithm

Encoding Dominance Scheme

Chromosome 1
Chromosome 2

Genotype
Phenotype
Fitness

External Environment
Evaluating

Same Phenotypic Alleles
Evaluating
Fitness

Genotypic Alleles:
Phenotypic Alleles:

- Each individual has a pair of chromosomes
- Dominance scheme maps genotype to phenotype
- Dominance scheme may change or be adaptive (Uyar & Harmanci, 2005)

Ng & Wong (1995)
Lewis et al. (1998)

Explicit Memory Approaches

Basic idea: use extra memory

- With time, store useful information of the pop into memory
- When a change occurs, use memory to track new optimum

Direct memory: store good solutions (Branke, 1999)
Associative memory: store environmental information + good solutions (Yang & Yao, 2008)

Explicit Memory: Direct vs Associative

Associative Memory Based Genetic Algorithm

Idea: Use allele distribution (AD) $\vec{D}$ to represent environmental info.

- Use memory to store $<\vec{D}, S>$ pairs
- Update memory by similarity policy
- Re-evaluate memory every generation. If change detected
  - Extract best memory AD: $\vec{D}_M$
  - Create solutions by sampling $\vec{D}_M$
  - Replace them into the pop randomly

Details:
Diversity Approaches: Random Immigrants

- Convergence is the key problem in metaheuristics for DOPs
- Random immigrants:
  - Each generation, insert some random individuals (called random immigrants) into the population to maintain diversity
  - When optimum moves, random immigrants nearby take action to draw the population to the new optimum

Shengxiang Yang (De Montfort University) Tutorial: EC for DOPs GECCO’15, 11/7/2015 37 / 84

Memory-Based Immigrants

- Random immigrants maintain the diversity while memory adapts an algorithm directly to new environments
- Memory-based immigrants: uses memory to guide immigrants towards current environment
  - Re-evaluate the memory every generation
  - Retrieve the best memory point \( E(t) \) as the base
  - Generate immigrants by mutating \( E(t) \) with a prob.
  - Replace worst members in the population by these immigrants

Shengxiang Yang (De Montfort University) Tutorial: EC for DOPs GECCO’15, 11/7/2015 38 / 84

Experimental Results: Immigrants Based GAs

- Cyclic Dynamic OneMax Function, \( \tau = 25, \rho = 0.1 \)
- Random Dynamic OneMax Function, \( \tau = 25, \rho = 0.1 \)

Shengxiang Yang (De Montfort University) Tutorial: EC for DOPs GECCO’15, 11/7/2015 39 / 84

Hybrid Immigrants Approach

- Combines elitism, dualism and random immigrants ideas
- Dualism: Given \( \vec{x} = (x_1, \cdots, x_l) \in \{0, 1\}^l \), its dual is defined as \( \vec{x}^d = \text{dual}(\vec{x}) = (x_1^d, \cdots, x_l^d) \in \{0, 1\}^l \)
  where \( x_i^d = 1 - x_i \)
- Each generation \( t \), select the best individual from previous generation, \( E(t-1) \), to generate immigrants
  - Elitism-based immigrants: Generate a set of individuals by mutating \( E(t-1) \) to address slight changes
  - Dualism-based immigrants: Generate a set of individuals by mutating the dual of \( E(t-1) \) to address significant changes
  - Random immigrants: Generate a set of random individuals to address medium changes
  - Replace these immigrants into the population

Shengxiang Yang (De Montfort University) Tutorial: EC for DOPs GECCO’15, 11/7/2015 40 / 84
Experimental Results: Hybrid Immigrants GA

- Hybrid immigrants improve GA's performance for DOPs efficiently

Multi-Populations: Shifting Balance

- Multi-population scheme uses co-operating sub-populations
- Shifting Balance GA (Oppacher & Wineberg, 1999):
  - A core population exploits the promising area
  - Several colonies explore the search space

