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

Tutorial presented at the 2017 IEEE Congress on Evolutionary Computation
(IEEE CEC 2017), Donostia-San Sebastian, Spain, 5-8 June, 2017
Copyright is held by the author/owner(s)
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 230 publications and £2M funding for research

AE/Editorial Board Member for 8 journals, including IEEE Trans Cybern, Evol Comput, Inform Sci, Neurocomputing, and Soft Comput

Chair of two IEEE CIS Task Forces
- EC in Dynamic and Uncertain Environments
- Intelligent Network Systems
Centre for Computational Intelligence (CCI)

CCI (www.cci.dmu.ac.uk):
- Mission: Developing fundamental theoretical and practical solutions to real-world problems using a variety of CI paradigms
- Members: 18 staff, several research fellows, 30+ PhDs, visiting researchers
- Themes: EC, fuzzy logic, neural networks, data mining, robotics, game ...

Funding:
- Research Councils/Charities: EPSRC, ESRC, EU FP7 & Horizon 2020, Royal Academy of Engineering, Royal Society, Innovate UK, KTP, Innovation Fellowships, Nuffield Trust, etc.
- Government: Leicester City Council, DTI
- Industries: Lachesis, EMDA, RSSB, Network Rail, etc.

Collaborations:
- Universities: UK, USA, Spain, and China
- Industries and local governments

Teaching/Training:
- DTP-IS: University Doctor Training Programme in Intelligent Systems
- MSc Intelligent Systems, MSc Intelligent Systems & Robotics
- BSc Artificial Intelligence with Robotics

YouTube page: http://www.youtube.com/thecci
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, Issues and Future Work
- EC enhancement approaches for DOPs
- Case studies
- Relevant issues
- Future work
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

Black Box Solver

Problem to solve → Evolutionary Algorithm → A set of solutions
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 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
  - ......
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
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 a 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; Alba et al., 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 (2004–2017): part of Evo*
- EvoDOP (’99, ’01, ’03, ’05, ’07, ’09): part of GECCO
- IEEE Competitions: within IEEE CEC’09, CEC’12 & CEC’14
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 \( \geq 2 \) 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
The DF1 Generator

- Proposed by Morrison & De Jong (1999)
- The base landscape in the $D$-dimensional real space:

$$f(\vec{x}) = \max_{i=1, \ldots, p} \left[ H_i - R_i \times \sqrt{\sum_{j=1}^{D} (x_j - X_{ij})^2} \right]$$

- $\vec{x} = (x_1, \cdots, x_D)$: a point in the landscape; $p$: number of peaks
- $H_i, R_i, X_i = (X_{i1}, \cdots, X_{iD})$: height, slope, center of peak $i$
- The dynamics is controlled by a logistics function:

$$\Delta_t = A \cdot \Delta_{t-1} \cdot (1 - \Delta_{t-1})$$

- $A \in [1.0, 4.0]$: a constant; $\Delta_t$: step size of changing a parameter
Moving Peaks Benchmark (MPB) Problem

- Proposed by Branke (1999)
- The MPB problem in the $D$-dimensional space:

$$ F(\vec{x}, t) = \max_{i=1, \ldots, p} \frac{H_i(t)}{1 + W_i(t) \sum_{j=1}^{D} (x_j(t) - X_{ij}(t))^2} $$

- $W_i(t)$, $H_i(t)$, $X_i(t) = \{X_{i1} \cdots X_{iD}\}$: height, width, location of peak $i$ at $t$
- The dynamics:

$$ H_i(t) = H_i(t-1) + \text{height\_severity} \times \sigma $$
$$ W_i(t) = W_i(t-1) + \text{width\_severity} \times \sigma $$

$$ \vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|} \left( (1 - \lambda)\vec{r} + \lambda \vec{v}_i(t-1) \right) $$

$$ \vec{X}_i(t) = \vec{X}_i(t-1) + \vec{v}_i(t) $$

- $\sigma \sim N(0, 1)$; $\lambda$: 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 $s$
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:

\[
\text{Max } f(\vec{x}(t), t) = \sum_{i=1}^{n} p_i(t) \cdot x_i(t), \quad \text{s. t. } \sum_{i=1}^{n} w_i(t) \cdot x_i(t) \leq C(t)
\]

- \( \vec{x}(t) \in \{0, 1\}^n \): a solution at time \( t \)
- \( x_i(t) \in \{0, 1\} \): indicates whether item \( i \) is included or not
- \( p_i(t) \) 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:

1. Create a template $T_k$ with $\rho \ast l$ ones
2. Create a mask $\vec{M}(k)$ incrementally
   
   $\vec{M}(0) = \vec{0}$ (the initial state)
   
   $\vec{M}(k + 1) = \vec{M}(k) \oplus \vec{T}(k)$
3. 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
Can extend the XOR DOP generator to create cyclic environments:

1. Construct $K$ templates $\tilde{T}(0), \cdots, \tilde{T}(K-1)$
   - Form a partition of the search space
   - Each contains $\rho \times l = l/K$ ones

