

Lecture 15

Evolutionary Dynamic Optimisation

Multi-populations

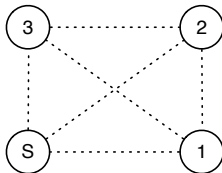
Dr. Philipp Rohlfshagen
P.Rohlfshagen@cs.bham.ac.uk

November 20, 2009

Dynamic TSP I

Classic TSP: Visit every city once, return to origin

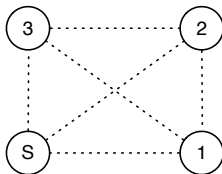
- ▶ Minimise some cost of travel (e.g., time, distance)
- ▶ Consider $n = 4$, start and finish at S



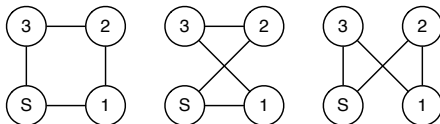
Dynamic TSP I

Classic TSP: Visit every city once, return to origin

- ▶ Minimise some cost of travel (e.g., time, distance)
- ▶ Consider $n = 4$, start and finish at S

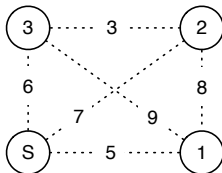


$3!/2 = 3$ unique solutions



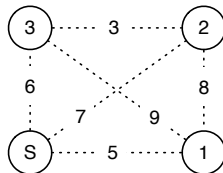
Dynamic TSP II

Each edge has a value: Time

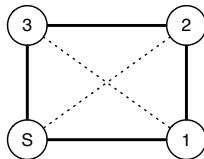


Dynamic TSP II

Each edge has a value: Time



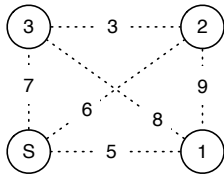
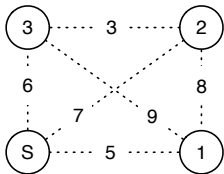
Optimal solution: $\mathbf{x}^* = S - 1 - 2 - 3 - S$ with $f(\mathbf{x}^*) = 22$



Dynamic TSP III

Time of travel depends on traffic

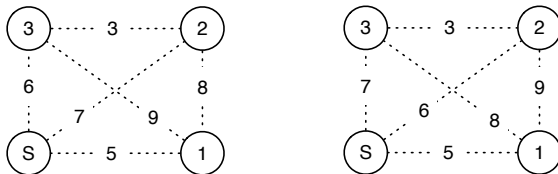
- ▶ Traffic patterns are dynamic and change with time: $I(T) \rightarrow I(T + 1)$



Dynamic TSP III

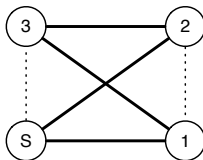
Time of travel depends on traffic

- ▶ Traffic patterns are dynamic and change with time: $I(T) \rightarrow I(T + 1)$



New optimal solution: $\mathbf{x}^*(T + 1) = S - 1 - 3 - 2 - S$ with $f(\mathbf{x}^*(T + 1)) = 22$

- ▶ **Previous** optimal solution now has $f(\mathbf{x}^*(T)) = 24$



Introduction

Most optimisation problems considered are static/stationary

Most real-world problems change over time

- ▶ Machines fail unexpectedly, dynamic traffic patterns, etc.
- ▶ New technology allows us to capture and process data (e.g., GPS)

Dynamic problems usually need to be solved online

- ▶ Adjust solution as time goes by; do not know future function values
- ▶ Ensure solution quality/feasibility

History of the field

- ▶ Initial efforts as early as 1966 but most progress in the last 10 years
 - ▶ Special sessions/workshops at major conferences (CEC, EvoSTAR)
 - ▶ Special issues in journals

Definition 1

DOPs have an additional parameter: Time t

- ▶ Time is discrete and advances with every function evaluation

Time-variant sequence of *instances* of the same *problem*

$$I(T = 0) \longrightarrow I(T = 1) \longrightarrow I(T = 2) \longrightarrow \dots$$

