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

http://www.sigevo.org/gecco-2015/

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 author/owner(s).

GECCO'15 Companion, July 11–15, 2015, Madrid, Spain.
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
- AE/Editorial Board Member for 7 journals, including *IEEE Trans. Cybern.*, *Evol. Comput.*, *Inform. Sci.*, 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: 16 staff, 4 research fellows, 30+ PhDs, visiting researchers
  - Three labs: Robotics, Eye Gaze, Game Programming
- 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

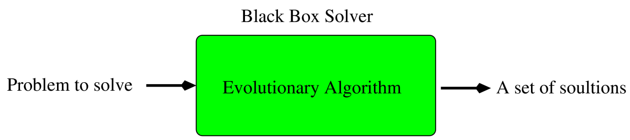
Part I: Fundamentals
- Introduction to evolutionary computation (EC)
- EC for dynamic optimization problems (DOPs): Concept and motivation
- Benchmark and test problems
- Performance measures

Part II: Approaches 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



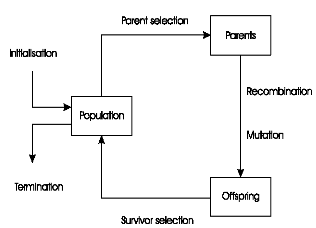
Design and Framework of an EA

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

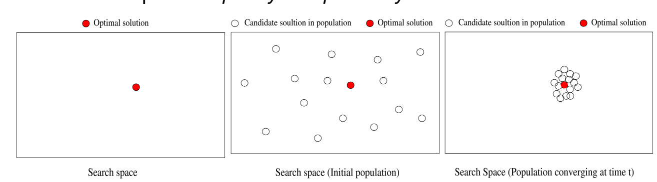


EC Applications

- 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
  - ......

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*

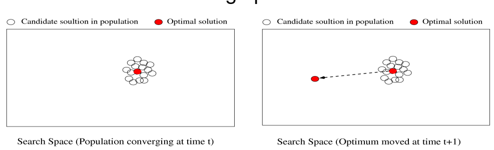
$$F = f(\vec{x}, \vec{\phi}, t)$$

– $\vec{x}$: 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:
  - Mavrovouniotis, 2013; Helbig, 2012; du Plessis, 2012; Li, 2011; Nguyen, 2011; Simoes, 2010
- 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*
  - ECiDUE (2004–2015): part of IEEE CEC
  - EvoDOP (’99, ’01, ’03, ’05, ’07, ’09): part of GECCO
- IEEE Symposium on CIDUE (2011, 2013, 2014, 2015)
- IEEE Competitions: within IEEE CEC 2009 & CEC 2012

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

The DF1 Generator

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

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

– $\vec{x} = (x_1, \cdots, x_D)$: 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; Δ_t: step size of changing a parameter

Moving Peaks Benchmark (MPB) Problem

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

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

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

- The dynamics:

$$H_i(t) = H_i(t-1) + height\_severity * \sigma$$
$$W_i(t) = W_i(t-1) + width\_severity * \sigma$$
$$\vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|}((1-\lambda)\vec{r} + \lambda \vec{v}_i(t-1))$$
$$\vec{X}_i(t) = \vec{X}_i(t)(t-1) + \vec{v}_i(t)$$

– $\sigma \sim N(0,1)$; λ: 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)

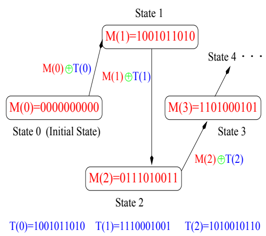
- Static knapsack problem:
  - Given n items, each with a weight and a profit, and a knapsack with a fixed capacity, select items to fill up the knapsack to maximize the profit while satisfying the knapsack capacity constraint
- The DKP:
  - Constructed by changing weights and profits of items, and/or knapsack capacity over time as:

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

– $\vec{x}(t) \in \{0,1\}^n$: 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 "$\oplus$" (Yang, 2003; Yang & Yao, 2005)
- Suppose the environment changes every τ generations
- For each environmental period $k = \lfloor t/\tau \rfloor$, do:



1. Create a template T_k with $\rho * l$ ones
2. Create a mask $\vec{M}(k)$ incrementally

$$\vec{M}(0) = \vec{0} \text{ (the initial state)}$$
$$\vec{M}(k+1) = \vec{M}(k) \oplus \vec{T}(k)$$

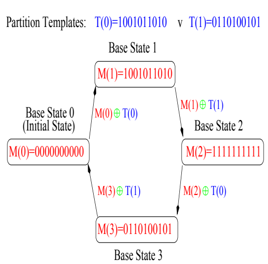
3. Evaluate an individual:

$$f(\vec{x}, t) = f(\vec{x} \oplus \vec{M}(k))$$

- τ and ρ controls the speed and severity of change respectively

Constructing Cyclic Dynamic Environments

Can extend the XOR DOP generator to create cyclic environments:



