Towards Flexible and Robust User Interface Adaptations With Multiple Objectives

Christoph Albert Johns
Aarhus University
Denmark
cajohns@cs.au.dk

João Marcelo Evangelista Belo
Aarhus University
Denmark
Saarland University
Germany
jbelo@cs.uni-saarland.de

Anna Maria Feit
Saarland University
Germany
feit@cs.uni-saarland.de

Clemens Nylandsted Klokmose
Aarhus University
Denmark
clemens@cs.au.dk

Ken Pfeuffer
Aarhus University
Denmark
ken@cs.au.dk

ABSTRACT

This paper proposes a new approach for online UI adaptation that aims to overcome the limitations of the most commonly used UI optimization method involving multiple objectives: weighted sum optimization. Weighted sums are highly sensitive to objective formulation, limiting the effectiveness of UI adaptations. We propose ParetoAdapt, an adaptation approach that uses online multi-objective optimization with *a posteriori* articulated preferences—that is, articulation of preferences after the optimization has concluded—to make UI adaptation robust to incomplete and inaccurate objective formulations. It offers users a flexible way to control adaptations by selecting from a set of Pareto optimal adaptation proposals and adjusting them to fit their needs. We showcase the feasibility and flexibility of ParetoAdapt by implementing an online layout adaptation system in a state-of-the-art 3D UI adaptation framework. We further evaluate its robustness and run-time in simulation-based experiments that allow us to systematically change the accuracy of the estimated user preferences. We conclude by discussing how our approach may impact the usability and practicality of online UI adaptations.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing systems and tools**.

KEYWORDS

multi-objective optimization, Pareto frontier, online UI adaptation

ACM Reference Format:
Christoph Albert Johns, João Marcelo Evangelista Belo, Anna Maria Feit, Clemens Nylandsted Klokmose, and Ken Pfeuffer. 2023. Towards Flexible and Robust User Interface Adaptations With Multiple Objectives. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29–November 01, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3586183.3606799

UIST '23, October 29–November 01, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0132-0/23/10…$15.00
https://doi.org/10.1145/3586183.3606799

1 INTRODUCTION

Optimization-based adaptive user interfaces (UIs) promise to provide user experiences that are tailored to individual needs and contexts. They automatically adjust the UI at run-time based on models of interaction cost [39], and apply to a variety of use cases and application domains; from changing the UI's layout [3, 24] and level of detail [31] to timing of notifications [51], and many others. The use of optimization methods in adaptive systems was motivated by their success supporting designers and researchers in UI development and automation of design processes [17, 38, 39]. However, applying optimization to online adaptation, it becomes evident that the objective function that determines the goodness of a design is only an approximation of the subjective criteria that determine an individual user's experience — that is, the utility of the design as perceived by the end-user. It is impossible to include and accurately model all the criteria that determine this utility for any user in any situation — a necessity, however, for successful UI adaptation. Thus, we require optimization approaches that are robust toward modeling errors and flexible to allow for various situational and individual requirements.

We propose ParetoAdapt, an optimization-based adaptation approach that generalizes across specific use cases and offers a first step toward more robust and flexible UI adaptation. Grounded in operations research, evolutionary algorithms research, and research on multi-criteria decision-making, it generalizes existing adaptation methods and is used to produce fully and semi-automatic online adaptations of 3D layouts. ParetoAdapt is motivated by reflections on the limitations of prior work that used weighted sums to balance multiple adaptation objectives. These works implicitly assume that the objective criteria are accurate and complete, that is, that they fully align with the user's perceived satisfaction criteria and that the perceived utility of the UI can be captured by a linear combination of the modeled objectives. In practice, these assumptions are often violated as subjective criteria are not modeled accurately or are missed entirely, which we further discuss below.

With ParetoAdapt, we apply multi-objective optimization with *a posteriori* articulation of preferences to online UI adaptation. This ensures three fundamental qualities that are critical for adaptation approaches to be useful in practice:

- *Robustness*: In situations where the optimization objectives are inaccurate or incomplete representations of the criteria that influence how a user perceives the utility of the UI, we show that ParetoAdapt provides better starting points to semi-automatically adapt a UI than a weighted sum approach.
- *Flexibility*: ParetoAdapt can be used for both fully and semi-automatic UI adaptation based on multiple objectives. It allows users to directly interact with the proposed adaptations to fit their individual and situational needs rather than specifying preferences over objective criteria.
- *Responsiveness*: ParetoAdapt can provide adaptation proposals in real-time to allow for online adaptation and integration of user feedback at run-time.

We demonstrate and evaluate our approach by implementing a 3D layout adaptation system based on the Adaptive User Interfaces Toolkit (AUIT) [3]. We examine how the choice of objectives, optimization method, and maximum number of adaptation proposals affect the effectiveness and runtime of ParetoAdapt compared to a weighted sum approach in simulation-based experiments that allow us to systematically vary the accuracy and completeness of the modeled objectives in relation to simulated user preferences.

In summary, we propose ParetoAdapt, a UI adaptation approach based on multi-objective optimization with *a posteriori* articulated preferences; that is, indication of preferred solutions after the optimization has concluded. ParetoAdapt makes UI adaptation more robust against modeling errors, requires minimal input about user preferences, and offers users control over the adaptation process. This is a first step toward making UI adaptation more robust and flexible and thus better applicable in practice.

2 BACKGROUND

We first introduce the central concepts and terminology used throughout this paper, before presenting related work on optimization-based approaches to UI adaptation and multi-objective optimization with *a posteriori* articulated preferences outside the domain of UI adaptation.

2.1 Multi-Objective Optimization

Multi-objective optimization, in the context of this work, refers to the optimization of a collection of objective functions that aim to model an end-user's preferences regarding the appearance or behavior of a UI. These preferences are opinions about the order or relative importance of subjective quality criteria and are primarily articulated either *before* the optimization has concluded over the set of objectives (*a priori*) or *after* the optimization over a set of potential solutions (*a posteriori*). Both approaches are explained in further detail in Section 2.2.

Problem Formulation. A general multi-objective optimization problem may be formulated as follows:

$$\begin{aligned} &\underset{\mathbf{x}}{\operatorname{argmin}}\, \mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}), F_2(\mathbf{x}), ..., F_k(\mathbf{x})]^T \\ &\text{subject to } g_j(\mathbf{x}) \leq 0 \text{ for } j = 1, 2, ..., m \\ &\text{and } h_l(\mathbf{x}) = 0 \text{ for } l = 1, 2, ..., n, \end{aligned} \tag{1}$$

where $\mathbf{F}$ is the set of k objective functions to be minimized using the design vector $\mathbf{x}$ as constrained by m inequality constraints g and n equality constraints h that define the feasible design space $\mathbf{X}$. In the context of UI optimization, the objective functions $\mathbf{F}$ may, for example, include ergonomic [15] or performance criteria [27] that depend on the UI's configuration $\mathbf{x}$ such as the UI's layout [3] or timing [51] and are constraint by inequalities g and equalities h that capture the limitations of the user's environment [15] or cognitive resources [31].

Pareto Optimality and Pareto Frontier. The solution to a multi-objective UI optimization problem is a set of designs $\mathbf{X}^*$ that satisfy some chosen definition of efficiency or optimality, most commonly the definition of Pareto optimality: A point in the design space (i.e., a UI configuration) is said to be Pareto optimal if no other feasible point exists that improves upon performance in one objective without degrading another [33]. The entire set of Pareto optimal UI adaptations, representing the various efficient trade-offs between conflicting design objectives, is called the Pareto front or Pareto frontier [33]. How this Pareto frontier is constructed and shaped can have a major effect on the effectiveness of adaptive UIs as we will demonstrate throughout this paper.

2.2 A Priori and A Posteriori Articulation of Preferences

Various methods, techniques, and algorithms have been developed to solve multi-objective optimization problems (see, for example, [33] for an overview). Two categories of these optimization methods are of particular interest to this study: optimization with *a priori* articulated preferences and optimization with *a posteriori* articulated preferences (cf. [33]).

A Priori Articulation of Preferences. In UI optimization with *a priori* articulated preferences, the end-user (or the designer acting as their representative) states their preferences *after* a set of relevant optimization objectives has been chosen and formulated but *before* any solutions have been identified [33]. In weighted sum optimization, the end-user may, for example, state their preferences regarding the layout of a text entry UI by choosing how each of the pre-defined layout objectives (e.g., related to input speed or accuracy) is weighted in a global linear sum representing the design's total utility or cost by anticipating the weights' effect on the generated adaptations. The globally optimal designs may then be identified by optimizing such a global criterion; for example, using exact methods like commercial linear solvers [31] or using approximate solvers like simulated annealing [3].

A Posteriori Articulation of Preferences. By contrast, multi-objective UI optimization with *a posteriori* articulated preferences aims to first generate a representation of the entire Pareto frontier and then to present all or a decomposed sample from this Pareto optimal set of potential adaptations to the end-user for selection [33]. In other

words, the user can state their preferences by choosing or ordering concrete UI designs, for example selecting a final keyboard layout from a collection of potential layouts trading off input speed and accuracy with varying priorities. Commonly, optimization with *a posteriori* articulated preferences involves evolutionary or genetic algorithms (e.g., NSGA-III [9], RVEA [12], or SMS-EMOA [4]) as these have been shown to successfully retrieve the Pareto optimal set for a wide range of multi-objective optimization problems. Our work advocates for the use of such *a posteriori* techniques in online UI adaptation as these may be more effective in producing desired trade-offs between conflicting design objectives. Our proposed approach retains the option, however, to utilize *a priori* articulated preference statements if desired.

Ref.	Pref. Artic.	Offline/Online	Applic. Domain
[5]	*A priori**	Offline	Text entry
[6]	*A priori**	Offline	Text entry
[14]	*A priori**	Offline	Text entry
[24]	*A priori**	Offline	Text entry
[40]	*A priori**	Offline	Text entry
[44]	*A priori**	Offline	Text entry
[45]	*A priori**	Offline	Text entry
[50]	*A priori**	Offline	Text entry
[2]	*A priori**	Offline	2D UI layouts
[20]	*A priori**	Offline	2D UI layouts
[21]	*A priori**	Offline	2D UI layouts
[3]	*A priori**	Online	3D UI layouts
[13]	*A priori**	Online	3D UI layouts
[18]	*A priori**	Online	3D UI layouts
[22]	*A priori**	Online	3D UI layouts
[31]	*A priori**	Online	3D UI layouts
[37]	*A priori**	Online	3D UI layouts
[49]	*A priori**	Online	3D UI layouts
[11]	*A posteriori*	Offline	3D touch interaction
[27]	*A posteriori*	Offline	Text entry
[28]	*A posteriori*	Offline	Physical input
[29]	*A posteriori*	Offline	Haptic feedback
[43]	*A posteriori*	Offline	2D UI layouts
[51]	*A posteriori*	Offline	Notification timing
Ours	*A posteriori*	Online	3D UI layouts

* Uses weighted sum optimization categorized as *a priori* by [33] also if weights are varied to obtain multiple solutions.

