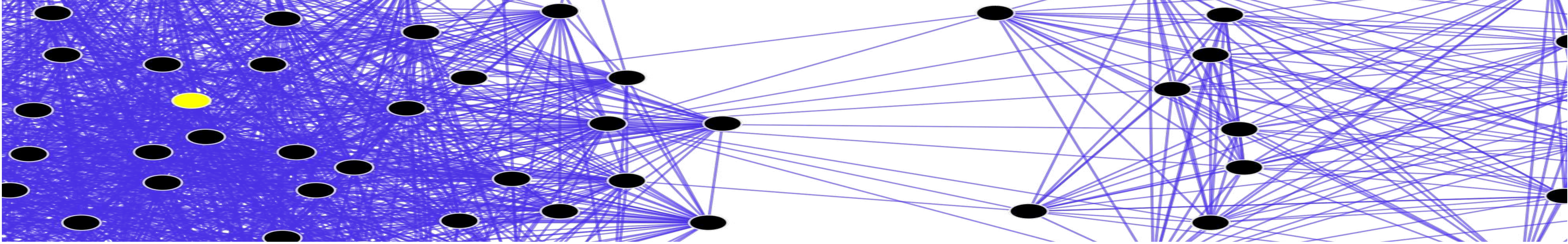


Computational Intelligence

Master in Artificial Intelligence

2016-17

Lluís A. Belanche
belanche@cs.upc.edu

Extra: tips for doing experimental work

Soft Computing Research Group

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Tips for doing experimental work

Experimentation

1. Has a goal(s) and conjecture(s) about the expected results

2. Involves algorithm design and implementation

3. Needs problems/data to run the algorithm on

4. Needs establishing what is variable and what is held fixed

5. Needs running the algorithm(s) on the problem(s) and collect the results

6. Needs evaluating the results in the light of the given goal(s) and conjectures

Tips for doing experimental work

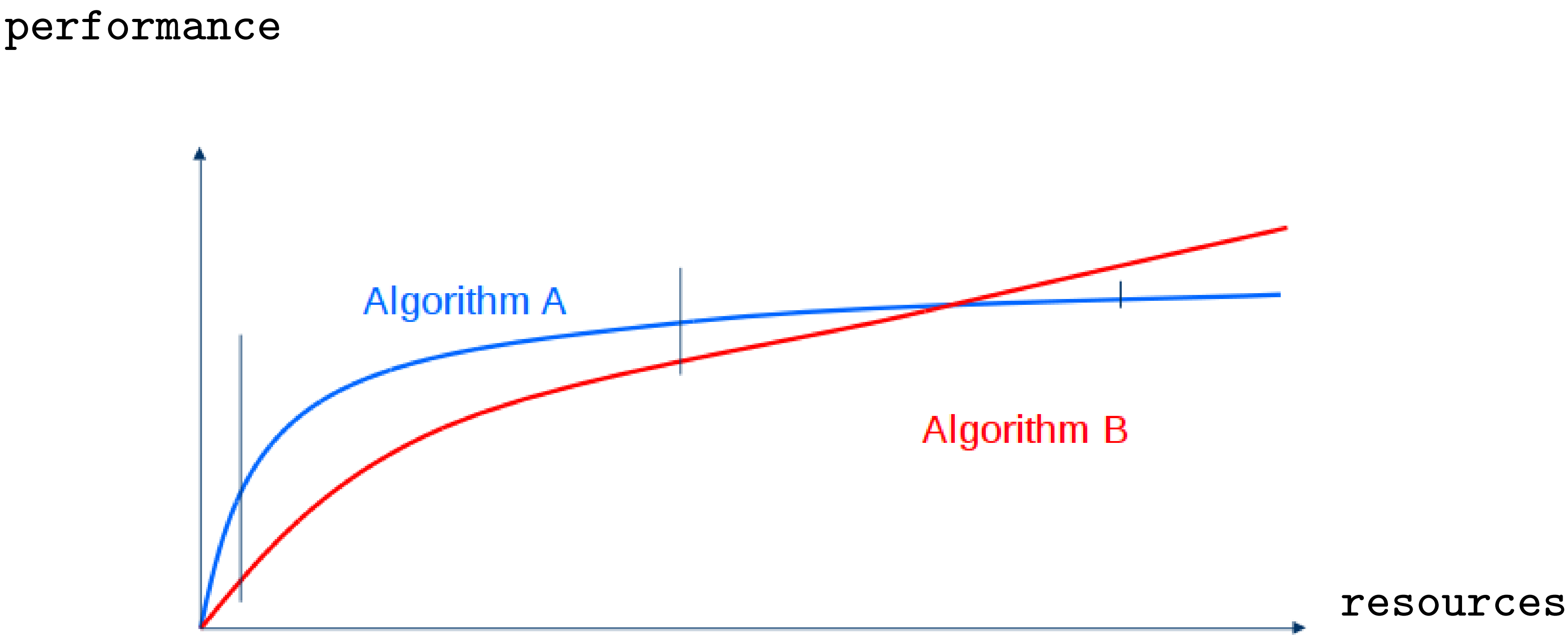
Possible goals

- Get a good solution for a given problem
- Show that an algorithm is applicable to a problem or class thereof
- Show that an algorithm is better than another one in some respect
- Find “optimal” setup for parameters of a given algorithm
- Understand algorithm behavior:
  - how it scales up with problem size
  - how it performs under extreme conditions

  $\rightarrow$ how performance is influenced by parameters (criticality)