1. Construct K templates $\vec{T}(0), \cdots, \vec{T}(K-1)$
   - Form a partition of the search space
   - Each contains $\rho \times l = l/K$ ones
2. Create $2K$ masks $\vec{M}(i)$ as *base states*

$$\vec{M}(0) = \vec{0} \text{ (the initial state)}$$
$$\vec{M}(i+1) = \vec{M}(i) \oplus \vec{T}(i\%K), i = 0, \cdots, 2K-1$$

3. Cycle among $\vec{M}(i)$'s every τ generations

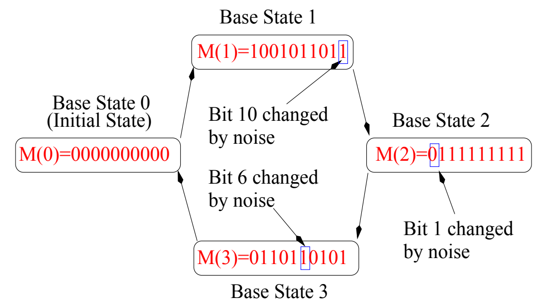
$$f(\vec{x}, t) = f(\vec{x} \oplus \vec{M}(I_t)) = f(\vec{x} \oplus \vec{M}(k\%(2K)))$$

– $k = \lfloor t/\tau \rfloor$: environmental index
– $I_t = k\%(2K)$: mask index

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



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

$$D(t) = \{d_{ij}(t)\}_{n*n}$$

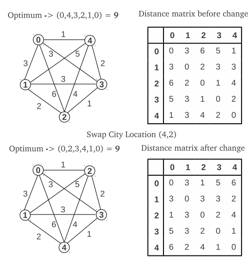
– $d_{ij}(t)$: cost from city i to j; n: the number of cities
  - The aim is to find a minimum-cost route containing all cities at time t
  - DTSP can be defined as $f(x, t)$:

$$f(x, t) = Min(\sum_{i=1}^{n} d_{x_i, x_{i+1}}(t))$$

where $x_i \in 1, \cdots, n$. 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:

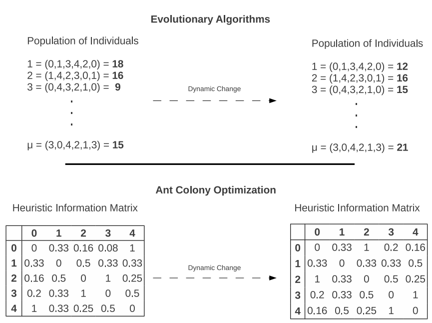


1. Generate a random vector $\vec{r}(T)$ that contains all objects every f iterations
2. Generate another randomly re-order vector $\vec{r'}(T)$ that contains only the first $m \times n$ objects of $\vec{r}(T)$
3. Modify the encoding of the problem instance with $m \times n$ pairwise swaps

- More details: M. Mavrovouniotis, S. Yang, & X. Yao (2012). *PPSN XII*, Part II, LNCS 7492, pp. 508–517

Effect on Algorithms

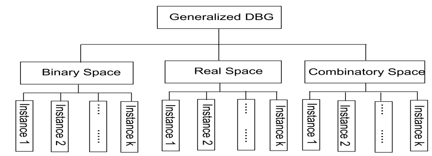
- Similar with the XOR DOP generator, DBGP shifts the population of an alg. to new location in the fitness landscape
- The individual with the same encoding as before a change will have a different cost after the change



- Can extend for cyclic and cyclic with noise environments

Generalized DOP Benchmark Generator (GDBG)

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



- In GDBG, DOPs are defined as:

$$F = f(x, \phi, t),$$

– ϕ: system control parameter

- Dynamism results from tuning ϕ 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: $\Delta\phi = \alpha \cdot \|\phi\| \cdot rand()$
  2. Large step: $\Delta\phi = \|\phi\| \cdot (\alpha + (1-\alpha)rand())$
  3. Random: $\Delta\phi = \|\phi\| \cdot rand()$
  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: $\phi(t+1) = \phi(t\%P) + \alpha \cdot \|\phi\| \cdot rand()$
  7. ......
- More details:
  - C. Li & S. Yang (2008). *SEAL'08*, LNCS 5361, pp. 391–400

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):

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

– 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)

$$Ada = \frac{1}{T} \sum_{t=1..T} (f_{best}(t)/f_{opt}(t))$$

- Accuracy (Trojanowski and Michalewicz, 1999)

$$Acc = \frac{1}{K} \sum_{i=1..K} (f_{best}(i) - f_{opt}(i))$$

– $f_{best}(i)$: 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

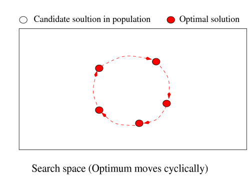
- Recap: traditional EAs are not good for DOPs
- Goal: to track the changing optimum
- How about restarting an EA after a change?
  - Natural and easy choice
  - But, not good choice because:
    1. It may be inefficient, wasting computational resources
    2. It may lead to very different solutions 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

