



UiO : Department of Informatics  
University of Oslo

**INF3490 - Biologically inspired computing**  
Lecture 4: Eiben and Smith,  
Working with evolutionary algorithms (chpt 9)  
Hybrid algorithms (chpt 10)  
Multi-objective optimization (chpt 12)

Jim Tørresen

UiO : Department of Informatics  
University of Oslo

**Chapter 9:  
Working with Evolutionary Algorithms**

1. Experiment design
2. Algorithm design
3. Test problems
4. Measurements and statistics
5. Some tips and summary

2

UiO : Department of Informatics  
University of Oslo

**Experimentation**

- Has a **goal** or goals
- Involves **algorithm** design and implementation
- Needs **problem(s)** to run the algorithm(s) on
- Amounts to **running** the algorithm(s) on the problem(s)
- Delivers **measurement data**, the results
- Is concluded with **evaluating** the results in the light of the given goal(s)
- Is often **documented** (thesis, papers, web,...)

3

UiO : Department of Informatics  
University of Oslo

**Experimentation:  
Goals for Research**

- Show that EC is applicable in a **(new) problem domain** (real-world applications)
- Show that *my\_EA* is **better than** *benchmark\_EA*
- Show that EAs outperform **traditional** algorithms
- Optimize or study **impact of parameters** on the performance of an EA
- Investigate **algorithm behavior** (e.g. interaction between selection and variation)
- See how an EA **scales-up** with problem size
- ...

4

## Example: Repetitive Problems

- Optimising Internet shopping delivery route
  - Need to **run regularly/repetitively**
  - Different destinations each day
  - **Limited time** to run algorithm each day
  - **Must *always* be *reasonably* good route in limited time**



5

## Example: Design Problems

- Optimising spending on improvements to national road network
  - Total cost: billions of Euro
  - Computing **costs negligible**
  - Six months to run algorithm on hundreds computers
  - Many runs possible
  - **Must produce *very* good result just *once***



6

## Algorithm design

- Design a representation
- Design a way of mapping a genotype to a phenotype
- Design a way of evaluating an individual
- Design suitable mutation operator(s)
- Design suitable recombination operator(s)
- Decide how to select individuals to be parents
- Decide how to select individuals for the next generation (how to manage the population)
- Decide how to start: initialization method
- Decide how to stop: termination criterion

7

## Test problems

1. Recognized **benchmark problem** repository (typically “challenging”)
2. Problem instances made by **random generator**
3. Frequently encountered or otherwise important variants of given **real-world problems**

Choice has severe implications on:

- generalizability and
- scope of the results

8

## Getting Problem Instances (1/3) Benchmarks

- Standard data sets in problem **repositories**, e.g.:
  - OR-Library  
[www.brunel.ac.uk/~mastijb/jeb/info.html](http://www.brunel.ac.uk/~mastijb/jeb/info.html)
  - UCI Machine Learning Repository  
[www.ics.uci.edu/~mlearn/MLRepository.html](http://www.ics.uci.edu/~mlearn/MLRepository.html)
- Advantage:
  - Well-chosen problems and instances (hopefully)
  - Much other work on these → results comparable
- Disadvantage:
  - Not real – might miss crucial aspect
  - Algorithms get tuned for popular test suites

9

## Getting Problem Instances (2/3) Problem instance generators

- **Problem instance generators** produce simulated data for given parameters, e.g.:
  - GA/EA Repository of Test Problem Generators  
<http://visicad.eecs.umich.edu/BK/Slots/cache/www.cs.uwo.edu/~wspears/generators.html>
- Advantage:
  - Allow very systematic comparisons for them
    - can produce many instances with the same characteristics
    - enable gradual traversal of a range of characteristics (hardness)
  - Can be shared allowing comparisons with other researchers
- Disadvantage
  - Not real – might miss crucial aspect
  - Given generator might have hidden bias

