A Comparison of Nelder-Mead and Bayesian Optimization for Mixed-Initiative Exploration of Machine Parameters

Andrew Head
UC Berkeley
Berkeley, CA, USA
andrewhead@eecs.berkeley.edu

ABSTRACT

People frequently work with machines to produce artifacts. While they know their end goal, it can be difficult to express it with the parameters the machine makes available. This paper studies algorithms to help users explore a space of unfamiliar machine parameters in pursuit of a known goal. It aims to understand the performance of two algorithms in terms of speed of convergence to the human's goal, and the human perceptions of randomness, convergence, and success. The algorithms are Nelder-Mead and a variant of Bayesian optimization. I build for front-end for each one where a user provides comparison-based ratings to guide the algorithm. With a 28-participant Mechanical Turk study, I compare the algorithms. While a Bayesian Optimization algorithm appeared more random to participants, they felt that it better achieved the goal. Nelder-Mead appeared to get better over time to participants, but also looks like it gets stuck. The study justifies a systematic evaluation of the affordances, human perceptions, and performance of algorithms for exploring parameter spaces.

ACM Classification Keywords

H.5.2. User Interfaces: Training, help, and documentation. Interaction Styles.

Author Keywords

mixed-initiative interaction; user preference modeling; numerical optimization; fabrication;

INTRODUCTION

We frequently use machines to tailor things to our subjective preferences. We may begin our day by adjusting a dial on a toaster to cook it to the right crispness, and later adjust the brightness of an image in a photo-editing program using a slider. Fabrication devices with complex mechanical operation are increasingly entering home and hobby spaces. Having a set of 'go-to' presets is unlikely to be helpful when experimenting with new materials and designs. To achieve their subjective preferences in unfamiliar circumstances, users will need a way to quickly experiment with parameters.

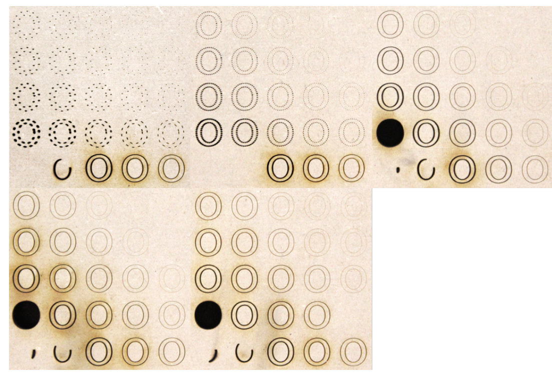


Figure 1: The letter 'O' engraved into a piece of particle board with a laser cutting machine. Each 'O' is engraved with a different configuration. Each of the five cells was engraved with a different PPI ('pulse per inch') In each cell, the laser's speed grows along the x-axis, and power along the y-axis. If a user wants fine-grained control over appearance, they need to express these parameters with a slider-based interface.

Today's interfaces are not well-equipped to help users who are seeking to configure these machines for the first time. One challenge is that users are reluctant to read documentation on how to use technologies. It is well established that software users don't read detailed documentation [5]. Another problem is that these machines have unfamiliar conceptual models (see Norman's definition [17]). It is hard for users to conceptualize the impact that parameters like laser power, speed, and "PPI" will have on a workpiece (see Figure 1).

This paper adapts two algorithms to help a user explore a parameter space to achieve their subjective preferences. It reports on user perceptions of trustworthiness, randomness, and accuracy of the methods. This is the first step in a systematic evaluation of algorithms for this problem. It focuses on the output of a laser cutter, a fabrication machine some scholars envision as being part of the home workshop of the future.

Both algorithms take *comparisons* of configurations as input. To justify this, I refer to a discussion from [3]. Human evaluation of single choices can be unreliable due to drift and anchoring. In addition, research in psychology [20] and economics [12] has established human models of choice, and explored phenomena such as preference refinement. This past work suggests humans are capable of producing good comparisons. For future work, it provides a lens for evaluating the cost of providing comparison-based inputs to algorithms.

