Evolutionary Computation for Dynamic Optimization Problems: Fundamentals

Shengxiang Yang

Centre for Computational Intelligence (CCI)
De Montfort University, Leicester LE1 9BH, UK
http://www.tech.dmu.ac.uk/~syang
Email: syang@dmu.ac.uk

DE MONTFORT UNIVERSITY LEICESTER

CCI Centre for Computational Intelligence

Lecturer: Shengxiang Yang

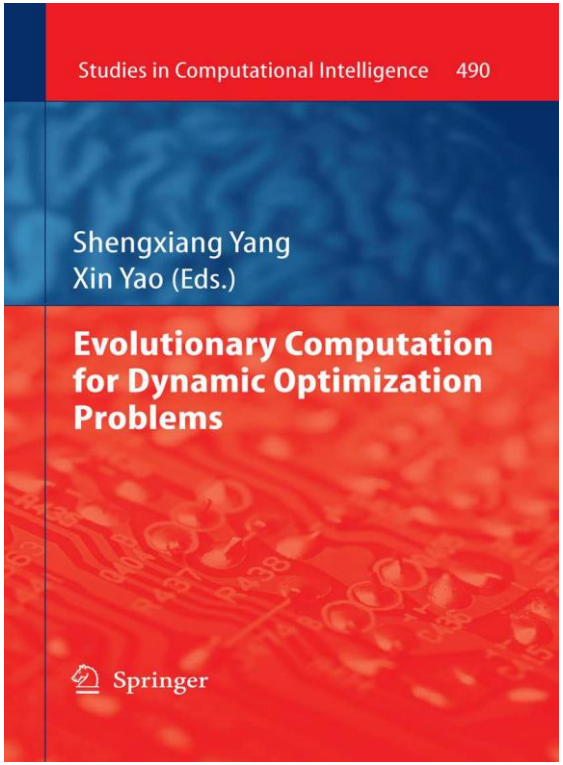
- Education and career history:
  - PhD, Northeastern University, China, 1999
  - Worked at King's College London, University of Leicester, Brunel University, 1999-2012
  - Joined De Montfort University (DMU) as Professor in Computational Intelligence (CI) in July 2012
  - Director, Centre for Computational Intelligence (CCI), since 2013
- Research interests:
  - Evolutionary Computation (EC) and nature-inspired computation
  - Dynamic optimisation and multi-objective optimisation
  - Relevant real-world applications
- Over 290 publications and over £2M funding
- AE/Editorial Board Member for 8 journals
- Chaired two IEEE CIS Task Forces
  - EC in Dynamic and Uncertain Environments (2011-2017)
  - Intelligent Network Systems (2012-2017)

Centre for Computational Intelligence, DMU

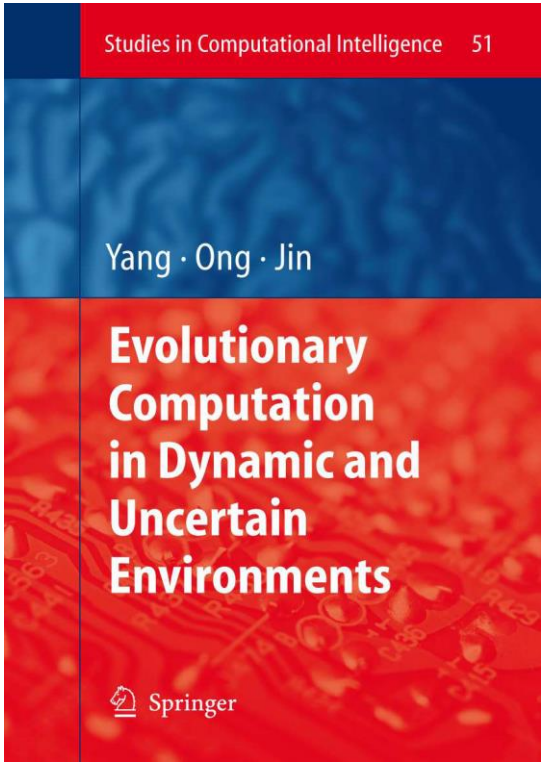
- CCI (www.cci.dmu.ac.uk):
  - Mission: Developing fundamental theoretical and practical solutions to real-world problems using a variety of CI paradigms
  - Members: 16 staff, research fellows, ~30 PhDs, visiting researchers
  - Themes: EC, fuzzy logic, neural networks, data mining, robotics, game …
- Funding:
  - Research Councils/Charities: EPSRC, ESRC, EU FP7 & Horizon 2020, Royal Society, Royal Academy of Eng., Innovate UK, KTP, Nuffield Trust …
  - 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 (& Robotics); Data Analytics; BI & Data Mining
  - BSc: AI with Robotics; Computer Game Programming; Math; Comp Sci
- YouTube page: http://www.youtube.com/thecci

Two Relevant Books

- S. Yang and X. Yao (eds.), Evolutionary Computation for Dynamic Optimization Problems, in the series Studies in Computational Intelligence, vol. 490, Springer, 2013



- S. Yang, Y.-S. Ong, Y. Jin (eds.), Evolutionary Computation in Dynamic and Uncertain Environments, in the series Studies in Computational Intelligence, vol. 51. Springer, March 2007

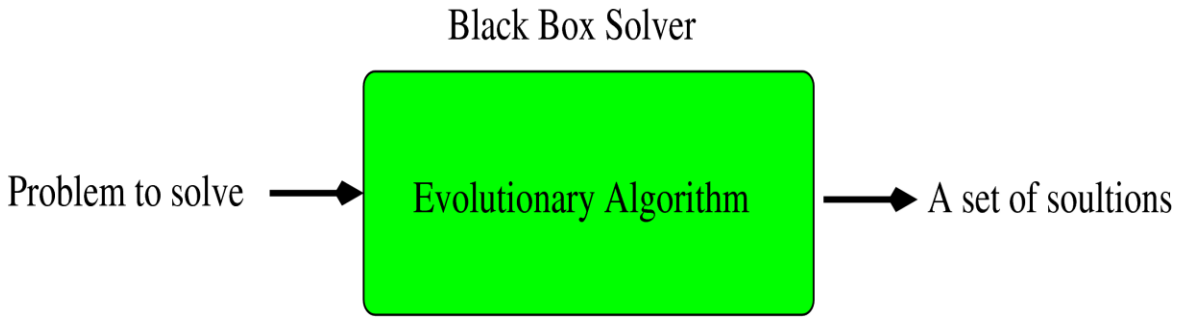


Outline of the Lecture

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

What is Evolutionary Computation (EC)?