Table 1: Related work on multi-objective UI optimization by preference articulation (*a priori* or *a posteriori*), when the optimization is applied (*offline* or *online*), and their application domain.

2.3 Optimization of UIs

Adaptive UIs have been heavily influenced by developments in the field of UI optimization. Two important milestones for applying optimization techniques in UI design were the representation of design factors as decision variables in problem formulations and the modeling of design heuristics and psychological models in cost functions [39, sec. 4.2]. Pioneering work proposed novel objective functions and used existing optimization techniques to optimize keyboard layouts [10, 30, 52] (see [16] for an overview) and menu designs [1, 25, 32, 34].

Since then, multi-objective optimization approaches have been applied to the design and adaptation of, for example, text entry methods (e.g., [6, 14, 24, 40, 44, 45]) and 2D and 3D UI layouts (e.g., [13, 18, 22, 31, 37, 47, 49]). A key work in this area was the multi-objective optimization method proposed by Smith et al. [44] which utilized a weighted sum-based scalarization with varying weights to generate Pareto optimal keyboard layouts. This approach has since been applied by, for example, [6] and [24] to generate keyboard layouts for constrained gesture typing and one-handed text entry respectively.

In the domain of GUI optimization, Supple [19–21] framed GUI design as a scalarized multi-objective optimization problem based on navigation costs and demonstrated the effectiveness of this approach in adapting to both device and usage characteristics using an equally-weighted sum [19]. Since Supple, several works have utilized similar approaches to, for example, optimize the layout of menu systems [2] or websites [47]. Recently, alternative multi-objective optimization methods have been utilized in the domains of keyboard layout optimization [27], GUI layout optimization [43], computational design of physical input devices [28, 29] and parameter search for 3D touch interaction [11]. These efforts employ more complex optimization methods (i.e., genetic algorithms [27, 43] and Bayesian multi-objective optimization [11, 28, 29]) which overcome the limitations of weighted sum approaches and can find Pareto optimal configurations for a variety of problem definitions. Such methods have, however, not yet been applied to online adaptation.

Overall, multi-objective optimization of UIs has been dominated by optimization with *a priori* articulated preferences, specifically weighted sum-based methods, used at design time (see Table 1) and as such are subject to the limitations that these methods entail. We highlight and reflect on these limitations in a separate section below. As the recent uses of genetic algorithms and multi-objective Bayesian optimization demonstrate, however, alternative approaches exist that aim to address some of the shortcomings of UI optimization with *a priori* articulated preferences and weighted sum optimization in particular. This paper contributes to this growing area of research by exploring the use of multi-objective optimization techniques with *a posteriori* articulated preferences for online UI adaptation.

3 LIMITATIONS OF WEIGHTED SUM UI ADAPTATION

Online adaptation at run-time requires to determine a user's specific interaction needs or preferences in context. The objective functions modeled in optimization-based adaptation are thus likely only *inaccurate* and *incomplete* approximations of the true criteria that determine how an individual user perceives the utility of a UI in a specific situation. This can be particularly problematic if adaptations are computed using a weighted sum approach, which is the

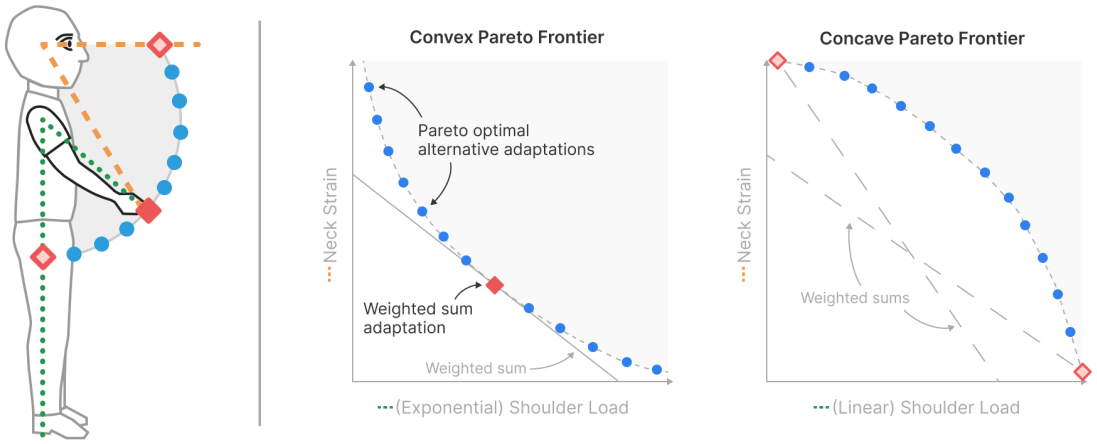


Figure 1: Left: Illustration of a neck strain and a shoulder load objective for UI optimization. Right: Pareto plots for an exponential and a linear formulation of the shoulder load objective.

basis of many optimization-based adaptive systems, as discussed in the previous section. In particular, we see three major issues:

- *Sensitivity to objective formulation*: Weighted sum optimization cannot yield UI configurations on non-convex regions of the Pareto frontier, no matter how the weights are changed [33]. This may be problematic if these Pareto optimal configurations would be preferred by the user.
- *Poor predictability*: Stating preferences using weights before the optimization is run can be intransparent and indirect for the decision-maker (i.e., the designer or end-user) since they need to anticipate the effects of the weights on the resulting adaptations. These effects are highly dependent on the objective formulations and may thus cause frustration for the user.
- *Poor scaling*: Weighted sum UI optimization scales poorly with the number of adaptation proposals that are desired by the designer or end-user as it needs to be run repeatedly, potentially making semi-automatic adaptations, experimentation with weights, and exploration of adaptation trade-offs impractical.

To illustrate these points, consider the following example.

Example: Adaptive Mixed Reality Layouts. A designer wants to create an adaptive mixed reality application that continuously optimizes the ergonomics of its UI. The position of a UI element should adapt to minimize the stress on the user's neck and on the user's shoulder when interacting with the application (see Figure 1, left). The designer creates two objective functions: a neck strain objective dependent on the head tilt required to view the UI (in orange), and a shoulder load objective as a function of the arm lift required to touch the UI (in green). Both objectives cannot be optimized simultaneously but trade off against each other. If the UI is placed closer to eye level, decreasing neck strain, it necessarily increases the shoulder load and vice versa. Next, the designer decides to ensure general visibility and reachability of the UI by adding constraints that limit the UI's potential adaptations to those in front of the user at arm's length. Finally, to increase the level of control that users can exert over the adaptation process, the designer decides to let users set the optimization weights, representing their preferences over the objectives, themselves. How exactly the designer chooses to model the ergonomic criteria can now heavily affect the user experience and adaptation quality.

The designer may choose a linear form for both the neck and shoulder objective, that is, as the head tilt/arm lift increases, the estimated load steadily increases. The Pareto frontier for such a formulation (i.e., the set of adaptations representing the efficient trade-offs between the shoulder and neck strain) will be concave (see Figure 1, right). Choosing an exponential form for either the neck or the shoulder load objective (e.g. the higher the arm is lifted the more the perceived load increases), the resulting Pareto frontier will be convex (see Figure 1, center). Which formulation is chosen may depend on the specific metric the designer is consulting, their modeling accuracy, or their domain knowledge. Whether it accurately represents the perception of the user will vary between individuals and different situations.

In the mixed reality application, the UI element's positions along the Pareto frontier will look identical for both scenarios (see Figure 1, left). In the convex case, a weighted sum optimization will be able to reliably achieve compromise solutions, that is, choose a UI position on the Pareto frontier that trades off neck and shoulder load depending on the chosen weights. In the concave case, however, the weighted sum approach can only yield two different positions, either at eye or at waist level (the red open diamonds in Figure 1, left), which represent the optimal points for one of the two objectives. Other points on the concave Pareto frontier cannot be reached via a linear combination of the two objective criteria. Thus, with a weighted sum adaptation, the system is lacking **robustness** to the scaling and shape of the optimization objectives.

If the user systematically changes the weight combinations from 0 to 1, the position of the UI element will not change until it suddenly jumps from eye level to waist level or vice versa. Thus, the system poorly reacts to changes in the objective weights, limiting the system's **flexibility** towards the needs of different use cases and users' priorities. Furthermore, in any of the above cases, the user will have to repeatedly run the optimization to gain an understanding of the relationship between the weights and the

resulting adaptations. Depending on their computational resources, this may cause the **responsiveness** of the adaptive UI (i.e. speed and efficiency) to drop.

4 PARETOADAPT: PARETO OPTIMAL ONLINE ADAPTATION PROPOSALS

To provide more *robust*, *flexible*, and *responsive* adaptations, we propose ParetoAdapt—a general UI adaptation approach based on multi-objective optimization and *a posteriori* articulated preferences. By generating diverse Pareto optimal adaptation proposals—for example, presented in the form of interactive virtual widgets—, users can easily select their preferred adaptations and further adjust them as needed. In this section, we describe the general adaptation pipeline of the ParetoAdapt approach, before introducing a prototypical system alongside 3D UI layout adaptation examples in the next section. ParetoAdapt reduces a set of potential UI adaptations along three major steps until one is picked by the user through interaction or the set of solutions is reduced to a single adaptation (see Figure 2). The steps are: (1) Multi-Objective Optimization, (2) Solution Set Reduction, and (3) Preference Articulation. Each of the three steps is explained in further detail below.

4.1 Multi-Objective Optimization

In the first step, we utilize vectorized multi-objective optimization algorithms—that is, algorithms that optimize a vector of objective functions—to generate a wide range of Pareto optimal potential adaptations. Given a multi-objective problem formulation defined by the designer, these algorithms continuously refine a set of adaptation candidates by recognizing conflicting objectives, discarding infeasible designs, and identifying a wide range of efficient trade-off solutions. The result is a high-resolution approximation of the entire Pareto frontier of optimal adaptations that can further be filtered in the following steps. This vectorized approach has several benefits for online adaptation:

- It is *robust* to the scale and shape of the individual objective functions, including any non-convexity along the Pareto frontier, as current multi-objective optimization algorithms can effectively approximate a variety of complex Pareto frontier shapes.
- It allows for *flexible* adaptations that can fit diverse situational and individual user's needs as it provides a range of efficient trade-off solutions, any of which might provide a good starting point for selection or further refinement given a certain usage context.
- It retains *responsiveness* as multi-objective optimization algorithms have been designed for efficiency, even given complex optimization problems, scaling to many objectives and constraints, allowing for parallel evaluation of potential adaptations, or utilizing surrogate models to limit expensive adaptation evaluations.

