Experimental Analysis of Optimization Algorithms: Tuning and Beyond

January 17, 2020

Outline

Introduction

- As in many sciences, research on metaheuristics and especially evolutionary computation (EC) mainly rests on two pillars: theory and practice.
- It seems that during the last decades, two motivations for experimental works have been predominant:
  - Solving a real-world problem or at least showing that it could be solved by some EC based method
  - Demonstrating the ability of a (preferably new and self-defined) algorithm
- Performing experiments in computer science can address the following (related) tasks:
  - Find the best parameters or algorithms given k sets of random numbers representing the outcome of some experiments
  - Find the best assignment for a set of real variables representing the parameters of the algorithm (within a given range) for a problem class
  - Given m possible ways to modify algorithm a (e.g., by using extra operators) find the best combination for a problem class

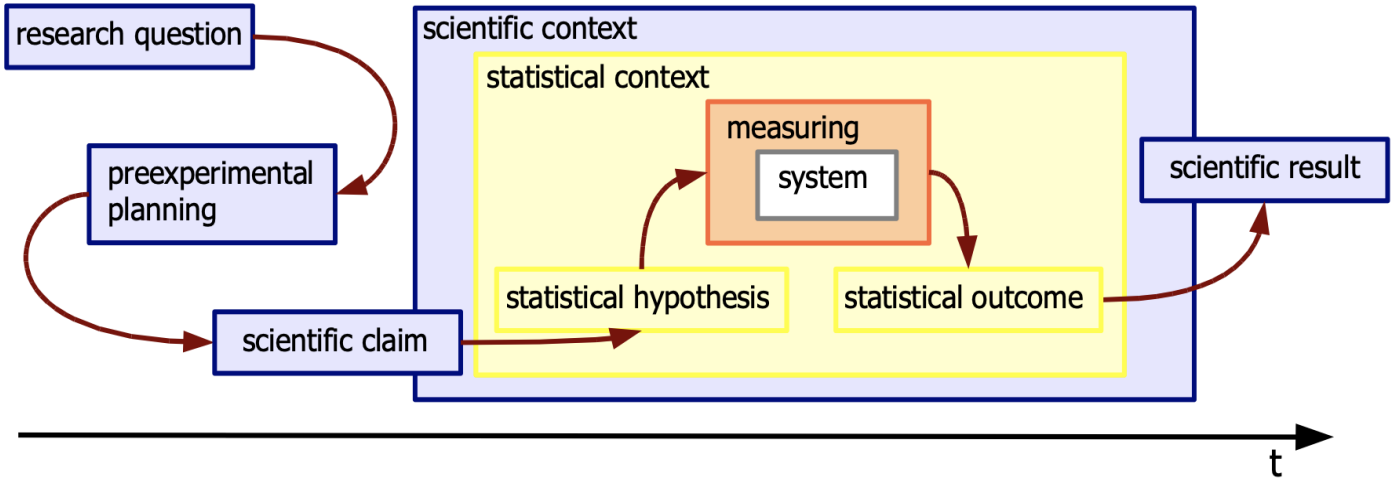

Fig. 1 Steps and contexts of performing an experiment from research question to scientific result

Towards an Experimental Methodology

- In theoretical computer science, are very important pessimistic generalizations, in knowing what an algorithm does in the worst possible case. Experimental results are considered with a certain amount of skepticism. This may have two reasons:
  - Many experimental works of the past are not very well crafted
  - Experimental investigations rarely care for worst cases

Towards an Experimental Methodology

- The following questions or aspects of questions are fundamental for experiments in computer science and may serve as a guideline for setting up new experiments. The experimenter should clearly state if experiments are performed to:
  - Demonstrate the performance of one algorithm
  - Verify the robustness of one algorithm on several problem instances
  - Compare two (or several) known algorithms
  - Explain and understand an observed behavior
  - Detect something new
- Each of these five research goals, which can also be characterized as demonstration, robustness, comparison, understanding, and novelty detection, requires a different experimental setup.

Active Experimentation

- Active experimentation (AEX) is a framework for tuning and understanding of algorithms. AEX employs methods from error statistics to obtain reliable results. It comprises the folllowing elements:
  - AEX-1: Scientific questions
  - AEX-2: Statistical hypotheses
  - AEX-3: Experiments
  - AEX-4: Scientific meaning

Active Experimentation

- These elements can be explained as follows. Starting point of the investigation is a scientific question (AEX-1). This question often deals with assumptions about algorithms, e.g., influence of parameter values or new operators. This (complex) question is broken down into (simple) statistical hypotheses (AEX- 2) for testing. Next, experiments can be performed for each hypothesis:
  - Select a model, e.g., a linear regression model to describe a functional relationship.
  - Select an experimental design.
  - Generate data, i.e., perform experiments.
  - Refine the model until the hypothesis can be accepted/rejected.

  Finally, to assess the scientific meaning of the results from an experiment, conclusions are drawn from the hypotheses. This is step (AEX-4) in the active experimentation framework

Severity - Motivation

- Severity provides a meta-statistical principle for evaluating proposed statistical inferences.
- Severity should be calculated after the test procedure is finished.

Severe tests

Definition 3 (Severe Test). A statistical hypothesis H passes a severe test T with data x_0 if:

- S-1 x_0 agrees with H
- S-2 with very high probability, test T would have produced a result that accords less well with H than x_0 does, if H were false.

Severity in the Case of Acceptance of the Null:

$$SEV(\mu \leq \mu_1) = 1 - \Phi(\frac{\bar{x_0} - \mu_0}{\sigma_x})$$

Severity in the Case of Rejection of the Null Hypothesis:

$$SEV(\mu > \mu_1) = 1 - SEV(\mu \leq \mu_1)$$

In case of a rejection of the alternative, the power of a test provides a lower bound for the severity.