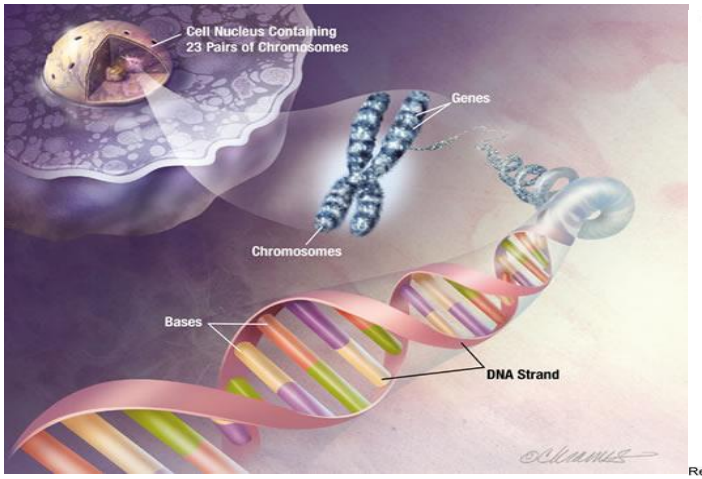
- EC encapsulates a class of stochastic optimisation algorithms, dubbed Evolutionary Algorithms (EAs)
- An EA is an optimisation algorithm that is
  - Generic: a black-box tool for many problems
  - Population-based: evolves a population of candidate solutions
  - Stochastic: uses probabilistic rules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

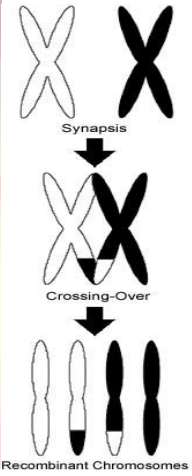


Principles of Biological Evolution

- For each organism, a population of individuals exists in a natural environment with limited resources and evolves as:
  - Individuals compete for resources: selection of the fittest to reproduce offspring
  - They reproduce offspring by genetic inheritance with variation (recombination and mutation)
  - Offspring have their fitness evaluated by the environment and compete for survival

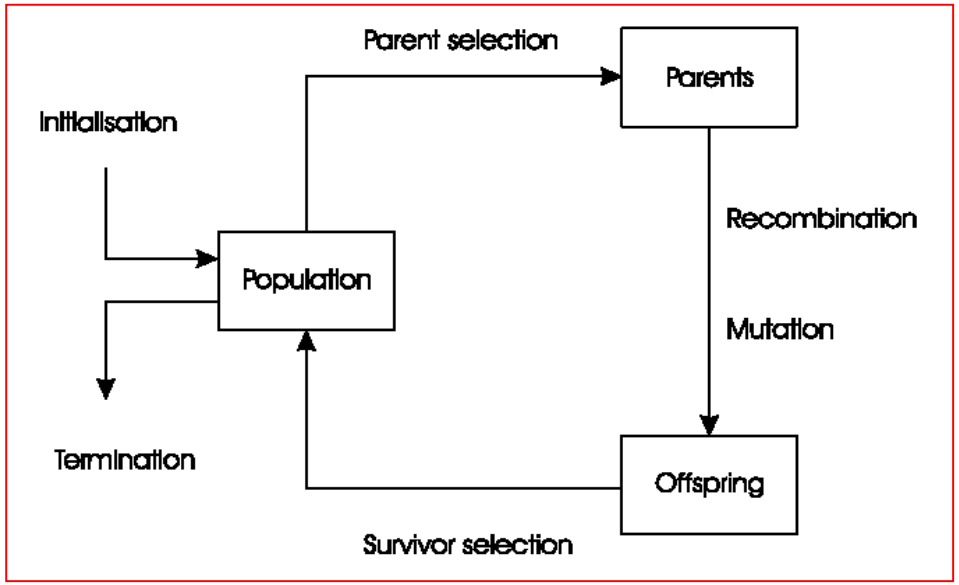
- Over time, natural selection drives the population to be better and better





Design and Framework of an EA

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

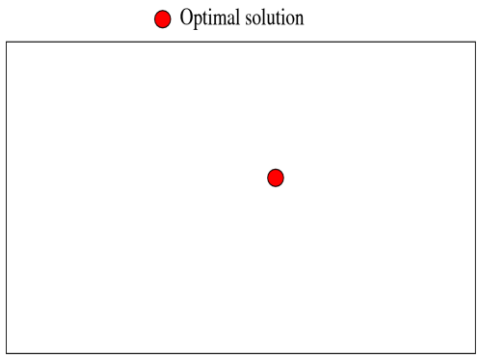


EC Applications

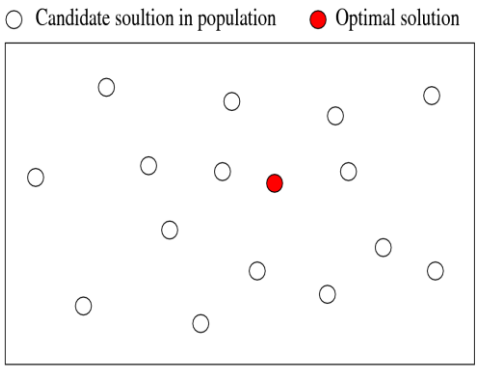
- Easy to use: No strict requirements to problems
- Widely used for optimisation and search problems
  - Search-based software engineering
  - Financial and economical systems
  - Transportation and logistics systems
  - Automatic programming, art and music design
  - Industrial engineering
  - ……

EC for Dynamic Optimisation Problems: Motivation

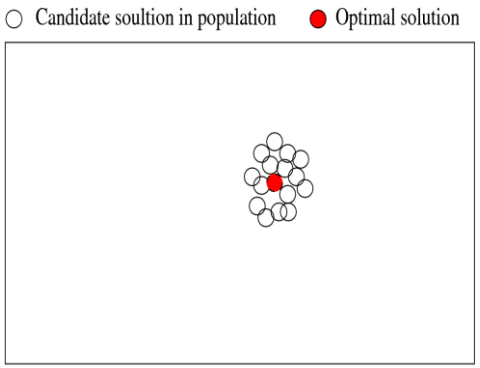
- Traditionally, EAs are mainly applied for static problems
  - Aim: find optimum quickly and precisely in the search space



Search space



Search space (Initial population)



Search Space (Population converging at time t)

- But, many real-world problems are dynamic optimisation problems (DOPs), where changes occur over time
  - Transport systems: travel time between nodes may change
  - Logistics: customer demands may change

What Are DOPs?

- In general terms, “optimisation problems that change over time” are called dynamic problems or time-dependent problems:

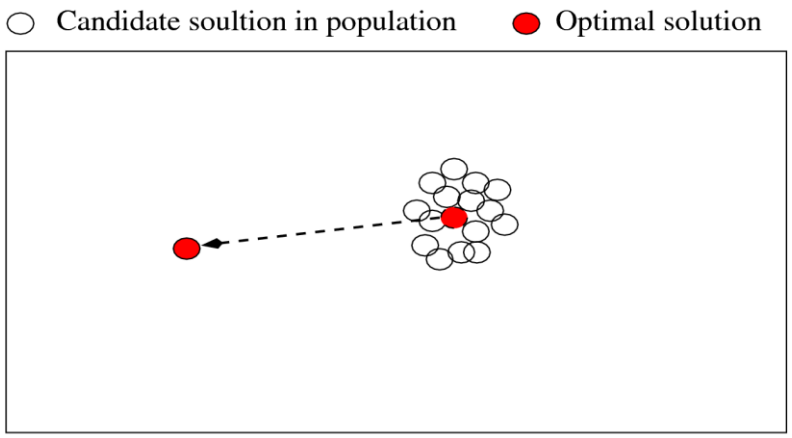
$$F = f(X, S, t)$$

where X: decision variable(s); S: parameters; t: time

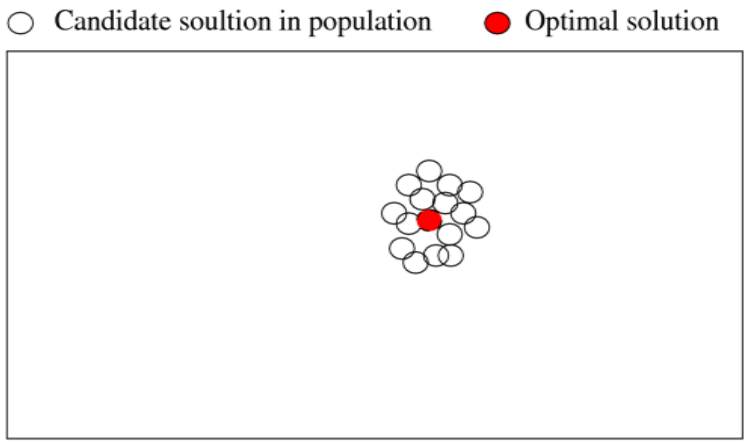
- DOPs: a special class of dynamic problems that are solved online by an algorithm as time goes by

Why DOPs Challenging EC?

- For DOPs, optima may move over time in search space
  - We need to track the moving optima over time
- DOPs challenge traditional EAs
  - Once converged, hard to escape from the old optimum



Search Space (Optimum moved at time t+1)



Search Space (Population converging at time t)

Why EC for DOPs?

- Many real-life problems are DOPs
  - Desirable to present solutions to decision makers over time
- EAs, once properly enhanced, are good choice
  - Inspired by biological behaviour, always in dynamic environments
  - Intrinsically, should be fine to deal with DOPs
- Research on EC for DOPs rises recently
  - Books, PhD Theses
  - Journal special issues
  - Workshops and conference special sessions
  - IEEE Symposium on CIDUE (2011, 2013-2020)
  - IEEE Competitions: within 2009 & 2012 IEEE CEC

Benchmark and Test DOPs

- Basic idea: change base static problem 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:

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

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

- The dynamics:

$$H_i(t) = H_i(t-1) + height\_severity * \sigma$$

$$W_i(t) = W_i(t-1) + width\_severity * \sigma$$

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

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

  – $\sigma \sim N(0,1)$; 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)

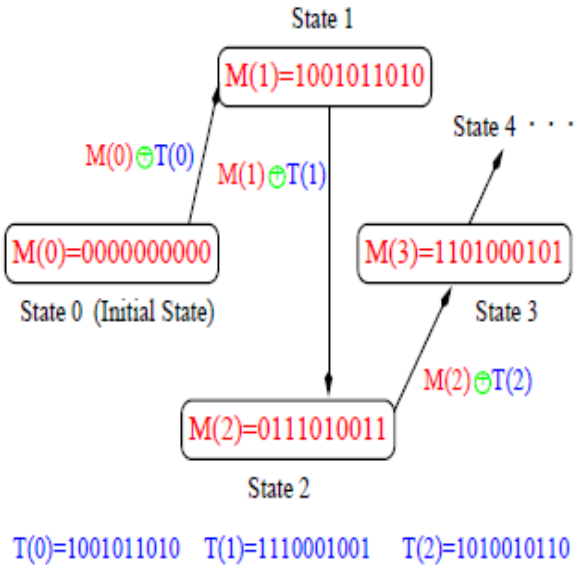
- Static knapsack problem:
  - Given *n* items, each with a weight and a profit, and a knapsack with a fixed capacity, select items to fill up the knapsack to maximize the profit while satisfying the knapsack capacity constraint
- The DKP:
  - Constructed by changing weights and profits of items, and/or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

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

– $\vec{x}(t) \in \{0,1\}^n$: 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:

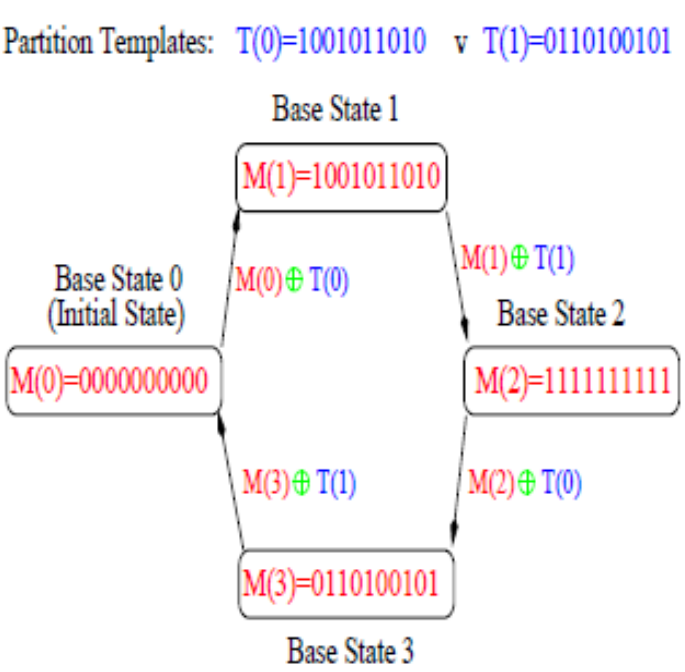
$$f(\vec{x}, t) = f(\vec{x} \oplus \vec{M}(k))$$

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

S. Yang and X. Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Computing, 9(11): 815-834, November 2005.

Constructing Cyclic Dynamic Environments

- Extend 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

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

– $k = \lfloor t/\tau \rfloor$: environmental index

– $l_t = k\%(2K)$: mask index

S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. IEEE Transactions on Evolutionary Computation, 12(5): 542-561, October 2008.

Dynamic Traveling Salesman Problem

- 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: 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 $x_{n+1} = x_1$

Performance Measures

- For EC for static 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
    - Satisfiability
    - 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)

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

- 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)

EC for DOPs: Things to Do

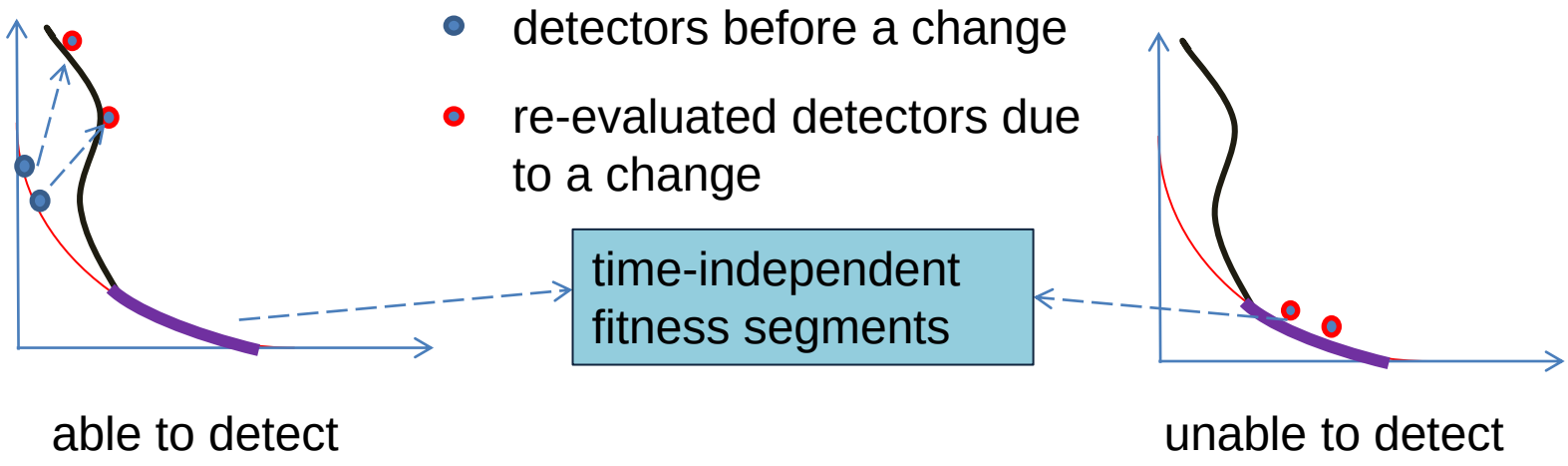
- Change detection: detect potential environmental changes
  - Success rate of detection
  - Cost of detection
- Response or enhancing approaches: track the changing optima
  - To expect a steady and fast change response
  - To reduce the cost of tracking (given the budget limit, i.e., time, memory)

