Oulasvirta, Antti

Optimizing user interfaces for human performance

Published in:
Intelligent Human Computer Interaction - 9th International Conference, IHCI 2017, Proceedings

DOI:
10.1007/978-3-319-72038-8_1

Published: 01/01/2017

Optimizing User Interfaces for Human Performance

Antti Oulasvirta(✉)

School of Electrical Engineering, Aalto University, Espoo, Finland
antti.oulasvirta@aalto.fi

Abstract. This paper summarizes an invited talk given at the 9th International Conference on Intelligent Human Computer Interaction (December 2017, Paris). Algorithms have revolutionized almost every field of manufacturing and engineering. Is the design of user interfaces the next? This talk will give an overview of what future holds for algorithmic methods in this space. I introduce the idea of using predictive models and simulations of end-user behavior in combinatorial optimization of user interfaces, as well as the contributions that inverse modeling and interactive design tools make. Several research results are presented from gesture design to keyboards and web pages. Going beyond combinatorial optimization, I discuss self-optimizing or "autonomous" UI design agents.

Talk Summary

The possibility of mathematical or algorithmic design of artefacts for human use has been a topic of interest for at least a century. Present-day user-centered design is largely driven by human creativity, sensemaking, empathy, and creation of meaning. The goal of computational methods is to produce a full user interface (e.g., keyboard, menu, web page, gestural input method etc.) that is good or even "best" for human use with some justifiable criteria. Design goals can include increases in speed, accuracy, or reduction in errors or ergonomics issues. Computational methods could speed up the design cycle and improve quality. Unlike any other design method, some computational methods offer a greater-than-zero chance of finding an optimal design. Computational design offers not only better designs, but a new, rigorous understanding of interface design. Algorithms have revolutionized almost every field of manufacturing and engineering. But why has user interface design remained isolated?

The objective of this talk is to outline core technical problems and solution principles in computational UI design, with a particular focus on artefacts designed for human performance. I first outline main approaches to algorithmic user interface (UI) generation. Some main approaches include: (1) use of psychological knowledge to derive or optimize designs [1–3], (2) breakdown of complex design problems to constituent decisions [4], (3) formulation of design problems as optimization problems [5], (4) use of design heuristics in objective functions [6], (5) use of psychological models in objective functions [7,8], (6) data-driven

P. Horain et al. (Eds.): IHCI 2017, LNCS 10688, pp. 3–7, 2017.
https://doi.org/10.1007/978-3-319-72038-8_1

methods to generate designs probabilistically, (7) formulation of logical models of devices and tasks to drive the transfer and refinement of designs [9], and (8) learning of user preferences via interactive black-box machine learning methods [10]. I ask: Why is there no universal approach yet, given the tremendous success of algorithmic methods across engineering sciences, and what would a universal approach entail? I argue that successful approaches require solving several hard, interlinked problems in optimization, machine learning, cognitive and behavioral sciences, and design research.

I start with an observation of a shared principle across the seemingly different approaches: The shared algorithmic basis is *search*: "To optimize" is the act and process of obtaining the best solution under given circumstances. Design is about the identification of optimal conditions for human abilities. To design an interactive system by optimization, a number of decisions is made such that they constitute as good whole as possible. What differentiates these approaches is what the design task is, how it is obtained, and how it is solved. Four hard problems open up.

The first problem is the definition of design problems: algorithmic representation of the atomic decisions that constitute the design problem. This requires not only abstraction and mathematical decomposition, but understanding of the designer's subjective and practical problem. I show several definitions for common problems in UI design and discuss their complexity classes. It turns out that many problems in UI design are exceedingly large, too large for trial-and-error approaches. To design an interactive layout (e.g., menu), one must fix the types, colors, sizes, and positions of elements, as well as higher-level properties, such as which functionality to include. The number of combinations of such choices easily gets very large. Consider the problem of choosing functionality for a design: If for n functions there are $2^n - 1$ candidate designs, we already have 1,125,899,906,842,623 candidates with only 50 functions, and this is not even a large application.

The second problem is the definition of meaningful objective functions. The objective function is a function that assigns an *objective score* to a design candidate. It formalizes what is assumed to be 'good' or 'desirable' – or, inversely, undesirable when the task is to minimize. In applications in UI design, a key challenge is to formulate objective functions that encapsulate goodness in both designer's and end-users' terms. In essence, defining the objective function "equips" the search algorithm with design knowledge that tells what the designer wants and predicts how users interact and experience. This can be surface features of the interface (e.g., visual balance) or expected performance of users (e.g., 'task A should be completed as quickly as possible'), users' subjective preferences, and so on. However, it is tempting but naive to construct objective function based on heuristics. Those might be easy to express and compute, but they might have little value in producing good designs. It must be kept in mind that the quality of a interface is determined not by the designer, nor some quality of the interface, but by end-users, in their performance and experiences. I argue that an objective function should be essentially viewed as a predictor:

a predictor of quality for end users. It must capture some essential tendencies in the biological, psychological, behavioral, and social aspects of human conduct. This fact drives a departure from traditional application areas of operations research and optimization, where objective functions have been based on natural sciences and economics. I discuss the construction of objective function based on theories and models from cognitive sciences, motor control, and biomechanics.

A key issue we face in defining objective functions for interface design is the emergent nature of interaction: the way the properties of the design and the user affect outcomes in interaction unfolds dynamically over a period of time in the actions and reactions of the user. A key issue is people's ability to adapt and strategically change. The way they deploy their capacities in interaction complicates algorithmic design, because every design candidate generated by an optimizer must be evaluated against how users may adapt to it. I discuss approaches from bounded agents and computational rationality toward this end. Computational rationality (CR) [11] assumes an ideal agent performing under the constraints posed by the environment. This assumption yields good estimates in performance-oriented activities, but complicates computation remarkably.