Many multi-objective optimization solvers exist that can generate effective approximations of the Pareto frontier (see Section 2). For example, evolutionary algorithms like NSGA-III [9] or SMS-EMOA [4] can effectively generate the Pareto frontier for a wide set of many-objective optimization problems. These algorithms take inspiration from natural evolution to generate a set of optimal solutions by evolving (i.e., crossing, selecting, and mutating) a population of potential solutions. The optimal solutions are those that achieve the highest "evolutionary fitness" as evaluated against the set of optimization objectives. How evolutionary fitness is defined and evaluated and how the evolutionary processes—that is, crossover, selection, and mutation—are constructed differentiates the algorithms. Which algorithms (and which hyperparameter settings) can and should be chosen, thus depends on the specific optimization problem in question (e.g., the number of objectives) and on the adaptive application's performance requirements.

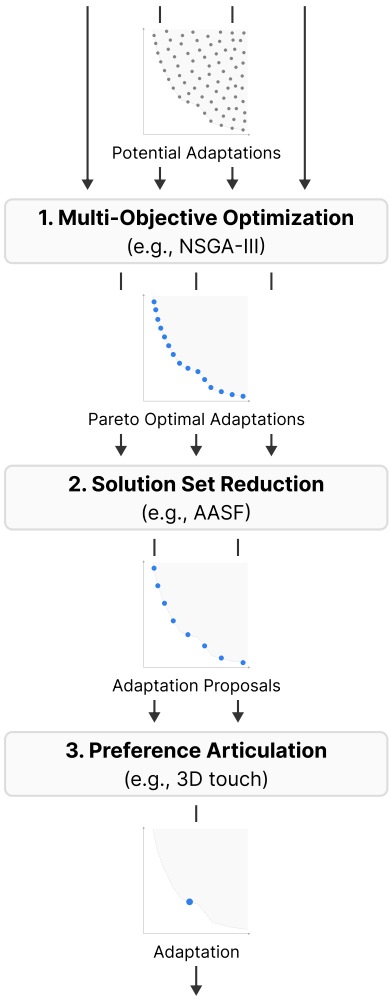


Figure 2: ParetoAdapt consists of three steps that continuously reduce the set of potential adaptations until one is chosen by the end-user or the set is reduced to a single point.

4.2 Solution Set Reduction

With the large set of Pareto optimal solutions as input, the second step aims to select a limited number of qualitatively different solutions from the Pareto frontier. To achieve this goal, the designer chooses decomposition techniques that capture their criteria for a useful set of adaptation proposals. There are a variety of

multi-criteria decision-making methods that can serve this function, differing in the specific qualities of the resulting adaptation proposals (cf. [46] for an overview).

A developer may choose, for example, to employ the Augmented Achievement Scalarization Function (AASF) [48] to identify a small set of extreme adaptations as well as a sample of compromise solutions as basis for selection and correction. Such a choice would emphasize the representation of the range of optimal trade-off solutions as well as control over the maximum number of adaptation proposals. AASF can further target solutions on non-convex regions of the Pareto frontier, improving *robustness* of the overall adaptation procedure. For other scenarios, the designer may decide to instead present a set of Pareto optimal solutions that possess certain desired properties (e.g., local efficiency [42]) without restricting the resulting number of adaptation proposals. Such a choice would emphasize *flexibility* as no further assumptions about the trade-offs that may be desired by users are made at design time, including the number of proposals that are deemed appropriate in a given context or their distribution along the Pareto frontier. Notably, Solution Set Reduction can also feature weighted sum decomposition with *a priori* specified weights. Our approach could thus be used to replicate previous UI adaptation examples or to incorporate user-specified weights if desired and appropriate.

Overall, algorithmic Solution Set Reduction contributes to the goals for an effective adaptation technique in the following ways:

- It retains *robustness* to errors in modeling of the subjective adaptation criteria as decomposition techniques can be chosen that are capable of targeting points on non-convex regions of the Pareto frontier while retaining a wide range of Pareto optimal adaptations.
- It allows for *flexible* adaptations as it aims to generate a varied set of optimal trade-off proposals, any of which may fit a given usage context.
- It retains adaptation *responsiveness* as algorithmic Solution Set Reduction can be very efficient, adding little computational overhead to the adaptation process.

4.3 Preference Articulation

In this step, the user makes their preferences for certain solutions explicit by choosing which proposed adaptations to interact with. The result is an adaptation that is applied to the UI. This adaptation could be further adjusted by the user and provides the starting point for the next iteration of the optimization. Preference Articulation may be skipped, if the previous Solution Set Reduction yields a single optimal solution (e.g., in the case of a single global optimum) or a fully automatic adaptation approach is preferred (e.g., in time-critical situations). But we emphasize the promise of this step to improve user control [26], increase robustness and flexibility of the adaptation.

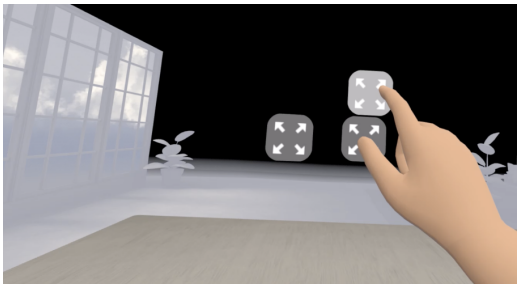
In 3D UI adaptation, a designer may represent a reduced set of Pareto optimal UI placements using floating widgets placed in the user's environment, as demonstrated in Figure 3 below. If the user activates one of these widgets, a new instance of the UI element is faded in at that position and other instances are faded out. Depending on the specific adaptation and application context, other presentation and selection methods may be more appropriate, such as showing semi-transparent duplicates of the UI to enable a better preview of the suggested solutions. If a higher level of automation is desired (e.g. because an application is highly time-sensitive or otherwise occupies the end-user), a default adaptation could be automatically suggested and the user would only have to select from the displayed alternatives if necessary. Which presentation and selection format is appropriate for a particular adaptive application is highly dependent on the chosen design variables, specific usage domain, and application characteristics. We anticipate, however, that UI properties related to appearance — e.g., position, color, or orientation — lend themselves well to duplication or to the use of symbols as presentation techniques. Overall, Preference Articulation contributes to effective adaptations in the following ways:

- It increases *robustness* against incomplete or inaccurate models as the user selects from a varied set of adaptation proposals and can further adjust them to fit their individual needs. The optimization objectives do not have to be perfect representations of the user's preferences but are sufficient if they produce solutions that are close enough to a desired configuration.
- It allows for *flexible* adaptations as users can state their individual preferences for their unique context and can change them over time.
- It retains adaptation *responsiveness* as preferences are articulated during usage, after the optimization has completed. As a result, there is no delay between the articulation of preferences and the consequent adaptation.

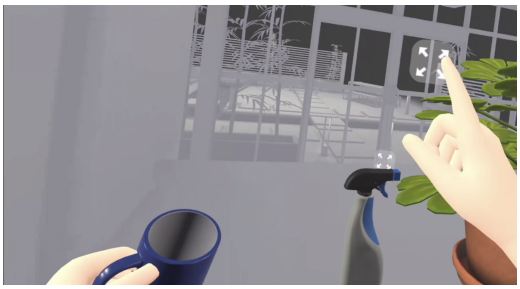
5 IMPLEMENTATION

To implement our proposed UI adaptation approach, we build upon AUIT [3], a toolkit to create adaptive UIs within the Unity development platform. It allows creators to develop real-time adaptations using multi-objective optimization, interactively constructing a vector of objective functions capturing the adaptation goals. However, AUIT does not currently support many of the concepts proposed in our approach: notably, the current solver and architecture are implemented focusing on a single adaptation solution, and preferences are articulated *a priori* using a weighted sum. To demonstrate our approach, we extend AUIT with the following features: (1) support for solutions containing multiple adaptation proposals, (2) a new solver to approximate the Pareto frontier and reduce the solution set, and (3) an interaction technique that allows users to visualize the solution set as well as to select and correct one of the adaptation proposals. The solver is implemented in Python 3.9 using the pymoo [7] package and communicates with the Unity-based adaptation system using socket-based networking. By default, it uses NSGA-III [9] (population size: 100, number of generations: 100) in combination with AASF [48] decomposition (ρ: 10^{-4}) initialized with 10 reference vectors corresponding to well-spaced objective weight combinations calculated via Riesz s-Energy [8]. Our custom and extended AUIT implementation was developed using Unity 2021.3 and further includes the Oculus Integration Package v50 to demonstrate an adaptive layout system on a Meta Quest Pro head-mounted display (see Section 6 below). We provide both the source code and further documentation of our implementation in an online repository[1].

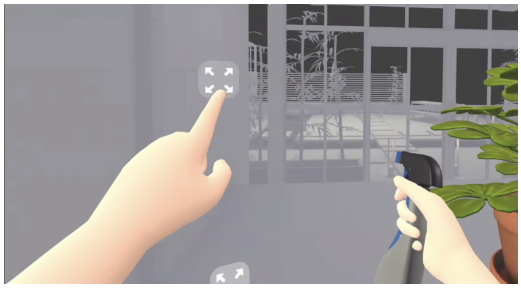
[1] https://github.com/christophajohns/auit-pareto-solver



(a) Center proposal preferred



(b) Right proposal preferred (left hand occupied)



(c) Left proposal preferred (right hand occupied)

Figure 3: Whether an adaptation is perceived by the user as appropriate is dependent on the current usage context. With ParetoAdapt we offer a set of approximate Pareto optimal solutions for the user to choose the UI position that best fits their current context. For example, the user might prefer a browser window to be positioned in the middle of their field of view (left). If their left hand is occupied, however, they may find it most comfortable if the UI is placed to the right (middle), or to their left if their right hand is occupied (right).

6 EXAMPLE APPLICATIONS: ONLINE 3D UI LAYOUT ADAPTATION

To illustrate the *flexibility* of our proposed approach to fit to many contexts and individual users, we apply our implementation to the problem of online 3D UI layout adaptation (cf. Section 3). In 3D UI layout adaptation, the position of a UI element in the user's environment is adapted as the user dynamically changes their location, pose, and activity. This problem has previously been studied by focusing on specific use cases and adaptation criteria (see Section 2). Online adaptation of 3D UIs remains, however, a hard problem to solve due to the many contextual factors that may affect the user's preferences and the many potential adaptations that can be chosen. In this demonstration, we present two worked use case examples for adaptive 3D UI systems: one simple and general, adapting the location of an application launcher in an open, uncluttered environment, and one more complex and specific, adapting the location of an interactive 3D object viewer in a cluttered office environment with complex objectives. The two examples serve to illustrate the benefits of the ParetoAdapt approach for both end-users and developers.

General Problem Definition. We define the 3D UI layout problem as a multi-objective optimization problem. Our design variable is a 3-dimensional vector denoting the position of the UI. As objectives, we employ two sets of adaptation measures related to visibility [3, 15], reachability [15], and semantics [13] that are inspired by prior literature on 3D UI layout adaptation. Our primary solver is the ParetoAdapt implementation described in Section 5 above. The adaptation is triggered by the user. The adaptation proposals are presented in the form of small 3D buttons displaying a shortcut icon and can be selected via touch. The user can adjust the position and rotation of the UI using pinch-and-drag. See the video for details.