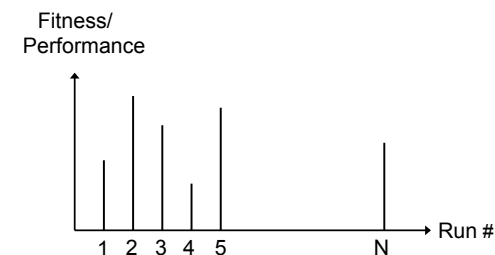
10

## Getting Problem Instances (3/3) Real-world problems

- Testing on (own collected) **real data**
- Advantages:
  - Results could be considered as very relevant viewed from the application domain (data supplier)
- Disadvantages
  - Can be over-complicated
  - Can be few available sets of real data
  - May be commercial sensitive – difficult to publish and to allow others to compare
  - Results are hard to generalize

11

## Typical Results from Several EA Runs



12

## Basic rules of experimentation

- **EAs are stochastic →**  
**never draw any conclusion from a single run**
  - perform sufficient number of independent runs
  - use statistical measures (averages, standard deviations)
  - use statistical tests to assess reliability of conclusions
- **EA experimentation is about comparison →**  
**always do a fair competition**
  - use the same amount of resources for the competitors
  - try different comp. limits (to cope with turtle/hare effect)
  - use the same performance measures

13

## Things to Measure

Many different ways. Examples:

- Average result in given time
- Average time for given result
- Proportion of runs within % of target
- Best result over  $n$  runs
- Amount of computing required to reach target in given time with % confidence
- ...

14

## What time units do we use?

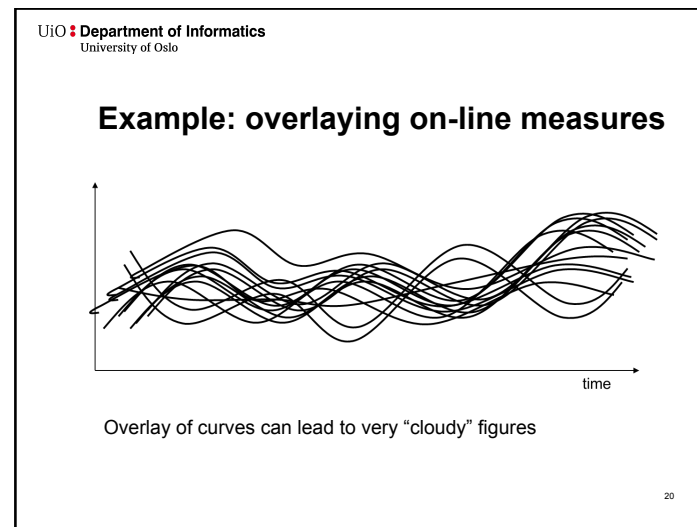
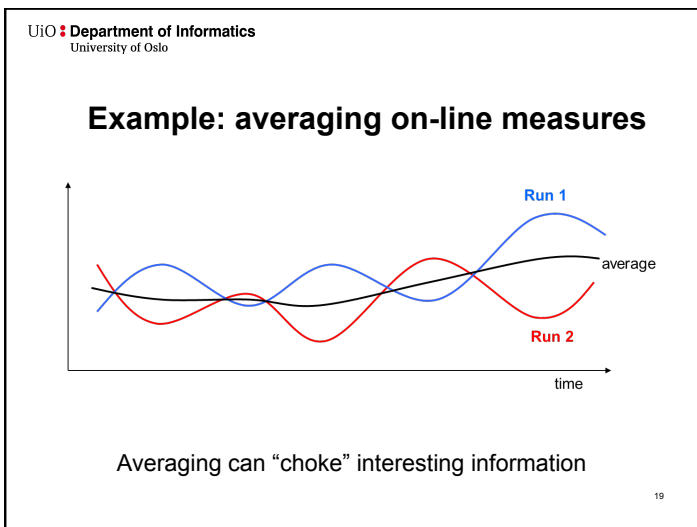
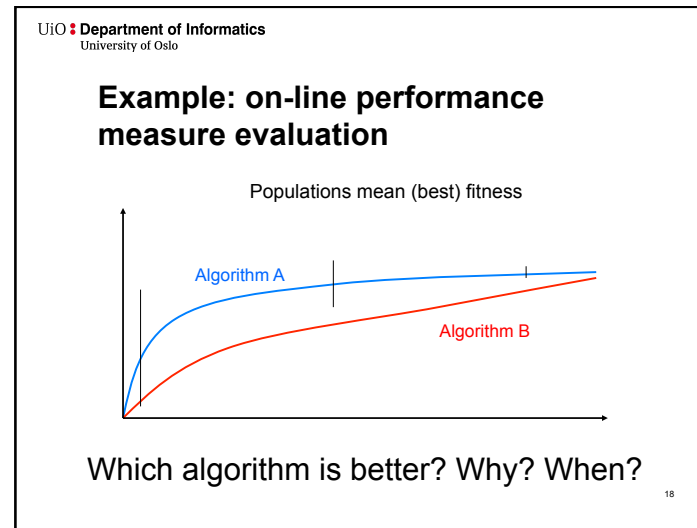
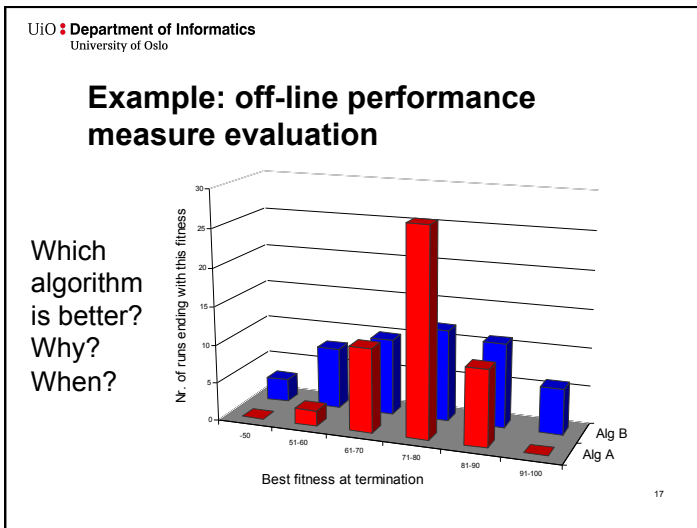
- **Elapsed time?**
  - Depends on computer, network, etc...
- **CPU Time?**
  - Depends on skill of programmer, implementation, etc...
- **Generations?**
  - Incomparable when parameters like population size change
- **Evaluations?**
  - Evaluation time could depend on algorithm, e.g. direct vs. indirect representation
  - Evaluation time could be small compared to other steps in the EA (e.g. genotype to phenotype translation)