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

Memory Approaches

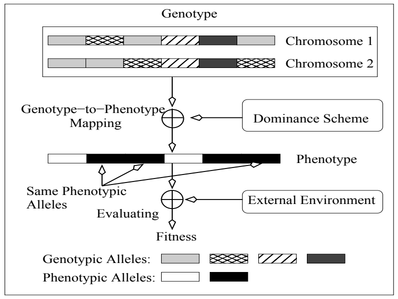
- Cyclic DOPs: change cyclically among a fixed set of states



- 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

Implicit Memory: Diploid Genetic Algorithm

Encoding



Dominance Scheme

	0	o	1	i
0	0	0	0/1	0
o	0	0	1	0/1
1	0/1	1	1	1
i	0	0/1	1	1

Ng & Wong (1995)

	A	B	C	D
A	0	0	0	1
B	0	0	0	1
C	0	0	1	1
D	1	1	1	1

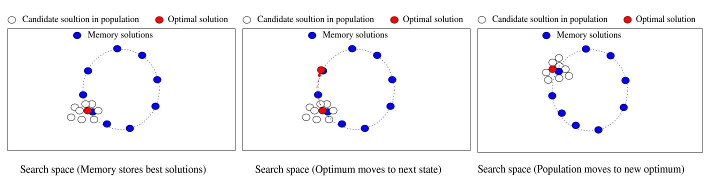
Lewis et al. (1998)

- 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

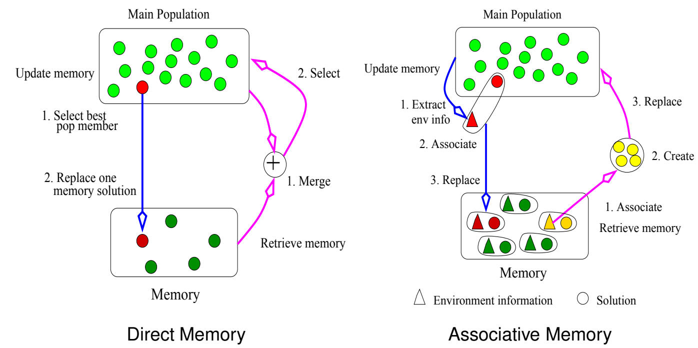
Basic idea: use extra memory

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



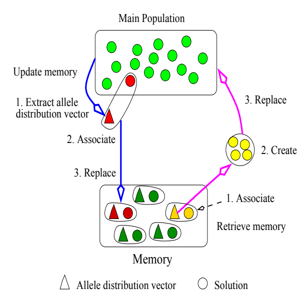
Explicit Memory: Direct vs Associative

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



Associative Memory Based Genetic Algorithm

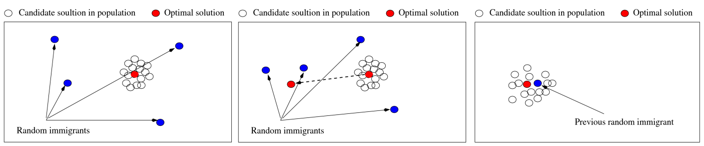
Idea: Use *allele distribution* (AD) $\vec{D}$ to represent environmental info.



- 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:
  - S. Yang (2006). *EvoWorkshops'06*, pp. 788–799

Diversity Approaches: Random Immigrants

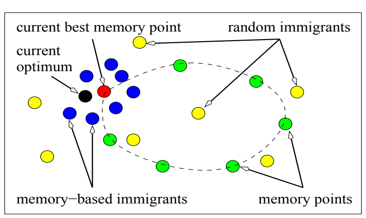
- Convergence is the key problem in metaheuristics for DOPs
- Random immigrants:
  - Each generation, insert some random individuals (called *random immigrants*) into the population to maintain diversity
  - When optimum moves, random immigrants nearby take action to draw the pop to the new optimum



Search Space (Population converging at time t) Search Space (Optimum moved at time t+1) Search Space (Population moves to new optimum)

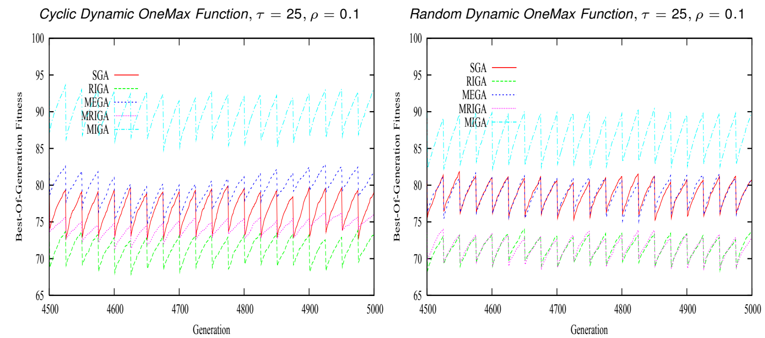
Memory-Based Immigrants

- 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

Experimental Results: Immigrants Based GAs



- Memory-based immigrants GA (MIGA) significantly beats other GAs
- More details:
  - S. Yang (2008). *Evol. Comput.*, 16(3): 385–416