- ▶ $T \cdot \tau \leq t < (T + 1) \cdot \tau$ where $1/\tau$ is the frequency of change
- ▶ The update period τ could be a function of time itself

Definition 1

DOPs have an additional parameter: Time t

- ▶ Time is discrete and advances with every function evaluation

Time-variant sequence of *instances* of the same *problem*

$$I(T = 0) \longrightarrow I(T = 1) \longrightarrow I(T = 2) \longrightarrow \dots$$

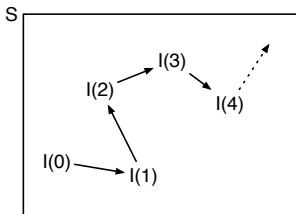
- ▶ $T \cdot \tau \leq t < (T + 1) \cdot \tau$ where $1/\tau$ is the frequency of change
- ▶ The update period τ could be a function of time itself

In practice, **any** component of the problem may be variant

1. Domain of the variables; e.g., from $[0, 1]$ to $[-1, 1]$
2. Dimensionality of the problem; e.g., from $\{0, 1\}^n$ to $\{0, 1\}^{n+m}$
3. Constraints of the problem; e.g., $g(\mathbf{x}) \leq y$ to $g(\mathbf{x}) \leq y + q$
4. Parameters of the function; e.g., $f(\mathbf{x}) = x^2 + 1$ to $f(\mathbf{x}) = x^{1.5} + 0.5$

Definition II

The dynamics describe a trajectory through the space of all instances



- ▶ Typical case stationary optimisation: $f(x) = 2x^2$, $x \in [l, u]$
- ▶ Dynamic case: $f(x; a, b) = ax^b$ (a class of functions)

Some mapping $\mathcal{T} : S \times \mathbb{N} \rightarrow S$ such that

$$I(T + 1) = \mathcal{T}(I(T), t)$$

Motivation

Dynamic problem: Time-series of problem instances

- ▶ Could treat each instance as a new problem
- ▶ Random re-starts whenever a change occurs
- ▶ This may be slow and costly

Key idea is to “**transfer knowledge from the past**” (Branke)

- ▶ Speed up the process of re-optimisation
- ▶ Successive instances need to be related to some degree

Motivation

Dynamic problem: Time-series of problem instances

- ▶ Could treat each instance as a new problem
- ▶ Random re-starts whenever a change occurs
- ▶ This may be slow and costly

Key idea is to “transfer knowledge from the past” (Branke)

- ▶ Speed up the process of re-optimisation
- ▶ Successive instances need to be related to some degree

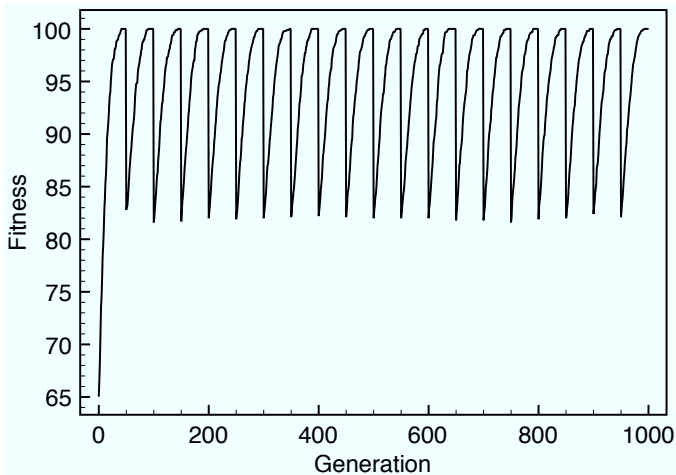
Commonly used performance measure reflects this

$$\hat{f} = \frac{1}{G} \sum_{i=1}^G e'_i$$

$$e'_i = \max\{e_\omega, e_{\omega+1}, \dots, e_i\}$$

- ▶ where ω is last time step smaller than i at which a change occurred
- ▶ e_i is the best at iteration (generation) i

