Introduction to Metaheuristics
Conclusions & Bonus Track

DrC. Alejandro Piad Morffis

Review: The metaheuristic framework

- given a black box function F
- define stop criteria
- while not stop do:
  - generate solutions
  - evaluate solutions
  - update global best
  - learn something about F
- return global best

Review: What can we learn...

... in local search?:

- Gradients
- Features of local optima

... in evolutionary search?:

- Components of good solutions

... in swarm optimization?:

- Features of the global landscape
- Features of local optima

... in EDAs?:

- Explicit models of the function

Which metaheuristic is the best?

All metaheuristics learn an implicit or explicit model of F. That model encodes an *inductive bias* from the designer of the metaheuristic.

So which is the best?

Which metaheuristic is the best?

All metaheuristics learn an implicit or explicit model of F. That model encodes an *inductive bias* from the designer of the metaheuristic.

So which is the best?

No Free Lunch Theorem (informally): *in search and optimization problems, the computational cost of finding a solution, averaged over all problems, is the same for any solution method.*

What does this mean?

Which metaheuristic is the best?

All metaheuristics learn an implicit or explicit model of F. That model encodes an *inductive bias* from the designer of the metaheuristic.

So which is the best?

No Free Lunch Theorem (informally): *in search and optimization problems, the computational cost of finding a solution, averaged over all problems, is the same for any solution method.*

What does this mean?

On average, there is no objectively better solution **unless** you exploit some characteristic of the problem.

Is there really no free lunch?

To formally prove no-free-lunch we need to average over **all** possible functions (mappings), the vast majority of which are not interesting (e.g., are Kolmogorov random, i.e., uncompressible).

So the practical question remains: for the set of problems that *we care about*, is there a best algorithm?

Is there really no free lunch?

To formally prove no-free-lunch we need to average over **all** possible functions (mappings), the vast majority of which are not interesting (e.g., are Kolmogorov random, i.e., uncompressible).

So the practical question remains: for the set of problems that *we care about*, is there a best algorithm?

If the *inductive biases* encoded in your metaheuristic are superior (more predictive of a solution's performance) than alternative biases, for most functions that your care about, then your metaheuristic is on average "better" (in terms of number of search steps).

Let's review some inductive biases

Local search:

- *Locality*: good solutions are close to other good solutions.

Evolutionary search:

- *Building blocks hypothesis*: good solutions are made up from good components.

Swarm intelligence:

- *Distributed knowledge*: global structure can be predicted from local structure.

EDAs:

- *Modeling fit*: the function can be accurately approximated by the chosen model.

Are there better inductive biases

Objectively, we cannot prove there is a universal set of inductive biases that work better than any other. This is the well-known philosophical problem of induction in Science.

Are there better inductive biases

Objectively, we cannot prove there is a universal set of inductive biases that work better than any other. This is the well-known philosophical problem of induction in Science.

However, some inductive biases can be said to be better than others, in a sense, if the first set is a superset of the second. That is, if bias A encodes a more general proposition that implies bias B.

Are there better inductive biases

Objectively, we cannot prove there is a universal set of inductive biases that work better than any other. This is the well-known philosophical problem of induction in Science.

However, some inductive biases can be said to be better than others, in a sense, if the first set is a superset of the second. That is, if bias A encodes a more general proposition that implies bias B.

In this sense, estimation of distribution algorithms often encode more general biases than other metaheuristics, because we can generalize a specific constructivist hypothesis to a probabilistic one (e.g., from GA to PBIL).

Comparing metaheuristics

Most metaheuristics are stochastic algorithms, running with the same hyperparameters on the same function doesn't guarantee the same result.

How do we compare random distributions?

Comparing metaheuristics

Most metaheuristics are stochastic algorithms, running with the same hyperparameters on the same function doesn't guarantee the same result.

How do we compare random distributions?

Using statistical hypothesis testing!

But comparing metaheuristics is tricky:

- Most commonly, people use a t-test.

Comparing metaheuristics

Most metaheuristics are stochastic algorithms, running with the same hyperparameters on the same function doesn't guarantee the same result.

How do we compare random distributions?

Using statistical hypothesis testing!

But comparing metaheuristics is tricky:

- Most commonly, people use a t-test.
- However, there's no guarantee fitness distributes normal!

Comparing metaheuristics

Most metaheuristics are stochastic algorithms, running with the same hyperparameters on the same function doesn't guarantee the same result.

How do we compare random distributions?

Using statistical hypothesis testing!

But comparing metaheuristics is tricky:

- Most commonly, people use a t-test.
- However, there's no guarantee fitness distributes normal!
- If so, you need non-parametric tests.

Comparing metaheuristics

Most metaheuristics are stochastic algorithms, running with the same hyperparameters on the same function doesn't guarantee the same result.

How do we compare random distributions?

Using statistical hypothesis testing!

But comparing metaheuristics is tricky:

- Most commonly, people use a t-test.
- However, there's no guarantee fitness distributes normal!
- If so, you need non-parametric tests.
- It gets worse if you're comparing multiple metaheuristics.