Change Detection Approaches

- Two ways of detecting changes:
  - Individual-level detection: fast but not robust
  - Population-level detection: slow but robust
  - Both methods could fail to detect changes (not 100% guaranteed)

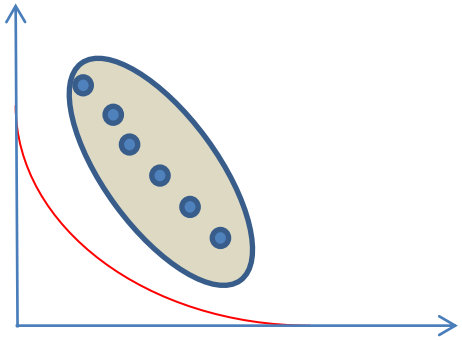
Individual-level Change Detection

- Re-evaluate some individuals' fitness every generation
  - Check the discrepancy between their current and previous fitness
- Success rate of detection depends on
  - Detectability of environmental changes
  - Location of detectors placed

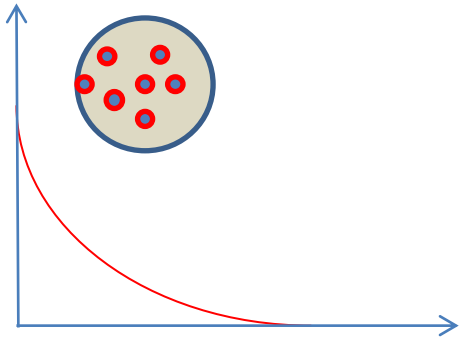


Population-level Change Detection

- Population-related statistical information, i.e., distribution, is assessed in every generation
  - Check the significance of variation in statistical information



population distribution
before a change



population distribution
after a change

- Less sensitive to noise but possibly higher computational cost

Response/Enhancing: 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

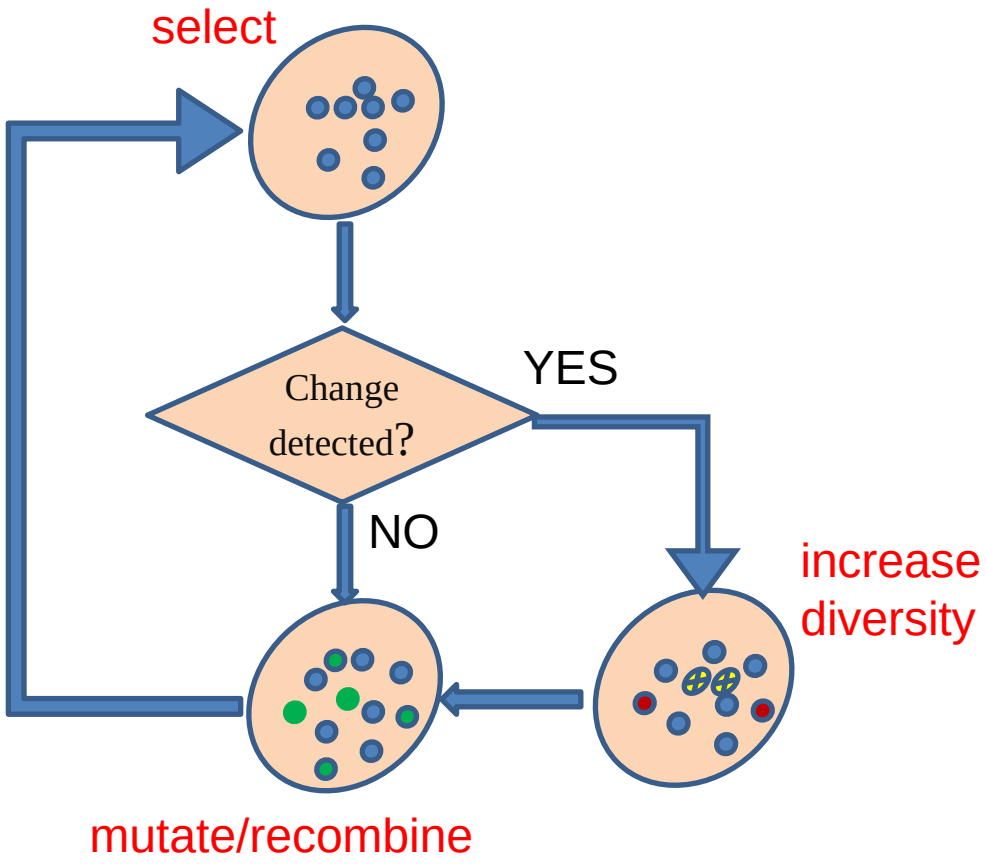
Response/Enhancing: General Approaches

- Many approaches developed to enhance EAs for DOPs
- Typical approaches:
  - Diversity: handle convergence directly
  - Memory: store and reuse useful information
  - 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

1. M. Mavrovouniotis, C. Li, and S. Yang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. Swarm and Evolutionary Computation, 33: 1-17, April 2017
2. T. T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. Swarm and Evolutionary Computation, 6: 1-24, October 2012