Performance



Change happens every 50 generations

Change Detection & Outdated Fitness Values

Need to make sure fitness values are kept **up-to-date**

- ▶ Values stored may represent outdated information
- ▶ Generational EAs are commonly used but could still pose a problem

Change Detection & Outdated Fitness Values

Need to make sure fitness values are kept **up-to-date**

- ▶ Values stored may represent outdated information
- ▶ Generational EAs are commonly used but could still pose a problem

May need to detect change

- ▶ **Population-based**
 - ▶ Monitor fitness values of population (use statistical tests)
- ▶ **Sensor-based**
 - ▶ Monitor fitness of stationary points
 - ▶ Placement of points?
- ▶ Neither method may guarantee detection
 - ▶ $\gamma = |Y|/|X|$ where $Y = \{\mathbf{y}^i\}$ such that $\forall_i f(\mathbf{y}^i, T + 1) \neq f(\mathbf{y}^i, T)$
- ▶ Dimensionality and domain changes should be signalled by the system
 - ▶ They require a change in the representation

Common Approaches

Diversity

- ▶ Constant or reactive
- ▶ **Hyper-mutations, Random immigrants**, Thermo-dynamic GA

Common Approaches

Diversity

- ▶ Constant or reactive
- ▶ **Hyper-mutations, Random immigrants**, Thermo-dynamic GA

Memory

- ▶ Implicit or **explicit**, direct or **indirect** (abstraction)
- ▶ Diploidy, external memory, memory-based diversity

Common Approaches

Diversity

- ▶ Constant or reactive
- ▶ **Hyper-mutations, Random immigrants**, Thermo-dynamic GA

Memory

- ▶ Implicit or **explicit**, direct or **indirect** (abstraction)
- ▶ Diploidy, external memory, memory-based diversity

Multi-populations

- ▶ Closely related to speciation
- ▶ Self-organising Scouts (SOS; Branke)

Common Approaches

Diversity

- ▶ Constant or reactive
- ▶ **Hyper-mutations**, **Random immigrants**, Thermo-dynamic GA

Memory

- ▶ Implicit or **explicit**, direct or **indirect** (abstraction)
- ▶ Diploidy, external memory, memory-based diversity

Multi-populations

- ▶ Closely related to speciation
- ▶ Self-organising Scouts (SOS; Branke)

Adaptive variation operators

- ▶ Adapt operators to the current environment
- ▶ Operators sometimes able to capture dynamics

Hypermutations

Proposed by Cobb in 1990

Increase mutation rate after change

- ▶ Mutation probability p_m is usually $1/n$
- ▶ Increase rate for m generations

Pros and Cons?

Hypermutations

Proposed by Cobb in 1990

Increase mutation rate after change

- ▶ Mutation probability p_m is usually $1/n$
- ▶ Increase rate for m generations

Pros and Cons?

- + Does not interfere with ongoing search
- + Introduces diversity into a possibly converged population
- Need to be able to detect change
- How much diversity is needed?
- May destroy valuable information

Hypermutations

Proposed by Cobb in 1990

Increase mutation rate after change

- ▶ Mutation probability p_m is usually $1/n$
- ▶ Increase rate for m generations

Pros and Cons?

- + Does not interfere with ongoing search
- + Introduces diversity into a possibly converged population
- Need to be able to detect change
- How much diversity is needed?
- May destroy valuable information

```
 $i \leftarrow 0$   
Initialise and evaluate  $P(i)$   
  
while terminate  $\neq$  true do  
   $i \leftarrow i + 1$   
  select  $P(i)$  from  $P(i - 1)$   
  cross( $P(i)$ ,  $p_c$ )  
  if  $i - \delta \leq m$  then  
    mutate( $P(i)$ ,  $p_d$ )  
  else  
    mutate( $P(i)$ ,  $p_m$ )  
  end if  
  evaluate  $P(i)$   
  if change is detected then  
     $\delta \leftarrow i$   
  end if  
end while
```

Random Immigrants

Proposed by Grefenstette in 1992

- ▶ **Prevent** complete convergence at all times

Replace γN individuals with random samples

- ▶ Replace random individuals or the weakest

Pros and Cons?