RELATED WORK

Optimization to Match User Preferences

This paper considers algorithms that help a user find an ideal configuration with a small number of tests. Methods to serve this purpose have been around for a long time. Some early attempts at this problem are in the domain of optimal experiment design [2]. Optimal experiment design aims to reduce the number of scientific tests necessary to develop an empirical model of a phenomenon. Because of the high cost of experiments, techniques including steepest ascent have been proposed to help researchers rapidly converge on an input space of interest [2].

Appropriate techniques for optimizing unknown, costly objective functions have been proposed in many additional directions. These include the Nelder-Mead method [16], evolutionary strategies (e.g., CMA-ES [9]), Bayesian optimization [3], and reinforcement learning [19]. Active learning, similar to optimal experiment design, aims to learn a model with a small number of strategically-chosen samples [6, 18].

This work ultimately aims to lessen the effort to train a machine to learn a human-provided goal. In this aim, it is similar to past work on robotic "clicker training" [11] [7]. Clicker training aims to teach robots complex behavior with a single binary input channel that can reward partial behaviors until a robot builds up the full behavior. This is a modified version of reinforcement learning, where a human judge offers rewards to a robot so that it can learn an ideal policy of how to act when it's in a certain state.

This work builds on past reviews of optimization techniques (e.g. [8, 13]). Rather than focusing on performance in this paper, we focus on human perception of algorithms, with an additional discussion of performance.

Interactive Fabrication

User activities in this paper focus on fabrication machines. This is due to the progressively lower cost of these machines, and predictions that they may someday become home and workshop appliances. A body of peripherally related work has focused on improving peoples' ability to express their design intentions with fabrication machines. In these works, the machine collects a user's intent based on models and sketches they provide and modifies a workpiece as a result.

FreeD [22, 21] is a semi-autonomous tool for carving shapes in 3D. A user provides a virtual 3D model to the machine as their goal. As they work with an uncarved workpiece, a user makes large-scale motions. The milling device changes its own speed to remove more or less material, based on its position relative to the virtual 3D model. The authors' take effort to minimize fabrication risk while allowing authentic engagement with raw material [21]. *Constructable* [15] augments a laser cutter, aiming to enable interactive construction, where users interact by sketching on a workpiece with a laser pointer. The system observes laser pointer motions and converts them into constraint sets for actions, which it performs on the workpiece [15, 14]. My work, in contrast to past work, enables user to achieve some optimal goal with a workpiece, rather than enabling creative engagement with material or speeding up the expression of toolpaths.

ALGORITHMS

I implement two algorithms that enable the discovery of user preferences. Here, I provide a brief introduction to each of the algorithms. I describe the steps taken to modify the algorithms to accept comparisons as input to drive optimization. I also describe how bounds are enforced for each one, the initial points of each algorithm, and the hyperparameters used.

The input space for each algorithm was discretized, with a log-scale for each of three dimensions: the laser cutter's settings for power, speed, and PPI. Power and speed ranged from 1–100%. Possible values for these two dimensions were: $[1\%, 3\%, 10\%, 32\%, 100\%]$. PPI ranged within 10–1000, with possible values: $[10, 32, 100, 316, 1000]$. In the space below, configurations are described as three-tuples: $(Power, Speed, PPI)$. For a discussion of how each of these parameters impact the laser cutter's operation, see [1].

Nelder-Mead Optimization

Nelder-Mead is an optimization method that iteratively moves vertices of a simplex towards the best-rated vertex [16]. For reflection, expansion, and contraction, we chose coefficients observed as the best in the original Nelder-Mead work [16]: $\alpha = 1$, $\gamma = 2$, $\beta = -1/2$, and a reduction coefficient of 1/2.

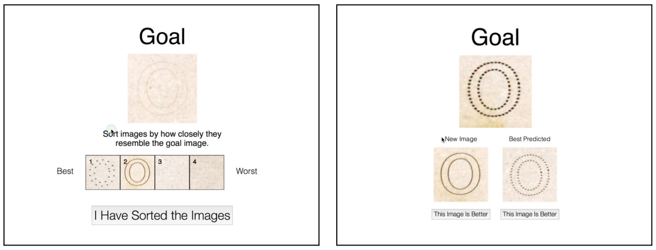
To perform Nelder-Mead optimization, the algorithm doesn't need to know an exact evaluation for each vertex. It only needs to know which is the best, second-to-worst, and worst points out of each set of vertices.

