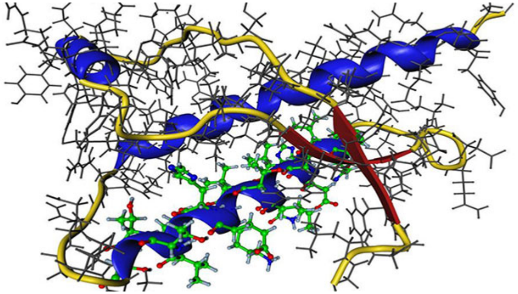
NTNU

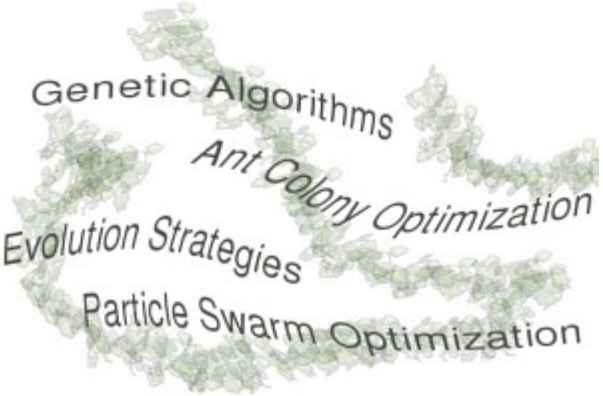
Norwegian University of Science and Technology



Lecture 3

Working with EA

Håken Jevne,
Kazi Ripon and Pauline Haddow



Outline

- No free lunch
  - Problem to be solved?
- Experimental Design and Parameter Tuning
  - Example of Parameter Tuning
  - Dynamic Parameter Tuning
- Expectation to your EA
- Performance Measures
- Rules of fair experimentation
- Exploration vs Exploitation

Outline

- No free lunch
  - Problem to be solved?
- Experimental Design and Parameter Tuning
  - Example of Parameter Tuning
  - Dynamic Parameter Tuning
- Expectation to your EA
- Performance Measures
- Rules of fair experimentation
- Exploration vs Exploitation

No Free Lunch Theorems

- No free lunch (NFL) Theorems apply to EC algorithms.
  - Theorems imply there can be no universally efficient EC algorithm.
  - Performance of one algorithm when averaged over all problems is identical to that of any other algorithm.
- IN LAYMAN'S TERMS,
  - Averaged over all problems.
  - For any performance metric related to number of distinct points seen.
  - All non-revisiting black-box algorithms will display the same performance.

No free lunch – but better?

- *Better?*
  - *What do you want your algorithm to solve?*
  - *What performance do you need?*
  - *How can I tune my algorithm to*
    - *be more efficient?*
    - *reach more optimal solution?*
  - *How much Application knowledge do I need to add?*

Outline

- No free lunch
  - Problem to be solved?
- Experimental Design and Parameter Tuning
  - Example of Parameter Tuning
  - Dynamic Parameter Tuning
- Expectation to your EA
- Performance Measures
- Rules of fair experimentation
- Exploration vs Exploitation

Motivation: working with parameters

An EA has many strategy parameters, e.g.

- mutation operator and mutation rate
- crossover operator and crossover rate
- selection mechanism and selective pressure (e.g. tournament size)
- population size

Good parameter values facilitate good performance

How to find good parameter values ?

Going Deep: working with parameters

EA parameters are rigid (constant during a run)

BUT

an EA is a dynamic, adaptive process

THUS

optimal parameter values may vary during a run

How to vary parameter values?

Working with Parameters

- Parameter tuning.
- Parameter control.

Parameter Tuning

- Traditional way of testing and comparing different values before the "*real*" run

- Problems:
  - Users mistakes in settings can be sources of errors or sub-optimal performance.
  - Costs much time.
  - Parameters interact: exhaustive search is not practicable.
  - Good values may become bad during the run.

Parameter Control

- Setting values on-line, during the actual run, e.g.
  - Predetermined time-varying schedule $p = p(t)$
  - Using feedback from the search process
  - Encoding parameters in chromosomes and rely on natural selection

- Problems:
  - Finding optimal p is hard, finding optimal $p(t)$ is harder
  - Still user-defined feedback mechanism, how to ``optimize"?
  - When would natural selection work for strategy parameters?

Parameter Control: Example

Task to solve:

- min $f(x_1,\ldots,x_n)$
- $L_i \le x_i \le U_i$ for $i = 1,\ldots,n$ bounds
- $g_i(x) \le 0$ for $i = 1,\ldots,q$ inequality constraints
- $h_i(x) = 0$ for $i = q+1,\ldots,m$ equality constraints

Algorithm:

- EA with real-valued representation $(x_1,\ldots,x_n)$
- arithmetic averaging crossover
- Gaussian mutation: $x'_i = x_i + N(0, \sigma)$

  standard deviation σ is called mutation step size

Example: option-1

- Replace the constant σ by a function $\sigma(t)$

$$\sigma(t) = 1 - 0.9 \times \frac{t}{T}$$

$0 \leq t \leq T$ is the current generation number

- Features:
  - changes in σ are independent from the search progress
  - strong user control of σ by the above formula
  - σ is fully predictable
  - a given σ acts on all individuals of the population

Example: option-2

- Replace the constant σ by a function $\sigma(t)$ updated after every n steps by the 1/5 success rule (ES):

$$\sigma(t) = \begin{cases} \sigma(t-n)/c & \text{if } p_s > 1/5 \\ \sigma(t-n)\cdot c & \text{if } p_s < 1/5 \\ \sigma(t-n) & \text{otherwise} \end{cases}$$

- Features:
  - changes in σ are based on feedback from the search progress
  - some user control of σ by the above formula
  - σ is not predictable
  - a given σ acts on all individuals of the population

Example: option-3

- Assign a personal σ to each individual
- Incorporate this σ into the chromosome: (x_1, …, x_n, σ)
- Apply variation operators to x_i's and σ

$$\sigma' = \sigma \times e^{N(0,\tau)}$$

$$x'_i = x_i + N(0,\sigma')$$

- Features:
  - changes in σ are results of natural selection
  - (almost) no user control of σ
  - σ is not predictable
  - a given σ acts on one individual

Classification of Control Techniques

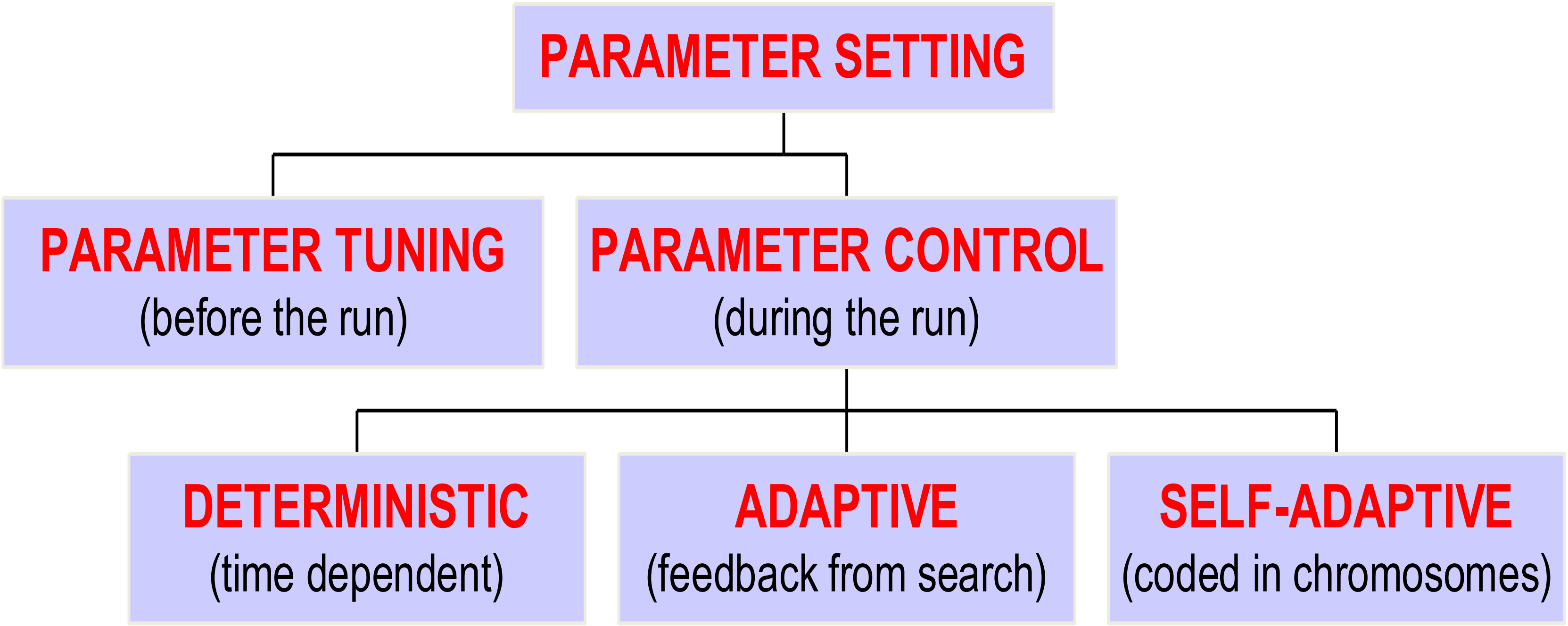
Various forms of parameter control can be distinguished by:

- primary features:
  - what component of the EA is changed.
  - how the change is made.
- secondary features:
  - evidence/data backing up changes.
  - level/scope of change.

What is Changed?

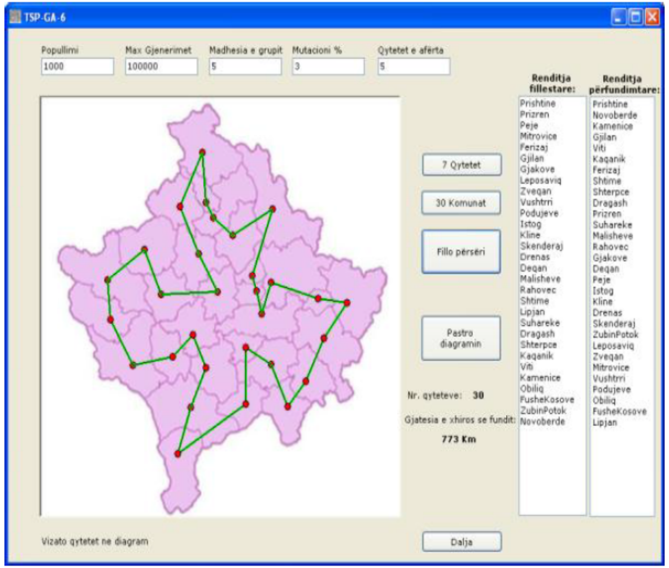
- Practically any EA component can be parameterized and thus controlled on-the-fly:
  - representation
  - evaluation function
  - variation operators
  - selection operator (parent or mating selection)
  - replacement operator (survival or environmental selection)
  - population (size, topology)

How are Changes Made?



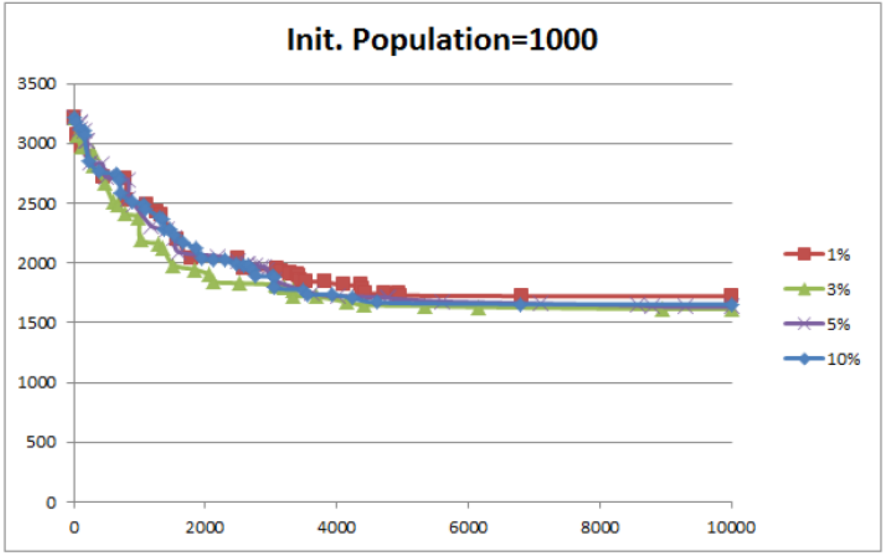
Example Parameter Tuning for TSP

- TSP for Kosovo municipalities.
- Genetic Algorithm
- Parameters tuned:
  1. size of initial population.
  2. mutation probability.
  3. number of generations.



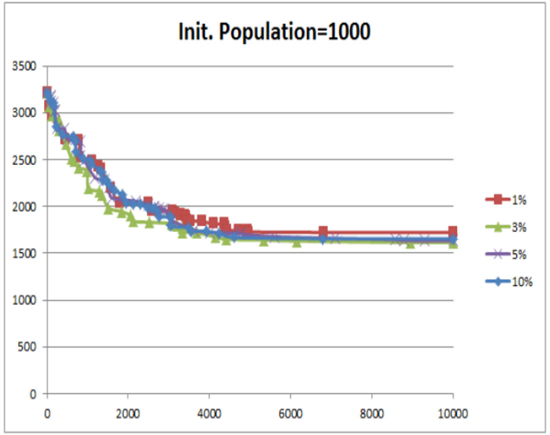
Varying Mutation Rate

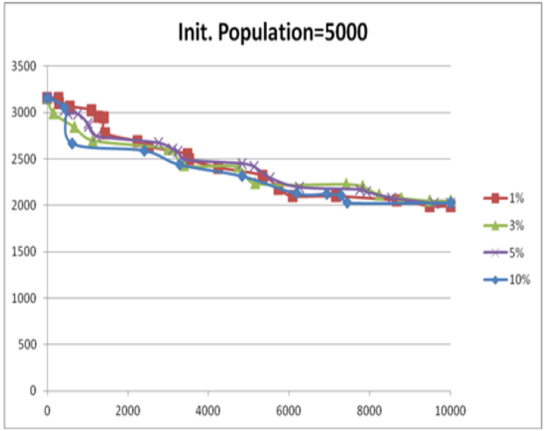
- # generations fixed **10,000**.
- Population fixed 1000
- Vary mutation: :
  - 1%, 3%, 5% and 10%

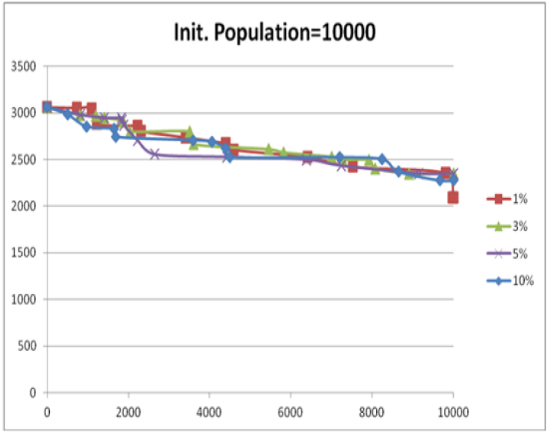


Varying Mutation Rate, Changing Population

- Fixed maximal number of generations to 10,000.
- Vary Mutation rate
  - 1%, 3%, 5% and 10%
- Vary population:
  - 1000, 5000 and 10000.