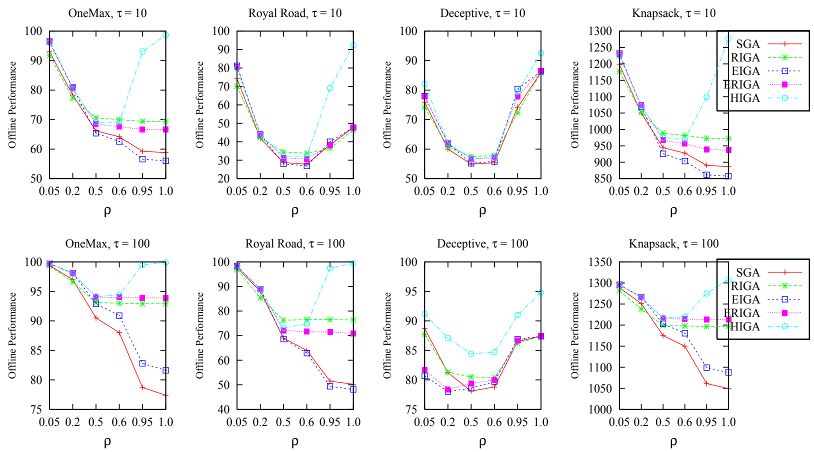
Hybrid Immigrants Approach

- 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

$$\vec{x}^d = dual(\vec{x}) = (x_1^d, \cdots, x_l^d) \in \{0,1\}^l$$

  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:
  - S. Yang & R. Tinos (2007). *Int. J. of Autom. & Comp.*, 4(3): 243–254

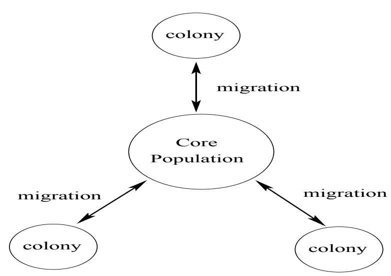
Experimental Results: Hybrid Immigrants GA



- Hybrid immigrants improve GA's performance for DOPs efficiently

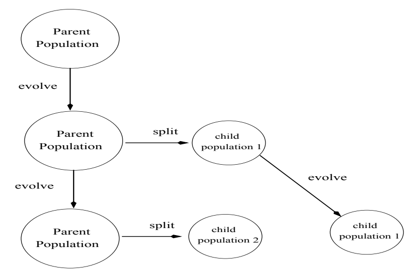
Multi-Populations: Shifting Balance

- 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



Multi-Populations: Self-Organizing Scouts

- 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

Prediction Approaches

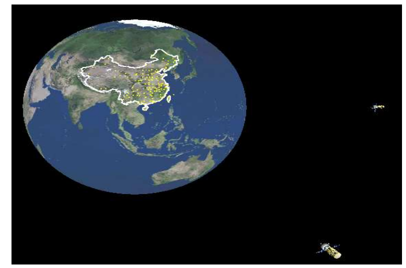
- For some DOPs, changes exhibit predictable patterns
- Techniques (forecasting, Kalman filter, etc.) can be used to predict
  - The location of the next optimum after a change
  - When the next change will occur and which environment may appear
- Some relevant work: see Simões & Costa (2009)

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 saterllite, 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:
  - C. Li, M. Yang, & L. Kang (2006). *SEAL'06*, LNCS 4247, pp. 236–243



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:
  - S. Yang, H. Cheng, & F. Wang (2010). IEEE Trans SMC Part C: Appl. & Rev., 40(1): 52–63

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

$$v'^{d}_{i} = \omega v^{d}_{i} + c_1 \cdot r_1 \cdot (pbest^{d}_{i} - x^{d}_{i}) + c_2 \cdot r_2 \cdot (gbest^{d} - x^{d}_{i})$$

$$x'^{d}_{i} = x^{d}_{i} + v'^{d}_{i}$$

  - x'^{d}_{i} and x^{d}_{i}: 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
  - $pbest_i$ and $gbest$: 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 *pbest* to the current position
    - Reevaluate *pbest* 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 *nbest* 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
  - Overlapping and convergence check
  - Strategies to response to changes
- More details:
  - Li & Yang, CEC 2009: 439-446
  - Yang & Li, IEEE Trans Evol Comput, 14(6): 959-974, 2010