Initially, the vertices were: (1%, 1%, 10), (1%, 1%, 1000), (1%, 100%, 10), (100%, 1%, 10), These points were chosen for two reasons. First, at least four points are needed to migrate to all possible points in 3D space. (With any three points, all three and their centroid would reside on a single plane. Any reflection, expansion, contraction computed as a transformation of a difference vector on that plane yields another point on that plane.) Second, it was clear from viewing this collection of points that extremes of appearance were shown. For example, one image consisted of very sparse dots, another of very light lines, and another showed dark, thick lines.

Bayesian Optimization

Each iteration of Bayesian optimization comprises two steps [3]: fitting a Gaussian process to the data seen so far, and sampling a new value likely to maximize the unknown cost function. Typically, selection of the sample in the second step is performed by maximizing *expected improvement*. Based on the work by Brochu et al. [3], I express this as combination of two terms for exploration and exploitation:

$$EI(x) = (\mu(x) - f(x^+))\Phi(Z) + \sigma(x)\phi(Z) \quad (1)$$



(a) Nelder-Mead (b) Bayesian Optimization

Figure 2: Interaces to the optimization algorithms.

where $Z = \frac{\mu(x) - f(x^+)}{\sigma(x)}$ is in the domain of a normal distribution, representing the probability of improvement at x; $\mu(x)$ is the predicted value of a new input x; $f(x^+)$ is highest value seen so far; $\sigma(x)$ is the standard deviation at x; $\phi(Z)$ and $\Phi(Z)$ are the probability and cumulative density functions for Z.

Brochu et al. [3] [4] propose a variant on Bayesian optimization that takes comparisons between sampled points as inputs. I implement this algorithm for this study. I used a squared exponential kernel with $\sigma = 0.25$. In performing Newton-Rhapson optimization according to Brochu et al.'s formulation, σ_{noise} was set to 10, and ten iterations were performed to optimize the Gaussian process. These parameters were chosen by trial and error, observing what appeared to enable convergence in the target 3D input space for this study. Caching and limited iterations were implemented to ensure the algorithm would update to reflect user input in around one second, even after twenty comparisons had been given.

The algorithm was seeded with one comparison: (3%, 3%, 32) and (32%, 32%, 316). These two points were chosen to cover two distinct ends of the input space with distinct appearances.

In addition to the static bounds, the algorithm was restricted to select examples where the material had not burned during cutting. 'O's that had fallen out of the material, or incomplete jobs that were aborted due to flames, could not be chosen by the acquisition function. I expected that users viewing these samples would be confused as to why the jobs weren't complete and provide comparisons that did not fit nicely to a smooth model of user preference based on the input space.

INTERFACES

Nelder-Mead Rank Sorting Interface

When choosing how to move the worst point, the Nelder-Mead algorithm may need to know any of the following: the worst vertex, the best vertex, or the second-to-worst vertex. All of these can be collected at once. The current interface (see Figure 2a) asks a user to produce a ranked list of the vertices from best to worst. They can do this by clicking and dragging examples to put the best examples at the front of the queue.

When a user finishes ranking, they click a button. The simplex chooses candidate next points. If the algorithm needs clarification about candidate points (e.g., to determine if an expansion is better than a reflection), it inserts them into the list. The user ranks them among the original vertices to resolve the ambiguity. Once it replaces the worst point, the list is cleaned to show only the simplex's four current vertices.

Bayesian Optimization Pair Comparison Interface

With Brochu et al.'s variant on Bayesian optimization [3], an optimal point for a cost function that's expensive to evaluate can be found by collecting comparison ratings between pairs of examples. In their interface, a user was shown the best example so far, and an examples that maximized expected improvement. The user was prompted to make a choice about which of the two was better.

My interface for Bayesian Optimization is very similar to Brochu et al.'s (see Figure 2b). A user is shown the best rated example, and an example that maximizes the expected improvement. With each rating, both the best rated example and that which maximizes expected improvement are updated.