15

## Measures

- **Performance measures (off-line)**
  - **Efficiency** (alg. speed, also called performance)
    - Execution time
    - Average no. of evaluations to solution (AES, i.e., number of generated points in the search space)
  - **Effectiveness** (solution quality, also called accuracy)
    - Success rate (SR): % of runs finding a solution
    - Mean best fitness at termination (MBF)
- **“Working” measures (on-line)**
  - Population distribution (genotypic)
  - Fitness distribution (phenotypic)
  - Improvements per time unit or per genetic operator
  - ...

16



## Statistical Comparisons and Significance



- Algorithms are stochastic, results have element of “luck”
- If a claim is made “Mutation A is better than mutation B”, need to show **statistical significance** of comparisons
- Fundamental problem: two series of samples (random drawings) from the SAME distribution may have DIFFERENT averages and standard deviations
- Tests can show if the differences are significant or not

21

## Example

Trial	Old Method	New Method
1	500	657
2	600	543
3	556	654
4	573	565
5	420	654
6	590	712
7	700	456
8	472	564
9	534	675
10	512	643
Average	545.7	612.3

Is the new method better?

22

## Example (cont'd)

Trial	Old Method	New Method
1	500	657
2	600	543
3	556	654
4	573	565
5	420	654
6	590	712
7	700	456
8	472	564
9	534	675
10	512	643
Average	545.7	612.3
SD	73.5962635	73.5473317
T-test	<b>0.07080798</b>	

- Standard deviations supply additional info
- T-test (and alike) indicate the chance that the values came from the same underlying distribution (difference is due to random effects) E.g. with 7% chance in this example.

