From benchmarking optimization heuristics to dynamic algorithm configuration

Vermetten, D.L.

Chapter 7

Conclusions

Throughout this thesis, we have explored iterative optimization heuristics for continuous optimization. We have shown that rigorous benchmarking does not only improve our understanding of the algorithm but also highlights avenues for further exploration. Through the use of modular design spaces for algorithms, we can fine-tune our optimizers to specific landscape properties, or create high-level selectors which exploit complementarity between the available solvers. In fact, observing the anytime performance of large algorithm portfolios shows that a dynamic approach to the algorithm selection problem has the potential to lead to even larger performance gains.

In Chapter 3, we addressed our first research question: *How can robust benchmarking pipelines be made accessible and resulting data be made usable by the wider community?* We introduced IOHprofiler as a modular environment for benchmarking iterative optimization heuristics, which provides both a way for setting up benchmarking studies via IOHexperimenter as well as an accessible interface for the analysis of the resulting benchmark data in IOHanalyzer. Using this framework, we showed how a robust benchmarking pipeline can be used in combination with a wide variety of problems. By focusing on the BBOB suite, we investigated some commonly overlooked aspects of the instance generation procedure and how this interacts with commonly used landscape analysis methods. This highlights the relation between the setup of a benchmark study and the ways in which we draw conclusions from the resulting data.

To close out Chapter 3, we note that benchmarking is more than just performance-oriented comparisons between algorithms. By looking at the concept of structural bias, we illustrate how behavior-based benchmarking can be used to gain insights into the inner workings of an algorithm, and the potential biases therein.

In Chapter 4, we explored our second research question: *How can a modular design aid in the exploration of interactions between different algorithmic ideas?* We discussed two modular algorithms: modCMA and modDE, each of which encompasses a design space with thousands of potential configurations. By making use of algorithm configuration techniques, we illustrated the complementarity which exists between different modules, since the best-performing configuration differs significantly per benchmark function. We also highlight that tuned configurations of these modular algorithms can clearly outperform existing versions of the respective algorithms. Additionally, the tuning procedure can be used as a way of incrementally assessing a new modules contribution to the existing design space. While these results show a promising direction for future assessment of algorithmic ideas, there are several challenges inherent to our proposed approach. Most critically, we showed that the inherent stochasticity of the considered algorithms has a drastic impact on the stability of the results obtained by algorithm configuration methods, suggesting a need for better noise-hanling methods.

The research question discussed in Chapter 5 was: *To what extent can we exploit performance complementary between different algorithms by switching between them?* Starting from a large set of benchmarking data, we showed that different algorithms perform well during different parts of the optimization process. By assuming we can freely switch between them, we observed significant potential performance gains from dynamic algorithm selection. By first focussing on switching between configurations of a modular algorithm, we sidestepped the question of warmstarting to show that performance can indeed improve by changing the configuration during the search. We then investigated per-run dynamic algorithm selection, where we utilize information from the first algorithm to determine which algorithm to switch to, where the switch incorporates a warmstart of the state of the secondary algorithm. Finally, we tackled the question of when the switch should occur, by transitioning to a sliding-window approach where we predict the relative benefit of performing a switch versus sticking with the original algorithm. Such a model could in future be used to create fully dynamic algorithm selectors which can switch multiple times throughout the search.

The final research question, discussed in Chapter 6, was: *How can we fairly judge the performance of meta-learning methods in the context of black-box optimization?* This question was inspired by observations from the previous chapter, where different algorithm selection techniques showed promising performance on the BBOB suite, but failed to generalize to a similar suite from a different platform. To better understand this seeming lack of generalizability, we introduced MA-BBOB, a problem generator based on the BBOB suite which uses affine recombination to create new benchmark

problems with known global optima. This generator was subsequently used to investigate the link between function landscapes and algorithm performance, resulting in an experiment which shows that generalizability of algorithm selection results is still lacking.

7.1 Key Findings

7.1.1 Chapter 3

- Benchmarking can be made accessible, and by doing so we can gain new insights into both algorithms and problems. Specifically, IOHprofiler allows us to investigate problems from new domains, such as star-discrepancy computation, and tackle them using state-of-the-art solvers. This highlights where further algorithm development is required, since the used algorithm portfolio failed to convincingly outperform a random search baseline.

- Benchmarking environments can not only function to interface problems with algorithms, with data annotation and sharing options such as ontologies, which allow for a wide variety of ways in which to combine data from separate sources and gain new insights.

- Instance generation mechanisms such as the one used in the BBOB suite can be very useful to test algorithm invariances, but care should be taken when using them in a box-constrained setting, since the used transformations necessarily change the part of the landscape the algorithm interacts with. This can be seen when looking at the ELA features of different BBOB instances, many of which are different in a statistically significant way. Looking at the instance generation procedure also highlights potential biases in the BBOB suite, e.g. with regard to the possible locations of the global optima, which should be kept in mind when using them in future benchmarking studies.

- Benchmarking is not limited to only looking at the performance of an algorithm. We can design more behavior-oriented benchmarks to gain an understanding of different algorithmic aspects, for example, its structural bias to certain regions of its domain. Knowing whether an algorithm is biased towards e.g. the center of the domain can be combined with knowledge about the biases of different benchmark suites to identify potentially misleading comparisons.