Both interfaces show the goal configuration at the top-center to guide users' exploration during the experiments.

EXPERIMENT

I recruited 28 participants on Mechanical Turk. We accepted workers for whom 95% of past tasks were approved. The design was between-subjects. Each participant was assigned either the Nelder-Mead interface or the Bayesian optimization interface, and one of two goal configurations. They were shown an image of an engraved 'O' produced with the "goal" configuration (see Figure 2). They were asked to rank example images based on their closeness to the goal as they attempted to lead the algorithm to the goal.

They completed the activity when they had either submitted 20 rankings, or had marked the goal configuration as the best. I chose 20 rankings for three reasons. First, it appeared reasonable from personal testing that participants could achieve an image close to the provided goals with the initial points chosen with that many rankings, with both interfaces. Second, the Nelder-Mead method seemed to converge to a set of very similar examples before reaching the 20th iteration. I wanted to avoid participant attrition when they felt like the algorithm was no longer responding to their rankings. Third, I needed to impose a limit on the number of rankings a participant submitted to let them finish the task an claim their compensation in a reasonable amount of time.

Participants were not allowed to end their participation by just reporting that they had reached the goal configuration—the goal configuration actually had to be presented to them by the algorithm, and they had to report it as the best one. This was to prevent participants from falsely reporting having achieved the goal in order to claim early compensation. Each participants were paid $1.00 for this HIT.

In a follow-up questionnaire, participants were rated their agreement with six statements on a 5-point Likert scale:

- I trusted that the algorithm would show me better images in each round.
- I don't understand why the algorithm chose the examples it did.

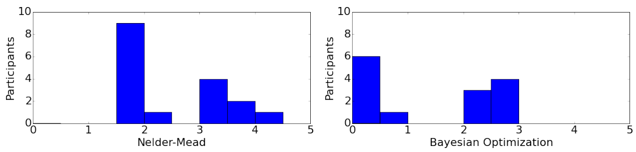


Figure 3: Distance of a participant's best-rated point and the goal configuration for each algorithm.

- The algorithm got stuck. It kept picking images that weren't what I wanted.
- The last image I marked as best was very similar to the goal image.
- The algorithm's guesses got better over time.
- The algorithm made random choices about what to show me.

These statements were chosen to capture participants' feelings of the algorithms randomness, trustworthiness and success in attaining a goal, and change in behavior over time. They were based on Hoffman's recommended metrics for evaluating robot fluency [10]. Although I was not evaluating fluency, I was interested in assessing perceptions of an algorithm's contribution, trust, and satisfaction, for which Hoffman provided initial ideas. The order of the questions was not randomized, but it should be in future versions of this study.

RESULTS

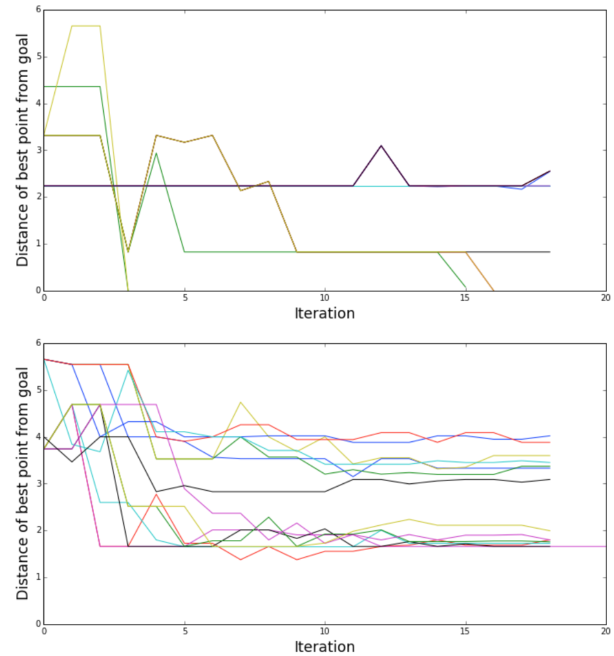
When discussing results, I refer to conditions with acronyms NM for Nelder-Mead, and BO for Bayesian optimization.