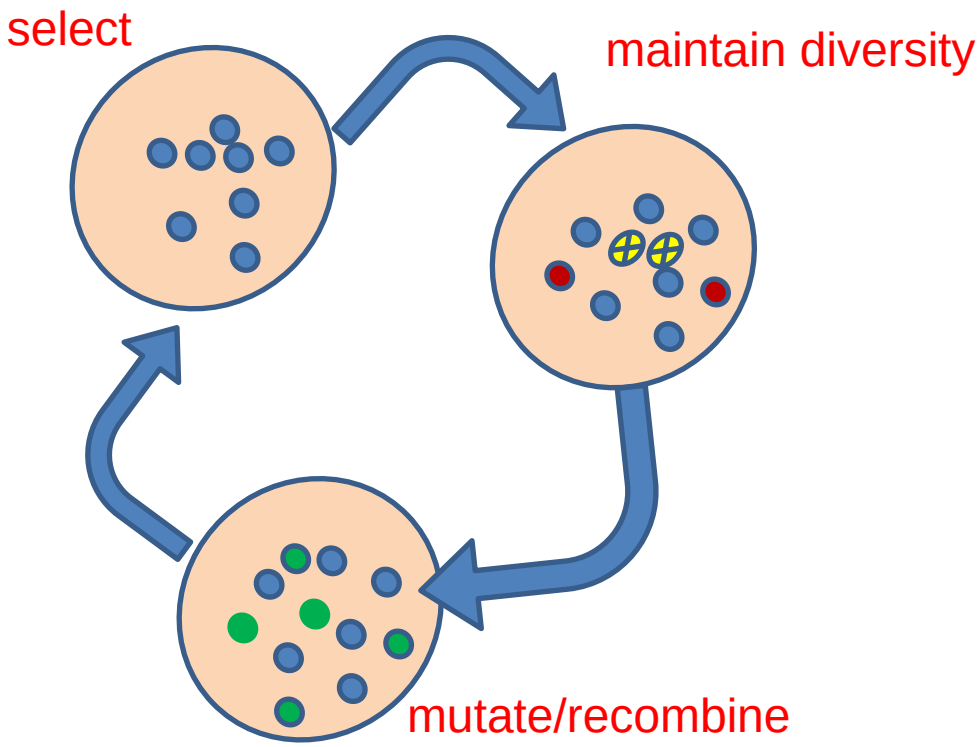
Diversity Approaches

- Diversity increase: introduce diversity after a change
  - Partially random restart, hyper-mutation, variable local search



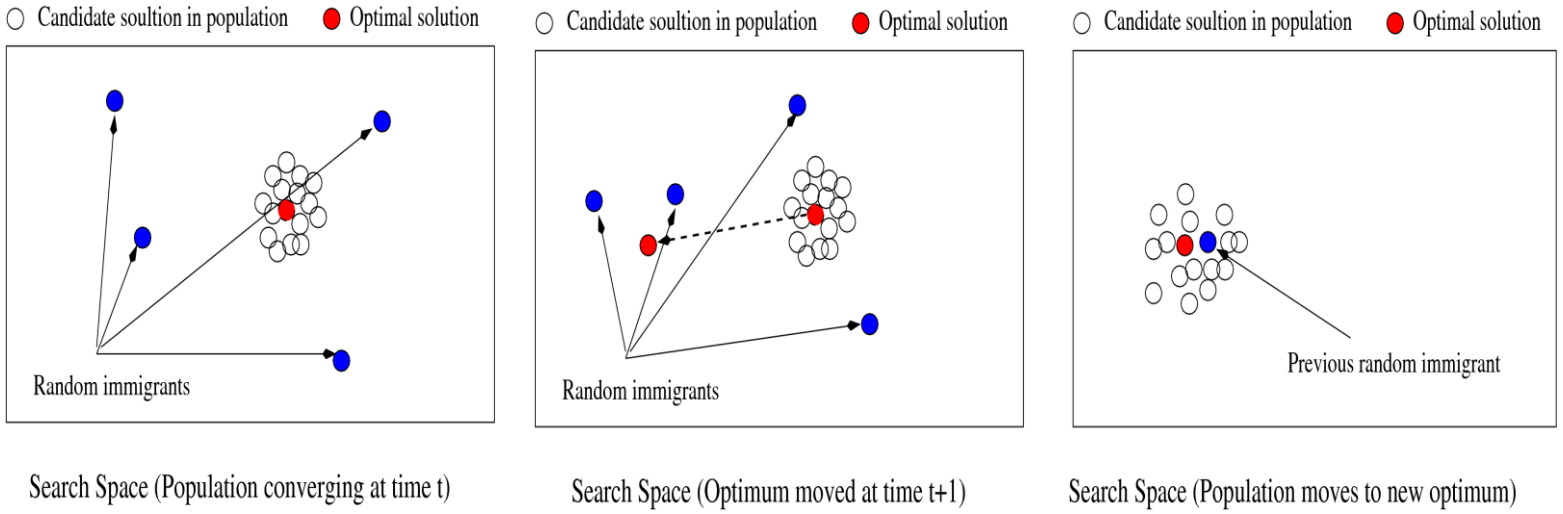
Diversity Approaches

- Diversity maintenance: maintain diversity throughout the run (even if no change occurs)
  - Random immigrants



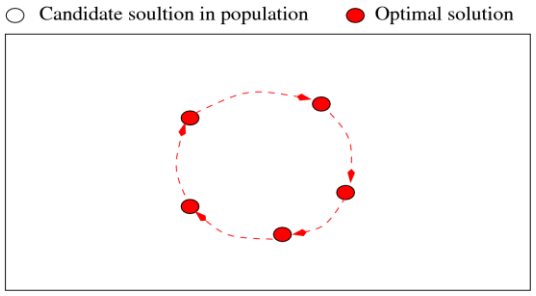
Diversity Approaches: Random Immigrants

- Diversity approaches address convergence directly
  - Each generation, generate some random solutions (random immigrants) into the population to maintain the diversity
  - When optimum moves, random immigrants nearby take action
  - They will draw the population to the new optimum



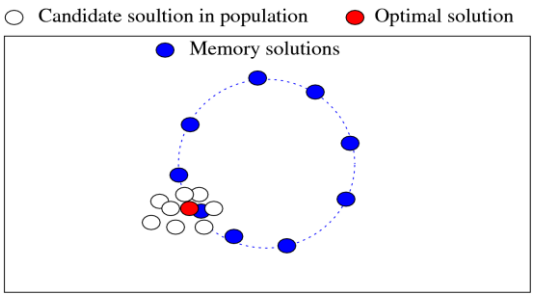
Memory Approaches

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

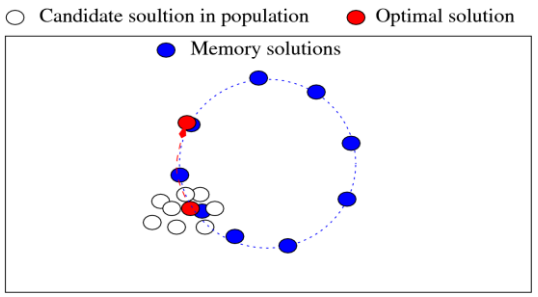


Search space (Optimum moves cyclically)

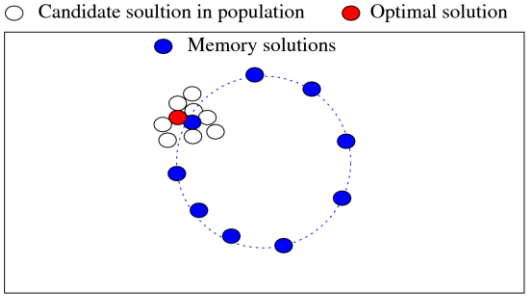
- For cyclic DOPs, use memory to store & reuse good solutions
  - With time, store the best solution of population into memory
  - When a change occurs, memory helps to track new optimum



Search space (Memory stores best solutions)