Adaptive Approaches

- Aim: Adapt operators/parameters, usually after a change
  - Hypermutation (Cobb & Grefenstette, 1993): raise the mutation rate temporarily
  - Hyper-selection (Yang & Tinos, 2008): raise the selection pressure temporarily
  - Hyper-learning (Yang & Richter, 2009): raise the learning rate for Population-Based Incremental Learning (PBIL) temporarily
  - Combined: Hyper-selection and hyper-learning with re-start or hypermutation
Prediction Approaches

- For some DOPs, changes exhibit predictable patterns
- Techniques (forecasting, Kalman filter, etc.) can be used to predict
  - The location of the next optimum after a change
  - When the next change will occur and which environment may appear
- Some relevant work: see Simões & Costa (2009)

Remarks on Enhancing Approaches

- No clear winner among the approaches
- Memory is efficient for cyclic environments
- Multi-population is good for tracking competing peaks
  - The search ability will decrease if too many sub-populations
- Diversity schemes are usually useful
  - Guided immigrants may be more efficient
- Different interaction exists among the approaches
- Golden rule: balancing exploration & exploitation over time

Case Study: GA for Dynamic TSP

- **Dynamic TSP:**
  - 144 Chinese cities, 1 geo-stationary satellite, and 3 mobile satellites
  - Find the path that cycles each city and satellite once with the minimum length over time
- Solver: A GA with memory and other schemes
- More details:

Case Study: GAs for Dynamic Routing in MANETs – 1

- Shortest path routing problem (SPRP) in a fixed network:
  - Find the shortest path between source and destination in a fixed topology
- More and more mobile ad hoc networks (MANETs) appear where the topology keeps changing
- Dynamic SPRP (DSPRP) in MANETs:
  - Find a series of shortest paths in a series of highly-related network topologies
- We model the network dynamics as follows:
  - For each change, a number of nodes are randomly selected to sleep or wake up based on their current status
Case Study: GAs for Dynamic Routing in MANETs – 2

- A specialized GA for the DSPRP:
  - Path-oriented encoding
  - Tournament selection
  - Path-oriented crossover and mutation with repair
- Enhancements: Immigrants and memory approaches
- Experimental results:
  - Both immigrants and memory enhance GA's performance for the DSPRP in MANETs.
  - Immigrants schemes show their power in acyclic environments
  - Memory related schemes work well in cyclic environments
- More details:

Case Study: PSO for Continuous DOPs

- PSO was inspired by models of swarming and flocking
- First introduced by Kennedy and Eberhart in 1995
- Standard PSO: particle position and velocity update rules

\[ v_i^d = \omega v_i^d + c_1 \cdot r_1 \cdot (p_{best}^d - x_i^d) + c_2 \cdot r_2 \cdot (g_{best}^d - x_i^d) \]

\[ x_i^d = x_i^d + v_i^d \]

- \( x_i^d \) and \( x_i^d \): the \( d \)-th dimension of the current and previous position of particle \( i \)
- \( v_i \) and \( v_i \): current and previous velocity of particle \( i \)
- \( p_{best} \) and \( g_{best} \): best so far position found by particle \( i \) and by the whole swarm
- PSO has been applied for many static optimization problems

PSO for Continuous DOPs

- Recently, PSO has been applied for continuous DOPs
- Two aspects to consider:
  - Outdated memory: Two solutions:
    - Simply set \( p_{best} \) to the current position
    - Reevaluate \( p_{best} \) and reset it to current position if it is worse than the current position
  - Diversity loss: Three solutions:
    - Introduce diversity after a change
    - Maintain diversity during the run
    - Use multi-swarms

Multi-swarm PSO for DOPs

- Aim: To enhance the diversity by maintaining multiple swarms on different peaks
- Key questions:
  - How to guide particles to different promising sub-regions
  - How to determine the proper number of sub-swarms
  - How to calculate the search area of each sub-swarm
  - How to create sub-swarms
- Algorithms:
  - Kennedy's \( k \)-means clustering algorithm
  - Brits's \( n_{best} \) PSO algorithm and niching PSO (NichePSO)
  - Parrott and Li's speciation based PSO (SPSO)
  - Blackwell and Branke's charged PSO (mCPSO) and quantum swarm optimization (mQSO)
- Potential problems:
  - There may be improper number of sub-swarms
  - One sub-swarm might cover more than one peaks
  - One peak might be surrounded by more than one sub-swarms
Multi-swarm: Clustering PSO (CPSO)