Random Immigrants

Proposed by Grefenstette in 1992

- ▶ **Prevent** complete convergence at all times

Replace γN individuals with random samples

- ▶ Replace random individuals or the weakest

Pros and Cons?

- + Prevents convergence of the population
- + May increase the rate of exploration of the search space
- May slow down process of optimisation
- Added diversity is uninformed
- Which individuals to replace and how many?

Random Immigrants

Proposed by Grefenstette in 1992

- ▶ **Prevent** complete convergence at all times

Replace γN individuals with random samples

- ▶ Replace random individuals or the weakest

Pros and Cons?

- + Prevents convergence of the population
- + May increase the rate of exploration of the search space
- May slow down process of optimisation
- Added diversity is uninformed
- Which individuals to replace and how many?

```
i ← 0  
Initialise and evaluate P(i)  
while terminate ≠ true do  
  i ← i + 1  
  select P(i) from P(i - 1)  
  cross(P(i), pc)  
  mutate(P(i), pm)  
  Replace  $\gamma N$  individuals in P(i) with  
  random samples  
  Evaluate P(i)  
end while
```

Abstract Memory I

Use explicit memory to memorise regions of high quality

- ▶ Need strategy to place individuals into memory
- ▶ Need strategy to retrieve individuals from memory
- ▶ This is known as a **memory management** scheme

Abstract memory due to Richter and Yang

- ▶ Information is stored indirectly, as an abstract approximation
- ▶ Continuous search space is discretised (grid size ϵ)
 - ▶ Memory matrix \mathbb{M} filled with counters
- ▶ High quality individuals encountered are assigned to partition cells
- ▶ Frequency count of partition cells functions as probability distribution

Tested on a continuous problem

- ▶ n-dimensional “field of cones on a zero plane”
- ▶ Classical benchmark problem

Abstract Memory II

At each iteration

- ▶ Select the q best individuals from the population

For each selected individual

- ▶ Obtain the index for \mathbb{M} and update count

Perform usual operators (selection, crossover, mutation)

If a change is detected

- ▶ Create adjunctive memory matrix \mathbb{M}_μ
- ▶ Fix number of individuals to be generated
- ▶ Calculate distribution of individuals across \mathbb{M}_μ
- ▶ Randomly fix position of individual in cell
- ▶ Merge generated individuals with population

Related Fields and Future Directions

Closely related fields

- ▶ Noisy optimisation, surrogate-assisted (approximated) optimisation, robust optimisation
- ▶ Also related is multi-objective optimisation and co-evolution

Modified offline performance is only one performance measure

- ▶ Solution similarity
 - ▶ For example, drivers prefer similar/identical routes for delivery
- ▶ Feasibility driven
 - ▶ Ensure you always have a feasible solution at all times
- ▶ Robustness
 - ▶ Small changes: Do not adapt; large changes: Adapt

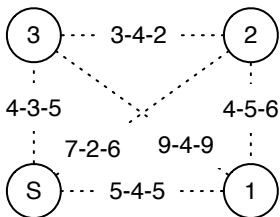
Related Fields and Future Directions

Closely related fields

- ▶ Noisy optimisation, surrogate-assisted (approximated) optimisation, robust optimisation
- ▶ Also related is multi-objective optimisation and co-evolution

Modified offline performance is only one performance measure

- ▶ Solution similarity
 - ▶ For example, drivers prefer similar/identical routes for delivery
- ▶ Feasibility driven
 - ▶ Ensure you always have a feasible solution at all times
- ▶ Robustness
 - ▶ Small changes: Do not adapt; large changes: Adapt



Solutions have values

- ▶ (16, 16, 18)
- ▶ (24, 14, 25)
- ▶ (24, 21, 26)
- ▶ First solution not always best, but reliable

Multi-Populations: Introduction

Almost all EAs use a single population

Any individual may reproduce with any other

Population may be seen as one **species**

In nature, one species may split into two

Geographical conditions (e.g., distance) may restrict mate selection

Individual groups evolve in different environments (Galapagos islands)

Impact of multiple species (populations):

Different niches are “explored” (and more quickly)