6.1 Example 1: Ergonomically Adapting an Application Launcher

The first example focuses on a common use case in mixed reality settings: launching an application from a floating menu panel and positioning it for comfortable interaction. Consider a scenario where a user is at work and wants to review the practical details of an upcoming conference. They tap on their wrist to request the system's application launcher. Dependent on the user's surroundings and on their planned application use, they may prefer certain locations for both the application launcher and for the browser window that will eventually be displayed in its place (see Figure 3).

To find appropriate locations for the application launcher and browser window, we refine the general problem definition above. We constrain the potential locations of the UI to room-scale and include the ergonomic objectives from Section 3. We also add two objectives that further support the user's comfortable viewing and touching of the UI: The first, *Field-of-View*, incentivizes UI positions near the center of the user's view frustum. The second, *Distance*, rewards locations at around arm's length distance from the user (ca. 80 cm). For formal definitions and implementation details of the objectives and constraints, see Appendix A.1.

The proposals that are selected via the decomposition and which are consequently displayed to the user offer meaningful adaptation options at different heights and at different distances to the center of the view frustum. Dependent on their personal and situational needs, the user may, for example, prefer a proposal in the center of their field of view at medium height (e.g., because the virtual content will be the focal point of their current activity) or may prefer an adaptation further down, up, left, or right (e.g., because one of their hands is currently occupied, because relevant information in their environment would be occluded, or because an otherwise preferred adaptation is currently inaccessible; see Figure 3).

The objectives define a true Pareto frontier that spans from the user's waist to their eye level with all optimal proposals lying in

front of the user at around arm's distance. With the default settings, the ParetoAdapt solver quickly converges upon large stretches of this Pareto frontier with many proposals lying near their optimal locations. Some proposals in the final generation are positioned further from the user than formally optimal. This is an artifact of poor convergence and can be mitigated through tighter constraints, more efficient solvers, or more effective objective definitions. In practice, most poor proposals can be ignored by the user without degrading their overall experience. Convergence performance should, however, be monitored as it may limit the system's usability. Notably, the adaptation system does not require full knowledge of the contextual demands. The user can freely choose a starting adaptation from the set of generated proposals and adjust it as needed.

6.2 Example 2: Semantically Adapting a 3D Model Viewer

The second example illustrates a more specific and challenging scenario for the ParetoAdapt solver: placing a UI around a dense and cluttered physical environment and leveraging semantic associations between the UI and objects in the user's environment (see Figure 4). Consider an architect's office where various office supplies and desk accessories are scattered across a user's workspace. The architect is currently working on a housing project and has a physical blueprint lying on their desk. To their right, a mixed reality messaging application is displaying chat messages from friends and family members. They decide to open a 3D viewer to inspect the CAD model for their current design project. Since both the user's environment and their current activities pose challenging contextual demands on the appropriate positioning of the model viewer, adaptations are required.

We constrain the potential locations for the model viewer to be in front of the user at a reachable distance (between 30 and 80 cm) at maximum 80 cm to the left or right of the view frustum's center point. Starting with the ergonomic objectives discussed in Section 3, we add two further objectives that leverage the relation between the adapted UI and the virtual and physical objects in the user's environment. These two objectives are inspired by the semantic objectives in [13] and [36] and constitute both a *Semantic "Pull"*, minimizing the distance between the UI and the most strongly positively associated anchor object in the user's environment, and a *Semantic "Push"*, maximizing the distance between the UI and the most strongly negatively associated anchor object in the user's environment.[2] For this example, we manually define positive and negative association weights for the objects in the example environment. Here, the blueprint on the user's desk receives the strongest positive association and the personal messaging application receives the strongest negative association.

The true Pareto frontier originally spans from the user's waist to their eye level, distributing proposals across the interaction space (see Figure 4a). As more objectives are included, it is refined to eventually represent locations at various heights and distances from the positive and negative anchors (see Figure 4c). All optimal adaptations for this final formulation are pushed to the opposite side of the negative anchor. They are positioned either on a line originating from the negative anchor and intersecting the positive anchor or are located on the edge of the feasible space at various heights. The ParetoAdapt solver slowly converges upon this complex true Pareto frontier, positioning most proposals to the left of the blueprint at various heights. Similar to Example 1, some presented proposals have not yet converged and represent formally dominated or sub-optimal solutions. Depending on their needs and preferences, the user may choose a proposal that lies close to the physical blueprint, taking advantage of the semantic association between the two objects, or may, for example, choose a position closer to eye level to spread their application windows out across the interaction space. By leveraging non-dominance as its optimality definition, the ParetoAdapt solver is able to accommodate these latent preferences with a wide and varied set of adaptation proposals.

Notably, even the simple ergonomic objectives presented in Section 3 suffice to create varied adaptation proposals at various heights and distances to the two semantic anchors. Even when the semantic criteria are not included in the optimization objectives, users can select adaptation proposals near preferred regions of the interaction space and further adjust them as needed. By focusing on a varied set of objectives that span a wide Pareto frontier across the interaction space, even heavily underspecified preferences can thus be accommodated by the adaptive system.

7 TECHNICAL EVALUATION

We conduct simulation-based experiments to examine whether our ParetoAdapt implementation sufficiently supports *robustness* and *responsiveness*, targeting three general research questions:

1. How can ParetoAdapt handle incomplete representation (i.e., underspecification) of the user's preferences in the objectives?
2. How can ParetoAdapt handle inaccurate representation (i.e., misspecification) of the user's preferences in the objectives?
3. How can ParetoAdapt work in real-time with many proposals?

There is a general need for flexible methods to evaluate UI adaptations, especially those aiming to fit to user preferences. A user study targeting the above questions would necessarily be tied to a specific use case and thus be highly vulnerable to specific implementation issues which may strongly bias the outcome. Therefore, we follow a simulation-based approach that allows us to measure and control the latent, complex, and highly context-dependent nature of real preferences.

7.1 Experimental Design

We represent end-user preferences as utility functions that linearly combine a set of satisfaction criteria expressed as cost functions (see Appendix A.2). These utility functions reflect *qualities* of real end-user preferences and are used to quantify an adaptation's fit to a simulated user's preferences. They are inspired by adaptation criteria from prior literature and include both workplace ergonomic

[2]To include this objective in the current problem formulation, we transform the semantic agreement criterion into a semantic mismatch criterion by flipping its sign. To further allow for comparison against a weighted sum-based adaptation in the following technical evaluation, we bound and normalize it to produce costs in the same range as the other objectives. Here, we use clipped sigmoid normalization with large max and minimum values. Alternative normalization techniques can be found in [33].

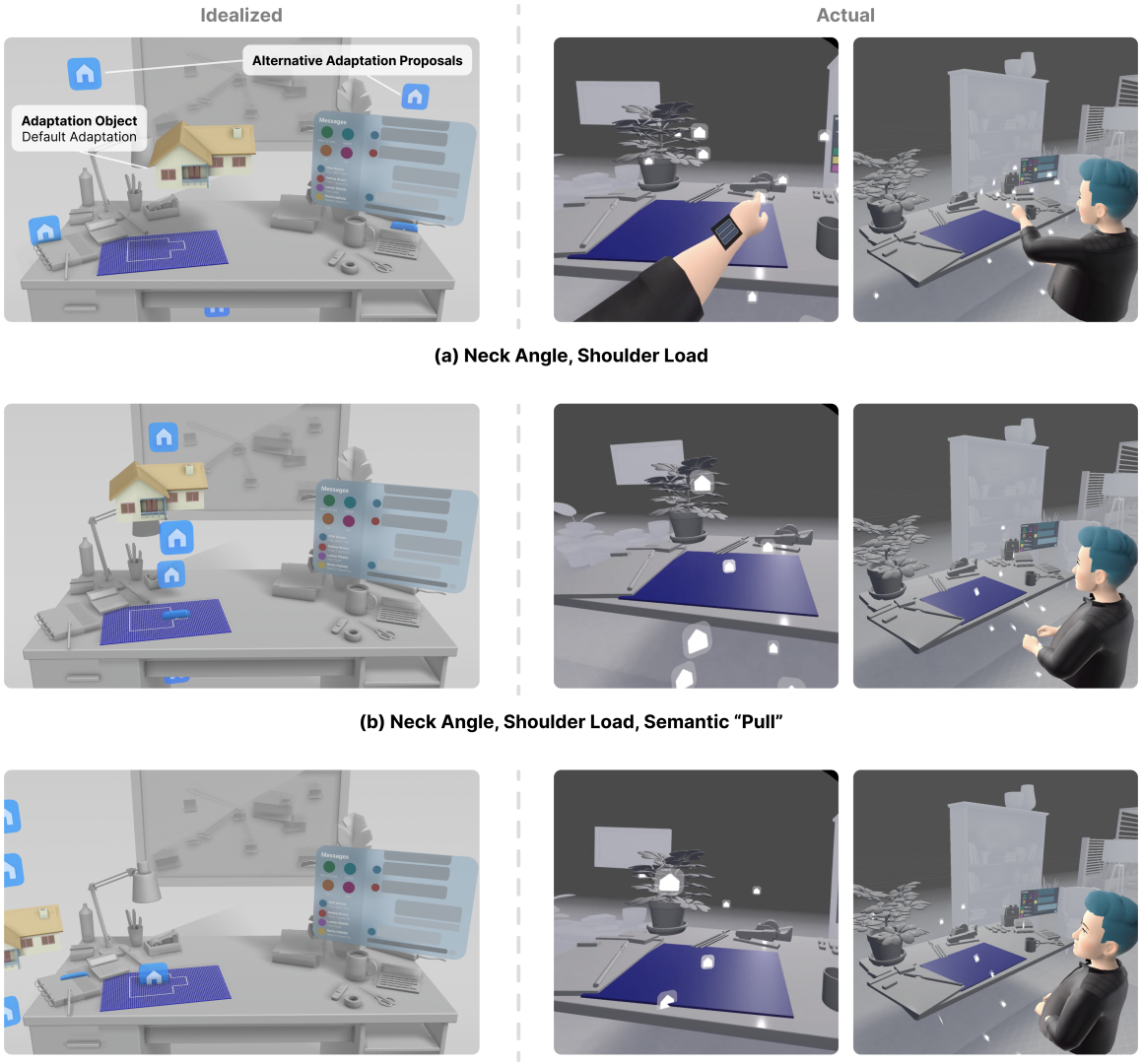


Figure 4: The positions of the adaptation proposals change as the included objectives reshape the Pareto frontier. In this example scenario, a virtual 3D model of a house is to be placed in a cluttered mixed reality environment. In each of the three objective configurations, the location of the virtual model represents a potential default solution targeting a fair trade-off between the objectives. Alternative Pareto optimal adaptation proposals are represented by blue 3D icons. Real-world objects in the environment are colored in grayscale. Each figure is annotated with the included optimization objectives. Both a simplified representation with samples from the true Pareto frontier and screenshots from its approximation in a live system with many proposals are included. Even two simple objectives offer varied and meaningful adaptation proposals to end-users while any single solution may disregard valid alternative adaptations. The number of proposals to be displayed can further be controlled via the chosen Solution Set Reduction technique.

metrics [3, 15, 35] (i.e., a neck, a shoulder, and a reachability objective) and semantic relatedness measures [13, 36] which are used to create a set of approximately naturalistic satisfaction criteria (see Appendix A.1). We parametrize the utility functions with a uniform probability distribution over their weight parameters to represent the variety and individuality of preferences that may be present in real end-user populations (cf. [41]). This linear form for the utility functions favors weighted sum-based adaptation, which will be the baseline in our evaluation.

