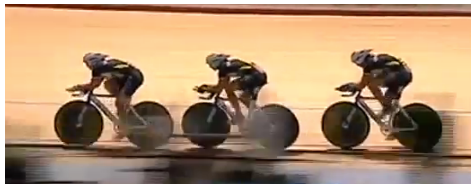
Real-World Optimisation, a reminder

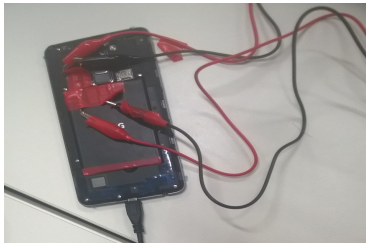


AUSTRALIAN INSTITUTE OF SPORT

Carnegie CLEAN ENERGY

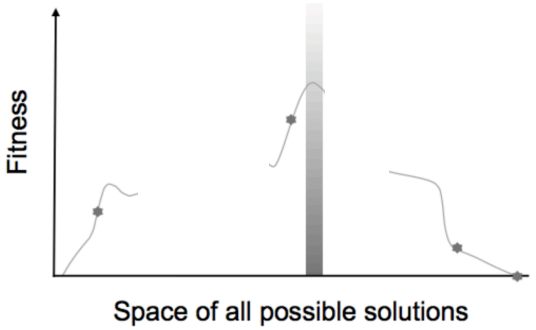
Premier's Research and Industry Fund: $14.6m Research Consortium "Unlocking Complex Resources through Lean Processing"

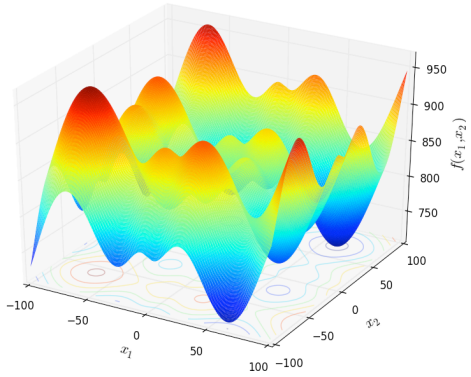
Partners: BHP, OZ Minerals, AMIRA International, Australian Information Industries Association (AIIA) IoT Cluster for Mining and Energy Resources, Australian Semi-Conductor Technology Company, Boart Longyear, Consilium Technology, CRC Optimising Resource Extraction, Datanet, Data to Decisions CRC, Eka, Innovyz, Magotteaux, Manta Controls, Maptek, METS Ignited Industry Growth Centre, Mine Vision Systems, Rockwell Automation, SACOME, SAGE Automation, Sandvik, Scantech, South Australian Mining Industry Participation Office (SA MIPO), SRA IT and Thermo Fisher Scientific Australia (Processing Instruments & Equipment), with the University of South Australia as a key research partner.



Australian Government
Australian Research Council

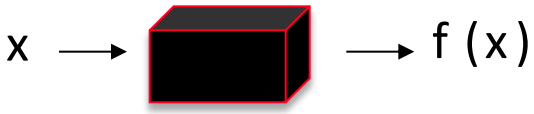
DE16: Dynamic adaptive software configurations





➔ Everybody is after decision support!

(to be faster, decrease wear, deliver on time, …)



Working with Stochastic Algorithms

Chapters 8 and 14 in Eiben/Smith

Issues considered

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

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 well documented (locally on the drive, in emails, in a scientific article, in an honours/masters/PhD thesis, …)

Example: Design Perspective

Optimising spending on improvements to national road network

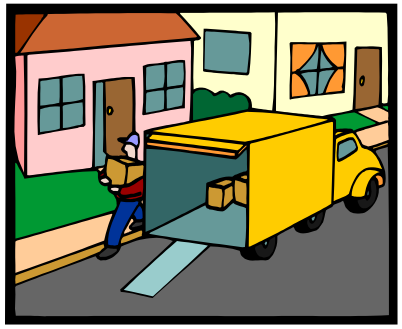
- Total cost: billions of AUD
- Computing costs negligible
- Six months to run algorithm on hundreds computers
- Many runs possible
- Must produce a very good result just once

Example: Production Perspective

Optimising Internet shopping delivery routes of hundreds of trucks given tens of distribution centres

- Different destinations each day
- Multiple locations can satisfy a request
- Limited time to run algorithm each day
- Must always be reasonably good route in limited time

The time constraint could be even stronger: think of "online algorithms"



Perspectives of an EA's goals

- Design perspective:

  find a very good solution at least once
- Production perspective:

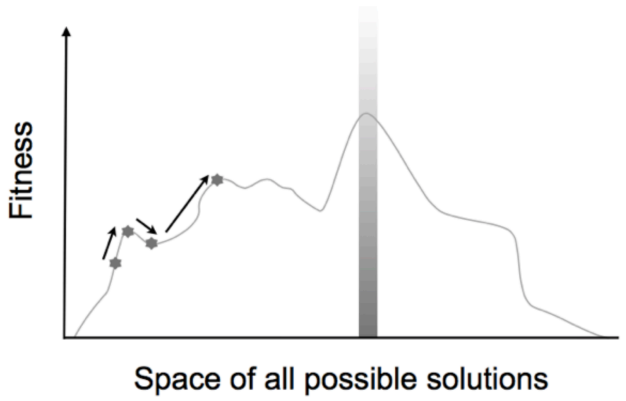
  find a good solution at almost every run
- Academic perspective:

  must meet scientific standards

These perspectives have very different implications when evaluating results.

Algorithm design

- Design a representation
- Design a way of evaluating a solution
- Design suitable neighbourhood operator(s)
  For Genetic/Evolutionary Algorithms: design suitable mutation and crossover (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: initialisation method
- Decide how to stop: termination criterion



Bad example

- I invented "tricky variation operator" for Genetic Algorithms (GA)
- 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!

- Q1: what did I learn from this experience?
- Q2: 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 generalised? (Is the tricky GA applicable for other problems? Which ones?)

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 generalise – the problem of "overfitting" to the scenario is very real

Getting Problem Instances 2

Standard data sets in problem repositories, e.g.:

- OR-Library
  http://people.brunel.ac.uk/~mastjjb/jeb/info.html
- UCI Machine Learning Repository
  http://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 (again overfitting)

Getting Problem Instances 3

Problem instance generators produce simulated data for given parameters

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 (can result in overfitting again)

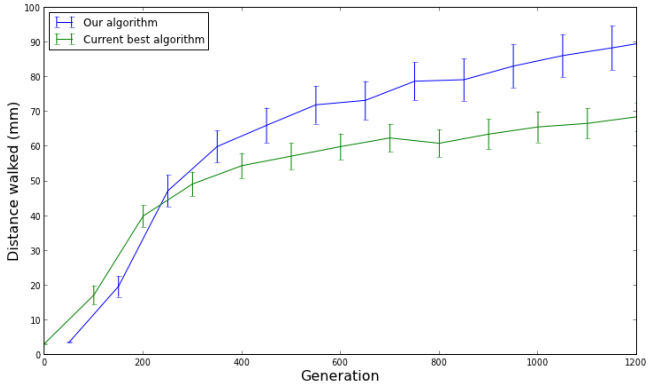
Basic rules of experimentation

Stochastic algorithms make use of random decisions → 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

Experimentation is about comparison → always do a fair competition

- use the same performance measures
- use the same amount of resources for the competitors
- try different competition limits



http://www.randalolson.com/2013/07/20/a-short-guide-to-using-statistics-in-evolutionary-computation/

Whiskers show standard deviation of the performance at particular points in time.

Things to Measure

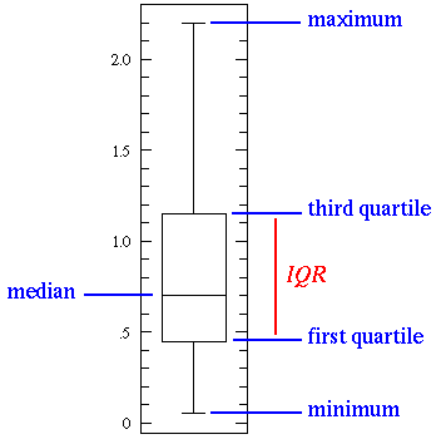
Many, many different ways.

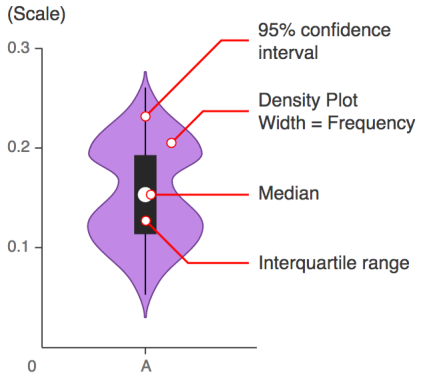
Examples:

- Mean/median result in given time
- Mean/median time for given result
- Proportion of runs within % of target
- Best result over n runs
- …

Many visualisations possible:

- Boxplots, violin plots, scatter plots, …





https://datavizcatalogue.com/methods/images/anatomy/SVG/violin_plot.svg
http://www.physics.csbsju.edu/stats/simple.box.defs.gif

What time units do we use?