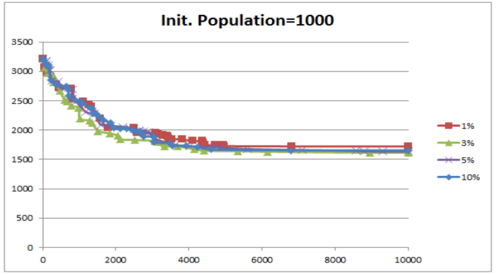


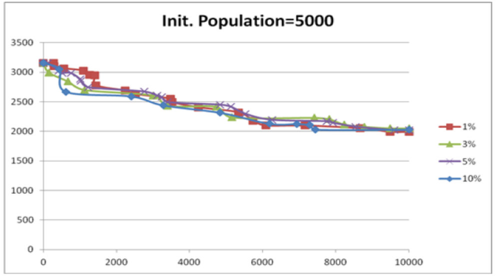


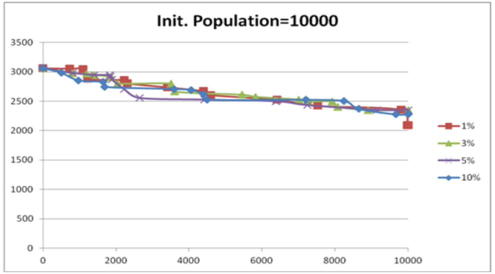


Different Initial Population, with Different mutation probability, for Different generation

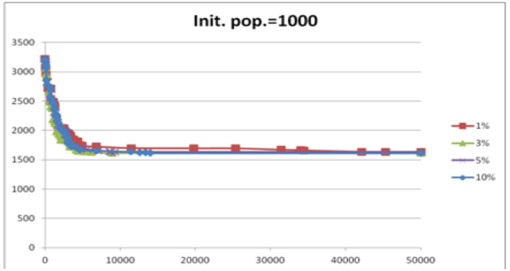
10,000 generations

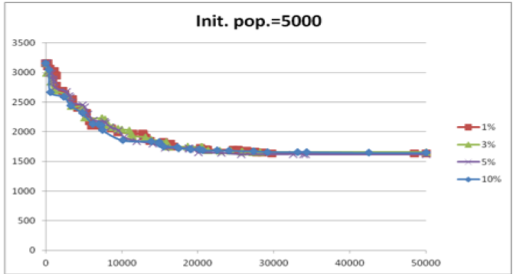


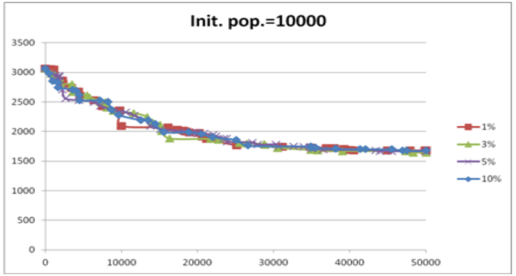




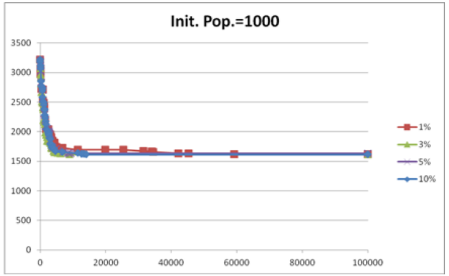
50,000 generations

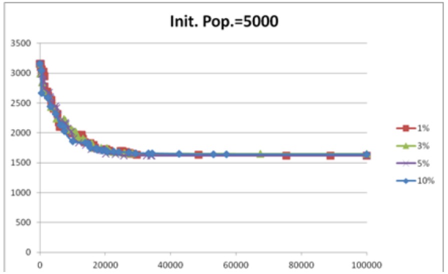


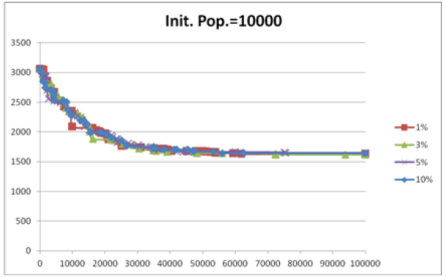




100,000 generations



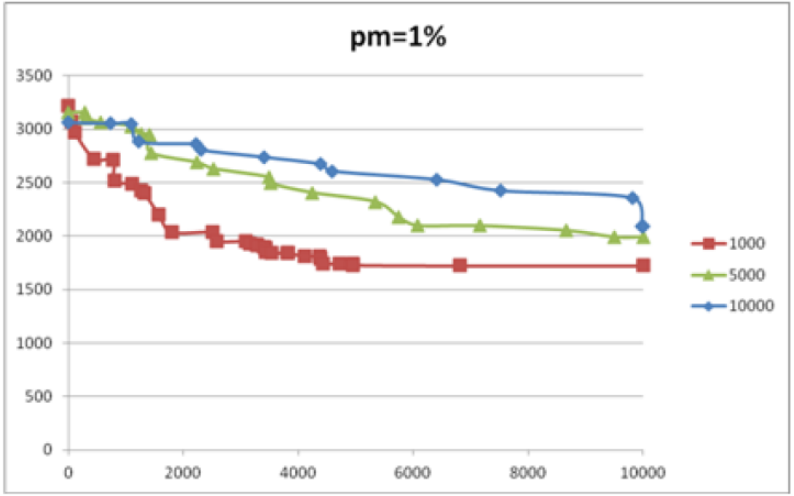




NTNU

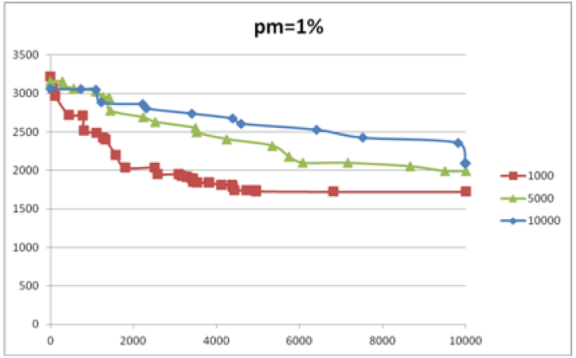
Varying Population, Fixed Mutation Rate

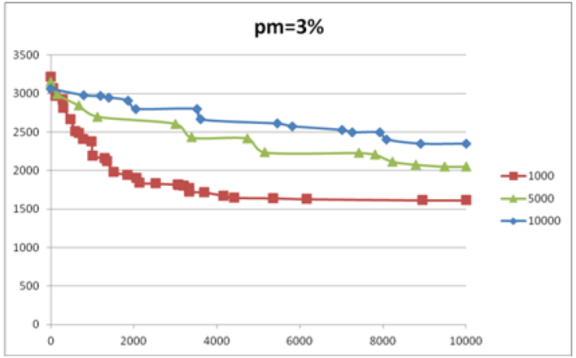
- Fixed # generations 10,000
- Vary Population :
  - 1000, 5000, 10000
- mutation probability (pm):
  - 1%

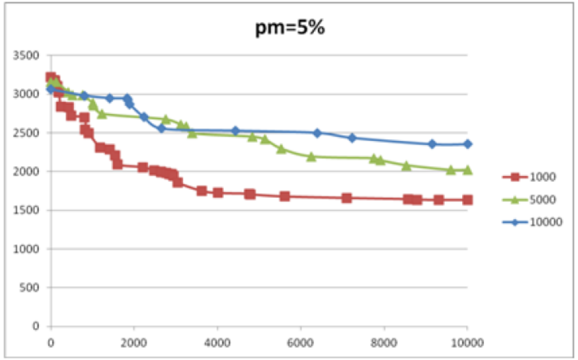


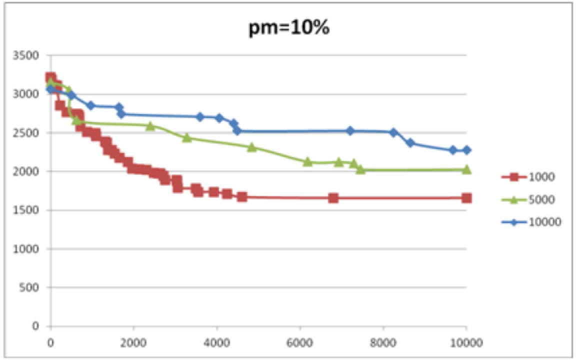
Varying Population, Fixed Mutation Rate

Fixed # generations 10,000,
Vary population: 1000, 5000, 10000 vary mutation rate (pm) : 1,3,5,10