7.1.1 Condition 1: Objective Completeness. To investigate robustness, we consider a scenario where a developer has modeled an adaptation's relevant satisfaction criteria but has incomplete information about the specific set and combination of satisfaction

criteria for a given user and context. We examine (1) a subset relation, where most but not all satisfaction criteria for a given user are known and have been included in the problem formulation (*incomplete*), and (2) an equal relation, where all relevant satisfaction criteria have been included (*complete*). These two relationships are least affected by the choice of objective function formulation and, in our view, constitute both a realistic scenario for a well-designed adaptive UI system in the case of the subset condition and a best-case scenario for weighted sum optimization in the equal condition (for more detail on relationships between optimization objectives and satisfaction criteria, see Appendix A.4). The *complete* case includes the three ergonomic objectives described above: neck strain, shoulder strain, and distance. The *incomplete* case only features the two conflicting objectives neck strain and shoulder strain while the simulated users' utility functions additionally include the distance (*incomplete*) and semantic "push" and "pull" objectives (cf. Section 6; *incomplete (incl. semantics)*). Formally, this condition tests whether the end-user's preferences can be accurately approximated using a linear combination of the optimization objectives.

7.1.2 Condition 2: Convex Pareto Frontier. In the second condition, we focus on concavity to evaluate robustness against non-convexity of the Pareto frontier. This constitutes a realistic scenario for an adaptive UI system since design problems typically are non-trivial (i.e., involve conflicting objectives), since both alternative non-convex shapes can be considered less realistic, and since local concavity is a feature of various complex Pareto frontier shapes. For a further explanation and discussion of the potential Pareto frontier shapes for adaptive UI systems, see Appendix A.5. The *convex* and *non-convex* conditions are achieved by altering the shape of the shoulder strain objective. In the *non-convex* condition, the shoulder strain grows linearly as the arm lift increases. In the *convex* condition, the shoulder strain grows exponentially as arm lift increases (see Appendix A.1). The difference between the absolute values produced by both functions for solutions along the Pareto frontier is kept to a minimum (< 20% in relative or < 0.02 in absolute difference between function values), which is favorable for weighted sum optimization.

7.1.3 Experimental Scenarios. We construct experimental scenarios that represent specific relationships between the optimization objectives and relevant satisfaction criteria.

1. *complete* and *convex*: Best-case scenario for the weighted sum as all satisfaction criteria are known and accurately modeled.
2. *complete* and *non-convex*: Optimistic scenario as all general satisfaction criteria are known, the utility function has the same form as the optimization objective (i.e., a weighted sum), and the relative weights in the global optimization criterion match the expected weights in the utility functions. Yet, one satisfaction criterion is modeled inaccurately; the shoulder strain objective uses a linear instead of an exponential form.
3. *incomplete* and *convex*: Optimistic scenario as most satisfaction criteria are known and have been accurately modeled. However, the distance criterion has been excluded, for example, because it was unknown, only occurs in certain individuals, or because it is specific to a context (e.g., where reach is particularly important).
4. *incomplete* and *non-convex*: A pessimistic scenario where the satisfaction criteria are partially modeled (i.e., the shoulder and neck strain objectives). However, the distance criterion has been excluded which adds noise to the utility, and the shoulder strain criterion has been modeled inaccurately resulting in a concave Pareto frontier shape.
5. *incomplete (incl. semantics)* and *non-convex*: The most pessimistic scenario where the satisfaction criteria are most heavily underspecified (i.e., the distance, semantic "pull" and semantic "push" objectives have not been included). The shoulder strain criterion is furthermore misspecified as a linear function, resulting in a concave Pareto frontier shape.

7.1.4 Dependent Variables. First, to evaluate robustness, we utilize the utility functions described above. An adaptation technique is considered to produce preferred adaptation proposals if the proposals achieve higher *maximum utility*. We assume that end-users ignore poor adaptation proposals and will only be affected by the quality of the best, and currently do not consider potential other impacts of presenting many proposals. Second, to evaluate the performance implications, we measure the *time* from the start of the optimization until the final set of adaptation proposals has been computed.

Based on preliminary experiments, we employ a set of concrete algorithms (NSGA-III), decomposition techniques (AASF), hyperparameter settings (population size: 100, number of generations: 100, $\rho = 10^{-4}$), and design variable ranges (room-scale with full element rotation, see Appendix A.3. The NSGA-III algorithm, in particular, was selected because it can be used to solve both single-objective and multi-objective optimization problems and thereby allows for performance comparisons between both weighted sum and vectorized problem formulations.

7.1.5 Independent Variables. We investigate the effect of:

- **Optimization Method**: *a priori* weighted sum and *a posteriori* ParetoAdapt.
- **Number of Proposals**: 1 and 10 as the maximum number of adaptation proposals.

In the *1 proposal* conditions, both the weighted sum and the AASF decomposition employ equal weights for all optimization objectives since it reflects the expected relative weights for our sampled utility functions. In the *10 proposal* conditions, we utilize Riesz s-Energy [8] to generate weight combinations corresponding to well-spaced compromise solutions for the AASF decomposition in the ParetoAdapt condition and return a maximum of 10 global optima generated via 10 total runs of the optimization for the equally weighted sum. To account for the randomness in the solvers and decomposition techniques, we further conduct 10 trials for each optimization run using different seeds. Finally, we evaluate both the effectiveness and real-time performance in a comparison with a sample of 100 utility functions, yielding 20,000 population evaluations with 110,000 utility evaluations and 1,000 maximum utility values per experimental condition in total. A random baseline is further included for each of the conditions. This baseline corresponds to the utilities of 1,000 randomly generated UI adaptations and can be used to anchor the results of our performance evaluations with regard to the scale and shape of the utility function. All experiments

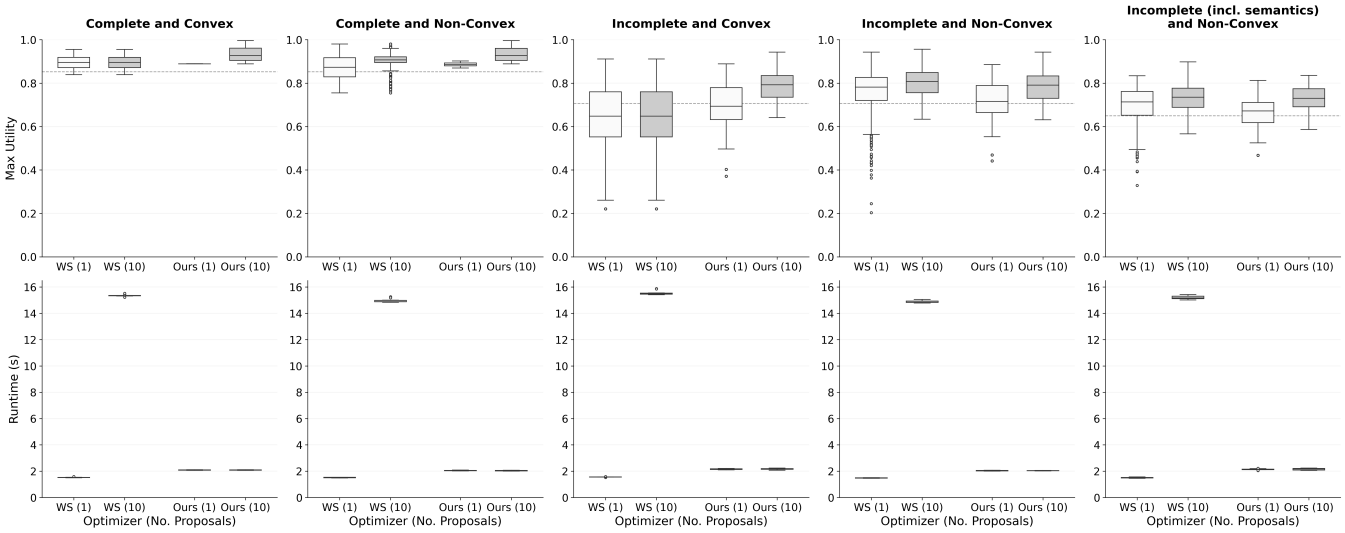


Figure 5: Results of our simulation-based evaluation for maximum utility and runtime performance. The expected utility for each scenario is marked with a dashed grey line.

are conducted on a MacBook Pro with an Apple M1 Pro CPU using Python 3.9.

7.2 Results

The results are displayed in Figure 5 and further detailed in Table 2 in Appendix A.6. They present the distributions of maximum utility for all utility functions and optimization runs per optimization method, maximum number of proposals, and experimental condition.

7.2.1 Robustness. Our implementation achieves similar or higher maximum utility compared to the weighted sum optimization for all experimental conditions. The number of proposals and accuracy/completeness of the optimization objectives strongly affect the adaptation quality and responsiveness. We find a wide distribution for the *incomplete/non-convex* and *incomplete (incl. semantics)/non-convex* scenarios as our approach cannot completely overcome the issue of poorly formulated and incomplete optimization objectives.

In the incomplete cases, when the utility functions do not include all preference criteria (i.e., the distance criterion and the semantic criteria are not included in the optimization objectives), the performance of all adaptation techniques is affected and all distributions of maximum utility are stretched. Our implementation is, however, least affected by this noisiness of end-user utility overall. It retains a distribution centered above the expected utility with a comparatively tight range. It further retains a similarly high maximum utility for some utility functions to repeated weighted sum optimization across all four experimental scenarios while maintaining interactive adaptation speed. Single-proposal adaptation techniques are more strongly affected by both weak and strong incompleteness of the optimization objectives. Heavy underspecification of the optimization objectives (i.e., the *incomplete (incl. semantics)/non-convex* scenario) negatively affects the quality of adaptations across all adaptation techniques without a clear difference between the weighted sum and our implementation.

Non-convexity affects all adaptation techniques except for our proposed implementation with multiple adaptation proposals. It is less affected across both number-of-proposal conditions than weighted sum optimization, retaining a similar distribution for both completeness conditions above the expected utility, which matches our earlier reflections on limitations of weighted sum UI optimization. By contrast, non-convexity considerably impacts the effectiveness of weighted sum optimization. This effect is, however, dampened when providing multiple adaptation proposals. When only considering optimization with single adaptation proposals (e.g., fully automatic adaptation), our technique offers more control than weighted sum optimization as represented by the tighter distribution anchored to a similar maximum utility. As a result of its wider distribution, the weighted sum with a single proposal does, however, achieve higher maximum utility for some utility functions in all conditions.