First, which algorithm was better able to help participants achieve their goals? In the BO condition, participants' best-rated configurations were much closer to their goal than in the NM condition (see Figure 3), By looking at longitudinal data (Figure 4a), it appears that by iteration 10, participants in the BO condition were, in general, closer to their goal than participants in the NM condition.

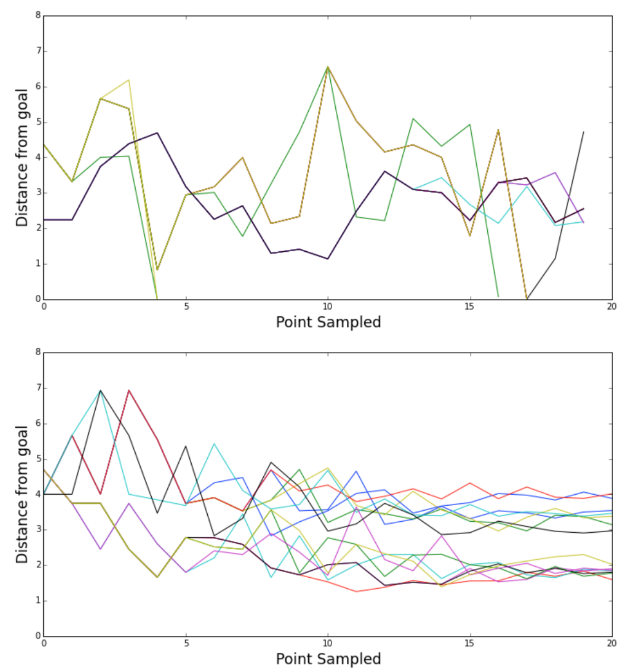
When we observe the paths of points provided by each of the algorithms, we see differing approaches in the points proposed. In its current form, participants using BO clearly covered more of the parameter space (see Figure 5). For NM, it appears that vertices were gradually pulled in toward some center where the simplex converged. In fact, this motion looks mostly planar. BO, in comparison, appears almost like random sampling. This may be in part due to our particular choice of kernel σ and noise standard deviation σ_{noise}.

When we consider the distance of each of the proposed points from the goal, we see why BO outperformed NM. Figure 4b shows how the distance of the latest point proposed changes over time for each algorithm. After about the tenth point sampled, it appears as if Nelder-Mead no longer suggests new examples radically farther from the goal. This may be an indication of NM prematurely converging to a local maximum. BO, on the other hand, continues to suggest points both near and far from the goal, and does not appear to converge at all.

Participants perceived these differences in how the algorithms worked (see Figure 6). Those in the BO condition were more



(a) With each successive ranking, the distance of a participant's best-rated point from the goal configuration.



(b) The distance of each new point an algorithm proposes from the goal configuration.

Figure 4: Distances of best-rated or proposed points from goal configurations. Each line represents one participant working with one algorithm. For each pair of plots, Bayesian optimization is shown on the top, and Nelder-Mead on the bottom.

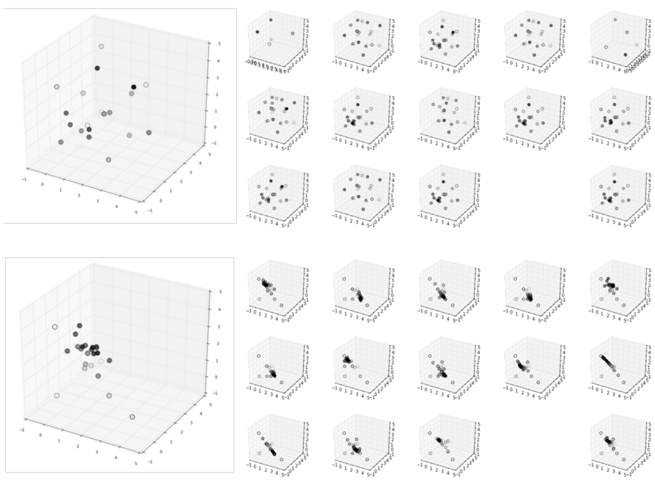


Figure 5: The points that each algorithm proposed. The first three rows show participants using Bayesian optimization. The bottom three rows show participants using Nelder-Mead. An enlarged exemplar for each method is shown to the left of the small multiples. The first point an algorithm proposes is shown in white; Each new point is more grey than the last, until the final point is black.

likely to report that the algorithm's choice of points seemed random. But these participants were also more likely to claim that the point they marked as the best was very close to the goal. Participants working with NM more often reported that the algorithm seemed to improve over time. However, it also seemed more likely to get stuck.