- Recently, we developed a Clustering PSO (CPSO) for DOPs
  - Training: Move particles toward different promising regions
  - Clustering: Use a Single Linkage Hierarchical Clustering to create sub-swarms
  - Local search: Each sub-swarm will search among one peak quickly

More details:
- Li & Yang, CEC 2009: 439-446

Experiments on GDBG System

<table>
<thead>
<tr>
<th>Tab. XVIII</th>
<th>OVERALL PERFORMANCE OF CPSO, mPSO, and MPSO ON ALL THE TEST CASES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>0.922403</td>
</tr>
<tr>
<td>T2</td>
<td>0.922403</td>
</tr>
<tr>
<td>T3</td>
<td>0.922403</td>
</tr>
<tr>
<td>T4</td>
<td>0.922403</td>
</tr>
<tr>
<td>T5</td>
<td>0.922403</td>
</tr>
<tr>
<td>T6</td>
<td>0.922403</td>
</tr>
<tr>
<td>T7</td>
<td>0.922403</td>
</tr>
<tr>
<td>T8</td>
<td>0.922403</td>
</tr>
<tr>
<td>T9</td>
<td>0.922403</td>
</tr>
<tr>
<td>T10</td>
<td>0.922403</td>
</tr>
</tbody>
</table>

Experiments on MPB Functions

- Comparison with mQSO and mCPSO on MPB with different shift severities

<table>
<thead>
<tr>
<th>s</th>
<th>CPSO</th>
<th>mCPSO</th>
<th>mQSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.89</td>
<td>1.18</td>
<td>1.18</td>
</tr>
<tr>
<td>1.0</td>
<td>1.49</td>
<td>2.05</td>
<td>1.75</td>
</tr>
<tr>
<td>2.0</td>
<td>2.63</td>
<td>2.80</td>
<td>2.40</td>
</tr>
<tr>
<td>3.0</td>
<td>3.96</td>
<td>3.57</td>
<td>3.00</td>
</tr>
<tr>
<td>4.0</td>
<td>2.05</td>
<td>4.18</td>
<td>3.59</td>
</tr>
<tr>
<td>5.0</td>
<td>2.24</td>
<td>4.89</td>
<td>4.24</td>
</tr>
<tr>
<td>6.0</td>
<td>2.29</td>
<td>5.53</td>
<td>4.79</td>
</tr>
</tbody>
</table>

Summary of CPSO for DOPs

- The nearest neighbour training strategy can efficiently guide randomly initialized particles to different promising sub-regions
- CPSO scales well regardless the number of peaks in the fitness landscape over other PSO algorithms
- The clustering method in CPSO is effective to generate sub-swarms
- It is still difficult to create accurate sub-swarms. More work should be done to solve this problem
Adaptive Multi-Swarm Optimizer (AMSO)

- Single linkage hierarchical clustering is used to create populations
  - All populations use the same search operator for local search
- An overcrowding scheme is used to remove unnecessary populations
- To find out proper moments to increase diversity without change detection, a special rule is designed according to the drop rate of the number of populations over a certain period of time
- To create a proper number of populations needed in each environment, an adaptive method is developed according to the information collected from the whole populations since the last diversity-increasing point