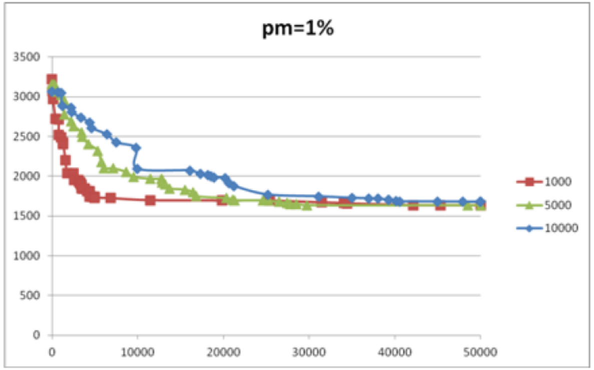


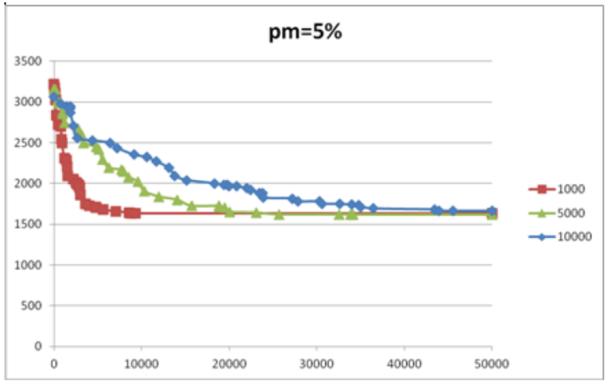


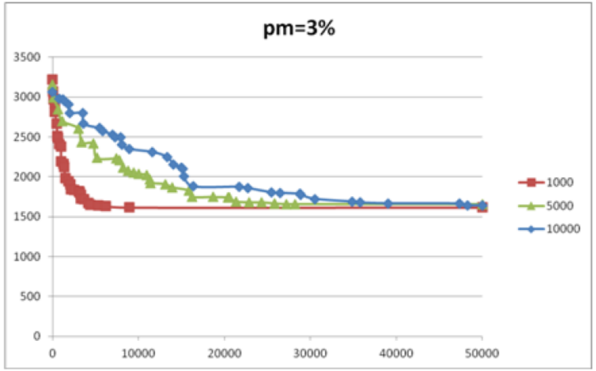
Varying Population, Fixed Mut. # Gen. 50k

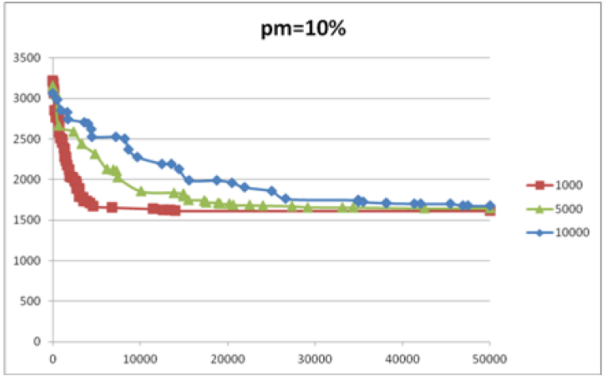
Fixed # generations 50,000!

Vary population: 1000, 5000, 10000 vary mutation rate (pm) : 1,3,5,10

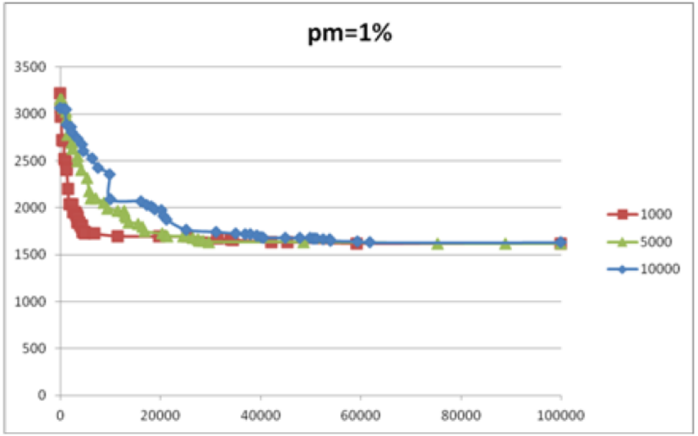


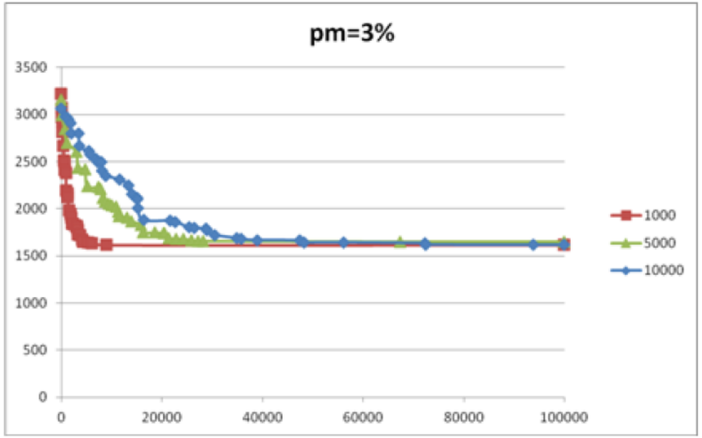




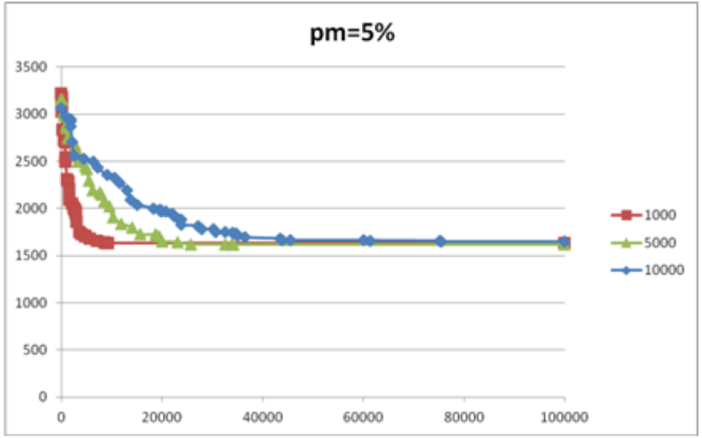


Varying Population, Fixed Mut. # Gen 100k





Fixed # generations 100,000!
Vary population: 1000, 5000, 10000
Vary mutation rate (pm) : 1,3,5,10



About Probabilities...

- General rule of thumb:
  - Average probability for individual to crossover: ~ 80%.
  - Average probability for individual to mutate: 1-2%.
- Probability of genetic operators follow the probability in natural systems
- Better solutions reproduce more often

Get the Balance Right

- Population size
  - Popsize too small → premature convergence
  - Popsize too large → too slow to compute
- Mutation – upholds diversity
  - Mutation rate too low → not enough exploring
  - Mutation rate too high → too much noise
- Crossover – often effective
  - Late in the search: crossover has smaller effect
  - Selective choice of crossover point

NTNU

Setting Parameters: sensitivity study

- Trial and error
- Apply general rule of thumb
  - Exploration vs exploitation trade-off
- Sensitivity study:
  - Vary one parameter at a time
  - Study sets of parameters
    - Study effect on convergence / time

Outline

- No free lunch
  - Problem to be solved?
- Experimental Design and Parameter Tuning
  - Example of Parameter Tuning
  - Dynamic Parameter Tuning
- Expectation to your EA
- Performance Measures
- Rules of fair experimentation
- Exploration vs Exploitation

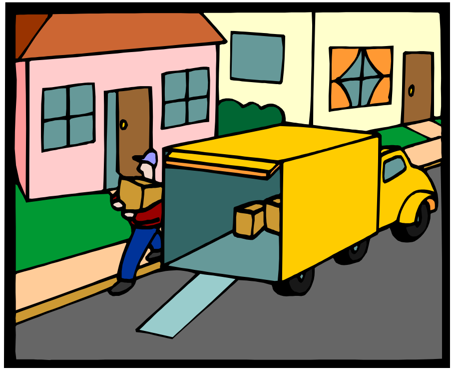
Best-ever, Worst-ever fitness

Design problems

- Best-ever fitness
- Looking for ONE '*excellent*' solution

Repetitive problems

- Worst–ever fitness
- requiring many '*good*' yet '*timely*' solutions.
- Includes on-line control problem as special case.