Experiments on MPB Functions

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

s	CPSO	mCPSO	mQSO
0.0	0.89	1.18	1.18
1.0	1.49	2.05	1.75
2.0	1.63	2.80	2.40
3.0	1.96	3.57	3.00
4.0	2.05	4.18	3.59
5.0	2.24	4.89	4.24
6.0	2.29	5.53	4.79

Experiments on GDBG System

TABLE XVIII
OVERALL PERFORMANCE OF CPSO, SGA, AND SPSO ON ALL THE TEST CASES

		$F1(p=10)$	$F_1(p=50)$	F_2	F_3	F_4	F_5	F_6
CPSO	T_1	0.966001	0.966917	0.794534	0.592024	0.76771	0.731467	0.649896
	T_2	0.922947	0.907395	0.635615	0.0440386	0.560921	0.715637	0.508471
	T_3	0.875388	0.856738	0.661482	0.0982846	0.575383	0.718818	0.531786
	T_4	0.987497	0.987119	0.913543	0.367912	0.888087	0.878806	0.663524
	T_5	0.922095	0.940859	0.648544	0.0741003	0.549243	0.690628	0.550686
	T_6	0.930797	0.940859	0.726883	0.174772	0.683067	0.69816	0.521062
	T_7	0.886248	0.86016	0.752913	0.200772	0.673421	0.711989	0.596812
	Mark	0.0929334	0.0925999	0.117181	0.0356395	0.107361	0.117796	0.0917592
SGA	T_1	0.910613	0.902603	0.442572	0.250439	0.383658	0.388381	0.359464
	T_2	0.84281	0.843315	0.257916	0.0255796	0.14769	0.340976	0.240533
	T_3	0.796255	0.771041	0.310793	0.060691	0.215296	0.369612	0.314915
	T_4	0.859011	0.865804	0.352983	0.116428	0.301695	0.310434	0.256899
	T_5	0.863522	0.90018	0.306424	0.0683818	0.193047	0.460397	0.374333
	T_6	0.804106	0.795939	0.3369	0.0800156	0.286556	0.30469	0.255455
	T_7	0.808816	0.802818	0.394271	0.116652	0.299491	0.37657	0.255455
	Mark	0.0842329	0.0842114	0.0544904	0.0163033	0.0414625	0.0582129	0.0473257
SPSO	T_1	0.888962	0.899623	0.491121	0.0288704	0.479559	0.562548	0.320394
	T_2	0.837838	0.842265	0.345149	0.0142884	0.279373	0.651761	0.374696
	T_3	0.826183	0.797807	0.430344	0.0172752	0.319256	0.655	0.392653
	T_4	0.826493	0.887341	0.402016	0.0170278	0.379976	0.476617	0.254912
	T_5	0.89435	0.924999	0.404052	0.0338731	0.329061	0.770762	0.490316
	T_6	0.748523	0.769592	0.391393	0.0136195	0.35754	0.45786	0.260332
	T_7	0.753283	0.760335	0.465614	0.0762798	0.468118	0.574794	0.38434
	Mark	0.0828681	0.0844278	0.0665876	0.00421938	0.0589642	0.0949859	0.0563887

Performance (sum the marks for all test cases and multiply by 100): CPSO 65.527, SGA 38.0564, SPSO 44.8442

Summary of CPSO for DOPs

- The nearest neighbour training strategy can efficiently guide randomly initialized particles to different promising sub-regions
- CPSO scales well regarding 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)

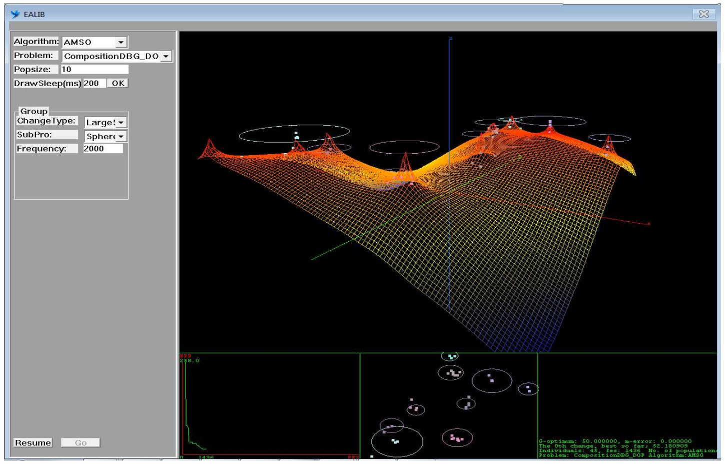
- Single linkage hierarchical clustering is used to create populations
  - All populations use the same search operator for local search
- An overcrowding scheme is used to remove unnecessary populations
- To find out proper moments to increase diversity without change detection, a special rule is designed 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

Experimental Results

Table : The offline error ($E_{offline}$) and best-before-change error (E_{BBC}) on the MPB with changing number of peaks

