Landscape analysis correlated with algorithm performance

Evolving algorithm selectors (or parameter tuners) which learn ~~[illegible]~~ which features are more important for their choices

Create EA Taxonomy of Landscapes with expected performance

Evolve explainable/interpretable selectors/tuners.

Bridging continuous and discrete optimization theory.
Can we make discrete spaces continuous? Would continuous optimizers work there?

- Especially for non-separable functions with interaction between discrete and cont. variables
- Analyze probability distributions over discrete spaces as a surrogate to solve discrete problems with continuous techniques (similar to EDAs)

- Closing the gap between $\Omega\left(\frac{n}{\log n}\right)$ and $O(n)$ binary unbiased complexity of OneMax

- Simplify existing work on lower bounds to accessible framework (powerful).

- Black box complexity results on multimodal benchmarks

- Runtime analysis under gray-box and white-box settings

Studing how local optima look like on real problems to build adequate theoretical benchmarks

- find theoretical problem models generating same properties, which can be analyzed GA drift etc.
- find generalizable theoretic toy problems where we can define the number of optima, the size of the attraction areas etc.
- Benchmarks of functions with the problem properties defined above (wher they can be parameterized)

↓ P-PEAKS

↓ Walsh decomp.

↓ (W)PUBO

Evolutionary Computation in quantum hardware

- Is it efficient to do co-evolution on quantum hardware?
- What aspects of quant. lend themselves best to EC?
- Can a basic EC be prototyped in existing emulators for quantum hardware?
- could completely new methods be found for quantum hardware?
- Or EAs to discover new quantum algorithms?

How can we better convey the significance of meta-heuristics given that a) they are abstract methods, and b) some applications (ex. combinatorial optimization) are hard for individuals to connect to their lives?

What can causal inference teach us about how problem instances (or algorithms and instances) relate to one another?

Could we use LLMs or OL to classify them or cluster them into similar groups given their descriptions

Can we develop a rigorous theory of asynchronous parallel algorithms, to better enable distributed apps?

Could we design as.. par.. alg.. that are easy to understand and therefore analyze to start developing their rigorous theory

Meta EAs

↳ changing the algorith itself using EA-appoaches

PERFORMANCE METRICS?

- self-adaptian in MOO and its routine anlysis

multi-objective optimisation

meta-meta-EA?

(can it be so meta that there is no free variable (i.e. it adapt itself))

Some theoretical tool develop or proof ideen shary.

→ for new proble in the middle zone like mix-integer problem.

→ or solvy continous problem with a discrete algoriteh.

↳ maybe find a way to do dynamic discretization of the problem so the discrete search space changes over time)

Landscape anaylysis for algorithms and ~~the~~ how understady is, what landscape is benefit for what question.

"Next-best Proof"
→ Automated recommender for incrementally generalizing proofs. (transfer learning from other proofs)

Gray-Box: How to incorporate problem knowledge

—

What is appropriate measure of the cost of gray box algorithm?

—

Transition boundaries from black to gray to white

Gray-box complexity

Can "landscape analysis" and "runtime analysis" be brought together?

Instead of "E(T)", what happens when we change encoding a feature?

—

Runtimes for sub-classes of optimization problems exhibiting certain features

—

Runtimes of landscape feature computations

Mathematical tools (like drift algorithms) to analyze runtime of [Evolutionary] Algorithms to solve continuous optimiz. problems.

What about mixed integer / real optimisation?

Theoretical runtime analysis of Evolutionary strategies on interesting test/Benchmark problems.

Simplified (1+1)-ES

- Multi-Criteria Runtime focusing on trade-offs of optimization criteria

ROBUSTNESS OF AI / Optimization Pipelines

Look at "typical" cases, instead of pathological ones.

WHAT IS A TYPICAL CASE?

- Robustness of Automated Algorithm Selection.
- **Multi-criteria-Aspect: Robustness as Crit.**

In discrete optimisation we look at different problem classes. In continuous optimisation all problems are treated as one big class - why?

(NO...) focus on feature-based optimization problem classes

What happens during the run of an algo? How does it optimize?

EXPAND BENCHMARKING TOOLS SUCH AS IOH PROFILER

How far does algorithm deviate from expected behaviour at each stage?

Dynamic automated algorithm configuration

What can we say about the performance of EAs on more complex/interesting problem classes?

Can we have general results holding for a large variety of problem classes?

What are the intresting variety of problem classes to look at?

Landscape features instead of problem classes

Understanding internal optimization criteria of generative large language models

---

More generally, given a system that makes decisions can you reverse engineer what it was trying to optimise?

Trade-off between costs of fitness evaluation vs "offline" computation.

To understand this, maybe theoretical results can be combined with empirical analysis which also talks about wall-clock time and not just about number of iterations?

VISUALIZATION OF RUNTIME RESULTS

Blend theoretical results with empirical results

Phases
Regions for Param

Deepen the theoretical understanding of Genetic Programming.