Severe tests

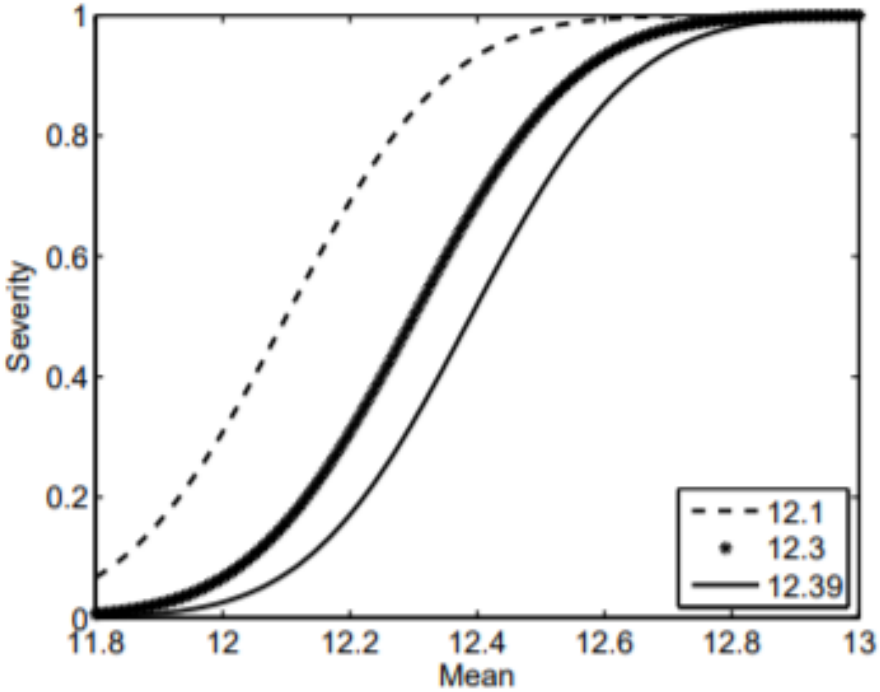

Fig. 5 Severity for three different results $\bar{x}_0$: 12.1, 12.3, and 12.39. These curves can be interpreted as follows: consider, e.g., $\bar{x}_0 = 12.3$, which gives $d(\mathbf{x}_0) = 1.5$: the assertion that $\mu \leq 13$ severely passes because $\text{SEV}(\mu \leq 13) = 0.9998$

Results from Default, Random, and Tuned Settings

- Experiments are performed at this stage (step AEX-3 from the active experimentation framework)
- We run SANN 100 times, first with default parameters (tmax = temp = 10), and second, with randomly chosen parameter values from the interval [1, 50]

Table 1 SANN results. Results from $n = 100$ repeats. Smaller values are better. The optimal function value is $y^* = 0.3979$

Model	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Default	0.3982	0.4037	0.4130	0.8281	0.5032	6.1120
Random	0.3988	0.5326	1.2160	2.0720	2.9820	8.8800
Tuned	**0.3979**	**0.3987**	0.4000	**0.4010**	**0.4022**	**0.4184**

Spurious Effects

Following Cohen, we define spurious effects as effects that suggest that a treatment is:

- effective when it is not
- not effective when it is

Two prominent examples for spurious effects are: *ceiling effect* and *floor effect*. If one wants to investigate performance differences between different methods, it is important to select the test problems so these differences indeed can occur.

- ceiling effect: the test problems are too easy, all algorithms "crash into the ceiling". (success rates of 100%)
- floor effect: test problems are too hard, most algorithms never obtain measurable progress. (all remain on "the floor")

Exploratory Landscape Analysis

Exploratory Landscape Analysis detects problem's properties first in order to make a reasonably information decision for some optimization algorithm. **Important Problem Properties**:

- **Multimodality**: In case of few local optima (Schwefel) a niching or time parallel method is suited and in case of many local optima (Rastrigin) one has to rely on multistarts or using large populations.
- **Global basin structure**: Rastrigin has a huge amount of local optima, but also a global basin structure due to the quadratic term. (it appears as a parabola). Problems without global structure are more difficult, we have to "look in every corner".
- **Separability**: If a problem is fully or partly separable, it may be partitioned into subproblems easier to solve.

Problem properties

- **Variable scaling**: The problem may behave very different in the single dimensions. It is important to perform small steps in some dimensions and large ones in others. CMA-ES algorithm handles this problems well.
- **Search space homogeneity**: Most benchmark sets are created by a structured, simple formula. Real-world problems don't behave like this.
- **Basin size homogeneity**: The basin size of the global optimum influences the hardness of a problem. Most niching EA methods assume similar basin sizes, so these methods are doomed to fail.
- **Size of plateaus**: Plateaus make optimization problems harder as they do not provide any information about the directions to turn to.

Exploratory Testing

- Attempts of experimentally acquiring property knowledge on expensive functions are usually performed manually.
- Sampling, dimension reduction techniques and especially visualisation are important techniques to obtain problem knowledge.
- Exploratory testing can focus either on global or local features of a problem. Global sampling may employ any space filling design which is useful to understand the nature of the problem.
- Testing locally makes sense if one needs to find out which properties of the problem lead to stagnation in the optimization process. However, for high-dimensional problems, running grid tests in all possible combinations is not feasible.

Conclusion

- If the problem properties are unknown, employing an exploratory landscape analysis is a good first step.
- Visualization has a key role in better understanding the algorithm performance.
- As a general conclusion, some more emphasis on experimental methodology is needed and much work is still left undone in this area.
- The cooperation between theory and practice should be improved. Theory should consider current experimental results as starting points, and established theory should be validated by means of structured experimental analysis.