Elapsed time?

- Depends on computer, network, etc…

CPU Time?

- Depends on skill of programmer, implementation, etc…

Generations?

- Difficult to compare when parameters like population size change

Evaluations?

- Evaluation time could depend on algorithm, e.g. direct vs. indirect representation

Measures

Performance measures (off-line)

- 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

"Working" measures (on-line)

- Solution set (population) distribution (decision/encoding space)
- Fitness distribution (objective space)
- Improvements per time unit or per variation operator application
- …

Performance measures

- Number of generated points in the search space
  = number of fitness evaluations
  (normally don't use number of generations!)
- AES: average number of evaluations to solution
- SR: success rate = % of runs finding a solution (individual with acceptable quality / fitness)
- MBF: mean best fitness at termination, i.e., best per run, mean over a set of runs
- SR ≠ MBF
  - Low SR, high MBF: good approximiser (more time helps?)
  - High SR, low MBF: "Murphy" algorithm (if it goes wrong, then it goes very wrong)

Fair experiments

Basic rule: use the same computational limit for each competitor

Allow each algorithm the same number of evaluations, but

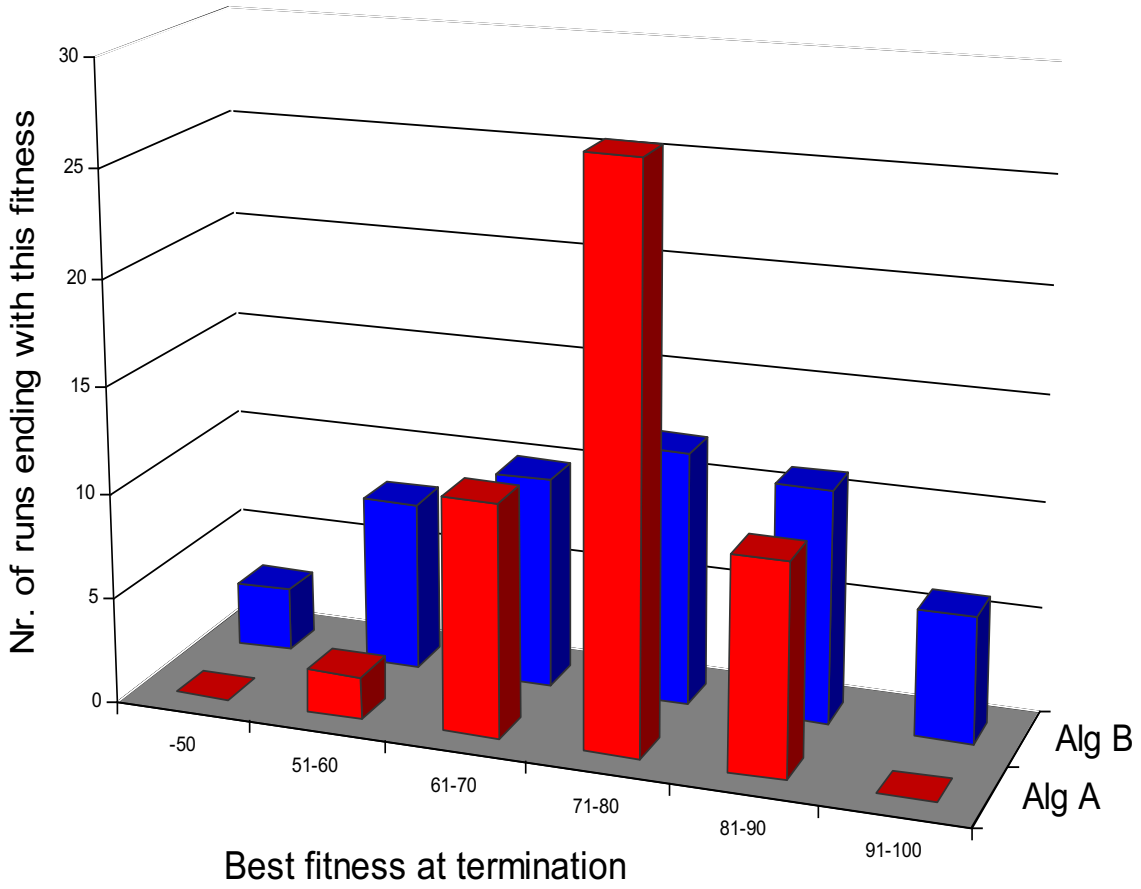
- Beware of hidden labour, e.g. in heuristic mutation operators
- Beware of possibly fewer evaluations by smart operators