Limitations

Participants were forced to stop after reporting 20 rankings to the algorithm. I expect that that the Bayesian optimization algorithm would have seemed less random over time. The choice of hyperparameters for NM and BO might impact user perceptions of randomness and the best point a participant finds after 20 rankings. Before drawing any conclusions about these algorithms, a wider, systematically-determined range of hyperparameters must be tested.

Many of the participants in the BO condition achieved exactly the goal after only a few rankings (see Figure 5). It's likely that one of the goals was "just right" for participants in the BO condition to discover precisely the right condition after only a few comparisons. This suggests that this one goal was better suited to BO than NM for these tests, and weakens any potential claims about inherent strengths of BO at finding configurations close to the goal. To counter this side effect and improve generalizability, a larger number of goal images should be tested. Better yet, goal images could be randomly selected from the domain of the input space.

A PRELIMINARY LAB STUDY

Before the experiment above, I ran an in-lab study with three participants. As the motivating domain for this paper is fabrication, I sought to compare one optimization technique—Nelder-Mead—with a baseline interface for discovering the

Figure 6: Participant responses to questions about their perception of each algorithm. Blue bars show participants in the Bayesian optimization condition. Orange bars show participants in the Nelder-Mead condition.

right engraving settings. In particular, participants would view and rank physical, laser-engraved examples.

Participants used either the Nelder-Mead interface or a slider-based interface to guide the laser cutter to choosing engraving parameters. The slider-based interface was a near-direct copy of the software user interface for the Universal Laser Systems VersaLaser 3.50, a modern laser cutter. Just like in the original interface, there was no indication of how these parameters would affect how the laser engraved, or how the result would appear. The experimenter provided no description of the parameters.

Similarly to the MTurk experiment, participants were provided with a goal— a physical 2cm$\times$2cm tile with an engraved 'O'. This was cut on a soft particle wood 3mm thick. With the slider-based interface, participants moved sliders to change the values of power, speed and PPI to alter the engraving appearance. They were given pen and paper to take notes as they worked to understand how these parameters affected the appearance. With the Nelder-Mead interface, similar to above, participants ordered engravings from left to right based on their closeness to the goal. Unlike the MTurk study, participants in the lab could actually do this ranking with the physical workpieces. After ordering them on the table, they entered their ranking into the software interface. The experimenter fed the suggested parameters from the algorithm into the laser cutter to engrave each new example.

While participants' screens were recorded and all examples saved, I have yet to process this data. My intuitions are that participants were able to try out more examples with less time in-between with the Nelder-Mead interface than with the baseline interface. However, participants were also frustrated, confused, or skeptical when the Nelder-Mead interface converged on an maximum that was not their goal. Feedback from this study also inspired the author to improve the ranking interface to support insertion sort-based ranking mechanics, to avoid showing a reflection, expansions and contraction all at once, and to improve bugs that were uncovered.

CONCLUSION

People frequently have to work with machines to achieve some subjective goal with parameters they do not understand. This paper compares two algorithms to help humans achieve such goals by systematically exploring parameter spaces. While the Nelder-Mead optimization method appears less random to users than Bayesian optimization, the latter appears less likely to prematurely converge. Future work should address three issues. First, it should provide a systematic review of algorithms equipped to solve this problem and a comparison of human perceptions of these methods. Second, it should explore a wider range of mechanisms for expressing rankings, beyond just pairs and sorted lists. Third, it should do this in a domain-independent way, beyond just fabrication machines.