23

## Summary of tips for experiments

- **Be organized**
- Decide what you want & define **appropriate measures**
- Choose **test problems** carefully
- Make an **experiment plan** (estimate time when possible)
- Perform sufficient number of runs
- Keep all experimental data (never throw away anything)
- Include in publications all necessary parameters to make **others able to repeat** your experiments
- Use **good statistics** (“standard” tools from Web, MS, R)
- Present results well (figures, graphs, tables, ...)
- Watch the **scope** of your claims
- Aim at **generalizable** results (use separate data set for training and testing)
- **Publish code** for reproducibility of results (if applicable)
- **Publish data** for external validation (open science)

24

## Chapter 10: Hybridisation with Other Techniques: Memetic Algorithms

1. Why to Hybridise
2. What is a Memetic Algorithm?
3. Where to hybridise
4. Local Search
  - Lamarckian vs. Baldwinian adaptation

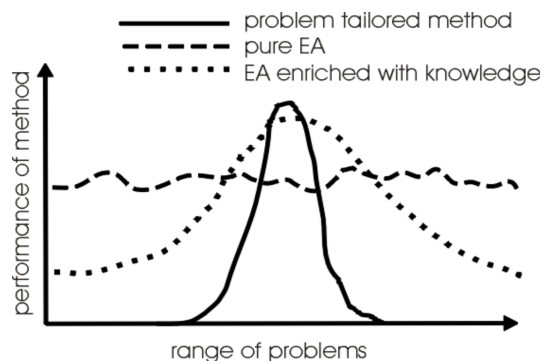
25

## 1. Why Hybridise

- Might be looking at **improving on existing techniques** (non-EA)
- Might be looking at **improving EA search** for good solutions

26

## 1. Why Hybridise Michalewicz's view on EAs in context



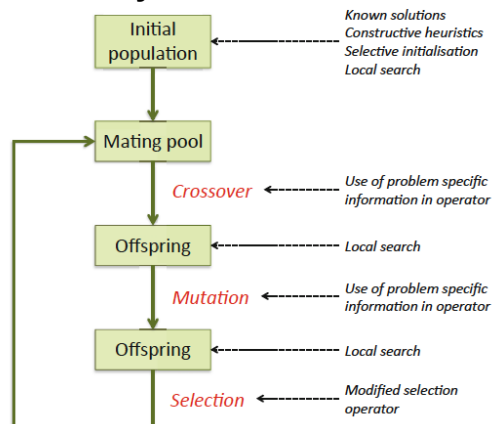
27

## 2. What is a Memetic Algorithm?

- The combination of Evolutionary Algorithms with **Local Search Operators** that work within the EA loop has been termed "**Memetic Algorithms**"
- Term also applies to EAs that use **instance-specific knowledge**
- Memetic Algorithms have been shown to be orders of magnitude **faster and more accurate** than EAs on some problems, and are the "state of the art" on many problems

28

### 3. Where to Hybridise:



29

### 3. Where to Hybridise: In initialization

- Seeding
  - Known good solutions are added
- Selective initialization
  - Generate  $kN$  solutions, keep best  $N$
- Refined start
  - Perform local search on initial population

30

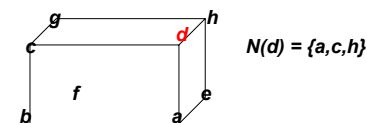
### 3. Where to Hybridise: Intelligent mutation and crossover

- Mutation bias
  - Mutation operator has bias towards certain changes
- Crossover hill-climber
  - Test all 1-point crossover results, choose best
- “Repair” mutation
  - Use heuristic to make infeasible solution feasible

31

### 4. Local Search: Local Search

- Defined by combination of *neighbourhood* and *pivot rule*
- Related to landscape metaphor
- $N(x)$  is defined as the set of points that can be reached from  $x$  with one application of a move operator
  - e.g. bit flipping search on binary problems



32



#### 4. Local Search: Pivot Rules

- Is the neighbourhood searched randomly, systematically or exhaustively ?
- does the search stop as soon as a fitter neighbour is found (**Greedy Ascent**)
- or is the whole set of neighbours examined and the best chosen (**Steepest Ascent**)
- of course there is no one best answer, but some are quicker than others to run .....