More details:
- Li, Yang & Yang, Evol Comput, 22(4): 559-594, 2014

Experiment Results

<table>
<thead>
<tr>
<th>Error</th>
<th>AMSO</th>
<th>CPSO</th>
<th>PSO</th>
<th>TPSO</th>
<th>SPBO</th>
<th>EPOS</th>
<th>mCPSO</th>
<th>mPSO</th>
<th>SAMO</th>
<th>DynDE</th>
<th>PopD</th>
<th>HmSO</th>
<th>SOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{\text{offline}}$</td>
<td>2.3±0.25</td>
<td>2.8±0.17</td>
<td>4±0.29</td>
<td>6±0.55</td>
<td>5.9±0.58</td>
<td>7.8±0.62</td>
<td>3.5±0.34</td>
<td>3.6±0.38</td>
<td>3.7±0.26</td>
<td>4.1±0.13</td>
<td>4.6±0.09</td>
<td>5.4±3.2</td>
<td></td>
</tr>
<tr>
<td>$E_{\text{BBC}}$</td>
<td>1.9±1.0</td>
<td>1.9±1.0</td>
<td>5.4±3.0</td>
<td>3.0±1.0</td>
<td>7.7±2.0</td>
<td>2.8±3.0</td>
<td>3.8±1.0</td>
<td>3.7±1.0</td>
<td>3.7±1.0</td>
<td>4.2±1.0</td>
<td>5.4±3.4</td>
<td>3.7±1.0</td>
<td></td>
</tr>
<tr>
<td>$E_{\text{offline}}$</td>
<td>2.8±0.74</td>
<td>3.3±0.3</td>
<td>5±1.0</td>
<td>4.6±0.67</td>
<td>3.9±0.89</td>
<td>7.3±0.92</td>
<td>3.4±0.68</td>
<td>3.5±0.81</td>
<td>4.2±0.75</td>
<td>4.2±1.2</td>
<td>4.6±4.3</td>
<td>4.3±1.3</td>
<td></td>
</tr>
<tr>
<td>$E_{\text{BBC}}$</td>
<td>2.9±1.2</td>
<td>2.4±1.0</td>
<td>2.7±1.0</td>
<td>2.7±1.0</td>
<td>2.8±1.0</td>
<td>2.8±1.0</td>
<td>2.8±1.0</td>
<td>2.8±1.0</td>
<td>2.8±1.0</td>
<td>2.8±1.0</td>
<td>2.8±1.0</td>
<td>2.8±1.0</td>
<td></td>
</tr>
<tr>
<td>$E_{\text{offline}}$</td>
<td>0.45±0.04</td>
<td>0.29±0.02</td>
<td>0.36±0.06</td>
<td>0.66±0.08</td>
<td>0.58±0.05</td>
<td>0.32±0.05</td>
<td>0.09±0.02</td>
<td>0.4±0.02</td>
<td>0.4±0.02</td>
<td>0.4±0.02</td>
<td>0.4±0.02</td>
<td>0.4±0.02</td>
<td></td>
</tr>
<tr>
<td>$E_{\text{BBC}}$</td>
<td>1.7±1.0</td>
<td>1.6±1.0</td>
<td>2.2±1.0</td>
<td>3.9±1.0</td>
<td>3.7±1.0</td>
<td>6.4±1.0</td>
<td>3.3±1.0</td>
<td>2.8±1.0</td>
<td>4.1±1.0</td>
<td>12±1.0</td>
<td>4.1±1.0</td>
<td>8.5±1.0</td>
<td></td>
</tr>
</tbody>
</table>

Case Study: ACO for DOPs

- ACO mimics the behaviour of ants searching for food
- The first ACO algorithm was proposed for TSPs (Dorigo et al’96)
- Generally, ACO was developed to be suitable for graph optimization problems, such as TSP and VRP
- The idea was to let ants “walk” on the arcs of the graph while “reading” and “writing” pheromones until they converge into a path
- Standard ACO consists of two phases:
  - Forward mode: Construct solutions
  - Backward mode: Pheromone update
- Conventional ACO cannot adapt well to DOPs due to stagnation behaviour
  - Once converged, it is hard to escape from the old optimum
Pheromone Evaporation

- Pheromone evaporation is the adaptation mechanism in ACO
- It helps to eliminate the high intensity of pheromone trails that may misguide ants to search in non-promising areas
- However, the pheromone evaporation rate depends on the magnitude of change and the problem size (Mavrovouniotis and Yang’13).