7.2.2 Responsiveness. Our illustrative implementation generates adaptation proposals in ca. 2 seconds independent of the number of adaptation proposals across all scenarios. By contrast, the runtime of the weighted sum optimization ranges from 1.5 seconds to around 15 seconds and scales nearly linearly with the number of adaptation proposals. Both techniques achieve near real-time performance and could substantially be improved in terms of software architecture, algorithm, hyperparameter, and hardware choice.

8 DISCUSSION

The core problem of effective adaptations lies in the approximation of the user's true preferences. Our evaluation shows that ParetoAdapt is more robust to inaccuracy in the form of non-convexity of the Pareto frontier and incompleteness of the optimization objectives in the form of unspecified preference criteria than weighted sum UI optimization in this regard. Especially when providing multiple adaptation proposals, it requires no prior knowledge about appropriate relative weights or conflicts between optimization objectives

or satisfaction criteria to provide effective adaptations. Our results further suggest that ParetoAdapt scales well with the number of adaptation proposals but does introduce a flat cost in terms of run-time performance compared to weighted sum optimization and requires efficient algorithms to quickly converge upon the true Pareto frontier. Effectiveness remains, furthermore, sensitive to the shape and scale of unrecognized satisfaction criteria as exemplified by the distance semantic objectives included in our *incomplete* experimental conditions.

We find that ParetoAdapt has great potential to provide effective adaptations when user-preferred adaptation solutions lie on or near the Pareto frontier in either the design space (e.g., in the user's environment) or in the criterion space, that is, the adaptation represents a similar or equivalent trade-off between conflicting satisfaction criteria. In these cases, the adaptation proposals constitute effective starting points for finer correction of the UI configuration, providing users with some benefits of fully-automated adaptation (e.g., reduced interaction effort) while retaining the precision of manual configuration. These benefits are the result of a basic underlying idea: Users recognize what they want from an adaptation for a given context and can correct the UI's configuration to fit those needs. This trust in user capabilities allows our approach to loosen the requirements and assumptions of the UI optimization problem and focus on variety of adaptation proposals rather than the precision of a single automatic adaptation.

Limitations and Future Work. While our approach provides a significant leap over the prior methods, how user preferences are captured can yet be improved as described in Figure 6. Preference approximation can potentially guide such efforts. By viewing the optimization problem formulation not as a static entity but as a hypothesis about satisfaction criteria to be continuously refined (e.g., through the addition or combination of objectives, the selection of design variables, or the addition of constraints), the adaptations may be continuously improved in turn. The user's selection from a set of adaptation proposals and consequent correction developed here may provide a nuanced and natural way to generate high-quality information for learning or optimization-based preference approximation approaches in this regard.

In this work, we have further focused on the technical limitations of optimization-based adaptation approaches. We, thus, developed a systematic evaluation setup that can reliably produce these conditions of interest at sufficient scale while remaining flexible and easy to extend. It offers empirical support *why* multiple proposals may be perceived as beneficial by end-users. It is, however, limited in the types of support it can provide. The effects of our adaptation approach on subjective criteria related to user satisfaction (e.g., cognitive load, transparency, or perceived control) remain unclear. An end-user study investigating the effect of factors such as the chosen presentation and selection techniques, number and distribution of adaptation proposals, and responsiveness of adaptations may aid in this effort. A key challenge will be to explore and understand the effect that *bundles* of adaptations have on these subjective criteria. While previous work has largely focused on the evaluation of a single adaptation's quality, this will require new research on the joint perception of multiple adaptations and adaptation proposals and may enable the development of novel adaptive systems that explore the opportunities of semi-automatic adaptation techniques and concurrent parallel adaptations.

Overall, ParetoAdapt has been developed focusing on semi-stationary environments where interaction episodes are long or involved enough to warrant careful setup of the UI but dynamic enough to warrant continuous adaptation. It represents a support system that aims to reduce the effort required to create effective UI configurations. We suspect, however, that fully dynamic environments, where bad adaptations do not strongly affect system usability but manual correction would introduce high interaction costs, do not lend themselves well to semi-automatic adaptations as proposed here. Instead, these conditions may benefit from fully-automatic default adaptations that may be corrected by the user or adaptation proposals that can be requested on demand. Whether ParetoAdapt can successfully cover such use cases remains an open question for future research.

Finally, we note that our approach is not limited to the specific 3D UI adaptation examples we have highlighted here. Instead, it can be applied to a wide range of UI optimization problems where multiple, potentially conflicting objectives need to be considered. In the context of 3D UI adaptation, other objectives related to, for example, visibility [3, 13, 31, 36], reachability [3, 13, 15], semantics [13, 36], spatio-temporal consistency [3], user-specified preference regions [36], or cognitive load [31] may be explored. Similarly, alternative design variables, for example related to a UI's level of detail [31], may be considered. Since ParetoAdapt does not require objective normalization or manual weighting and since it does not limit the choice of design variables, such measures and UI properties can easily be included in the adaptation problem formulation and allow ParetoAdapt to integrate well with plug-and-play UI optimization toolkits such as AUIT [3]. A human-in-the-loop vectorized multi-objective optimization approach may also be applied to optimization problems outside the domain of 3D UIs where objectives span meaningful sets of trade-off solutions. For example, in the domain of information visualization, so-called *degree-of-interest* criteria commonly formalize qualities of appropriate visualization configurations and are continuously optimized to identify interesting visualization opportunities (see, for example, [23]). These objectives are typically combined into a single global objective using a weighted sum but generally define a set of Pareto optimal visualizations that may offer diverse insights for data curators and analysts. Such visualizations would, for example, lend themselves well to parallel data exploration in the form of *multiverse analysis* where each analysis option represents a different Pareto optimal visualization configuration. Online multi-objective UI optimization with *a posteriori* articulated preferences may thus open interesting opportunities across application domains and warrants further exploration in future work.

9 CONCLUSION

In this paper, we proposed a novel approach for online UI adaptation based on multi-objective optimization with *a posteriori* articulated preferences titled ParetoAdapt. Our approach generates a set of Pareto-optimal adaptation proposals that users can choose from, rather than relying on global criterion optimization methods that may not meet their expectations. We investigated and discussed

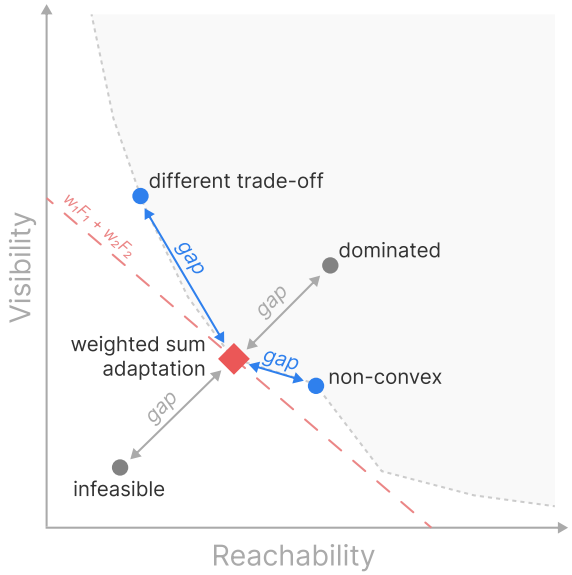


Figure 6: How can we best capture end-user preferences? An adaptation may (1) be dominated and not optimal, (2) be infeasible, (3) represent a different trade-off between conflicting objectives, or (4) lie on a non-convex part of the Pareto frontier. ParetoAdapt captures the latter two issues, and proposed manual correction as a remedy for the former. This leaves dominated and infeasible preferred adaptations for future work, potentially via preference approximation.

factors that can impact adaptation usability, including optimization method, inaccuracy and incompleteness of optimization objectives, and the number of adaptation proposals. Our experiments indicated that our approach can provide effective proposals for semi-automatic online UI adaptation. We have discussed several opportunities for future research, including investigating the effect on other aspects of users' experience such as cognitive load and refining the optimization problem formulation over time. While this work has demonstrated our proposed approach by adapting the layout of an adaptive mixed reality application, we believe that ParetoAdapt can generalize to problems beyond layout adaptation, offering a wealth of avenues for future research.

REFERENCES

[1] Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. 2013. MenuOptimizer: Interactive Optimization of Menu Systems. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (St. Andrews, Scotland, United Kingdom) *(UIST '13)*. Association for Computing Machinery, New York, NY, USA, 331–342. https://doi.org/10.1145/2501988.2502024

[2] Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. 2013. MenuOptimizer: interactive optimization of menu systems. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, St. Andrews Scotland, United Kingdom, 331–342. https://doi.org/10.1145/2501988.2502024

[3] João Belo, Matthias N. Lystbæk, Anna Maria Feit, Ken Pfeuffer, Peter Kán, Antti Oulasvirta, and Kaj Grønbæk. 2022. AUIT - the Adaptive User Interfaces Toolkit for Designing XR Applications. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22)*. Association for Computing Machinery, New York, NY, USA, 16. https://doi.org/10.1145/3526113.3545651

[4] Nicola Beume, Boris Naujoks, and Michael Emmerich. 2007. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research* 181, 3 (Sept. 2007), 1653–1669. https://doi.org/10.1016/j.ejor.2006.08.008

[5] Xiaojun Bi, Tom Ouyang, and Shumin Zhai. 2014. Both complete and correct?: multi-objective optimization of touchscreen keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Toronto Ontario Canada, 2297–2306. https://doi.org/10.1145/2556288.2557414

[6] Xiaojun Bi and Shumin Zhai. 2016. IJQwerty: What Difference Does One Key Change Make? Gesture Typing Keyboard Optimization Bounded by One Key Position Change from Qwerty. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose California USA, 49–58. https://doi.org/10.1145/2858036.2858421

[7] Julian Blank and Kalyanmoy Deb. 2020. Pymoo: Multi-Objective Optimization in Python. *IEEE Access* 8 (2020), 89497–89509. https://doi.org/10.1109/ACCESS.2020.2990567 Conference Name: IEEE Access.

[8] Julian Blank, Kalyanmoy Deb, Yashesh Dhebar, Sunith Bandaru, and Haitham Seada. 2021. Generating Well-Spaced Points on a Unit Simplex for Evolutionary Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 25, 1 (Feb. 2021), 48–60. https://doi.org/10.1109/TEVC.2020.2992387 Conference Name: IEEE Transactions on Evolutionary Computation.