(5-min break)

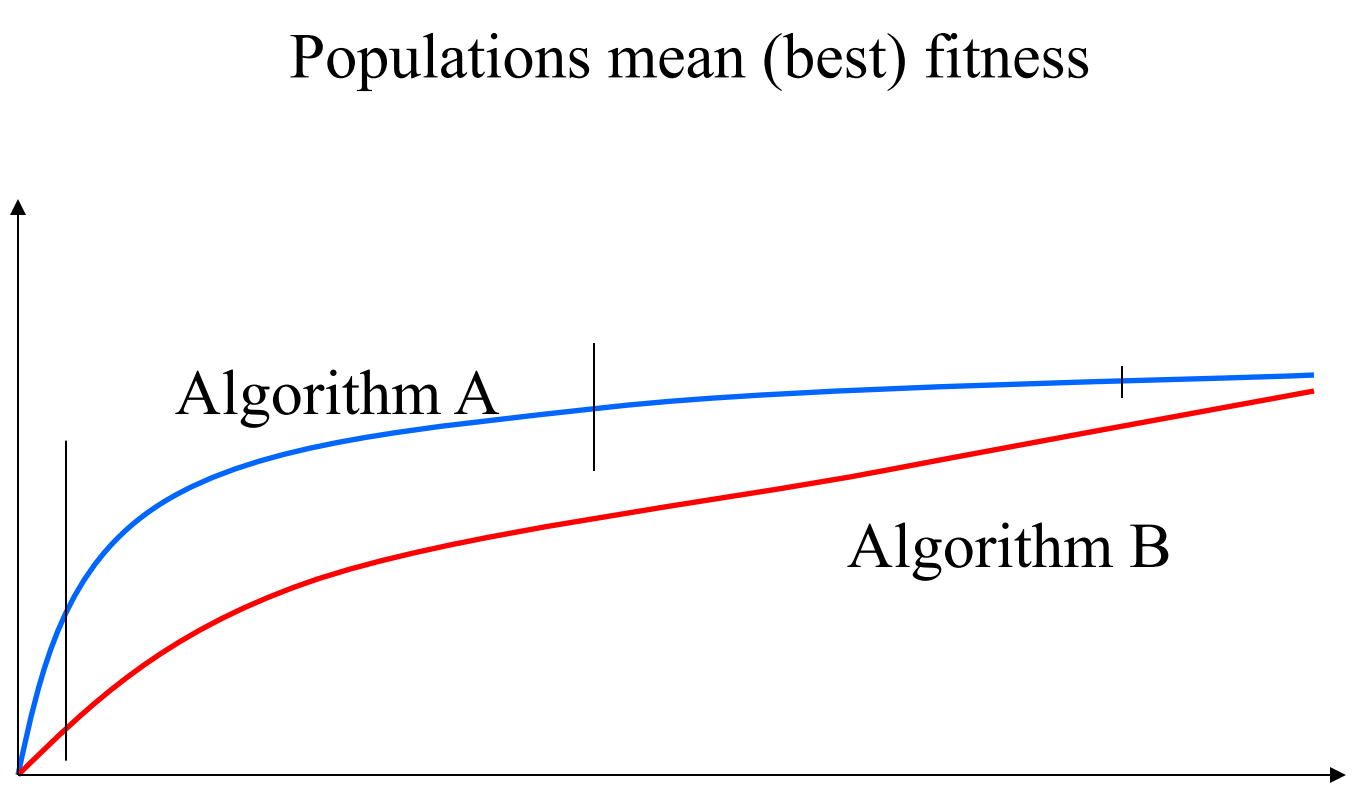
http://boxcar2d.com/

Example: off-line performance measure evaluation

Which algorithm is better?
Why?
When?