2. Create $2K$ masks $\tilde{M}(i)$ as base states
   
   \[
   \tilde{M}(0) = \tilde{0} \quad \text{(the initial state)}
   \]
   
   \[
   \tilde{M}(i + 1) = \tilde{M}(i) \oplus \tilde{T}(i \% K), \quad i = 0, \cdots, 2K-1
   \]

3. Cycle among $\tilde{M}(i)$’s every $\tau$ generations
   
   \[
   f(\vec{x}, t) = f(\vec{x} \oplus \tilde{M}(I_t)) = f(\vec{x} \oplus \tilde{M}(k \% (2K)))
   \]
   
   \[-\quad k = \lfloor t/\tau \rfloor: \text{environmental index}\]
   
   \[-\quad I_t = k \% (2K): \text{mask index}\]
We can also construct cyclic environments with noise:

- Each time before a base state is entered, it is bitwise changed with a small probability.
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
    
    \[ D(t) = \{d_{ij}(t)\}_{n \times n} \]
    
    – \( d_{ij}(t) \): 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 \( f(x, t) \):
    
    \[
    f(x, t) = \text{Min}(\sum_{i=1}^{n} d_{x_i, x_{i+1}}(t))
    \]
    
    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:

1. Generate a random vector $\vec{r}(T)$ that contains all objects every $f$ iterations
2. Generate another randomly re-order vector $\vec{r}'(T)$ that contains only the first $m \times n$ objects of $\vec{r}(T)$
3. Modify the encoding of the problem instance with $m \times n$ pairwise swaps

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.

Can extend for cyclic and cyclic with noise environments.
Generalized DOP Benchmark Generator (GDBG)

- Proposed by Li & Yang (2008), GDBG uses the model below:

\[ F = f(x, \phi, t), \]

- \( \phi \): 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:

\[ f(x, \phi, t + 1) = f(x, \phi(t) \oplus \Delta \phi, t) \]
GDBG: Dynamic Change Types

- **Change types:**
  1. Small step: $\Delta \phi = \alpha \cdot \| \phi \| \cdot \text{rand}()$
  2. Large step: $\Delta \phi = \| \phi \| \cdot (\alpha + (1 - \alpha)\text{rand}())$
  3. Random: $\Delta \phi = \| \phi \| \cdot \text{rand}()$
  4. Chaotic: $\phi(t + 1) = A \cdot \phi(t) \cdot (1 - \phi(t)/\| \phi \|)$
  5. Recurrent: $\phi(t + 1) = \phi(t \% P)$
  6. Recurrent with nosy: $\phi(t + 1) = \phi(t \% P) + \alpha \cdot \| \phi \| \cdot \text{rand}()$
  7. ...... 

- **More details:**
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

Shengxiang Yang (De Montfort University)
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):

\[
\overline{F}_{BOG} = \frac{1}{G} \times \sum_{i=1}^{G} \left( \frac{1}{N} \times \sum_{j=1}^{N} F_{BOG_{ij}} \right)
\]

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

- Adaptation performance (Mori et al., 1997)

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

- Accuracy (Trojanowski and Michalewicz, 1999)

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

  - \(f_{best}(i)\): best fitness for environment \(i\) (best before change)
Part II: Approaches, Issues and Future Work

- EC enhancement approaches for DOPs
- Case studies
- Relevant issues
- Future work
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:
    1. It may be inefficient, wasting computational resources
    2. 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
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
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
Each individual has a pair of chromosomes
- Dominance scheme maps genotype to phenotype
- Dominance scheme may change or be adaptive (Uyar & Harmanci, 2005)
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
Explicit Memory: Direct vs Associative

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

![Diagram of Direct Memory and Associative Memory](image-url)
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:**
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.
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 $B_M(t)$ as the base
  - Generate immigrants by mutating $B_M(t)$ with a prob.
  - Replace worst members in the population by these immigrants

![Search Space Diagram](image-url)
Experimental Results: Immigrants Based GAs

- **Cyclic Dynamic OneMax Function,** $\tau = 25, \rho = 0.1$
- **Random Dynamic OneMax Function,** $\tau = 25, \rho = 0.1$

- Memory-based immigrants GA (MIGA) significantly beats other GAs
- More details:
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
- More details:
Experimental Results: Hybrid Immigrants GA

Hybrid immigrants improve GA’s performance for DOPs efficiently
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
Self-organizing scouts (SOS) GA (Branke et al., 2000)
- The parent population explores the search space
- A child population is split under certain conditions
- Child populations search limited promising areas
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
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:
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
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:
Particle Swarm Optimization (PSO):
- Inspired by models of swarming and flocking
- First introduced by Kennedy and Eberhart in 1995
- PSO has been applied for many static optimization problems

Recently, PSO has been applied for continuous DOPs

Two aspects to consider for DOPs:
- Outdated memory. Two solutions:
  - Simply set $p_{best}$ to the current position
  - Reevaluate $p_{best}$ and reset it to the current position if current position is better
- Diversity loss. Three solutions:
  - Introduce diversity after a change
  - Maintain diversity during the run
  - Use multi-swarms
Multi-swarm PSO for DOPs