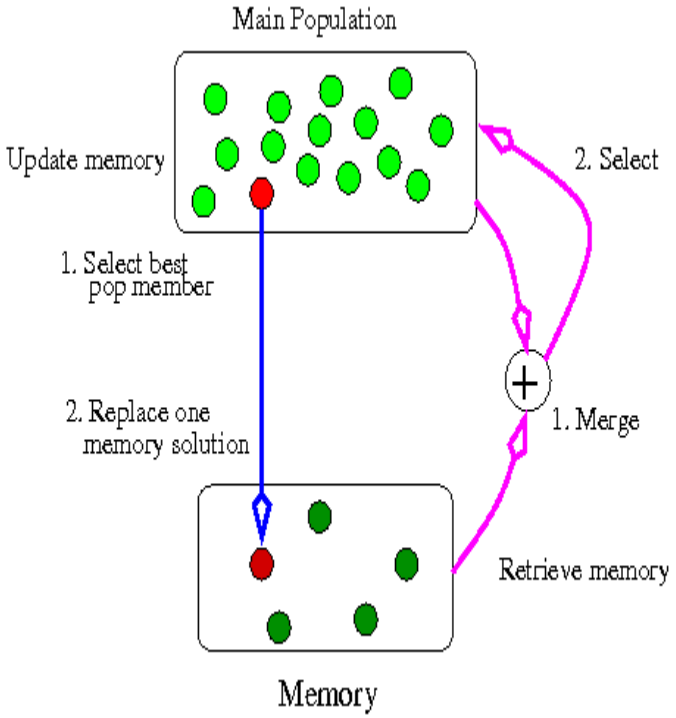
Search space (Optimum moves to next state)

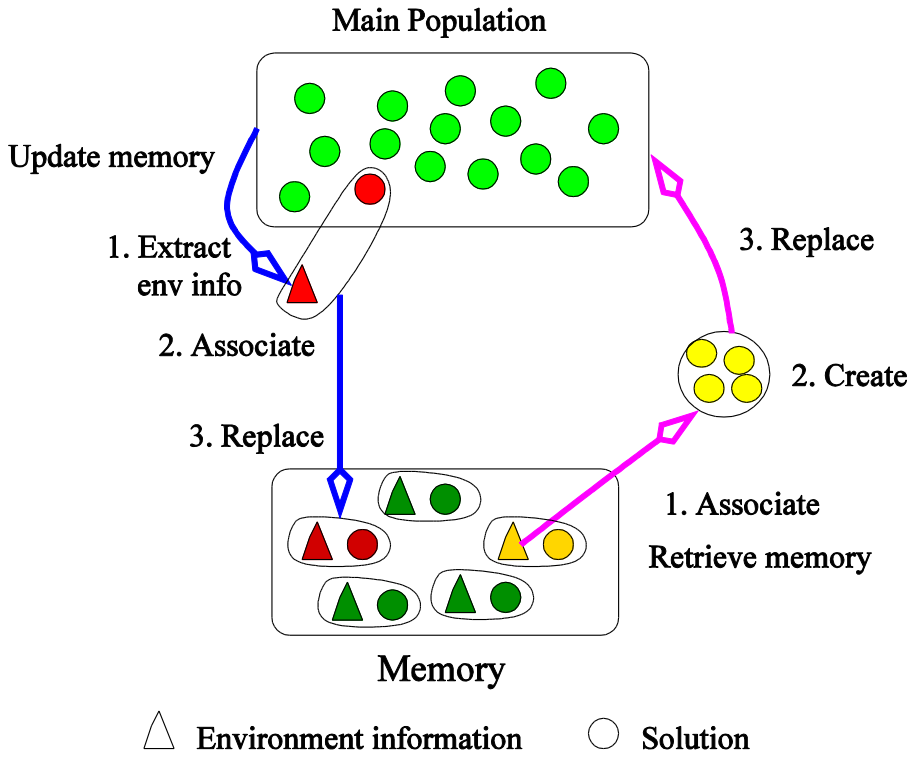


Search space (Population moves to new optimum)

Direct Memory vs Associative Memory

- Direct memory: store good solutions (Branke, CEC'99)
- Associative memory: store environmental information + good solutions

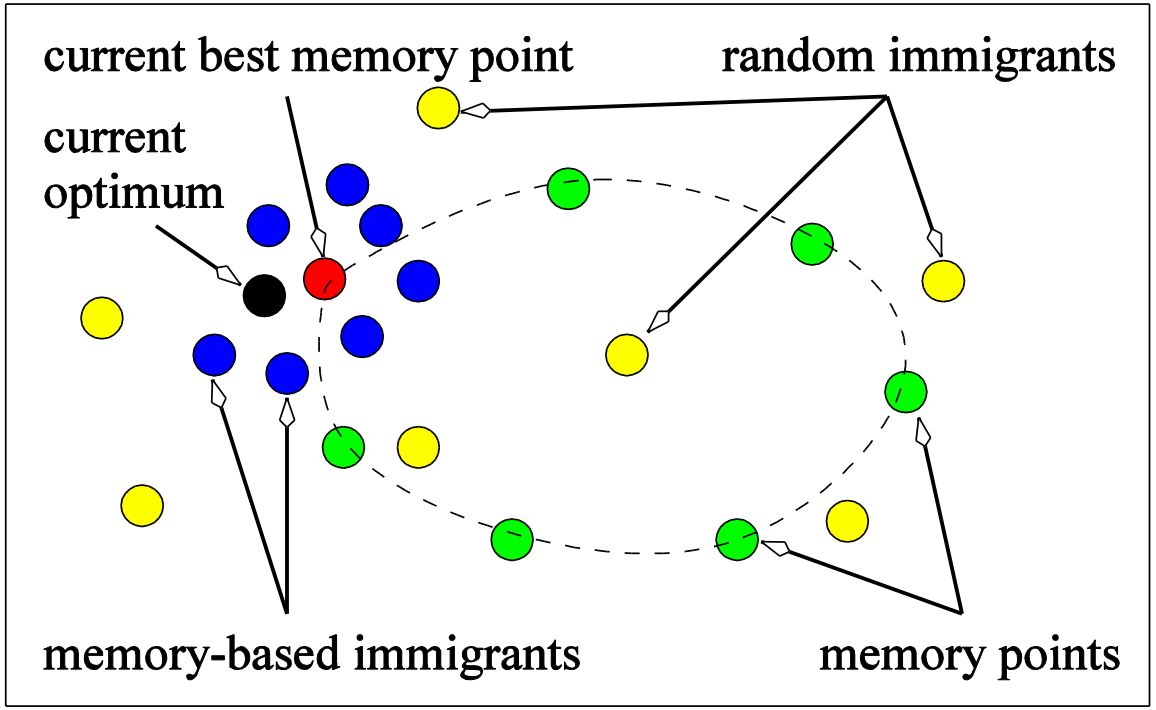




S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. IEEE Transactions on Evolutionary Computation, 12(5): 542-561, October 2008