ACO for DOPs: General Comments

- ACO’s knowledge transfer makes sense on slight changes; otherwise, it may misguide the search
  - A global restart is a better choice on more severe changes
- A global restart of ACO ⇒ pheromone re-initialization
- Moreover, ACO has to maintain adaptability, instead of stagnation behaviour, to accept knowledge transferred
- Recently, many approaches developed with ACO for DOPs
  - Pheromone modification after a change (Guntsch and Middendorf’01, Eyckelhof and Snoek’02)
  - Memory-based schemes (Guntsch and Middendorf’02)
  - Hybrid and memetic algorithms (Mavrovouniotis and Yang’11)
  - Pheromone modification during execution (Mavrovouniotis and Yang’12,’13)
  - Multi-colony schemes (Mavrovouniotis, Yang and Yao’14)

Pheromone Modification After a Change

- Pheromone strategies are applied to DTSP where cities are exchanged
- Global pheromone strategies ⇒ Initialize all pheromone trails equally
- Local pheromone strategies ⇒ Initialize pheromone trails where the change occurs
- The offended pheromone trails from the cities replaced are re-initialized according to a metric either based on different heuristic information
- Requires the detection of change. Even more challenging to detect the change locally!
- More details: Guntsch and Middendorf’01 for DTSP

ACO with Memory Schemes

- Population-based ACO (P-ACO) maintains an actual population of ants
- Applied to the DTSP where cities are exchanged
- Pheromone trails are removed or added directly when an ant exists or enters the population-list
- Solutions stored are repaired heuristically when a change occurs
- Requires prior knowledge to repair solutions stored in memory
**Hybrid/Memetic ACO Algorithms**

- The memetic ACO (M-ACO) uses the P-ACO framework
- Before the best ant enters the population-list it is improved by a local search operator (inversion).
- Local search operator provides strong exploitation.
- A diversity scheme is applied (triggered immigrants) as follows:
  - If the population-list contains identical solutions, a random immigrant replaces one existing ant
- Inherits the disadvantages of P-ACO.

**ACO with Pheromone Strategies: Adapting Evaporation**

- Different evaporation rate perform better under different DOPs
- Solution ⇒ Adaptive pheromone evaporation rate
- Starts with an initial $\rho$ and modifies it as follows:
  - When stagnation behaviour is detected, the value is increased to help ants forget the current solution; otherwise, it is decreased to avoid randomization
  - $\lambda$-branching is used to detect stagnation behaviour
- Measures the distribution of pheromone trails
  - Example: if only a single path contains extreme pheromone whereas the remaining have lower pheromone ⇒ stagnation
- Performs better than fixed evaporation rate. However, a restart strategy performs better in severely changing environments
- More details: Mavrovouniotis and Yang (2013) for both DTSP and DVRP

**Experiments: M-ACO vs P-ACO and ACS**

- P-ACO performs better than the conventional ACO
- M-ACO achieves better performance than P-ACO

**Experiments: Adaptive vs Fixed (Optimized)**

- Adaptive often performs than fixed in some cases
- Sometimes is outperformed by the fixed evaporation
- Considering the tedious work to optimize evaporation; the adaptive mechanism is a good choice

More details: Mavrovouniotis and Yang (2012) for DVRP.
ACO with Pheromone Strategies: Immigrants

- Integrate immigrants schemes to ACO
- A short-term memory is used to store the best \( k \) ants and generated immigrant ants
- The memory is updated every iteration
  - No ant can survive in more than one iteration
- Pheromone trails are synchronized with short-term memory
- Any changes to the memory applied also to pheromone trails
- Pheromone evaporation is not used because pheromone trails are removed directly

Experiments: Immigrants Schemes – 1

- RIACO, EIACO and MIACO outperform conventional ACO algorithms
- EIACO performs better than, followed by MIACO
- M-ACO performs better than RIACO

Experiments: Immigrants Schemes – 2

- RIACO does not perform well on DOPs that change slightly
- Generate higher diversity than EIACO and MIACO
- Higher diversity does not always achieve better performance