Design Problems

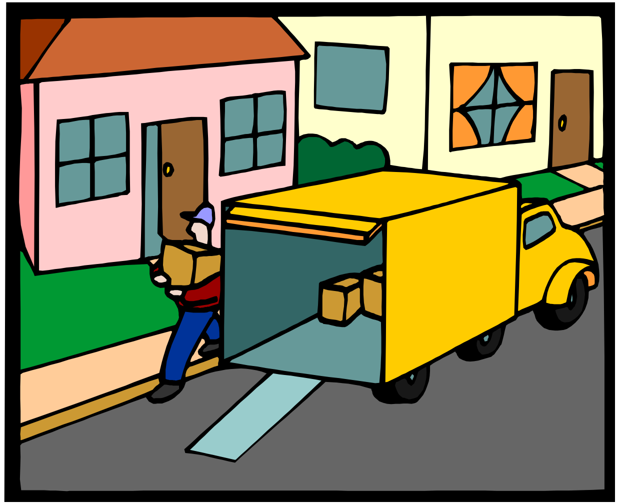
- Optimizing spending on improvements to national road network
  - Total cost: billions of Euro
  - Computing costs negligible
  - Six months to run algorithm on hundreds computers
  - Many runs possible
  - Must produce *very* good result just *once.*

Design Problems

- Quality is the most important.
  - Performance (speed) is secondary.
- Algorithm does not need to be fast.
  - It can run for several months of computing time
    - Performing several runs
    - Keeping the best result.
- Very specific.
  - No need to be generally applicable.

Repetitive Problems

- Optimizing Internet shopping delivery routes.
  - Different destinations each day.
  - Limited time to run algorithm each day.
  - Must *always* be *reasonably* good route in limited time.



NTNU

Repetitive Problems

- Solutions must be good (better than hand-made ones),
  - but *not optimal*.
- Speed is very crucial.
- Speed vs quality trade-off.
- It is important that the **performance is stable**.
- Applies repeatedly for *different instances* of the problem.
  - Wide applicability of the algorithm.

On-Line Control Problem

- Repetitive problem with extremely tight time constraints.
  - Traffic light optimization of a single crossing with four crossroads.

- Traffic light – GA controller
  - Streaming sensory information
  - One full cycle (turns to green)
    - Few minutes
    - Population of individuals, # generations → good result

Academic Research

- Different but important type of context.
- Not application oriented.

Measures - online

- Off-line measures
  - Efficiency and Effectivity measures

- “Working” measures (on-line)
  - Population distribution (genotypic)
  - Fitness distribution (phenotypic)
  - Improvements per time unit
  - Improvements per genetic operator
  - …

Outline

- No free lunch
  - Problem to be solved?
- Experimental Design and Parameter Tuning
  - Example of Parameter Tuning
  - Dynamic Parameter Tuning
- Expectation to your EA
- Performance Measures
- Rules of fair experimentation
- Exploration vs Exploitation

Algorithm Quality

- 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 competition limits
  - use the same **Performance Measures**

Things to Measure

- Many different ways:
  - 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
  - …

Performance Measures

- Efficiency (alg. speed)
  - CPU time
  - No. of steps, i.e., generated points in the search space
  - …….
- Effectivity (alg. quality)
  - Success rate
  - Solution quality at termination
  - …………

Performance Measures : Efficiency

- Algorithm Speed
- TIME
  - Elapsed time?
    - Depends on computer, network etc
  - CPU Time?
    - Depends on programmer skill, implementation…
  - # Generations?
    - Difficult to compare when parameters like population size change
    - Difficult to compare against non evolutionary results.
  - # Fitness Evaluations?
    - Evaluation time could depend on algorithm, e.g. direct vs. indirect representation

Performance Measures: Effectivity

- Algorithm quality
- Measured with fixed computation resources

Performance Measures: Effectivity

- Algorithm quality
- Measured with fixed computation resources

Success Rate (SR)

- % runs terminating in success
- Not always measurable, no known optimal
  - timetabling:
  - benchmark last years

Performance Measures: Effectivity

- Algorithm quality
- Measured with fixed computation resources

Success Rate (SR)

- % runs terminating in success
- Not always measurable, no known optimal
  - timetabling:
  - benchmark last years

Mean Best Fitness (MBF)

- Explicit fitness
- Best fitness each run i, F_i i :1..n
- ***MBF = AVG*** $\boldsymbol{\Sigma F_i}$
- Always valid measure

Performance Measures: Effectivity

- Algorithm quality
- Measured with fixed computation resources

Success Rate (SR)

- % runs terminating in success
- Not always measurable, no known optimal
  - timetabling:
  - benchmark last years

Mean Best Fitness (MBF)

- Explicit fitness
- Best fitness each run
- ***BF = AVG*** $\boldsymbol{\Sigma\, F_i}$
- Always valid measure

Average # of Evaluations to Solution (AES)

- AVG # fitness evaluation in solution over n runs
- counts AVG runs reaching **solution**

Combination of SR and MBF

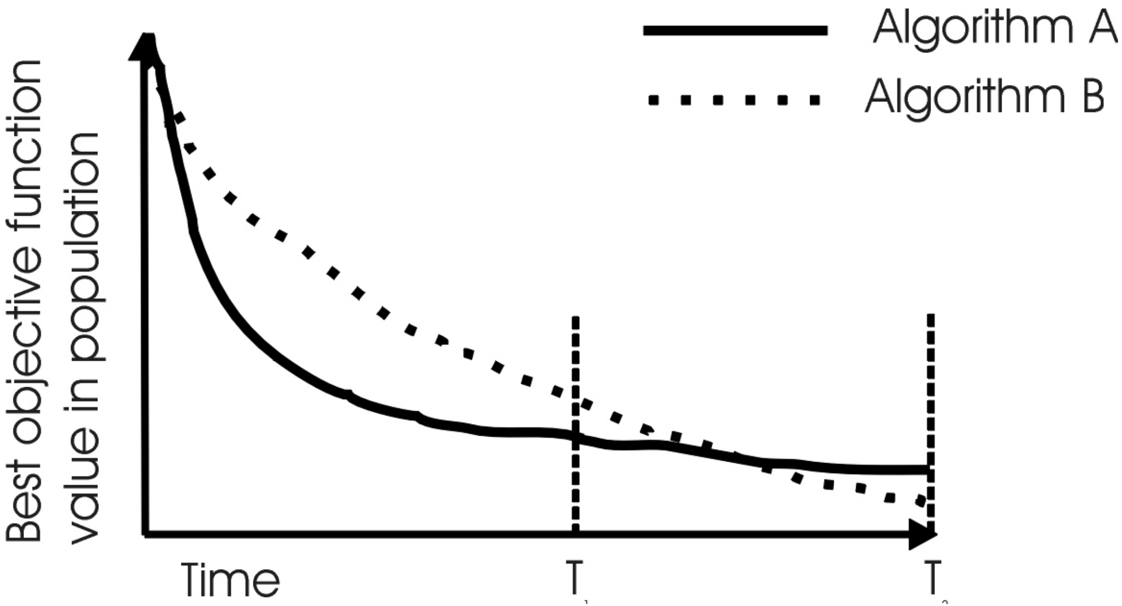
Good Approximiser

- ↓ SR+ ↑MBF
  - Try ↑ # generations → ↑SR ie allow search to finish
  - Optimal solution?
- Problem type
  - timetabling

‘Murphy’ algorithm

- ↑ SR+ ↓ MBF
  - When algorithm goes wrong, goes very wrong
- Problem Type
  - 3_SAT , # unsatisfied clauses as fitness

Performance Measures



Comparing algorithms A and B by after terminating at time T_1 and T_2 (for a minimization problem).