	Error	AMSO	CPSOR	CPSO	rSPSO	SPSO	mCPSO	mQSO	SAMO	DynDE	DynPopDE	ESCA	HmSO	SOS
Var1	$E_{offline}$	**2.3** ±0.25	2.8^w ±0.17	4^w ±0.28	6^w ±0.55	5.9^w ±0.58	7.8^w ±0.62	4.5^w ±0.27	3.5^w ±0.24	3.6^w ±0.38	3.7^w ±0.26	13^w ±1.3	4.5^w ±0.19	11^w ±3.2
	E_{BBC}	**1.5**	1.6^w	1.9^w	5.2^w	5.1^w	7^w	3.7^w	2.8^w	3^w	3.1^w	12^w	3.5^w	9.7^w
Var2	$E_{offline}$	**2.9** ±0.74	3.3^w ±0.63	5^w ±1	4.6^w ±0.65	4.9^w ±0.67	7.3^w ±0.89	4.4^w ±0.92	4^w ±0.68	3.5^w ±0.81	4.2^w ±0.75	13^w ±1.2	5.4^w ±0.69	9.4^w ±4.3
	E_{BBC}	2	$\mathbf{1.9}^t$	2.6^w	3.7^w	3.9^w	6.3^w	3.4^w	3.1^w	2.9^w	3.6^w	13^w	4^w	8.4^w
Var3	$E_{offline}$	**2.7** ±0.45	2.9^w ±0.29	4.5^w ±0.36	4.9^w ±0.68	4.8^w ±0.66	7.4^w ±0.98	4.1^w ±0.55	3.7^w ±0.32	3.4^w ±0.5	4.8^w ±0.59	13^w ±2	5.3^w ±0.4	9.6^w ±3.5
	E_{BBC}	1.7	$\mathbf{1.6}^t$	2.2^w	3.9^w	3.7^w	6.4^w	3.3^w	2.8^w	2.7^w	4.1^w	12^w	4.1^w	8.5^w

Demo: CPSO & AMSO for DOPs



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

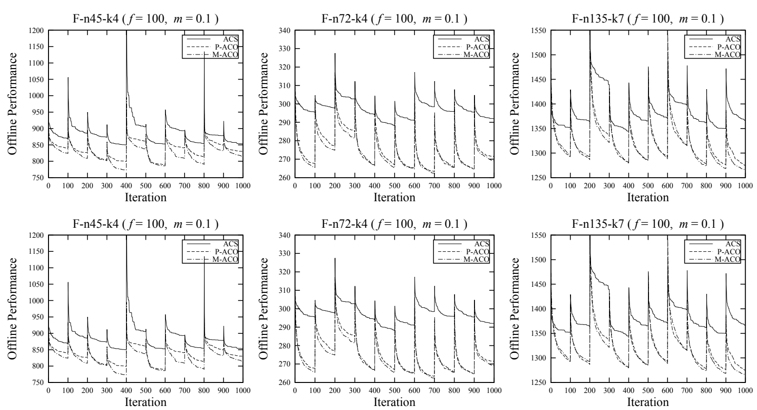
- Population-based ACO (P-ACO) maintains an actual population of ants
- Applied to the DTSP where cities are exchanged
- Pheromone trails are removed or added directly when an ant exists or enters the population-list
- Solutions stored are repaired heuristically when a change occurs
- Requires prior knowledge to repair solutions stored in memory
- More details: Guntsch and Middendorf (2002) for DTSP and Mavrovouniotis and Yang (2012) for DVRP

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.
- More details: Mavrovouniotis and Yang (2010) for DTSP and Mavrovouniotis and Yang (2012) for 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

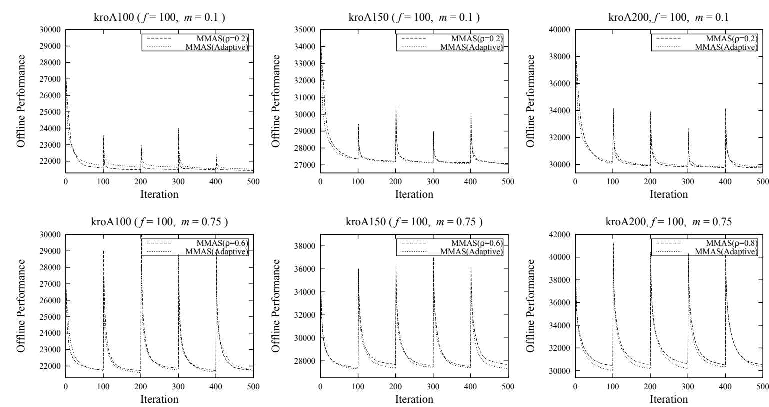


ACO with Pheromone Strategies: Adapting Evaporation

- Different evaporation rate perform better under different DOPs
- Solution ⇒ Adaptive pheromone evaporation rate
- Starts with an initial ρ 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
- λ-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: 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