Part III: Issues, future work, and summary

- Relevant issues
- Future work
- Summary and references
Theoretical Development

- So far, mainly empirical studies
- Theoretical analysis has just appeared
- Runtime analysis:
  - Stanhope & Daida (1999) first analyzed a (1+1) EA on the dynamic bit matching problem (DBMP)
  - Droste (2002) analyzed the first hitting time of a (1+1) ES on the DBMP
  - Rohlfshagen et al. (2010) analyzed how the magnitude and speed of change may affect the performance of the (1+1) EA on two functions constructed from the XOR DOP generator
- Analysis of dynamic fitness landscape:
  - Branke et al. (2005) analyzed the changes of fitness landscape due to changes of the underlying problem instance
  - Richter (2010) analyzed the properties of spatio-temporal fitness landscapes constructed from Coupled Map Lattices (CML)
  - Tinos and Yang (2010) analyzed the properties of the XOR DOP generator based on the dynamical system approach of the GA

Challenging Issues

- Detecting changes:
  - Most studies assume that changes are easy to detect or visible to an algorithm whenever occurred
  - In fact, changes are difficult to detect for many DOPs
- Understanding the characteristics of DOPs:
  - What characteristics make DOPs easy or difficult?
  - The work has started, but needs much more effort
- Analyzing the behavior of EAs for DOPs:
  - Requiring more theoretical analysis tools
  - Addressing more challenging DOPs and EC methods
  - Big question: Which EC methods for what DOPs?
- Real world applications:
  - How to model real-world DOPs?

Future Work

- The domain has attracted a growing interest recently
  - But, far from well-studied
- New approaches needed: esp. hybrid approaches
- Theoretical analysis: greatly needed
- EC for DOPs: deserves much more effort
- Real world applications: also greatly needed
  - Fields: logistics, transport, MANETs, data streams, social networks, ...

EC for Dynamic Multi-objective Optimization

- So far, mainly dynamic single-objective optimization
- Dynamic multi-objective optimization problems (DMOPs): even more challenging
- A few studies have addressed EC for DMOPs
  - Farina et al. (2004) classified DMOPs based on the changes on the Pareto optimal solutions
  - Goh & Tan (2009) proposed a competitive-cooperative coevolutionary algorithm for DMOPs
  - Zeng et al. (2006) proposed a dynamic orthogonal multi-objective EA (DOMOEA) to solve a DMOP with continuous decision variables
  - Zhang & Qian (2011) proposed an artificial immune system to solve constrained DMOPs
  - Jiang & Yang (2014) proposed a new benchmark MDOP generator
Summary

EC for DOPs: challenging but important

The domain is still young and active:

- More challenges to be taken regarding approaches, theory, and applications
- More young researchers are greatly welcome!

Shengxiang Yang (De Montfort University) Tutorial: EC for DOPs GECCO'15, 11/7/2015 77 / 84

Acknowledgements

- Two EPSRC funded projects on EC for DOPs
  - “EAs for DOPs: Design, Analysis and Applications”
    - Linked project among Brunel Univ. (Univ. of Leicester before 7/2010),
    - Univ. of Birmingham, BT, and Honda
    - Funding/Duration: over £600K / 3.5 years (1/2008–7/2011)
    - http://www.cs.le.ac.uk/projects/EADOP/
  - “EC for Dynamic Optimisation in Network Environments”
    - Linked project among DMU, Univ. of Birmingham, RSSB, and Network Rail
    - Funding/Duration: ~£1M / 4 years (2/2013–2/2017)
    - http://www.cci.dmu.ac.uk/research-grants/

- Research team members:
  - Research Fellows: Dr. Hui Cheng, Dr. Crina Grosan, Dr. Changhe Li,
  - Dr. Michalis Mavrovouniotis
  - PhD students: Changhe Li, Michalis Mavrovouniotis, Lili Liu, Hongfeng
  - Wang, Yang Yan

- Research cooperators:
  - Prof. Xin Yao, Prof. Juergen Branke, Dr. Renato Tinos, Dr. Hendrik Richter,
  - Dr. Trung Thanh Nguyen, etc.

Shengxiang Yang (De Montfort University) Tutorial: EC for DOPs GECCO'15, 11/7/2015 78 / 84

References – 1

7. S. Droste (2002). Analysis of the (1+1) EA for a dynamically changing onemax-variant. CEC'02, pp. 55–60

Shengxiang Yang (De Montfort University) Tutorial: EC for DOPs GECCO'15, 11/7/2015 79 / 84