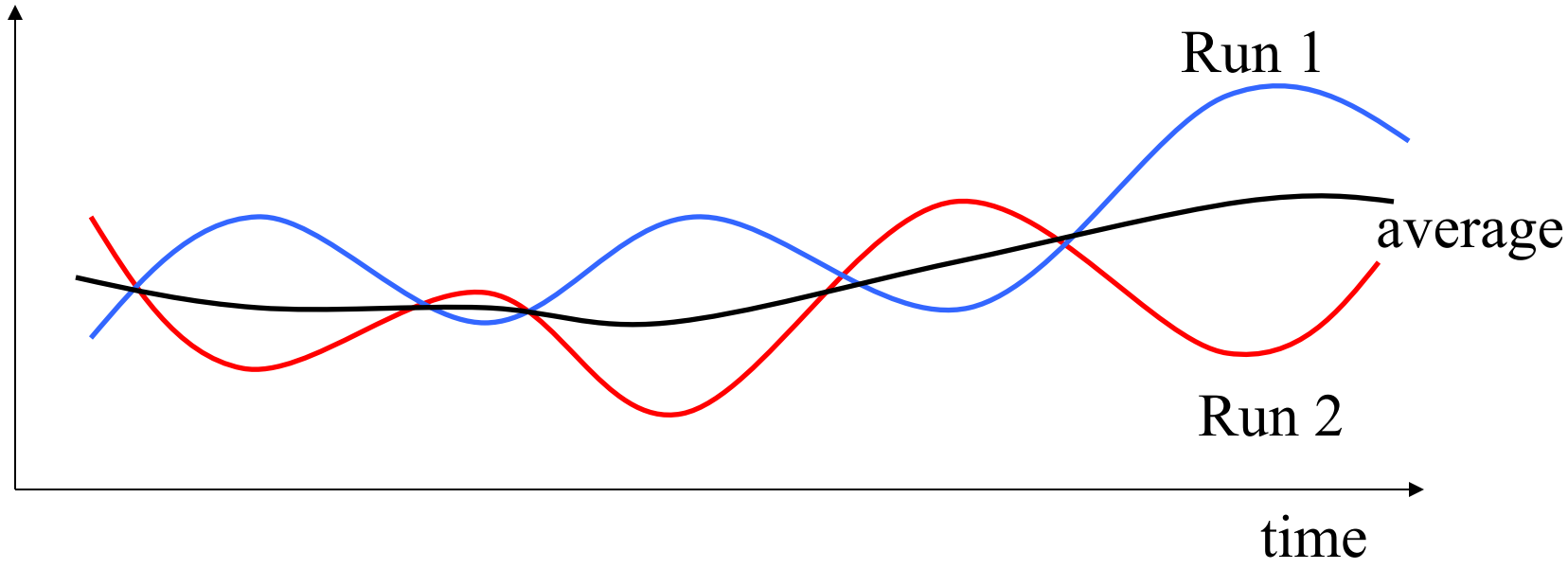


Example: on-line performance measure evaluation



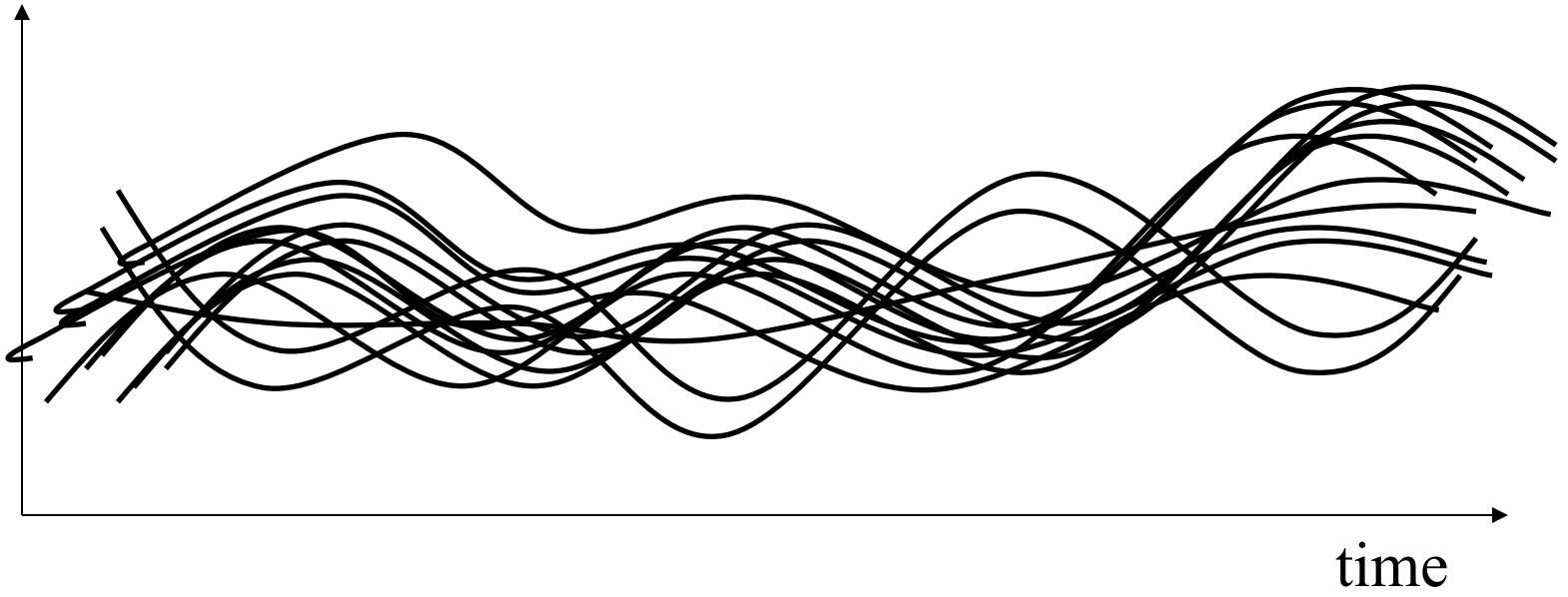
Which algorithm is better? Why? When?