Tips for doing experimental work

Example 1

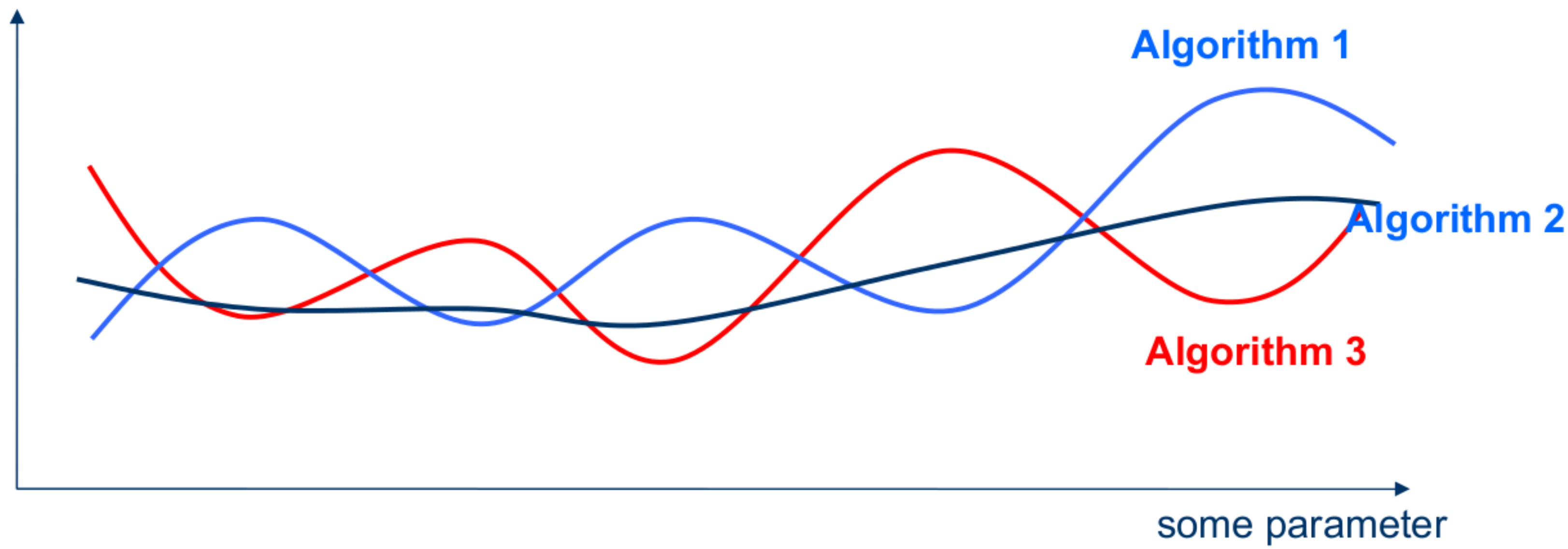


Which algorithm is better?

Tips for doing experimental work

Example 2

performance



What kind of algorithm we wanted?

1. very good in very specific situations
2. fairly good on average

Hallmarks of good scientific work

- Clear statement of target problem, goals and scope
- Large enough tests (number, difficulty, representativeness)
- Statistical analysis of results
- Reproducibility
- Insightful discussion of the results
- Conclusions: what have we achieved, what have we learned, whys and why nots, strenghts and limitations
- Future work: open problems, avenues for future investigation

Using real vs. synthetic problems/data

Advantages well-chosen problems (coming from real-world), comparison to previous work

Disadvantages not owner, might miss important absent information, problem feasibility?

Advantages allow systematic comparison: repetitions, sizes, hardness, ...; can target very specific issues; can be shared

Disadvantages not the "real thing"; might be too simplistic/hard; hidden biases?

Tips for doing experimental work

never draw any conclusion from a single run!

- perform sufficient number of independent runs
- use statistical measures (means, standard errors)
- use statistical tests to assess reliability of conclusions (t-test, F-test, Wilcoxon, ...)

always do a fair competition!

- use same amount of resources for the competitors
- use same performance measures

Tips for doing experimental work

Bad example

I invented a "Cool Super Algorithm" (CSA) for problem X

- I showed it really "works" by running it against a "standard" solution
- I chose a bunch of 10 problems (rather randomly)
- I run the two methods once (I invested 10 times more time in tuning mine)
- I found that my CSA was "better" on 7 problems, "equal" on 1, and "worse" on 2

I wrote everything down in a paper and got it published!

Tips for doing experimental work

Bad example

Ask yourself ...

1. How relevant are these results? (honest, trustable, reproducible, representative, ...)
2. What is the scope of my claims? (and what are my claims ...)
3. Why do I get the 7 successes (and why these 2 failures)? Is there a common explanation?
4. What did I learned from this work? What will others learn?

Tips for doing experimental work

Good example

I invented a new algorithm/variation “Quasi New Algorithm” (QNA) for problem X; then I ...

- Looked in the literature and selected 3 other methods and a baseline heuristic for problem X

- Asked myself when/why is my QNA better than any/some of these

- Found/designed a problem generator for problems of the X type with two parameters:

  - n (problem size)

  - k (problem-specific indicator: *e.g.*, hardness)

Tips for doing experimental work

Good example

1. Selected some values for k and some for n (with some criterion)

2. Generated 100 problem instances for each combination

3. Executed all methods on each instance

4. Recorded all performance measures (accuracies, CPU times, ...)

5. Put my program code and the instances on the Web $\rightarrow$ facilitate reproducing my results $\rightarrow$ further research (and learned a lot about problem X and its solvers)

Tips for doing experimental work

Good example

1. Arranged results "in 3D": (n,k,performance)

2. Assessed statistical significance of results

3. Found the *niche* for my QNA: "Weak in ____ cases, strong in ____ cases, comparable otherwise."

   Thereby I answered the -when- question

4. Analyzed the specific features and the niches of each algorithm, thus answering the -why- question

5. Achieved generalizable results, or at least claims with well-identified scope based on solid data