Landscape analysis of search spaces in GP

New methods for runtime analysis of algorithms with a variable individual size (like semantic trees in GP problems)

Runtime analysis on non-trivial multi-objective combinatorial optimisation problems, e.g. multi-obj. MST.

I would add the multi-objective shortest path problem

Analysis of both black-box optimizers and algorithms which are tailored to these problems (e.g., two-bits-flip mutation for MST)

↓

supported solution?

- Systematic Experimental Analysis on which EA's give a 2-APX in reasonable time on combinatorial problems.
- Extend to analysis on multi-obj. combinatorial → When do we reach an approximation of the Pareto-set in reasonable time?
- What is reasonable time? Polynomial? ~~High order polynomials not~~ Quadratic? ~~What~~ What level of approximation can be obtained in, say, quadratic time?
- This looks like a fixed-budget analysis ⇒ it can inspire new drift theorems or even other tools for analysis.

~~The~~ Algebraic results to improve the performance of search algorithms

New local search operators using local properties of a landscape. Or also global ones, which use some properties which are large-scaled (based on some distant points fitness)? Or even a combination of two?

... or even combinations of more than two.

~~What~~ When can we prove lower bounds on runtime that hold for:

- arbitrary mutation operators?
- arbitrary selection operators?

Is it possible for arbitrary operators? Is it possible to cluster / categorize operators & give bounds?

↳ Or to provide a general way to categorize problem instances into groups where lower bounds are easier to prove for operators?

Connections of operators & landscapes?
Discrete and continuous:
what is similar? what is different?
What operator capabilities we do NOT YET use?

- Switching to mathematical optim. from metaheuristic when possible.

- could approach this question in two contexts:
  - when we have access to gradient
  - when we do not, and must obtain estimates.

- could also discuss trade-offs - when should I use what opti-method?
  ↳ runtime? ↳ possibility?
  ↳ complexity?

- Or how to decompose complex instances into easy-to-solve and hard-to-solve parts: ex. to hybridize Meta-heuristics with, say, linear programming.

- Study how "grey box" operators could be designed from gradient information.

More Topics / Papers on basics on EA / GP would be cool
↳ And in particular, how to make EAs accessible and useful to non-experts (ex. mechanical engineers, operations researchers, physicists, etc.)

Talking about GP it would be interesting to understand better the impact of constraining the size of the programs or the size of the primitive function/set

Make classes of functions on which certain EAs work better than on functions from other classes. How to classify a real-world function?

Find new EA with the power of AI™ ?!?

↳ Synthesize 💡 ideas out of large language models ?

- The power of LLMs comes from massive exploitation of prior knowledge. How can we achieve the same with meta-heuristics ?
- Could we use LLMs as operators (some work has been made in neuroevolution)
- LLMs could be used as GP helpers creating/modifying code in different ways than we do with current GP types
- Can we make LLM specific for the field of EA?
- Is AI effective for parameter selection?

Metaheuristics: what to do with all those nature-"analogies" that flood the conferences?

↳ Can we replace them with a new taxonomy that unifies them into a few natural mathematical families? (In a more principled way than the current system of "EAs, PSO, DE, EDAs, ACO, = everything else?")

Is there away to classify their operators rather than the algorithms and an algorithm could be grouped based on what type of operators they use.

Is it possible to make a bijection between metaheur. and certain mathematical objects in order to understand what is different?

What can algebraic methods (ex. group theory) teach us about classes of combinatorial optimization problems?

- Formalisation of metaheuristics applied to comb. opt. with Group theory.

- Comparison with experiments
- Study the generalization capabilities of benchmark results
- Encourage more experimental papers at FOGA? Maybe many from this community aren't aware their papers would be welcome at FOGA.

Experiments can also create hints towards theoretical analysis of interesting phenomena. Requires good benchmarking tools.

Create ~~a~~ "empirical" and "theoretical" tracks reviewed separately - /AGREE

Pushing runtime analysis away from "toy problems" like OneMax toward more realistic problems that practitioners would be interested in...

Depends on problem class and instances; maybe involve parameterized complexity (k-bounded problems).

- Where are the limits of runtime analysis? How little "structural information" about a problem class is sufficient to give tight bounds for given algorithms? ~~FEar~~
- Identify problem „classes" and use their properties to adapt the EA
- Starting from a "basic problem": other similar (more complex) problems as perturbation of basic problem

- Blend EAs with other optimization paradigms.
- Particularly with exact methods, B&B, B&C
- Come up with theoretical frameworks for comparisons w/ exact methods.

Characterize problem classes for which this would be promising.

What are areas, where one can compete with exact methods.

- Algorithms for Optimization that learn from past problems solved and become better for new ones (all from the same "class" of problems).
  - What is the algorithm learning?
  - Identifying similarity of past problems
  - Precisely define the "class of problems"
  - It would be nice to have a standard way of encoding the information learned by an algorithm on past problems that could be used by future algorithms (maybe already exists?)

- Design and provide guarantees for our methods in the solution of machine learning problems

To add to this: create stronger connections to the machine learning community..