Memory-Based Immigrants

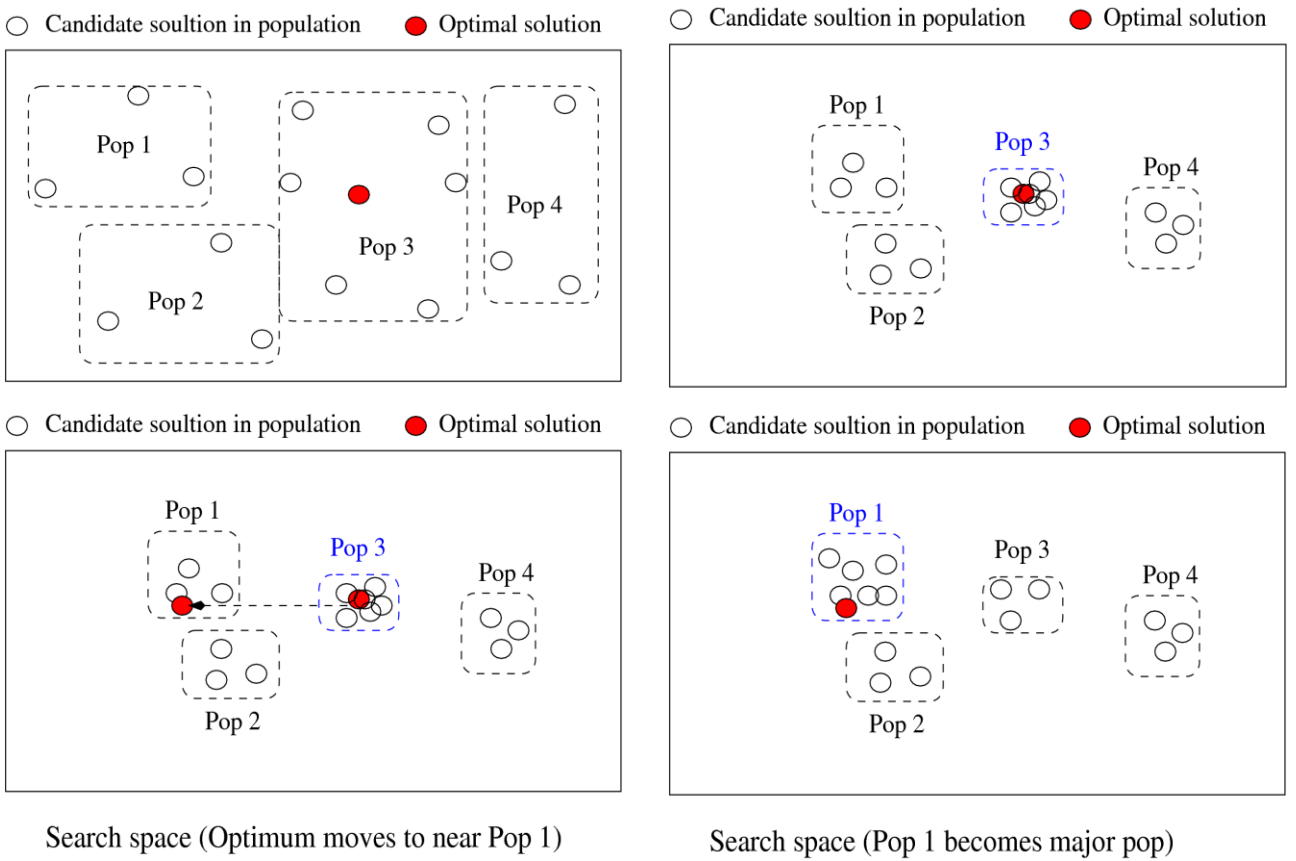
- Use memory to guide immigrants toward current environment



S. Yang. Genetic algorithms with memory- and elitism-based immigrants in dynamic environments. Evolutionary Computation, 16(3): 385-416, Fall 2008

Multi-population Approaches

- Idea: Use several cooperative populations
  - Populations evolve independently in different areas of search space
  - Populations exclude each other to avoid overlap
  - When optimum moves, nearby population will take action

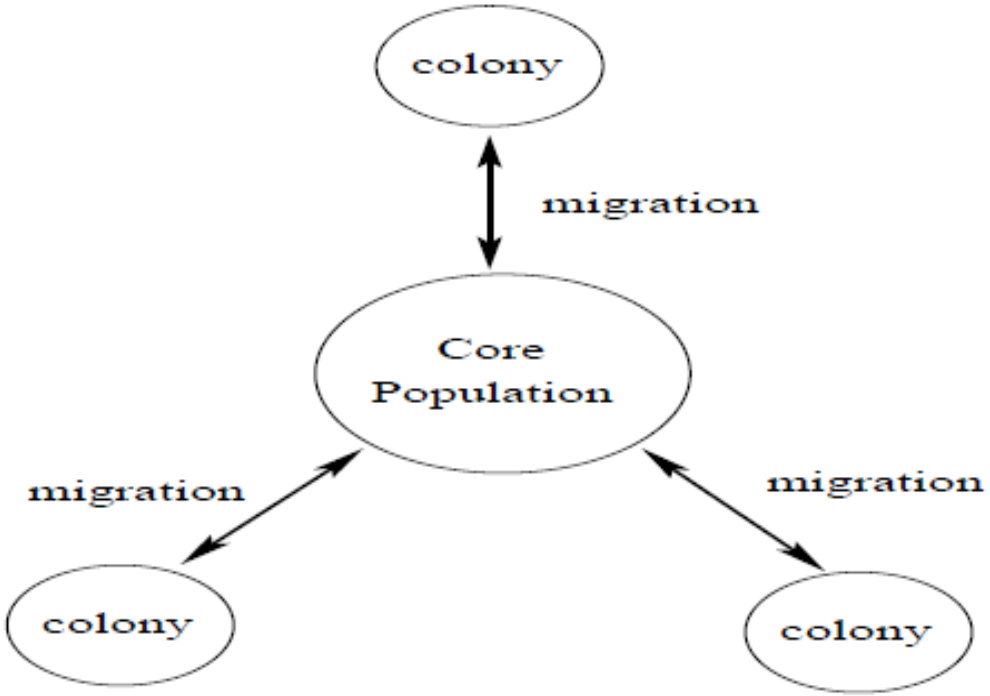


Search space (Optimum moves to near Pop 1)

Search space (Pop 1 becomes major pop)