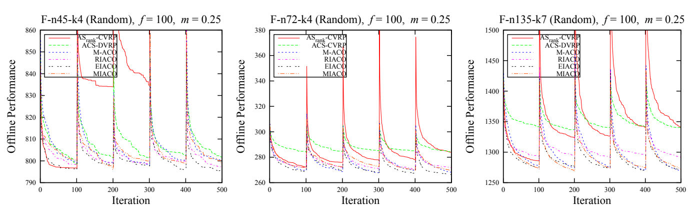


ACO with Pheromone Strategies: Immigrants

- 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
- More details: Mavrovouniotis and Yang (2010, 2013) for DTSP and Mavrovouniotis and Yang (2012a, 2012b) for DVRP

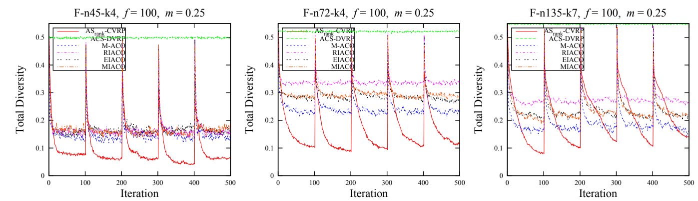
Experiments: Immigrants Schemes – 1

- 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

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

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, ...

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!

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.

Relevant Information

- 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
  - 2014 Competition on EC for DOPs: http://cs.cug.edu.cn/teacherweb/lichanghe/pages/organization/competition/ECDOP-Competition14.html

References – 1

1. J. Branke (1999). Memory enhanced evolutionary algorithms for changing optimization problems. *CEC'99*, pp. 1875–1882
2. J. Branke (2002). *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers
3. J. Branke, E. Salihoglu, S. Uyar (2005). Towards an analysis of dynamic environments. *GECCO'05*, pp. 1433–1439
4. H.G. Cobb, J.J. Grefenstette (1993). Genetic algorithms for tracking changing environments. *Proc. ICGA*, pp. 523–530
5. C. Cruz, J. Gonzanlez, D. Pelta (2011). Optimization in dynamic environments: A survey on problems, methods and measures. *Soft Comput.*, 15: 1427–1448
6. M. Dorigo, V. Maniezzo, A. Colorni (1996). Ant system: Optimization by a colony of cooperating agents, IEEE Trans SMC-Part B: Cybern 26(1): 29–41
7. S. Droste (2002). Analysis of the (1+1) EA for a dynamically changing onemax-variant. *CEC'02*, pp. 55–60
8. C. Eyckelhof, M. Snoek (2002). Ant systems for a dynamic TSP, *Proc 3rd Int Workshop on Ant Algorithms*, LNCS 2463, pp. 88–99
9. M. Farina, K. Deb, P. Amato (2004). Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Trans. Evol. Comput.*, 8(5): 425–442
10. C. Goh, K.C. Tan (2009). A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Trans. Evol. Comput.*, 13(1): 103–127
11. M. Guntsch, M. Middendorf (2001). Pheromone modification strategies for ant algorithms applied to dynamic TSP, *EvoWorkshops 2001*, LNCS 2037, pp. 213–222
12. M. Guntsch, M. Middendorf (2002). Applying population based aco to dynamic optimization problems, *Proc. 3rd Int. Workshop on Ant Algorithms*, LNCS 2463, pp. 111–122

References – 2

13. M. Guntsch, M. Middendorf, H. Schmeck (2001). An ant colony optimization approach to dynamic TSP, *GECCO 2001*, pp. 860–867
14. S. Jiang, S. Yang (2014a). A benchmark generator for dynamic multi-objective optimization problems. *UKCI 2014*, pp. 1–8
15. S. Jiang, S. Yang (2014b). A framework of scalable dynamic test problems for dynamic multi-objective optimization. *IEEE Symp. CIDUE 2014*, pp. 32–39
16. Y. Jin, J. Branke (2005). Evolutionary optimization in uncertain environments–A survey. *IEEE Trans. Evol. Comput.*, 9(3): 303–317
17. R.W. Morrison (2004). *Designing Evolutionary Algorithms for Dynamic Environments.* Springer
18. E.H.J. Lewis, G. Ritchie (1998). A comparison of dominance mechanisms and simple mutation on non-stationary problems. *PPSN V*, pp. 139–148
19. C. Li, S. Yang, M. Yang (2014). An adaptive multi-swarm optimizer for dynamic optimization problems. Evol Comput, 22(4): 559–594
20. M. Mavrovouniotis, S. Yang (2010). Ant colony optimization with immigrants schemes for dynamic environments, *PPSN XI*, LNCS 6239, pp. 371–380
21. M. Mavrovouniotis, S. Yang (2012). Ant colony optimization with immigrants schemes for the dynamic vehicle routing problem, EvoApplications 2012, LNCS 7248, pp. 519–528
22. M. Mavrovouniotis, S. Yang (2012). Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem, CEC 2012, pp. 2645–2652
23. M. Mavrovouniotis, S. Yang (2013). Adapting the pheromone evaporation rate in dynamic routing problems, EvoApplications 2013, LNCS 7835, pp. 606–615