[9] Julian Blank, Kalyanmoy Deb, and Proteek Chandan Roy. 2019. Investigating the Normalization Procedure of NSGA-III. In *Evolutionary Multi-Criterion Optimization*, Kalyanmoy Deb, Erik Goodman, Carlos A. Coello Coello, Kathrin Klamroth, Kaisa Miettinen, Sanaz Mostaghim, and Patrick Reed (Eds.). Vol. 11411. Springer International Publishing, Cham, 229–240. https://doi.org/10.1007/978-3-030-12598-1_19 Series Title: Lecture Notes in Computer Science.

[10] Rainer E Burkard and Josef Offermann. 1977. Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. *Zeitschrift für Operations Research* 21, 4 (1977), B121–B132.

[11] Liwei Chan, Yi-Chi Liao, George B Mo, John J Dudley, Chun-Lien Cheng, Per Ola Kristensson, and Antti Oulasvirta. 2022. Investigating Positive and Negative Qualities of Human-in-the-Loop Optimization for Designing Interaction Techniques. In *CHI Conference on Human Factors in Computing Systems*. ACM, New Orleans LA USA, 1–14. https://doi.org/10.1145/3491102.3501850

[12] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. 2016. A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 20, 5 (Oct. 2016), 773–791. https://doi.org/10.1109/TEVC.2016.2519378 Conference Name: IEEE Transactions on Evolutionary Computation.

[13] Yifei Cheng, Yukang Yan, Xin Yi, Yuanchun Shi, and David Lindlbauer. 2021. SemanticAdapt: Optimization-based Adaptation of Mixed Reality Layouts Leveraging Virtual-Physical Semantic Connections. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21)*. Association for Computing Machinery, New York, NY, USA, 282–297. https://doi.org/10.1145/3472749.3474750

[14] Mark Dunlop and John Levine. 2012. Multidimensional pareto optimization of touchscreen keyboards for speed, familiarity and improved spell checking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. Association for Computing Machinery, New York, NY, USA, 2669–2678. https://doi.org/10.1145/2207676.2208659

[15] João Marcelo Evangelista Belo, Anna Maria Feit, Tiare Feuchtner, and Kaj Grønbæk. 2021. XRgonomics: Facilitating the Creation of Ergonomic 3D Interfaces. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3411764.3445349

[16] Anna Maria Feit. 2018. *Assignment Problems for Optimizing Text Input*. Ph. D. Dissertation. Aalto University. http://urn.fi/URN:ISBN:978-952-60-8016-1

[17] Anna Maria Feit, Mathieu Nancel, Maximilian John, Andreas Karrenbauer, Daryl Weir, and Antti Oulasvirta. 2021. AZERTY AméLioré: Computational Design on a National Scale. *Commun. ACM* 64, 2 (jan 2021), 48–58. https://doi.org/10.1145/3382035

[18] Andreas Fender, Philipp Herholz, Marc Alexa, and Jörg Müller. 2018. OptiSpace: Automated Placement of Interactive 3D Projection Mapping Content. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–11. https://doi.org/10.1145/3173574.3173843

[19] Krzysztof Gajos and Daniel S. Weld. 2004. SUPPLE: automatically generating user interfaces. In *Proceedings of the 9th international conference on Intelligent user interfaces (IUI '04)*. Association for Computing Machinery, New York, NY, USA, 93–100. https://doi.org/10.1145/964442.964461

[20] Krzysztof Gajos and Daniel S. Weld. 2005. Preference elicitation for interface optimization. In *Proceedings of the 18th annual ACM symposium on User interface software and technology (UIST '05)*. Association for Computing Machinery, New York, NY, USA, 173–182. https://doi.org/10.1145/1095034.1095063

[21] Krzysztof Z. Gajos, Daniel S. Weld, and Jacob O. Wobbrock. 2010. Automatically generating personalized user interfaces with Supple. *Artificial Intelligence* 174, 12 (Aug. 2010), 910–950. https://doi.org/10.1016/j.artint.2010.05.005

[22] Ran Gal, Lior Shapira, Eyal Ofek, and Pushmeet Kohli. 2014. FLARE: Fast layout for augmented reality applications. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, Munich, Germany, 207–212. https://doi.org/10.1109/ISMAR.2014.6948429

[23] Stefan Gladisch, Heidrun Schumann, and Christian Tominski. 2013. Navigation Recommendations for Exploring Hierarchical Graphs. In *Advances in Visual Computing*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Baoxin Li, Fatih Porikli, Victor Zordan, James Klosowski, Sabine Coquillart, Xun Luo, Min Chen, and David Gotz (Eds.). Vol. 8034. Springer Berlin Heidelberg, Berlin, Heidelberg, 36–47. https://doi.org/10.1007/978-3-642-41939-3_4 Series Title: Lecture Notes in Computer Science.

[24] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang 'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. WristText: One-handed Text Entry on Smartwatch using Wrist Gestures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–14. https://doi.org/10.1145/3173574.3173755

[25] Mikhail V. Goubko and Alexander I. Danilenko. 2010. An Automated Routine for Menu Structure Optimization. In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Berlin, Germany) *(EICS '10)*. Association for Computing Machinery, New York, NY, USA, 67–76. https://doi.org/10.1145/1822018.1822030

[26] Kristina Höök. 2000. Steps to take before intelligent user interfaces become real. *Interacting with Computers* 12 (2000), 409–426. Issue 4. https://doi.org/10.1016/S0953-5438(99)00006-5

[27] DoYoung Lee, Jiwan Kim, and Ian Oakley. 2021. FingerText: Exploring and Optimizing Performance for Wearable, Mobile and One-Handed Typing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–15. https://doi.org/10.1145/3411764.3445106

[28] Yi-Chi Liao. 2021. Computational Workflows for Designing Input Devices. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems (CHI EA '21)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3411763.3443428

[29] Yi-Chi Liao, John J. Dudley, George B. Mo, Chun-Lien Cheng, Liwei Chan, Antti Oulasvirta, and Per Ola Kristensson. 2023. Interaction Design With Multi-objective Bayesian Optimization. *IEEE Pervasive Computing* (Jan. 2023), 1–10. https://doi.org/10.1109/MPRV.2022.3230597 Conference Name: IEEE Pervasive Computing.

[30] LW Light and Peter Anderson. 1993. Designing better keyboards via simulated annealing. *AI Expert* 8, 9 (1993), 20–27.

[31] David Lindlbauer, Anna Maria Feit, and Otmar Hilliges. 2019. Context-Aware Online Adaptation of Mixed Reality Interfaces. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. Association for Computing Machinery, New York, NY, USA, 147–160. https://doi.org/10.1145/3332165.3347945

[32] Baili Liu, Gregory Francis, and Gavriel Salvendy. 2002. Applying models of visual search to menu design. *International Journal of Human-Computer Studies* 56, 3 (2002), 307–330.

[33] R.T. Marler and J.S. Arora. 2004. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization* 26, 6 (April 2004), 369–395. https://doi.org/10.1007/s00158-003-0368-6

[34] Shouichi Matsui and Seiji Yamada. 2008. Genetic Algorithm Can Optimize Hierarchical Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 1385–1388. https://doi.org/10.1145/1357054.1357271

[35] Lynn McAtamney and E. Nigel Corlett. 1993. RULA: a survey method for the investigation of work-related upper limb disorders. *Applied Ergonomics* 24, 2 (April 1993), 91–99. https://doi.org/10.1016/0003-6870(93)90080-S

[36] Aziz Niyazov, Barrett Ens, Kadek Ananta Satriadi, Nicolas Mellado, Loic Barthe, Tim Dwyer, and Marcos Serrano. 2023. User-Driven Constraints for Layout Optimisation in Augmented Reality. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. Association for Computing Machinery, New York, NY, USA, 1–16. https://doi.org/10.1145/3544548.3580873

[37] Benjamin Nuernberger, Eyal Ofek, Hrvoje Benko, and Andrew D. Wilson. 2016. SnapToReality: Aligning Augmented Reality to the Real World. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose California USA, 1233–1244. https://doi.org/10.1145/2858036.2858250

[38] Antti Oulasvirta, Niraj Ramesh Dayama, Morteza Shiripour, Maximilian John, and Andreas Karrenbauer. 2020. Combinatorial Optimization of Graphical User Interface Designs. *Proc. IEEE* 108, 3 (March 2020), 434–464. https://doi.org/10.1109/JPROC.2020.2969687 Conference Name: Proceedings of the IEEE.

[39] Antti Oulasvirta, Per Ola Kristensson, Xiaojun Bi, and Andrew Howes. 2018. *Computational interaction*. Oxford University Press, Oxford, UK.

[40] Antti Oulasvirta, Anna Reichel, Wenbin Li, Yan Zhang, Myroslav Bachynskyi, Keith Vertanen, and Per Ola Kristensson. 2013. Improving two-thumb text entry on touchscreen devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Paris France, 2765–2774. https://doi.org/10.1145/2470654.2481383

[41] Ken Pfeuffer and Yang Li. 2018. Analysis and Modeling of Grid Performance on Touchscreen Mobile Devices. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173862

[42] Lily Rachmawati and Dipti Srinivasan. 2009. Multiobjective Evolutionary Algorithm With Controllable Focus on the Knees of the Pareto Front. *IEEE Transactions on Evolutionary Computation* 13, 4 (Aug. 2009), 810–824. https://doi.org/10.1109/TEVC.2009.2017515

[43] Morteza Shiripour, Niraj Ramesh Dayama, and Antti Oulasvirta. 2021. Grid-based Genetic Operators for Graphical Layout Generation. *Proceedings of the ACM on Human-Computer Interaction* 5, EICS (May 2021), 1–30. https://doi.org/10.1145/3461730

[44] Brian A. Smith, Xiaojun Bi, and Shumin Zhai. 2015. Optimizing Touchscreen Keyboards for Gesture Typing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3365–3374. https://doi.org/10.1145/2702123.2702357

[45] Srinath Sridhar, Anna Maria Feit, Christian Theobalt, and Antti Oulasvirta. 2015. Investigating the Dexterity of Multi-Finger Input for Mid-Air Text Entry. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3643–3652. https://doi.org/10.1145/2702123.2702136

[46] Jitesh J. Thakkar. 2021. Introduction. In *Multi-Criteria Decision Making*, Jitesh J. Thakkar (Ed.). Springer, Singapore, 1–25. https://doi.org/10.1007/978-981-33-4745-8_1

[47] Kashyap Todi and Antti Oulasvirta. 2016. Sketchplore: Sketch and Explore with a Layout Optimiser. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16)*. Association for Computing Machinery, New York, NY, USA, 543–555. https://doi.org/10.1145/2901790.2901817

[48] Andrzej P. Wierzbicki. 1982. A mathematical basis for satisficing decision making. *Mathematical Modelling* 3, 5 (Jan. 1982), 391–405. https://doi.org/10.1016/0270-0255(82)90038-0

[49] Robert Xiao, Scott Hudson, and Chris Harrison. 2017. Supporting Responsive Cohabitation Between Virtual Interfaces and Physical Objects on Everyday Surfaces. *Proceedings of the ACM on Human-Computer Interaction* 1, EICS (June 2017), 1–17. https://doi.org/10.1145/3095814