ACKNOWLEDGMENTS

Thanks to Jessica Hamrick for introducing me to psiTurk tool and helping me get access to it. Thanks also Anca Dragan, Chelsea Finn, Ben Caulfield and Manuel Sabin for their help choosing and tweaking the algorithms for this project.

REFERENCES

1. VersaLaser (VLS) User Guide, 2010.
2. BOX, G. E., AND DRAPER, N. R. *Empirical model-building and response surfaces*, vol. 424. Wiley New York, 1987.
3. BROCHU, E., CORA, V. M., AND DE FREITAS, N. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *arXiv:1012.2599 [cs]* (Dec. 2010). arXiv: 1012.2599.
4. BROCHU, E., FREITAS, N. D., AND GHOSH, A. Active Preference Learning with Discrete Choice Data. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2008, pp. 409–416.
5. CARROLL, J. M. *The Nurnberg funnel: designing minimalist instruction for practical computer skill*. MIT press Cambridge, MA, 1990.
6. COHN, D. A., GHAHRAMANI, Z., AND JORDAN, M. I. Active learning with statistical models. *Journal of artificial intelligence research* (1996).
7. GROLLMAN, D., AND JENKINS, O. Dogged Learning for Robots. In *2007 IEEE International Conference on Robotics and Automation* (Apr. 2007), pp. 2483–2488.
8. HANSEN, N., AUGER, A., ROS, R., FINCK, S., AND POSIK, P. Comparing Results of 31 Algorithms from the Black-box Optimization Benchmarking BBOB-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation* (New York, NY, USA, 2010), GECCO '10, ACM, pp. 1689–1696.
9. HANSEN, N., AND OSTERMEIER, A. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *, Proceedings of IEEE International Conference on Evolutionary Computation, 1996* (May 1996), pp. 312–317.
10. HOFFMAN, G. Evaluating fluency in human-robot collaboration. *International Conference on Human-Robot Interaction (HRI), Workshop on Human Robot Collaboration* (2013).
11. KAPLAN, F., OUDEYER, P.-Y., KUBINYI, E., AND MIKLÃŞSI, A. Robotic clicker training. *Robotics and Autonomous Systems 38*, 3 (2002), 197–206.
12. KINGSLEY, D. C. Preference uncertainty, preference refinement and paired comparison choice experiments. *Dept. of Economics. University of Colorado* (2006).
13. MERSMANN, O., PREUSS, M., AND TRAUTMANN, H. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In *Parallel Problem Solving from Nature, PPSN XI*, R. Schaefer, C. Cotta, J. KoÅĆodziej, and G. Rudolph, Eds., no. 6238 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Sept. 2010, pp. 73–82. DOI: 10.1007/978-3-642-15844-5_8.
14. MUELLER, S., KRUCK, B., AND BAUDISCH, P. LaserOrigami: Laser-cutting 3d Objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2013), CHI '13, ACM, pp. 2585–2592.
15. MUELLER, S., LOPES, P., AND BAUDISCH, P. Interactive Construction: Interactive Fabrication of Functional Mechanical Devices. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2012), UIST '12, ACM, pp. 599–606.
16. NELDER, J. A., AND MEAD, R. A Simplex Method for Function Minimization. *The Computer Journal 7*, 4 (Jan. 1965), 308–313.
17. NORMAN, D. A. *The design of everyday things*. Basic books, 2002.
18. SETTLES, B. Active learning literature survey. *University of Wisconsin, Madison 52*, 55-66 (2010), 11.
19. SUTTON, R. S., AND BARTO, A. G. *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.
20. TVERSKY, A., AND KAHNEMAN, D. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty 5*, 4 (1992), 297–323.

21. ZORAN, A., SHILKROT, R., NANYAKKARA, S., AND PARADISO, J. The Hybrid Artisans: A Case Study in Smart Tools. *ACM Trans. Comput.-Hum. Interact. 21*, 3 (June 2014), 15:1–15:29.

22. ZORAN, A., SHILKROT, R., AND PARADISO, J. Human-computer Interaction for Hybrid Carving. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2013), UIST '13, ACM, pp. 433–440.