References – 3

24. M. Mavrovouniotis, S. Yang (2013). Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors, Appl. Soft Comput. 13(10): 4023–4037
25. M. Mavrovouniotis, S. Yang (2013). Dynamic vehicle routing: A memetic ant colony optimization approach, in: A. Uyar, E. Ozcan, N. Urquhart (eds.), Automated Scheduling and Planning, Chap. 9, Springer, pp. 283–301
26. M. Mavrovouniotis, S. Yang, X. Yao (2012). A benchmark generator for dynamic permutation-encoded problems, PPSN XII, LNCS 7492, pp. 508–517
27. M. Mavrovouniotis, S. Yang, X. Yao (2014). Multi-colony ant algorithms for the dynamic travelling salesman problem, submitted to *IEEE SSCI 2014*
28. R. Montemanni, L. M. Gambardella, A. E. Rizzoli, A. V. Donati (2005). Ant colony system for a dynamic vehicle routing problem, Combinat. Optim. 10: 327–343
29. R.W. Morrison, K.A. De Jong (1999). A test problem generator for non-stationary environments. *CEC'99*, pp. 2047–2053
30. K.P. Ng, K.C. Wong (1995). A new diploid scheme and dominance change mechanism for non-stationary function optimisation. *ICGA 6*, pp. 159–166
31. T.T. Nguyen, S. Yang, J. Branke (2012). Evolutionary dynamic optimization: A survey of the state of the art. *Swarm & Evol. Comput.*, 6: 1–24
32. F. Oppacher, M. Wineberg (1999). The Shifting balance genetic algorithm: Improving the GA in a dynamic environment. *GECCO'99*, vol. 1, pp. 504–510
33. H. Richter (2010). Evolutionary optimization and dynamic fitness landscapes: From reaction-diffusion systems to chaotic cml. *Evolutionary Algorithms and Chaotic Systems*, Springer, pp. 409–446.

References – 4

34. P. Rohlfshagen, P.K. Lehre, X. Yao (2009). Dynamic evolutionary optimisation: An analysis of frequency and magnitude of change. *GECCO'09*, pp. 1713–1720
35. S.A. Stanhope, J.M. Daida (1999). (1+1) genetic algorithm fitness dynamics in a changing environments. *CEC'99*, vol. 3, pp. 1851–1858
36. R. Tinos, S. Yang (2010) An analysis of the XOR dynamic problem generator based on the dynamical system. *PPSN XI*, LNCS 6238, Part I, pp. 274–283
37. R. Tinos, S. Yang (2014). Analysis of fitness landscape modifications in evolutionary dynamic optimization. *Inform. Sci.*, 282: 214–236
38. A. Simões, E. Costa (2009). Improving prediction in evolutionary algorithms for dynamic environments. *GECCO'09*, pp. 875–882
39. K. Trojanowski, Z. Michalewicz (1999). Searching for optima in non-stationary environments. *CEC'99*, vol. 3, pp. 1843–1850
40. A.S. Uyar, A.E. Harmanci (2005). A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments. *Soft Comput.*, 9: 803–814
41. S. Yang (2003). Non-stationary problem optimization using the primal-dual genetic algorithm. *CEC'03*, pp. 2246–2253
42. S. Yang, Y. Jiang, T.T. Nguyen (2013). Metaheuristics for dynamic combinatorial optimization problems. *IMA J of Management Math.*, 24(4): 451–480
43. S. Yang, C. Li (2010). A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. IEEE Trans. Evol. Comput., 14(6): 959–974
44. S. Yang, Y.-S. Ong, Y. Jin (2007). *Evolutionary Computation in Dynamic and Uncertain Environments.* Springer

References – 5

45. S. Yang, H. Richter (2009). Hyper-learning for population-based incremental learning in dynamic environments. *CEC'09*, pp. 682–689
46. S. Yang, R. Tinos (2008). Hyper-selection in dynamic environments. *CEC'08*, pp. 3185–3192
47. S. Yang, X. Yao (2005). Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Comput.*, 9: 815–834
48. S. Yang, X. Yao (2008). Population-based incremental learning with associative memory for dynamic environments. *IEEE Trans Evol Comput*, 12: 542–561
49. S. Yang, X. Yao (2013). *Evolutionary Computation for Dynamic Optimization Problems.* Springer
50. K. Weicker (2003). *Evolutionary Algorithms and Dynamic Optimization Problems.* Der Andere Verlag
51. S. Zeng et al. (2006). A dynamic multi-objective evolutionary algorithm based on an orthogonal design. *CEC'06*, pp. 573–580
52. Z. Zhang, S. Qian (2011). Artificial immune system in dynamic environments solving time-varying non-linear constrained multi-objective problems. *Soft Comput.*, 15(7): 1333–1349