The third problem is posed by algorithmic methods. I discuss trade-offs among modern method, which can be divided into two main classes: (i) heuristics such as genetic algorithms and (ii) exact methods such as integer programming. Exact methods offer mathematical guarantees for solutions. However, they insist on rigorous mathematical analysis and simplification of the objective function, which has been successful in only few instances in HCI this far. Black-box methods, in contrast, can attack any design problem but typically demand empirical tuning of the parameters and offer only approximate optimality. Here the design of the objective function and design task come to fore. The choice of modeling formalism is central, as it determines how design knowledge is encoded and executed, and how interaction is represented.

Fourth is the definition of task instances. In optimization parlance, task instance is the task- and designer-specific parametrization of the design task: "What constitutes a good design in this particular case?" There are two main sources of information when determining a task instance. To capture a *designer's* intention, interactive optimization can be used. Characteristic of interaction design is that the objectives can be under-determined and choices subjective and tacit [12]. The known approaches in design tools can be divided according to four dimensions: (1) interaction techniques and data-driven approaches for specification of a design task for an optimizer, (2) control techniques offered for steering the search process, (3) techniques for selection, exploration and refinement of outputs (designs), (4) level of proactivity taken by the tool, for example in guiding the designer toward good designs (as determined by an objective function). Principled approaches like robust optimization or Bayesian analysis can be used. I discuss lessons learned in this area.

However, the designer may not always be able to report all design-relevant objectives. For a full specification of a design task, one may need to algorithmically elicit what *users* "want" or "can" from digitally monitorable traces.

This is known as the inverse modeling problem [13]. I discuss probabilistic methods for cognitive models. These may disentangle among beliefs, needs, capabilities, and cognitive states of users as causes of their observations. Alternatively, black box models can be used. The benefit of white-box models, however, is that they allow the algorithm in some cases to predict the consequences (costs, benefits) of changing a design on user.

To conclude, perhaps the most daring proposition made here is that essential aspects of design, which has been considered a nuanced, tacit, and dynamic activity, can be abstracted, decomposed, and algorithmically solved, moreover in a way that is acceptable to designers. I review empirical evidence comparing computationally to manually designed UIs. However, much work remains to be done to identify scalable and transferable solution principles.

Even more critical is the discussion of what "design" is. Interaction design is characterized as "the process that is arranged within existing resource constraints to create, shape, and decide all use-oriented qualities (structural, functional, ethical, and aesthetic) of a digital artefact for one or many clients" [14]. Some scholars go as far as claiming that interaction design is through-and-through subjective and experiential [15]. It is about conceptualizing product ideas and designing their behavior from a user's perspective. In this regard, computational methods still cover a limited aspect of design. Transcending beyond optimization, I end with a discussion of what *artificially intelligent UI design* might mean. I claim that "AI for Design" must meet at least five defining characteristics of design thinking: (1) agency, (2) problem-solving, (3) sense-making, (4) speculation, and (5) reflection. So far, no approach exists that – in a unified fashion and with good results – achieves this.

Acknowledgements. The work of AO has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 637991).

References

1. Dvorak, A., Merrick, N.L., Dealey, W.L., Ford, G.C.: Typewriting Behavior. American Book Company, New York (1936)
2. Fisher, D.L.: Optimal performance engineering: good, better, best. Hum. Factors J. Hum. Factors Ergon. Soc. **35**(1), 115–139 (1993)
3. Wickens, C.D., Kramer, A.: Engineering psychology. Ann. Rev. Psychol. **36**(1), 307–348 (1985)
4. Card, S.K., Mackinlay, J.D., Robertson, G.G.: A morphological analysis of the design space of input devices. ACM Trans. Inf. Syst. (TOIS) **9**(2), 99–122 (1991)
5. Burkard, R.E., Offermann, D.M.J.: Entwurf von schreibmaschinentastaturen mittels quadratischer zuordnungsprobleme. Z. für Oper. Res. **21**(4), B121–B132 (1977)
6. O'Donovan, P., Agarwala, A., Hertzmann, A.: Learning layouts for single-pagegraphic designs. IEEE Trans. Vis. Comput. Graph. **20**(8), 1200–1213 (2014)
7. Gajos, K., Weld, D.S.: Supple: automatically generating user interfaces. In: Proceedings of the 9th International Conference on Intelligent User Interfaces, pp. 93–100. ACM (2004)

8. Oulasvirta, A.: User interface design with combinatorial optimization. IEEE Comput. **50**, 40–47 (2017)
9. Eisenstein, J., Vanderdonckt, J., Puerta, A.: Applying model-based techniques to the development of UIS for mobile computers. In: Proceedings of the 6th International Conference on Intelligent User Interfaces, pp. 69–76. ACM (2001)
10. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: a review of Bayesian optimization. Proc. IEEE **104**(1), 148–175 (2016)
11. Gershman, S.J., Horvitz, E.J., Tenenbaum, J.B.: Computational rationality: a converging paradigm for intelligence in brains, minds, and machines. Science **349**(6245), 273–278 (2015)
12. Cross, N.: Designerly Ways of Knowing. Springer, Heidelberg (2006)
13. Kangasrääsiö, A., Athukorala, K., Howes, A., Corander, J., Kaski, S., Oulasvirta, A.: Inferring cognitive models from data using approximate Bayesian computation. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 1295–1306. ACM (2017)
14. Löwgren, J., Stolterman, E.: Thoughtful Interaction Design: A Design Perspective on Information Technology. The MIT press, Cambridge (2004)
15. Goodman, E., Stolterman, E., Wakkary, R.: Understanding interaction design practices. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1061–1070. ACM (2011)