Multi-Populations: Shifting Balance

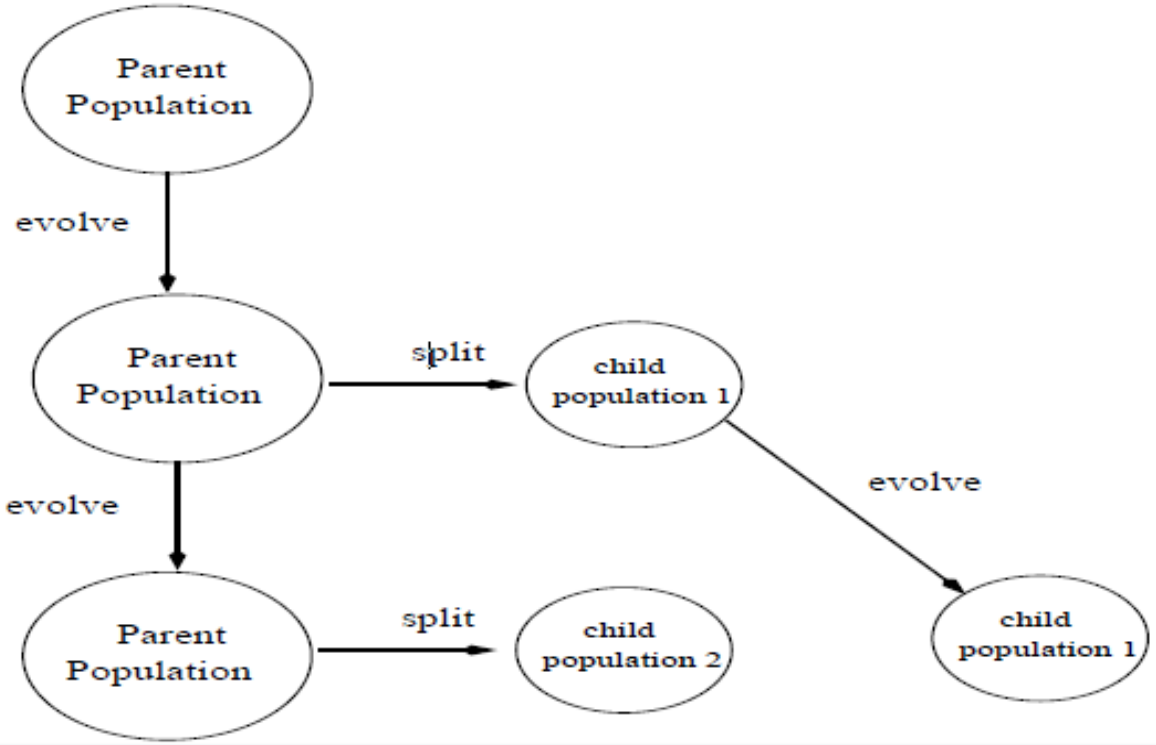
- Shifting Balance GA (Oppacher & Wineberg, 1999):
  - A core population exploits the promising area
  - Several colonies explore the search space



F. Oppacher, M. Wineberg (1999). The Shifting balance genetic algorithm: Improving the GA in a dynamic environment. GECCO'99, vol. 1, pp. 504–510

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



Branke J., Kaussler T., Smidt C., Schmeck H. (2000) A Multi-population Approach to Dynamic Optimization Problems. In: Parmee I.C. (eds) Evolutionary Design and Manufacture. Springer, London

Prediction Approaches

- For some DOPs, changes exhibit predictable patterns
- Techniques (e.g., forecasting, Kalman filter) can be used to predict
  - The location of the next optimum after a change
  - When the next change may occur and which environment may appear

Adaptive Approaches

- Aim: Adapt operators/parameters, usually after a change
  - Hypermutation (Cobb & Grefenstette'93): raise the mutation rate temporarily
  - Hyper-selection (Yang & Tinos'08): raise the selection pressure temporarily
  - Hyper-learning (Yang & Richter'09): raise the learning rate for Population-Based Incremental Learning (PBIL) temporarily

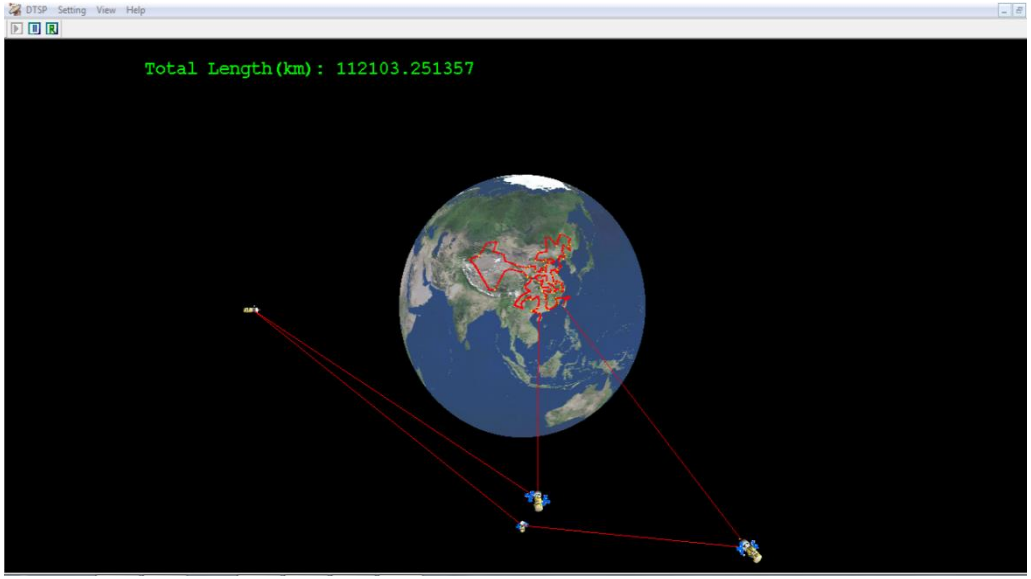
Hybridization Approaches

- Idea: Using hybridization technique to improve the performance of EC for DOPs
- Hybridize EC with local search + diversity schemes, e.g.:
  - P-ACO: Hybridize ACO with local search and random immigrants
  - Multi-strategy ensemble PSO (MEPSO): Hybridize PSO with Gaussian local search + differential mutation
- Hybridize EC with other meta-heuristic methods
  - PSO + Cellular Aotomata
  - PSO + Fuzzy C-means

- M. Mavrovouniotis, S. Yang, A memetic ant colony optimization algorithm for the dynamic travelling salesman problem, Soft Comput. 15 (7) (2011) 1405–1425
- W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Inf. Sci. 178 (15) (2008) 3096–3109
- A. Hashemi, M. Meybodi, A multi-role cellular PSO for dynamic environments, in: 14th International Computer Conference (CSICC 2009), 2009, pp. 412–417
- M. Kamosi, A. Hashemi, M. Meybodi, A hibernating multi-swarm optimization algorithm for dynamic environments, in: 2010 2nd World Congress on Nature and Biologically Inspired Computing, 2010, 363–369

Demo: Dynamic Travelling Salesman Problem

- Problem:
  - 144 Chinese cities, one geo-stationary satellite, and three mobile satellites
  - Find the path that cycles each city and satellite once with the minimal length over time
- Solver: EA with memory & immigrants schemes



Remarks on Enhancing Approaches

- No clear winner among the approaches
- Memory is efficient for cyclic environments
- Multi-population is good for multimodal problems
  - Able to maintain diversity
  - The search ability will decrease if too many sub-populations
- Diversity schemes are usually useful
  - Guided immigrants may be more efficient
- Thumb of rule: balancing exploration & exploitation over time

Summary

- EC for DOPs: important area
  - The domain is still young and active
  - Many challenges to be taken
- More young researchers are greatly welcome!

Thanks!

Relevant Information

- IEEE CIS Task Force on EC in Dynamic and Uncertain Environments
  - http://ieee-tf-ecidue.cug.edu.cn/
- Source codes:
  - http://www.tech.dmu.ac.uk/~syang/publications.html
- Books and survey papers:
  - M. Mavrovouniotis, C. Li, and S. Yang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. Swarm and Evolutionary Computation, 33: 1-17, April 2017
  - T. T. Nguyen, S. Yang, J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. Swarm and Evolutionary Computation, 6: 1-24, 2012
  - Y. Jin, J. Branke. Evolutionary optimization in uncertain environments–A survey. IEEE Trans Evol Comput, 9(3): 303–317, 2005
  - S. Yang, X. Yao. Evolutionary Computation for Dynamic Optimization Problems. Springer, 2013