7.1.2 Chapter 4

- In addition to enabling a fair comparison of different algorithmic ideas within the same overarching framework, modular algorithm design also creates opportunities to explore interactions between many algorithm modifications which have been proposed in isolation. By combining this design principle with algorithm configuration tools, we find that well-configured module settings can lead to significant improvement over common variants of the same base algorithm.

- While algorithms can be configured to perform well over a large set of benchmark problems, configuring for performance on individual functions leads to additional improvements, highlighting the inherent complementarity present in these large configuration spaces.

- Algorithm configuration is an inherently noisy problem, and while most algorithm configurations incorporate various strategies to overcome this noise, the relation between the level of variance and the best noisy selection technique is not yet clear, which can lead to 'lucky' configurations being selected over those with actual better performance.

7.1.3 Chapter 5

- Just as algorithm selection can take advantage of complementarity between solvers on different types of problems, dynamic algorithm selection can take advantage of complementarity on different parts of the search. While some of these advantages remain theoretical, we can obtain improvements in performance on several functions by switching between algorithms at certain stages.

- Switching between algorithm variants can be simplified by working with modular algorithms, where the question of warm-starting the second algorithm can be addressed by preserving the internal state of the algorithm. While finding optimal dynamic combinations of parameter settings becomes challenging because of the increased noise, we have shown that dynamic combinations can outperform their static counterparts on the majority of used benchmark problems.

- A dynamic algorithm combination does not always have to be determined before running the algorithm. By utilizing information collected from the trajectory of an initial algorithm, the most promising secondary algorithm can be selected

from a larger portfolio. This could allow us to take advantage of the per-run stochasticity of our algorithms, leading to a per-run algorithm selection scenario.

- If we can accurately predict an algorithm's performance from a small initial trajectory, we can use these models to identify whether a switch at any given point in the search would be worthwhile, which could form the basis for a truly online dynamic algorithm selector. Unfortunately, this adds another level of complexity to the meta-learning task and our current models are not robust enough for this purpose.

7.1.4 Chapter 6

- The Many-Affine BBOB generator provides a new set of problems to validate current results on the generalizability of algorithm selection methods trained on BBOB. By showcasing that this generalizability is severely lacking for simple ELA-based models, we highlight the importance of further research into both the features we use to represent problems based on limited samplings, and the potential over-reliance on the same set of problems to guide algorithmic developments.

7.2 Future Work

As this thesis takes a wide view on benchmarking and its implications for various meta-learning scenarios, many open questions and areas for further research remain. Here, we highlight a few of the most relevant ones:

- **Data Accessibility.** As this thesis illustrates, there are many potential uses of benchmark data beyond the comparison of algorithm performance. As such, data which is collected for one study can, and often should, be re-used in other areas. Benchmarking tools play a critical role in facilitating this aspect, and while tools such as OpenML [237] have been widely adopted in the machine learning community, examples of data repositories in the optimization domain are less common. Repositories such as COCO's data archive [4], while valuable sources of performance data, are still somewhat limited in their usability because of a lack of meta-data, specifically information about the used algorithm implementation, and available code. From the metadata perspective, data ontologies such as OPTION [136] could provide a way to link different repositories together, while

efforts to increase reproducibility [151] within the wider community might make sharing code and data the norm rather than an exception.

- **Judging New Algorithms.** As the field of iterative optimization heuristics continues to grow, an ever-increasing number of algorithms will be proposed. While many of these might contain interesting algorithmic novelties, the way in which these contributions are judged has to adapt to keep up with the increases in scale. Current papers too often rely on comparison to a very limited set of baselines, where only average performance over a large set of functions is considered. This comparison should be extended to include established algorithms and be judged not on aggregate performance, but for example on contribution to this overall portfolio of established algorithms. In addition to this, a larger focus should be placed on unambiguous descriptions of the algorithmic components, rather than metaphors used to motivate their novelty.

- **Achieving DynAS.** In Chapter 5, we showed the potential performance to be gained from different versions of dynamic algorithm selection. However, throughout the experiments described in that chapter, we notice that the resulting performance is not yet stable, sometimes failing to beat even the static algorithms. Since DynAS consists of several interacting components, improvements in each of these aspects are needed in order to achieve usable dynamic switching behavior, with the two core components being:

  - Warmstarting: when performing a switch, we need to be careful not to lose information obtained at the beginning of the search. Depending on the algorithms used, we might need to initialize a population, stepsizes, covariance information... The way in which this information transfer is performed has a significant impact on the final performance, and as such should be carefully designed.
  - Deciding when to switch: in order to achieve optimal performance, a switch between algorithms might need to occur at different points in the search for different runs. As such, the current algorithm needs to identify the point at which it is most promising to transition to a new algorithm, based only on the information it has obtained so far. While trajectory-based ELA features seem to be somewhat promising, their variance poses a significant challenge, and other techniques might need to be considered as well.

- **Understanding Generalizability.** With the growing popularity of methods

such as algorithm selection or even dynamic switching, the question of how to fairly judge these meta-algorithms becomes more prominent. We have seen that classical methods such as leave-one-problem-out don't fit well with the available benchmark sets, and performance on one suite does not necessarily translate to others. In order to create a larger testbed, more attention has to be placed on the analysis of different benchmark suites and their complementarity, resulting in larger sets of training and testing functions. While problem generators like MA-BBOB might be a useful component in this process, further analysis into the wide variety of benchmarks is required to make more informed decisions.