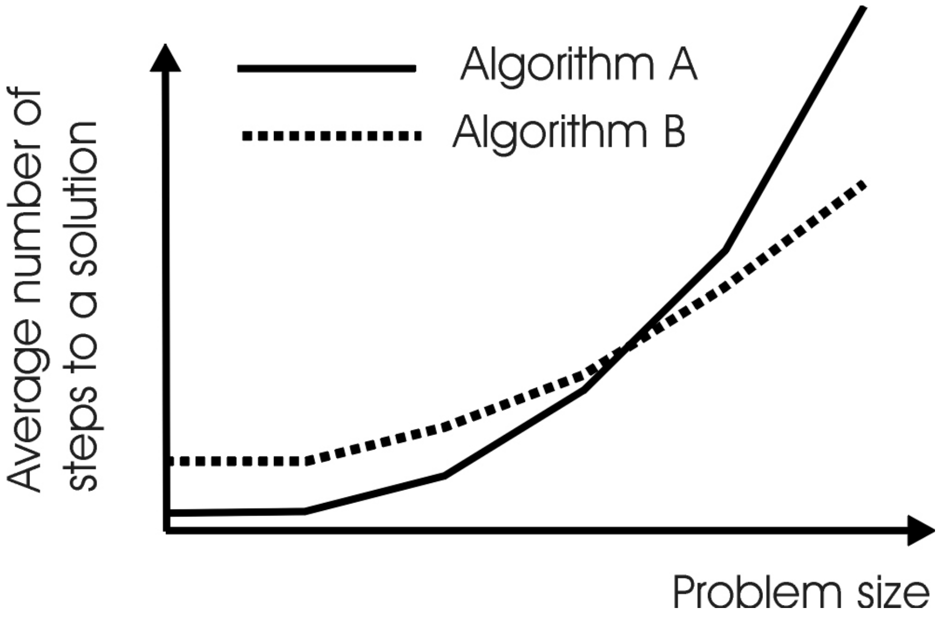
Fair Experiments

- **Basic rule: use the same computational limit for each competitor.**

- Allow each EA the same no. of evaluations, but
  - Beware of hidden labour, e.g. in heuristic mutation operators.
  - Beware of possibly fewer evaluations by smart operators.

- EA vs. heuristic: allow the same no. of steps:
  - Defining “step” is crucial, might imply bias!
  - Scale-up comparisons eliminate this bias.

Scale-Up Comparisons



Comparing algorithms A and B by their scale-up behaviour. Algorithm B can be considered preferable because its scale-up curve is less steep.

Peak vs Average Performance

- Typically in EA, average performance is more desirable.

- However, the best solution found in X runs or within Y hours/weeks is desirable in some applications.
  - Typically in design problems.

Outline

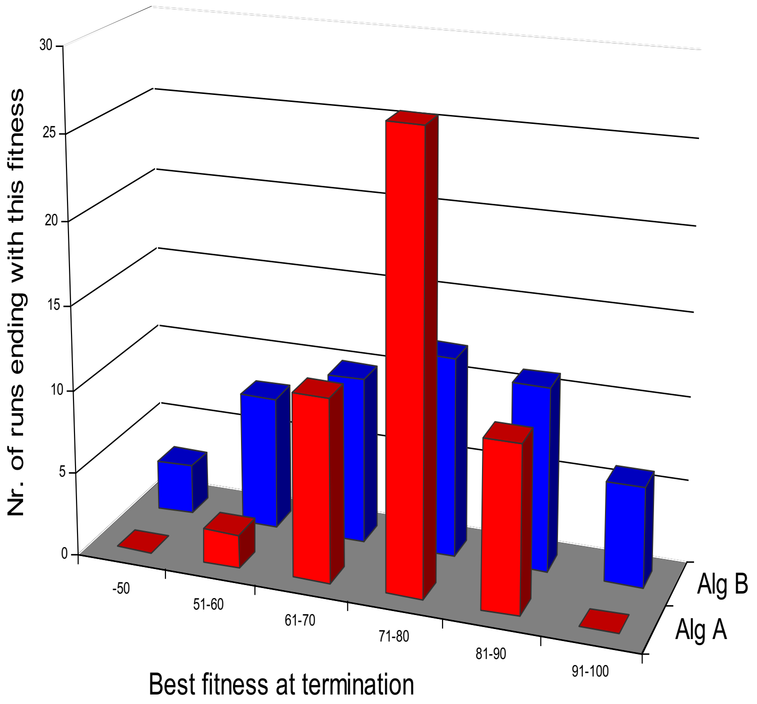
- No free lunch
  - Problem to be solved?
- Experimental Design and Parameter Tuning
  - Example of Parameter Tuning
  - Dynamic Parameter Tuning
- Expectation to your EA
- Performance Measures
- Rules of fair experimentation
- Exploration vs Exploitation

Basic Rules of Experimentation

- EAs are stochastic
  - *sufficient* # of independent runs
  - Apply statistics
    - measures (averages, standard deviations)
    - Tests (t-test…)

- Fair comparison
  - Same amount of resources per test
  - Different tests, varying resources
  - same performance measures

Off-line Algorithm Comparison

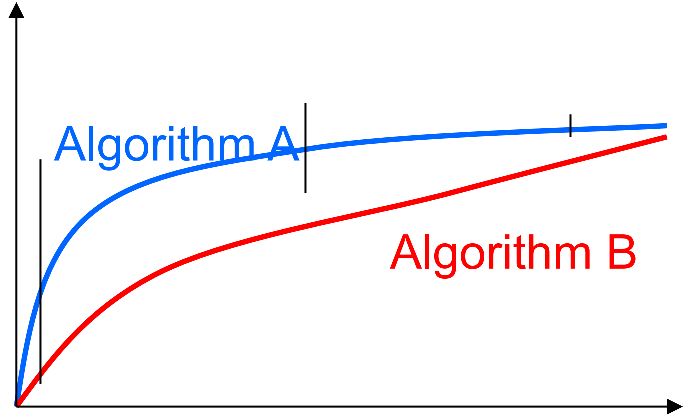


- A>B?
  - 50 runs
  - ↑ MBF
  - ↓ fitness variation
  - Yes: repetitive application
  - NO: design application
    - B has 6 runs achieving higher fitness
    - Good for timetabling once a year

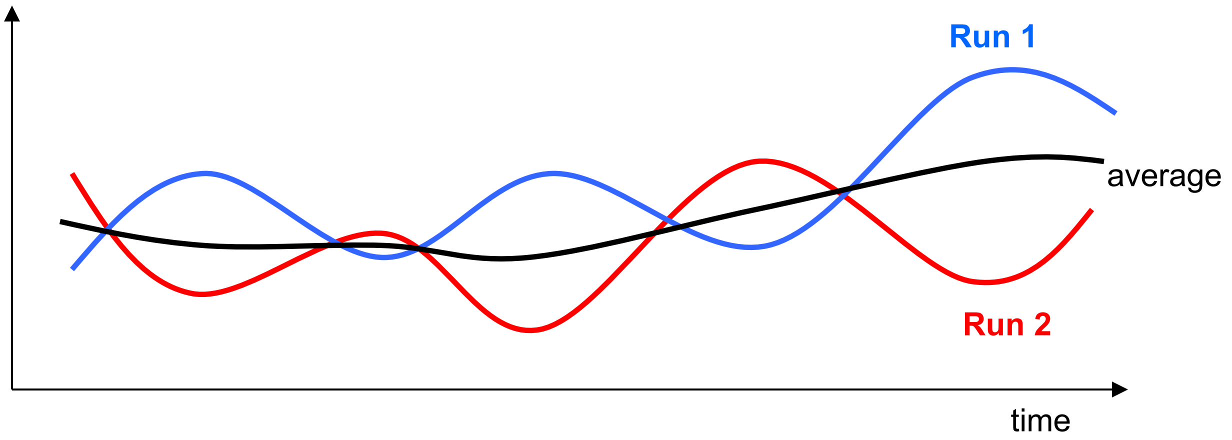
On-line Algorithm Comparison

- Same computational limit
  - All Performance Measures, SR, MBF etc
    - Same # evaluations /steps
    - Different # evaluations → different results
  - Averaging of algorithm's runs
    - Loss of information

Which algorithm is better?
Why? When?