Example: averaging on-line measures



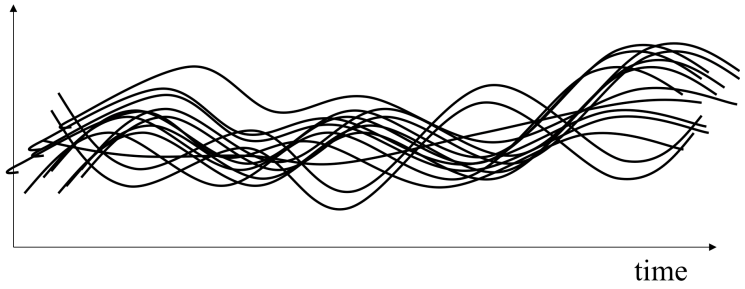
Averaging can "choke" interesting information

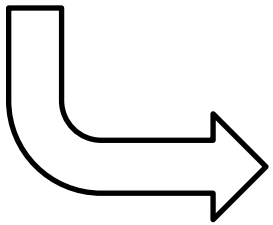
Example: overlaying on-line measures

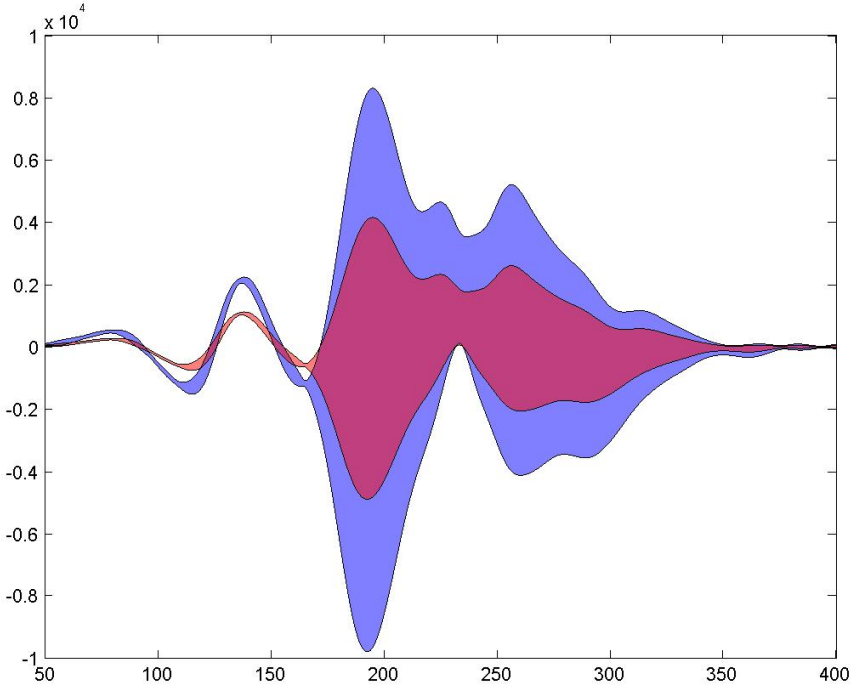


Overlay of curves can lead to very “cloudy” figures

Example: overlaying on-line measures



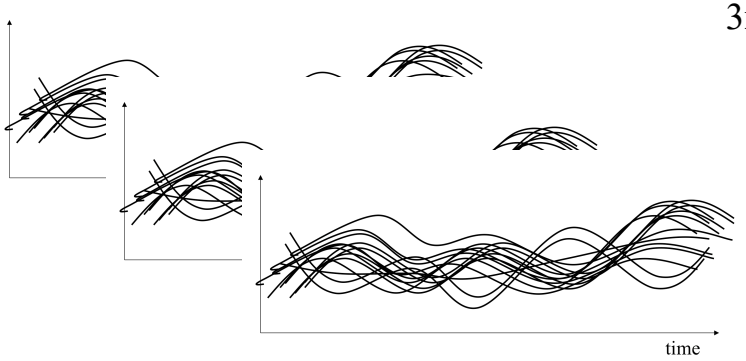




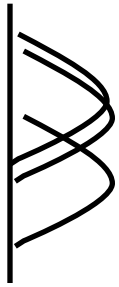
Plot of 2 approaches, confidence intervals
(Note: data is *different* from the top left)
https://au.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/13103/versions/1/screenshot.jpg

Statistical Comparisons and Significance

- Algorithms 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 than uniform crossover", need to show statistical significance of comparisons



3x 10 runs: are they from the same distribution? Yes/No?