For example: HPO tasks - compare "new" and "old" methods, hybridize, improve. Or

- What classes of ML problems are evolutionary (non-gradient based) methods competitive?
- ML-algorithm often depend crucially on the "right" parameter settings. Use EAs to find good ones.

Optimisation without an explicit objective function (learning the objective)

Focussing on precision of result, divergently runtime.

Characterization of potential objectives

Dynamic environments, changing objectives

The algorithm "on-the-fly"

How difficult are certain objectives to learn?

- Algorithms for Optimization that learn from past problems solved and become better for new ones (all from the same "class" of problems).
  - What is the algorithm learning?
  - Identifying similarity of past problems
  - Precisely define the "class of problems"
  - It would be nice to have a standard way of encoding the information learned by an algorithm on past problems that could be used by future algorithms (maybe already exists?)

- Design and provide guarantees for our methods in the solution of machine learning problems

To add to this: create stronger connections to the machine learning community..

For example: HPO tasks - compare "new" and "old" methods, hybridize, improve. Or

- What classes of ML problems are evolutionary (non-gradient based) methods competitive?
- ML-algorithm often depend crucially on the "right" parameter settings. Use EAs to find good ones.

Optimisation without an explicit objective function (learning the objective)

Focussing on precision of result, divergently runtime.

Characterization of potential objectives

Dynamic environments, changing objectives

The algorithm "on-the-fly"

How difficult are certain objectives to learn?

Experimental analysis of continous algorith.

WHAT IS THU USE CASE

- More routine anlysis or convergence anlysis in evolutionary strategy.
- Make doing experiments cheaper
- consideng the design of Experient as a topic More than just a sectn. why do you desig expim

More reserch about neural networks

will be nice if we discuss the - kernel design.
- or weight selection.

combine it with ~~machine~~ ML?

See how GAs compare to known algorithms in that field

GENETIC ALGORITHMS ARE GOOD AT PRUNING THESE, IT'D BE INTERESTING TO SEE, HOW SMALL WE CAN GET

EA PERFORMANCE COMPARISON

2) ~~How~~ More sensible ways to compare CoEAs in ~~MAXIMA~~ different solution concepts.

3) - More criteria for performance that is not just a single number

Let $\mathcal{P}(\mathbb{R})$ denote the set of probability distributions on $\mathbb{R}$.

Given a deterministic map

$$F: \mathcal{P}(\mathbb{R}) \to \mathcal{P}(\mathbb{R}),$$

what can we understand about D_0, $F(D_0)$, $F(F(D_0))$, ...?
(based on properties of F).

- When does the sequence converge? Rate of convergence?
- What metrics on $\mathcal{P}(\mathbb{R})$ to use.

Maybe useful for "better"/more precise statistical analysis and similar evaluation methods?

↳ What can we learn here that innovates beyond traditional EDA convergence theorems?

How can we make our field more attractive in the day and age of a booming ~~AI~~ neural network field - while everything else seems to get ignored? (Every other machine learning field)

Is is it just a subjective feeling?

- Is it possible to study the intersection of NN and EC as due in neuroevolution not only for weight search but also for architecture search.
- When is it better to use Metaheuristic to train NN than Adam/SGD?

~~• Benefits of EC~~

- Understandability of solutions as possible benefit of EC.

- Can EA be made more efficient by "partial" function evaluations? (Is possible this is already well-studied).
- Combine with algorithms to find "worst" performing coordinate?
- In the style of Successive halving or iRace — which use time series to decide when to "give up" on a solution.

~~[illegible]~~

More benchmarking frameworks for theoretical and experimental analysis

IDENTIFY WHAT IS MISSING IN CURRENT FRAMEWORKS

Design of benchmarks maximizing performance spread of algorithms

Feature analysis of problem classes.

Propose benchmarks that are suitable for both, theoretical and empirical studies

Game benchmarks for theoretical analysis

THEORY OF EA DESIGN FOR REAL WORLD APPLICATIONS

Adaptive and hybrid approaches, theory on concept drift detection

---

AUTOMATIC ALGO DESIGN & CONFIGURATION

How to capture (iteratively) "user" requirements?

Learning of user preferences

User-focused theory

Future FOGA shall...

Study theory-parameters around what it takes for EAs to be useful to users.

For example:

→ How diverse do solutions need to be for providing multiple options for decision support??

It might be useful to consider fixed-budget runtime analysis more since users generally know what budget they have and parameters might be different optimal for different budgets

also fixed-target runtime analysis for a specific application

PTO:

Matching properties of real-world problems to EAs that are theoretically understood to work well on such problems.

ERL

ALSO, LANDSCAPE ANALYSIS

1) Runtime Analysis or Regret Analysis ↓ Evolutionary reinforcement learning.

how to combine these two

NFL

2) No-Free-Lunch like analysis in multi-objective optimisation or Maximin optimisation.

more.

More Maths Tools can be introduced into the ~~ad~~ analysis in **Evolutionary Computation**