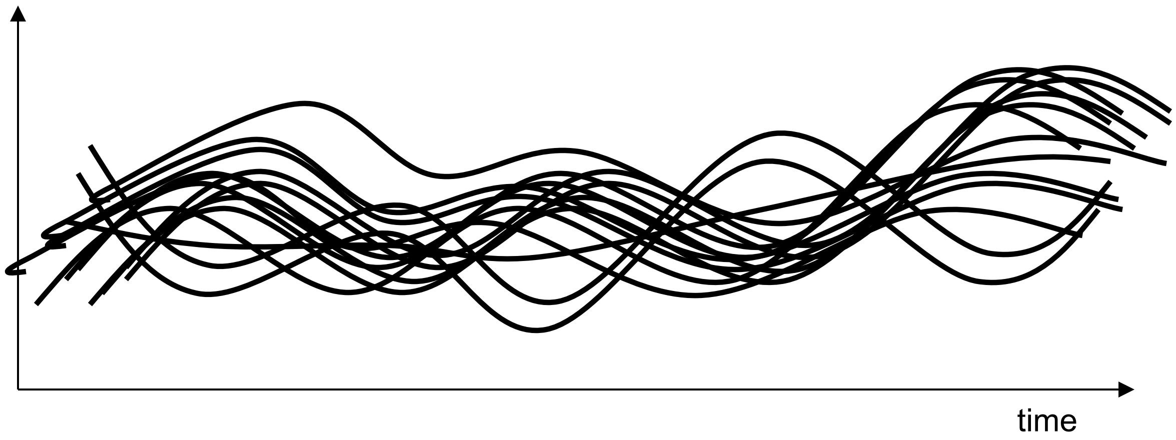


Example: Averaging On-line Measures



Averaging can “choke” interesting information

Example: Averaging On-line Measures



Overlay of curves can lead to very “cloudy” figures

Statistical Comparisons and Significance

- EAs are stochastic
- Results have element of “luck”
- Sometimes can get away with less rigour – e.g. parameter tuning
- For scientific papers where a claim is made: “Newbie recombination is better ran uniform crossover”, need to show statistical significance of comparisons

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?

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	**0.07080798**	

Statistical Comparison

- Mean and Standard Deviation
  - 2 values to describe a whole set of data
    - Randomness?

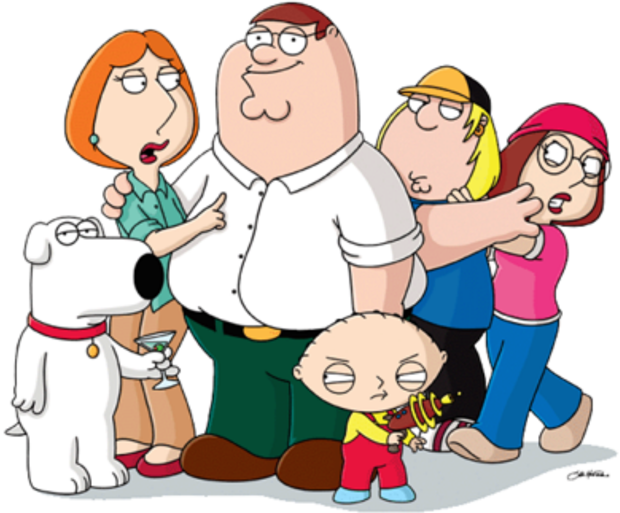
- TRUE statistical significance of differences
  - **T-test:**
    - Comparing 2 algorithms
  - **ANOVA test**
    - Comparing more than 2 algorithms

Statistical tests

- T-test assumes:
  - Data taken from continuous interval or close approximation
  - Normal distribution
  - Similar variances for too few data points
  - Similar sized groups of data points

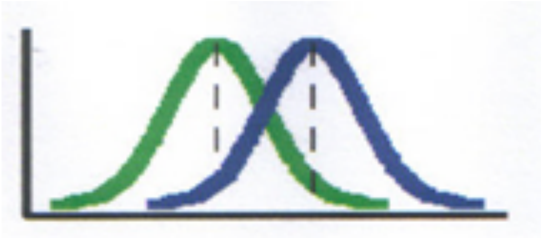
- Other tests:
  - Wilcoxon – preferred to t-test where numbers are small or distribution is not known.
  - F-test – tests if two samples have different variances.

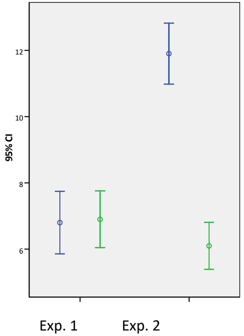
Comparison between Samples



Are these groups different?

t-tests





- Compare the **mean** between 2 samples/ conditions
- **if 2 samples are taken from the same population, then they should have fairly similar means**

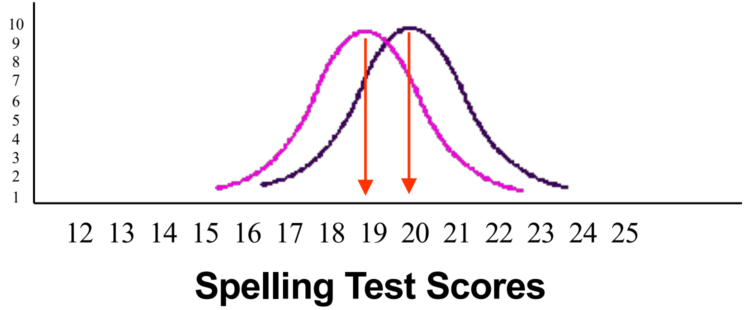
  ⇨ if **2 means are statistically different**, then the samples are likely to be drawn from 2 different populations, ie **they really are different**

Suppose we conducted a study to compare two strategies for teaching spelling.

Group A had a mean score of 19. The range of scores was 16 to 22, and the standard deviation was 1.5.

Group B had a mean score of 20. The range of scores was 17 to 23, and the standard deviation was 1.5.

How confident can we be that the difference we found between the means of Group A and Group B occurred because of differences in our teaching strategies, rather than by chance?



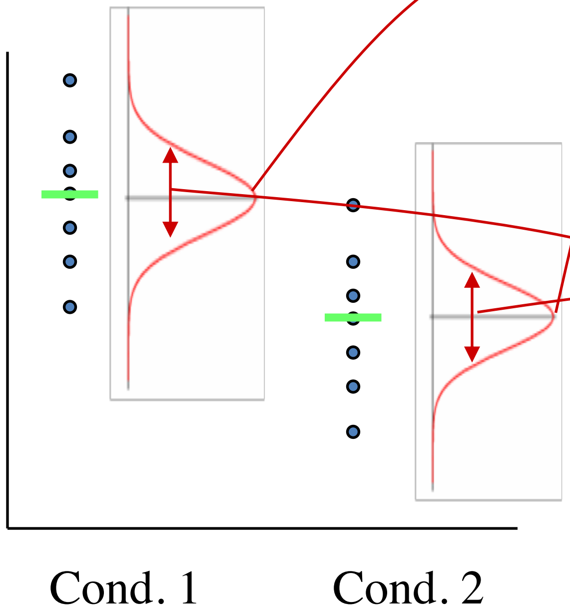
Formula

Difference between the means divided by the pooled standard error of the mean

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_{\bar{x}_1 - \bar{x}_2}}$$

Reporting convention: t= 11.456, df= 9, p< 0.001

Formula cont.