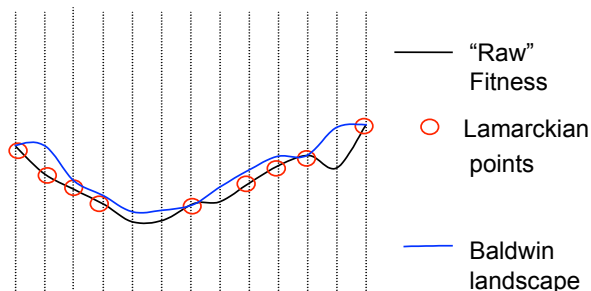
33

#### 4. Local Search and Evolution

- Do offspring inherit what their parents have learnt in life?
  - Yes - Lamarckian learning
    - Improved fitness and genotype
  - No - Baldwinian learning:
    - Improved fitness only

34

#### 4. Local Search: Induced landscapes



35

#### Hybrid Algorithms Summary

- It is **common practice to hybridise EA's** when using them in a real world context.
- This may involve the use of operators from other algorithms which have already been used on the problem, or the incorporation of domain-specific knowledge
- Memetic algorithms have been shown to be orders of magnitude faster and more accurate than EAs on some problems, and are the "state of the art" on many problems

36

## Chapter 12: Multiobjective Evolutionary Algorithms

- Multiobjective optimisation problems (MOP)
  - Pareto optimality
- EC approaches
  - Evolutionary spaces
  - Preserving diversity

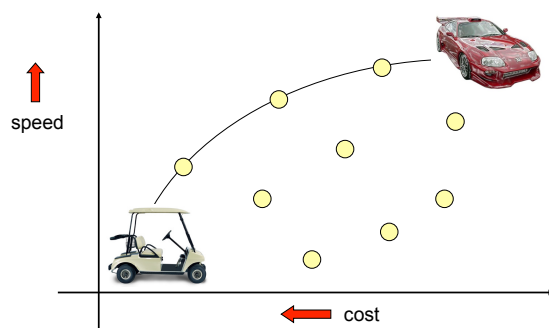
37

## Multi-Objective Problems (MOPs)

- Wide range of problems can be categorised by the presence of a number of ***n* possibly conflicting objectives**:
  - buying a car: speed vs. price vs. reliability
  - engineering design: lightness vs. strength
- Two problems:
  - finding set of good solutions
  - choice of best for the particular application

38

## An example: Buying a car



39

## Two approaches to multiobjective optimisation

- **Weighted sum (scalarisation)**:
  - transform into a **single objective** optimisation method
  - compute a weighted sum of the different objectives
- **A set of multi-objective solutions (Pareto front)**:
  - The **population-based** nature of EAs used to ***simultaneously*** search for a set of points approximating Pareto front

40

UiO Department of Informatics  
University of Oslo

### Comparing solutions

**Objective space**

- Optimisation task: Minimize both  $f_1$  and  $f_2$
- Then:
  - a is better than b
  - a is better than c
  - a is worse than e
  - a and d are incomparable

41

UiO Department of Informatics  
University of Oslo

### Dominance relation

- Solution  $x$  dominates solution  $y$ , ( $x \preceq y$ ), if:
  - $x$  is better than  $y$  in at least one objective,
  - $x$  is not worse than  $y$  in all other objectives

42

UiO Department of Informatics  
University of Oslo

### Pareto optimality

- Solution  $x$  is **non-dominated** among a set of solutions  $Q$  if no solution from  $Q$  dominates  $x$
- A set of non-dominated solutions from the entire feasible solution space is the **Pareto-optimal set**, its members Pareto-optimal solutions
- **Pareto-optimal front**: an image of the Pareto-optimal set in the objective space

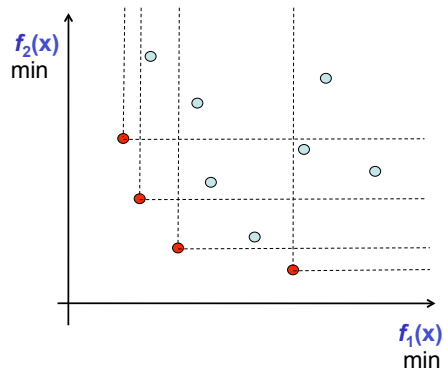
43

UiO Department of Informatics  
University of Oslo

### Illustration of the concepts

44

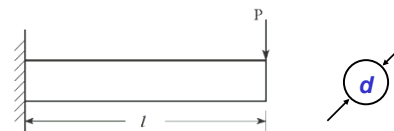
## Illustration of the concepts



45

A practical example:  
The beam design problem

Minimize weight and deflection of a beam (Deb, 2001):