Even if sub-populations converge, there is still diversity across populations

Parallel implementation

Each population can have its own CPU/core

Multi-Populations: The Island Model

Components of the island model

- ▶ Overall population P divided into N sub-populations
 - ▶ $P = \{P_1, P_2, \dots, P_N\}$
- ▶ Each sub-population P_i is of size μ_i

Use “standard” GA for each sub-population

- ▶ Evolve population i for G_i generations
- ▶ Have well-defined means of interaction

Communication Topology

- ▶ Sub-populations may interact with one another after each **epoch**
- ▶ Need to decide **magnitude** and **frequency** of interaction
- ▶ Determines isolation of each sub-population
 - ▶ Fully connected graph versus edgeless graph

Need to strike a good balance between:

- ▶ Isolated exploitation and collaborative exploration

The Island Model: Pseudo-code

Initialise and evaluate all $P(i)$

```
while terminate  $\neq$  true do  
  Evolve each  $P_i$  for  $G_i$  generations  
  for  $i = 1, 2, \dots, N$  do  
    for all neighbours  $j$  of  $i$  do  
       $migration(P_i, P_j)$   
    end for  
     $assimilate(P_i)$   
  end for  
end while
```

Migration

- ▶ Individuals are **moved** or **copied** to neighbouring populations

Assimilation

- ▶ Depends on model of migration and could include population size reduction

More in *Handbook of Evolutionary Computation*, section C.6.3.3

Multi-Populations: Other Algorithms

Many variations of these schemes:

- ▶ Dual Population Genetic Algorithm
- ▶ Forking Genetic Algorithm
- ▶ Self-organising Scouts (adaptation of FGA for DOPs)

Pros and Cons

- + Can be implemented in parallel
- + Helps maintain diversity
- + Can locate multiple optima simultaneously
- + Exploration versus exploitation
- Computationally expensive
- Parallel hardware often unavailable
- Numerous additional parameters require initialisation
 - ▶ Difficulty in choosing a model of migration

Student Projects

Field of evolutionary dynamic optimisation is relatively young

- ▶ Plenty of scope for new results

There are numerous possible student projects

- ▶ New problem benchmark generators
- ▶ Design of novel EAs for DOPs
 - ▶ Especially for dynamic networks / dynamic graphs
- ▶ Advance our understanding of dynamic optimisation
 - ▶ Analytical and empirical studies

For more details, see

- ▶ `www.cs.bham.ac.uk/~pZR/student-projects.html`

Dynamic Optimisation: Essential Reading

Jin, Y. & Branke, J. *Evolutionary Optimization in Uncertain Environment - A Survey* IEEE Transactions on Evolutionary Computation, 2005, 9, 303-317

Rohlfshagen, P. & Yao, X. *Dynamic Combinatorial Optimisation Problems: An Analysis of the Subset Sum Problem* To appear in: Soft Computing (Special issue on Evolutionary Optimization and Learning) 2009

Richter, H. *Detecting change in dynamic fitness landscapes* Proc. IEEE Congress on Evolutionary Computation, IEEE CEC 2009, (Ed. A. Tyrrell), IEEE Press, Piscataway, NJ, 1613-1620.

M-Level Bosman, P.A.N. *Learning, Anticipation and Time-Deception in Evolutionary Online Dynamic Optimization* Evolutionary Algorithms for Dynamic Optimization Problems EvoDOP Workshop at the Genetic and Evolutionary Computation Conference - GECCO-2005, pages 39-47, ACM Press, New York, New York, 2005.

Exercise

Non-assessed exercise for all to try

Adapt (improve) provided Java code

- ▶ Standard generational GA
- ▶ 2 functions: oneMax and twoMax
- ▶ Dynamic Framework: XOR

Implementations of Hypermutations and Random Immigrants

- ▶ Test these methods and improve them
- ▶ Implement and test more difficult binary functions

Some ideas

- ▶ Generate non-uniformly random immigrants (See Shengxiang Yang)
- ▶ Dynamically adjust hyper-mutation rate, etc.
- ▶ Develop your own technique and test and compare with the others