$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_{\bar{x}_1 - \bar{x}_2}}$$

$$s_{\bar{x}_1 - \bar{x}_2} = \sqrt{\frac{s_1^{\,2}}{n_1} + \frac{s_2^{\,2}}{n_2}}$$

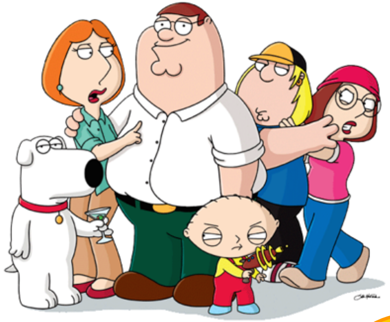
T Score

- The t score is a ratio between the difference between two groups and the difference within the groups.
  - A large (absolute) t-score tells you that the groups are different.
  - A small t-score (close to 0) tells you that the groups are similar.

T-Values and P-values

- How big is “big enough”?
- Every t-value has a p-value (0% to 100%.) to go with it.
- A p-value is the probability that the results from your sample data occurred by chance.
- They are usually written as a decimal.
  - For example, a p value of 5% is 0.05.
- Low p-values are good;
  - They indicate your data did not occur by chance.
  - A p-value of .01 means there is only a 1% probability that the results from an experiment happened by chance

Comparison of more than 2 samples



Tell me the difference between these groups… Thank God I have **ANOVA**

ANOVA

- ANalysis Of VAriance (ANOVA)
  - Still compares the differences in means between groups but it uses the variance of data to “decide” if means are different

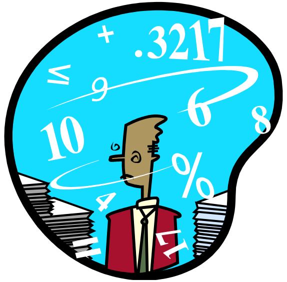
- Terminology (factors and levels)

- F- statistic
  - Magnitude of the difference between the different conditions
  - p-value associated with F is probability that differences between groups could occur by chance if null-hypothesis is correct
  - **need for post-hoc testing** (ANOVA can tell you if there is an effect but not where)

Reporting convention: $F= 65.58$, $df= 4,45$, $p< .001$

Statistical Resources

- http://fonsg3.let.uva.nl/Service/Statistics.html
- http://department.obg.cuhk.edu.hk/ResearchSupport/
- http://faculty.vassar.edu/lowry/webtext.html
- Microsoft Excel
- http://www.octave.org/



Test Problems for Experimental Comparisons

- Use problem instances from an academic repository.
- Use randomly generated problem instances.
- Use real life problem instances.

Getting Problem Instances 1

- Testing on ***real data***.
- Advantages:
  - Results are application oriented.
- Disadvantages
  - 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.

Getting Problem Instances 2

- Standard data sets in ***problem repositories***, e.g.:
  - OR-Library
  - http://www.ms.ic.ac.uk/info.html
  - UCI Machine Learning Repository
    www.ics.uci.edu/~mlearn/MLRepository.html
- Advantage:
  - Tried and tested 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.

Getting Problem Instances 3

- ***Problem instance generators*** produce simulated data for given parameters, e.g.:
  - GA/EA Repository of Test Problem Generators
    http://www.cs.uwyo.edu/~wspears/generators.html
- Advantage:
  - Allow systematic investigation of an objective function parameter range.
  - Can be shared allowing comparisons with other researchers
- Disadvantage:
  - Not real – might miss crucial aspect
  - Given generator might have hidden bias

Bad Practice : New Algorithm

- Overstatements based on simulation results
  - ‘Suitable’ performance metric
    - Different results from a different performance metric?
  - What if runs ran longer or shorter?
  - Scope of the superiority claim?
- ‘Selected’ test cases
  - is there a property in the ‘good’/bad results that tells you why they would be good/bad
- Generalisable results?
  - Sensitivity to parameter changes
    - Difficulty of achieving such results in other cases
- Statistical significance in results?

NTNU

Bad Example

- I invented “tricky mutation”
- Showed that it is a good idea by:
  - Running standard (?) GA and tricky GA
  - On 10 objective functions from the literature
  - Finding tricky GA better on 7, equal on 1, worse on 2 cases
- I wrote it down in a paper
- And it got published!
- Q: what did I learned from this experience?
- Q: is this good work?

Bad Example

- What did I (my readers) did not learn:
  - How **relevant** are these results (test functions)?
  - What is the **scope of claims** about the superiority of the tricky GA?
  - Is there a **property distinguishing** the 7 good and the 2 bad functions?
  - Can the results be **generalized**? (Is the tricky GA applicable for other problems? Which ones?)

Better Practice : New Algorithm

- Compare ~3 other algorithms
- Apply benchmark heuristic
- When and why new algorithm is better?
- Problem instance generator
  - Generate ~100 problem instances
- Execute **all** algorithms on **all** instances
- AES, SR and MBF (with SD, not on SR)
- Statistical significance

Outline

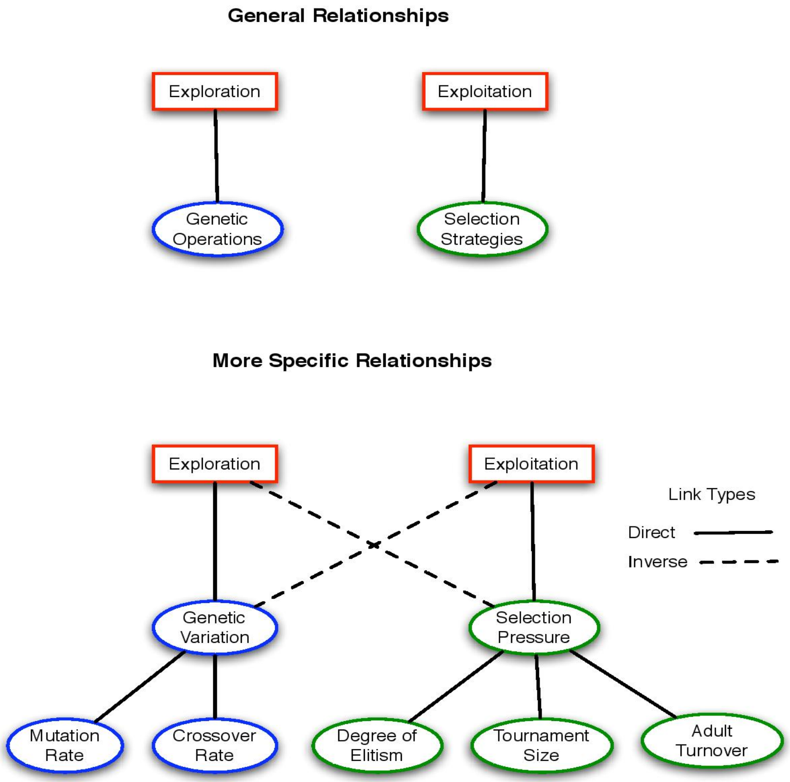
- No free lunch
  - Problem to be solved?
- Experimental Design and Parameter Tuning
  - Example of Parameter Tuning
  - Dynamic Parameter Tuning
- Expectation to your EA
- Performance Measures
- Rules of fair experimentation
- Exploration vs Exploitation

Exploration vs Exploitation

- Explorative search
  - new solutions, **quite different** from all previous probes,
  - Acquire information from **uncharted area**
    - **e.g. crossover**
- Exploitative Search
  - new solutions/probes, **slightly different form promising probes**
  - in an area of **known potential**
  - tries to zero in on the best solution in that region.
    - **mutation,**
    - **Selection** (prioritises best individuals)
- GA often needs both
- all parameters can be applied for exploration and/or exploitation

Frequency of crossover/mutation ~ measure of exploration vs exploitation

Exploration, Exploitation and Genetic Operators



Required Reading + References

- Floreano: chapters 1: 1-15
- Downing: Evolutionary Algorithms in Search and Problem Solving: sections 1-3
- Downing: Natural and Artificial Selection
- Eiben: chapters: 14, (14-1-14.3, 14.5); 8 (8.1, 8.2, 8.4)

- References:

  – [REX13] Rexhepi, A., Maxhuni, A., & Dika, A. (2013). Analysis of the impact of parameters values on the Genetic Algorithm for TSP. *International Journal of Computer Science Issues*, *10*(1), 158-164.