46

## Formal definition

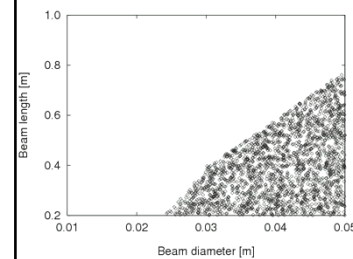
- Minimize  $f_1(d, l) = \rho \frac{\pi d^2}{4} l$  (beam weight)
- minimize  $f_2(d, l) = \delta = \frac{64 Pl^3}{3E\pi d^4}$  (beam deflection)
- subject to  $0.01 \text{ m} \leq d \leq 0.05 \text{ m}$   
 $0.2 \text{ m} \leq l \leq 1.0 \text{ m}$   
 $\sigma_{\max} = \frac{32 Pl}{\pi d^3} \leq S_y$  (maximum stress)  
 $\delta \leq \delta_{\max}$

where  $\rho = 7800 \text{ kg/m}^3$ ,  $P = 2 \text{ kN}$   
 $E = 207 \text{ GPa}$   
 $S_y = 300 \text{ MPa}$ ,  $\delta_{\max} = 0.005 \text{ m}$

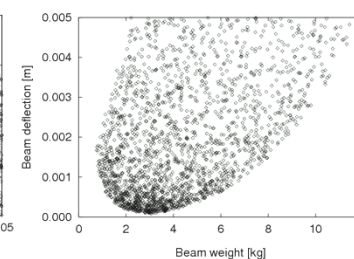
47

## Feasible solutions

Decision (variable) space

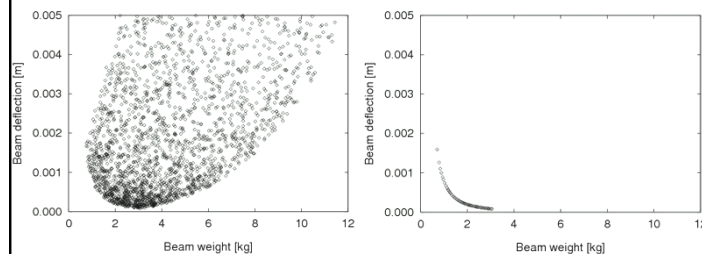


Objective space



48

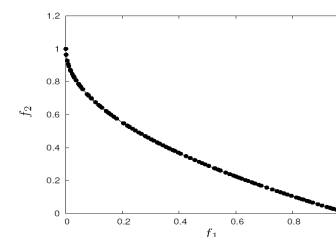
## Goal: Finding non-dominated solutions



49

## Goal of multiobjective optimisers

- Find a set of non-dominated solutions (**approximation set**) following the criteria of:
  - **convergence** (as close as possible to the Pareto-optimal front),
  - **diversity** (spread, distribution)



50

## EC approach: Requirements

1. Way of **assigning fitness**,
  - usually based on dominance
2. Preservation of a **diverse set of points**
  - similarities to multi-modal problems
3. Remembering all the **non-dominated points** you have seen
  - usually using elitism or an archive

51

## EC approach: 1. Fitness Assignment

- Could use aggregating approach and change weights during evolution
  - no guarantees
- Different parts of population use different criteria
  - no guarantee of diversity
- Dominance (made a breakthrough for MOEA)
  - ranking or depth based
  - fitness related to whole population

52

## EC approach: 2. Diversity maintenance

- Usually done by niching techniques such as:
  - fitness sharing
  - adding amount to fitness based on inverse distance to nearest neighbour (minimisation)
  - (adaptively) dividing search space into boxes and counting occupancy
- All rely on some distance metric in genotype / phenotype space

53

## EC approach: 3. Remembering Good Points

- Could just use elitist algorithm, e.g.
  - $(\mu + \lambda)$  replacement
  - crowding distance
- Common to maintain an archive of non-dominated points
  - some algorithms use this as a second population that can be in recombination etc.
  - others divide archive into regions too, e.g. PAES

54

## Multi objective problems - Summary

- MO problems occur very frequently
- EAs are very good in solving MO problems
- MOEAs are one of the most successful EC subareas

55