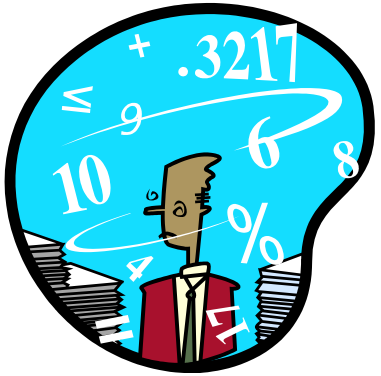
What if these are 3x 1 million runs: are they from the same distribution? Yes/No?

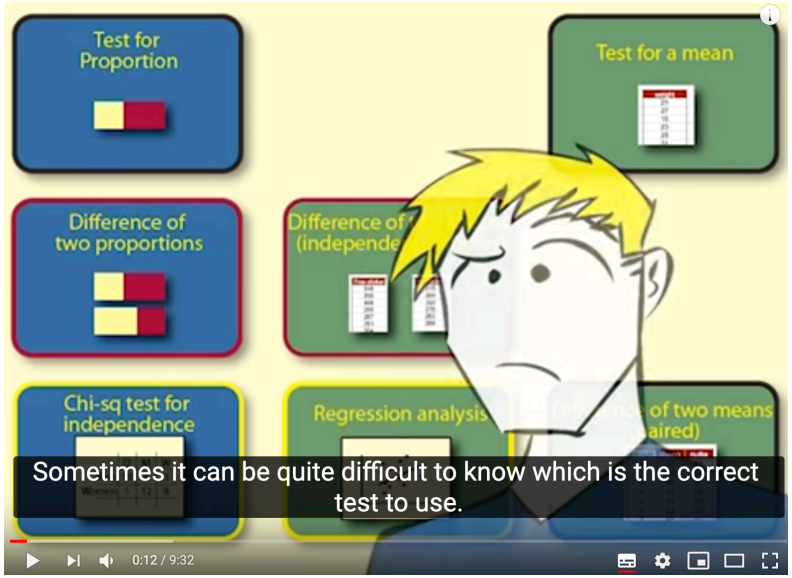
Statistical tests

- T-test assummes:
  - 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 (most likely preferred):
  - Wilcoxon ranksum test (or Mann-Whitney U test) – preferred to t-test where numbers are small, or distribution is not known.
  - F-test – tests if two samples have different variances.

Statistical Resources

- Web pages that perform statistical computations: http://statpages.org/
- Statistical Resources Links Page http://core.ecu.edu/psyc/wuenschk/statistics.htm
- Microsoft Excel
- http://www.octave.org
- http://www.r-project.org
- Entertaining video: "Choosing which statistical test to use" https://www.youtube.com/watch?v=rulIUAN0U3w





Better example: problem setting

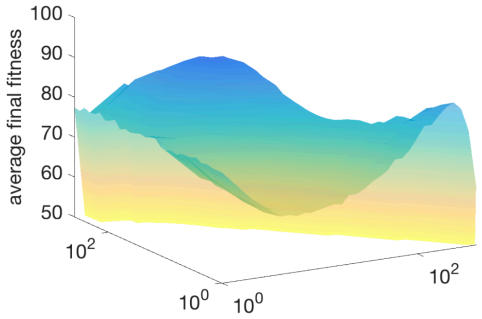
- I invented myEA for problem X
- Looked and found 3 other EAs and a traditional benchmark heuristic for problem X in the literature
- Asked myself when and why is myEA better

Better example: experiments

- Found/made problem instance generator for problem X with 2 parameters:
  - n (problem size)
  - k (some problem specific indicator)
- Selected 10 values for k and 10 values for n
- Generated 100 problem instances for each combination
- Executed all alg's on each instance 100 times (benchmark was also stochastic)
- Recorded AES, SR, MBF values with same computation limit (AES for generated instances?)
- Put my program code and the instances online