Other criteria to consider

- Are all metaheuristics run for the same number of function evaluations?

Other criteria to consider

- Are all metaheuristics run for the same number of function evaluations?
- Does computational cost matters? Or is function evaluation much more expensive?

Other criteria to consider

- Are all metaheuristics run for the same number of function evaluations?
- Does computational cost matters? Or is function evaluation much more expensive?
- How about memory cost?

Other criteria to consider

- Are all metaheuristics run for the same number of function evaluations?
- Does computational cost matters? Or is function evaluation much more expensive?
- How about memory cost?
- Did you run all the experiments, or reused published results?

Other criteria to consider

- Are all metaheuristics run for the same number of function evaluations?
- Does computational cost matters? Or is function evaluation much more expensive?
- How about memory cost?
- Did you run all the experiments, or reused published results?
- Did you fine-tune all metaheuristics with the same effort?

Avoiding premature convergence

The fundamental problem in metaheuristic design is balancing exploration and exploitation.

Exploration: search steps that discover new attraction basins.

Exploitation: search steps that improve a known attraction basin best estimate.

Some metaheuristics have explicit (hyper)parameters that control this balance:

- Temperature in SA
- Mutation rate in EAs
- Learning rate in EDAs

Why this balance matters

- If exploration is insuficient, you will converge to suboptimal attraction basins.
- If exploitation is insuficient, you will not converge at all.

Observations:

- Most metaheuristics have an implicit search step size, i.e., the average distance between newly created solutions and their parents.
- Search step sizes tend to diminish.
- Early exploitation cripples the capacity to search (catastrophic convergence).

Why this balance matters

- If exploration is insuficient, you will converge to suboptimal attraction basins.
- If exploitation is insuficient, you will not converge at all.

Observations:

- Most metaheuristics have an implicit search step size, i.e., the average distance between newly created solutions and their parents.
- Search step sizes tend to diminish.
- Early exploitation cripples the capacity to search (catastrophic convergence).

Key idea: Avoid small search steps early on.

Explicit convergence control

Thresheld Convergence:

- Introduce a step size metric (metaheuristic dependent)
- Measure the size of each search step to its parent(s)
- Reject or "push" solutions with step size smaller than some threshold
- Reduce the step size proportional to remaining FEs

This simple strategy has been shown to improve DE, PSO, and CMA-ES.

- Avoids catastrophic convergence.
- Requires knowing the search budget.

Optimizing stochastic functions

In a stochastic function, the value of $y = f(x)$ is a random variable.

Assume there's a deterministic function $g(x)$ such that

$$f(x) = g(x) + \epsilon$$

where $e \sim N(0, \sigma)$ for some unknown σ.

Why is this a problem?

Optimizing stochastic functions

In a stochastic function, the value of $y = f(x)$ is a random variable.

Assume there's a deterministic function $g(x)$ such that

$$f(x) = g(x) + \epsilon$$

where $e \sim N(0, \sigma)$ for some unknown σ.

Why is this a problem?

Comparing two solutions inside the metaheuristic loop now becomes comparing two distributions!

Optimizing stochastic functions

Some ideas:

- **Optimist**: assume $\sigma << f(x)$.

Optimizing stochastic functions

Some ideas:

- **Optimist**: assume $\sigma << f(x)$.
- **Naive**: evaluate $f(x)$ many times for every x.

Optimizing stochastic functions

Some ideas:

- **Optimist**: assume $\sigma << f(x)$.
- **Naive**: evaluate $f(x)$ many times for every x.
- **Sly**: assume $\sigma \to 0$ when $x \to x^*$.

Optimizing stochastic functions

Some ideas:

- **Optimist**: assume $\sigma << f(x)$.
- **Naive**: evaluate $f(x)$ many times for every x.
- **Sly**: assume $\sigma \to 0$ when $x \to x^*$.
- **Clever**: evaluate $f(x)$ incrementally to refine confidence intervals.

Multi-objective optimization

Instead of optimizing $F(x)$, we have $F_1(x), ..., F_n(x)$.

Key ideas:

- No globally optimal solution, but a Pareto frontier of non-dominated solutions.
- Can we weight-combine the functions?
- Maintain populations of non-dominated solutions, change comparison criteria.

Learning to search

Can we adjust hyperparameters automatically?

Learning to search

Can we adjust hyperparameters automatically?

Online: during the course a single search.

- ... adjust step size based on, e.g., rejected evaluations.
- ... combine several metaheuristics in hyperband.

Learning to search

Can we adjust hyperparameters automatically?

Online: during the course a single search.

- ... adjust step size based on, e.g., rejected evaluations.
- ... combine several metaheuristics in hyperband.

Offline: with previous knowledge about the function (gray-box optimization).

- Extract features from the function:
  - Domain and image ranges
  - Global structure
  - Landmarking
- Run many hyperparameter configurations for different functions.
- Train a machine learning model to predict optimal hyperparameters.

What's next?