- **Clustering PSO (CPSO):**
  - Training: Move particles toward different promising regions
  - Clustering: Single linkage hierarchical clustering to create sub-swarms
  - Local search: Each sub-swarm will search among one peak quickly
  - Overlapping and convergence check
  - Strategies to response to changes
  - Details: Li & Yang, CEC’09; Yang & Li, IEEE Trans Evol Comput, 14(6), 2010

- **Adaptive Multi-Swarm Optimizer (AMSO):**
  - Use single linkage hierarchical clustering to create populations
  - An overcrowding scheme to remove unnecessary populations
  - A special rule to decide proper moments to increase diversity without change detection
  - An adaptive method to create a proper number of populations needed
  - Details:
Demo: CPSO & AMSO for DOPs
ACO mimics the behaviour of ants searching for food

ACO was first proposed for TSPs (Dorigo et al., 1996)

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
ACO transfers knowledge via pheromone
- Make sense on slight changes; otherwise, may misguide the search
- For severe changes, a global restart is a better choice
- 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 (Mavrovouniotis, Li, & Yang 2017)
- Pheromone modification after a change (Guntsch & Middendorf, 2001, Eyckelhof & Snoek, 2002)
- Memory-based schemes (Guntsch & Middendorf, 2002)
- Hybrid and memetic algorithms (Mavrovouniotis, Muller & Yang, 2017)
- Pheromone modification during execution (Mavrovouniotis & Yang, 2013a)
- Multi-colony schemes (Mavrovouniotis, Yang & Yao, 2014)
ACO with Pheromone Strategies: Adapting Evaporation

- Pheromone evaporation is an adaptation mechanism in ACO
- Different evaporation rates perform better for different DOPs
- Solution ⇒ Adaptive pheromone evaporation rate
  - Starts with an initial $\rho$ and modifies it as follows:
    - When stagnation behaviour is detected, increase $\rho$ to help ants forget current solution; otherwise, decrease $\rho$ to avoid randomization
    - A $\lambda$-branching method is used to detect stagnation behaviour
- Performs better than fixed evaporation rate
  - However, a restart strategy performs better for severe changes
- More details:
  - Mavrovouniotis & Yang (2013a) for both DTSP and DVRP
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

More details:
  - Mavrovouniotis & Yang (2013b) for DTSPs
  - Mavrovouniotis & Yang (2015) for DVRPs
  - Eaton, Mavrovouniotis & Yang (2016) for the dynamic railway junction rescheduling problem
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, 2014) analyzed the properties of the XOR DOP generator based on the dynamical system approach of a GA
EC for Dynamic Multi-objective Optimization

- So far, mainly dynamic single-objective optimization
- Dynamic multi-objective optimization problems (DMOPs): even more challenging
- Recently, rising interest in studying 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 (2017a) proposed a new benchmark MDOP generator
  - Jiang & Yang (2017b) proposed a Steady-Generational EA (SGEA) for DMOPs
  - Ruan et al. (2017) analyzed the effect of diversity maintenance on prediction for DMOPs
  - Eaton et al. (2017) applied ACO for the dynamic multi-objective railway junction rescheduling problem
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

Analysing the behaviour 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 DMOPs: deserves much more effort
- Real world applications: also greatly needed
  - Fields: logistics, transport, MANETs, data streams, social networks, ...

Shengxiang Yang (De Montfort University)
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!
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://gtr.rcuk.ac.uk/project/B807434B-E9CA-41C7-B3AF-567C38589BAC
- “EC for Dynamic Optimisation in Network Environments”
  - Linked project among DMU, Univ. of Birmingham, RSSB, and Network Rail
  - Funding/Duration: ~£1M / 4.5 years (2/2013–8/2017)
  - http://gtr.rcuk.ac.uk/project/C43F34D3-16F1-430B-9E1F-483BBADCD8FA

Research team members:
- Research Fellows: Dr. Hui Cheng, Dr. Crina Grosan, Dr. Changhe Li, Dr. Michalis Mavrovouniotis, Dr. Yong Wang, etc.
- PhD students: Changhe Li, Michalis Mavrovouniotis, Shouyong Jiang, Jayne Eaton, Hongfeng Wang, etc.

Research cooperators:
- Prof. Xin Yao, Prof. Juergen Branke, Prof. Jinliang Ding, Dr. Renato Tinos, Dr. Hendrik Richter, Dr. Trung Thanh Nguyen, etc.
IEEE CIS Task Force on EC in Dynamic and Uncertain Environments
- http://www.tech.dmu.ac.uk/~syang/IEEE_ECIDUE.html
- Maintained by Shengxiang Yang

Source codes:
- http://www.tech.dmu.ac.uk/~syang/publications.html

IEEE Competitions:
- 2009 Competition on EC in Dynamic & Uncertain Environments:
  http://www.cs.le.ac.uk/people/syang/ECiDUE/ECiDUE-Competition09
- 2012 Competition on EC for DOPs:
  http://people.brunel.ac.uk/~csstssy/ECDOP-Competition12.html
References – 1


M. Mavrovouniotis, S. Yang, X. Yao (2014). Multi-colony ant algorithms for the dynamic travelling salesman problem, IEEE SSCI 2014, pp. 9–16