Average number of Evaluations to Solution
Success Rate
Mean Best Fitness

Better example: evaluation



- Arranged results "in 3D" (n,k) + performance
  (with special attention to the effect of n, as for scale-up)
- Assessed statistical significance of results

- Found the niche for my_EA:
  - Weak in … cases, strong in - - - cases, comparable otherwise
  - Thereby I answered the "when question"
- Analysed the specific features and the niches of each algorithm thus answering the "why question"
- Learned a lot about problem X and its solvers
- Achieved generalisable results, or at least claims with well-identified scope based on solid data
- Facilitated reproducing my results → further research

Other communities are learning from Optimisation Community

The following slides are taken from:

This is a Search-Based Software Engineering (SBSE) event

[simula . research laboratory]

Conducting and Analyzing Empirical Studies in Search-based Software Engineering

Lionel Briand

Simula Research Laboratory

and University of Oslo, Norway

SSBSE 2011, Szeged, Hungary

Over 24,000 citations

http://www.ssbse.org/2011/presentations/SSBSE_tutorial_Briand_final.pdf

Background

- Search techniques are increasingly used in SE to solve a variety of problems, ranging from requirements prioritization, re-engineering, to test generation and fault fixing
- Undecidable problems for which optimal solutions cannot be obtained within reasonable time
- Search -> Randomized algorithms
- The body of knowledge is increasing fast, as well as the research community.
- Now is the time to get our scientific procedures and reporting straight

Problem Definition

Note from Markus: this is no longer true in 2019

- For every problem addressed by SBSE, there is usually an alternative not based on search, e.g., constraint solver, model checker, static analysis
- Random search is always an alternative, and can be effective
- Many factors can affect the effectiveness and cost of a search technique, e.g., parameters, selected artifacts/problems, implementation
- Random variation (randomized algorithms)
- Designing, running, and analyzing SBSE empirical studies must be done with care to build a reliable body of knowledge
- Recent surveys shows this is not always the case
  (from 2009/2010)

Research Survey

- Snapshot of current practice
- Venues: TSE, ICSE and SSBSE
- Year 2009 and 2010
- Results:
  - Large number of papers involving randomized algorithms
    - TSE/ICSE: 7% (2009) and 21% (2010)
  - Randomness often not taken into account properly
  - Seldom/inappropriate use of *statistical tests*...

2009

Reference	Venue	Repetitions	Statistical Tests
[1]	TSE	1/5	U-test
[67]	TSE	1	None
[77]	TSE	1	None
[70]	ICSE	100	*t*-test, U-test
[99]	ICSE	100	None
[35]	ICSE	1	None
[56]	ICSE	1	None
[7]	SSBSE	1000	Linear regression
[39]	SSBSE	30/500	None
[26]	SSBSE	100	U-test
[38]	SSBSE	50	None
[60]	SSBSE	10	Linear regression
[54]	SSBSE	10	None
[66]	SSBSE	1	None
[57]	SSBSE	1	None
[91]	SSBSE	1	None

@ Lionel Briand

Reference	Venue	Repetitions	Statistical Tests
[37]	TSE	1000	None
[106]	TSE	100	*t*-test
[47]	TSE	60	U-test
[82]	TSE	32	U-test, $\hat{A}_{12}$
[24]	TSE	30	Kruskal-Wallis, undefined pairwise
[93]	TSE	20	None
[17]	TSE	10	U-test, *t*-test, ANOVA
[28]	TSE	3	U-test
[6]	TSE	1	None
[13]	TSE	1	None
[16]	TSE	1	None
[100]	TSE	1	None
[61]	ICSE	100	None
[107]	ICSE	50	None
[40]	ICSE	5	None
[74]	ICSE	5	None
[34]	ICSE	1	None
[45]	ICSE	1	None
[50]	ICSE	1	None
[104]	ICSE	1	None
[79]	ICSE	1	None
[89]	ICSE	1	None
[22]	SSBSE	100	*t*-test
[23]	SSBSE	100	None
[65]	SSBSE	50	*t*-test
[69]	SSBSE	50	U-test
[103]	SSBSE	30	U-test
[105]	SSBSE	30	*t*-test
[62]	SSBSE	30	Welch
[97]	SSBSE	30	ANOVA
[14]	SSBSE	3/5	None
[64]	SSBSE	3	None
[108]	SSBSE	1	None

2010

Note from Markus (2019):
currently, the average approach is decent, and sometimes code is online

Summary: some tips (yes, everything is red)

- Be organised
- 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)
- Use good statistics ("standard" tools from web, MS, …)
- Present results well (figures, graphs, tables, …)
- Watch the scope of your claims
- Aim at generalisable results
- Publish code for reproducibility of results (if applicable)