[50] Xin Yi, Chun Yu, Weinan Shi, Xiaojun Bi, and Yuanchun Shi. 2017. Word Clarity as a Metric in Sampling Keyboard Test Sets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, Denver Colorado USA, 4216–4228. https://doi.org/10.1145/3025453.3025701

[51] Difeng Yu, Ruta Desai, Ting Zhang, Hrvoje Benko, Tanya R. Jonker, and Aakar Gupta. 2022. Optimizing the Timing of Intelligent Suggestion in Virtual Reality. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22)*. Association for Computing Machinery, New York, NY, USA, 1–20. https://doi.org/10.1145/3526113.3545632

[52] Shumin Zhai, Michael Hunter, and Barton A Smith. 2002. Performance Optimization of Virtual Keyboards. *Human–Computer Interaction* 17, 2-3 (2002), 229–269. https://doi.org/10.1080/07370024.2002.9667315 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/07370024.2002.9667315

A APPENDIX

A.1 Optimization Objectives

For our demonstrative application and technical evaluation, we extend the objectives implemented in the AUIT [3] adaptation framework to include a neck strain objective F_{neck}, a shoulder strain objective $F_{shoulder}$ — both in a linear ($F_{shoulder,lin}$) and an exponential variant ($F_{shoulder,exp}$) —, a distance objective F_{dist}, and a set of semantic objectives, semantic "pull" ($F_{semantic,pull}$) and semantic "push" ($F_{semantic,push}$). The objectives are implemented as AUIT Adaptation Objectives [3] and interfaced with the off-the-shelf multi-objective optimization solvers provided by the pymoo [7] Python package using a custom AUIT Solver. The source code for the implementation of the objectives and solver are available at [Link omitted for anonymization].

Neck strain. The neck strain objective F_{neck} is based on the RULA [35] neck ergonomics metric and defined as the angle between a vector from the eyes to the target's projection on the xz-plane at the eye position's height (i.e., forward in the direction

of the element) and a vector from the eyes to the UI element (i.e., to the adaptation). The objective has a minimum value of 0 if the UI element is at eye level of the user and a maximum value of 1 if the UI element is directly above or below the user's eyes. If the element is at the user's eye position, the objective has a value of 1:

$$F_{neck}(\beta) = \begin{cases} 1 & \text{if } \beta = \pm\frac{\pi}{2} \\ \frac{|\beta|}{2\pi} & \text{otherwise} \end{cases} \tag{2}$$

where β is the angle between the vectors.

Shoulder load. The shoulder load objective $F_{shoulder}$ is based on the RULA [35] shoulder ergonomics metric and defined as the angle between a vector from the shoulder straight down and a vector from the shoulder to the UI element (i.e., adaptation). The objective has a minimum value of 0 if the UI element is directly below the shoulder of the user and a maximum value of 1 if the UI element is directly above the user's shoulders. If the element is at the user's shoulder position, the objective has a value of 1. To generate a non-convex (concave) Pareto frontier shape, we use a formulation where the shoulder strain cost grows linearly with the arm angle:

$$F_{shoulder,lin}(\theta) = \frac{\theta}{2\pi} \tag{3}$$

where θ is the angle between the vectors.

To generate a convex Pareto frontier shape, we use the formulation described below where the cost grows quasi-exponentially in periodic intervals with the arm angle:

$$F_{shoulder,exp}(\theta) = \begin{cases} \frac{e^{\frac{\alpha(\theta \mod 2\pi)}{\pi}} - 1}{e^{\alpha} - 1} & \text{if } \theta \mod 2\pi \leq \pi \\ \frac{e^{-\frac{\alpha(\theta \mod 2\pi - 2\pi)}{\pi}} - 1}{e^{\alpha} - 1} & \text{otherwise} \end{cases} \tag{4}$$

where θ is the angle between the vectors in radians and α is a parameter that controls the steepness of the curve. The closer α is to 1, the less concave the resulting Pareto frontier will be. The parameter α is set to 2 for all of our evaluations to limit the absolute and relative differences in function values between the linear and exponential objective formulation.

Distance. The distance objective F_{dist} is defined as the distance between the user's shoulder joint and the UI element (i.e., adaptation proposal). The objective has a minimum value of 0 if the UI element is within reach of the user as defined by a reachability threshold (i.e., arm length plus tolerance) and a maximum value striving toward 1 if the UI element is outside of the reachability threshold. If the element is at the user's shoulder position, the objective has a value of 1:

$$F_{dist}(d) = \begin{cases} 1 & \text{if } d \leq l + t \\ \frac{e^{-a(d-l-t)}}{e-1} & \text{otherwise} \end{cases} \tag{5}$$

where d is the distance between the user's shoulder joint and the UI element, l is the user's arm length, t is the reachability tolerance and a is a constant that controls the rate of decay. In our experiments, we set l to a value of 0.7, t to a value of 0.01 and a to a value of 0.5.

Semantics. The semantic mismatch criterion $F_{semantics}$ comprises two subobjectives combined into a single measure using weights [13]. Here, to allow for combination with other objectives in a weighted sum, we bound and normalize the objective using sigmoid normalization.

$$F_{semantics}(\mathbf{x}) = \frac{2}{1 + e^{w_{pull}F_{sem,pull}(\mathbf{x}) - w_{push}F_{sem,push}(\mathbf{x})}} \tag{6}$$

where $\mathbf{x}$ is the position of the UI element, w_{pull} is the weight of the semantic "pull" objective $F_{semantics,pull}$ and w_{push} is the weight of the semantic "push" objective $F_{semantics,push}$. We set $w_{pull} = 0.375$ and $w_{pull} = 0.625$ according to [13].

The semantic "pull" objective $F_{semantics,pull}$ minimizes the distance between the UI element and the most strongly positively associated object o.

$$F_{semantics,pull}(\mathbf{x}) = \max_o \left(\frac{a_o^+}{\delta_{\mathbf{x},o}^2} \right) \tag{7}$$

where $a_{x,o}^+$ is the positive association strength of the UI element and the object in the user's environment o and $\delta_{\mathbf{x},o}$ is the Euclidean distance between the UI element at position $\mathbf{x}$ and object o.

The semantic "push" objective $F_{semantics,push}$ minimizes the distance between the UI element and the most strongly negatively associated object o.

$$F_{semantics,push}(\mathbf{x}) = \max_o \left(\frac{a_o^-}{\delta_{\mathbf{x},o}^2} \right) \tag{8}$$

where $a_{x,o}^-$ is the negative association strength of the UI element and the object in the user's environment o and $\delta_{\mathbf{x},o}$ is the Euclidean distance between the UI element at position $\mathbf{x}$ and object o.

We manually define the associations between the UI element and objects in the user's environment so that the object with the strongest positive association of $a^+ = 1.0$ is positioned at $o^+ = (-0.4, -0.4, 0.3)$ and the object with the strongest negative association of $a^- = 1.0$ is positioned at $o^- = (0.4, -0.4, 0.2)$, replicating the configuration presented in [13].

A.2 End-User Utility Functions

The end-user utility function as employed in our technical evaluation is:

$$U = 1 - \frac{\sum_{i=1}^{k} w_i F_i(\mathbf{x})}{\sum_{i=1}^{k} w_i} \tag{9}$$

where U is the end-user utility (i.e., the value derived from) a UI configuration $\mathbf{x}$ computed as the sum of the partial utilities indicated by k cost functions $F(\mathbf{x})$ and weighted by the weights w. The UI configuration $\mathbf{x}$ is defined as a 7-dimensional vector combining the UI's three-dimensional position and four-dimensional rotation as described in Section 6. The utility is normalized to a range from 0 to 1 using the sum of all weights w and all cost functions $F(\mathbf{x})$ ranging from 0 to 1. Whether a UI configuration $\mathbf{x}_1$ is preferred over a configuration $\mathbf{x}_2$ is determined by the relationship between their two utilities:

$$\begin{aligned} \mathbf{x}_1 \succ \mathbf{x}_2 \quad & if \quad U(\mathbf{x}_1) > U(\mathbf{x}_2) \qquad (\mathbf{x}_1 \text{ is preferred}) \\ \mathbf{x}_1 \sim \mathbf{x}_2 \quad & if \quad U(\mathbf{x}_1) = U(\mathbf{x}_2) \quad (\mathbf{x}_1 \text{ is indifferent to } \mathbf{x}_2) \\ \mathbf{x}_1 \prec \mathbf{x}_2 \quad & if \quad U(\mathbf{x}_1) < U(\mathbf{x}_2) \qquad (\mathbf{x}_2 \text{ is preferred}) \end{aligned} \tag{10}$$

A.3 Optimization Constraints

We limit the feasible positions in our demonstration (see Section 6) to room-scale:

$$\begin{aligned} g_1 &: -3 \leq x_1 \leq 3 \\ g_2 &: -2 \leq x_2 \leq 2 \\ g_3 &: -3 \leq x_3 \leq 3 \end{aligned} \tag{11}$$

These constraints correspond to 3D position and rotation as used in the Unity development platform.

A.4 Relationships Between Optimization Objectives and Satisfaction Criteria

There are five potential relationships between the selected optimization objectives and the true satisfaction criteria if they belong to the same greater set of cost functions. The first four constitute an *incomplete* condition in the terminology of this study, the last a *complete* condition:

- **Subset** (*SUB*): The optimization objectives are a subset of the true satisfaction criteria; the satisfaction criteria have been underspecified and a relevant quality of satisfying adaptations has not been considered that adds noise to the estimation of end-user utility.
- **Superset** (*SUP*): The optimization objectives are a superset of the true satisfaction criteria; the satisfaction criteria have been overspecified and an irrelevant quality of UI configurations has been considered in the optimization that adds noise to the estimation of end-user utility.
- **Incomplete Intersection** (*INT*): The optimization objectives and the true satisfaction criteria intersect (i.e., have metrics in common) but both include further criteria; not all relevant satisfaction criteria were considered but additional irrelevant objectives were included in the optimization that add noise to the estimation of end-user utility.
- **Disjoint** (*DIS*): The optimization objectives and the true satisfaction criteria do not intersect; none of the relevant satisfaction criteria were considered but instead other irrelevant objectives were included in the optimization, the reported estimation of end-user utility is thus entirely determined by the specific objectives and objective formulations that were chosen.
- **Equal** (*EQU*): The optimization objectives and the true satisfaction criteria are identical; all relevant satisfaction criteria were considered and no others, end-user utility can perfectly be estimated using a linear combination of the optimization objectives.

In the evaluation presented in the body of this paper, we focus on the subset (*SUB*) and equal (*EQU*) relations to represent the *incomplete* and *complete* conditions respectively.

A.5 Pareto Frontier Shapes in UI Optimization

Four shapes of the Pareto frontier are of particular interest for UI optimization and adaptive UIs:

- **Point**: The Pareto frontier is collapsed to a single point that still may consist of multiple equivalent (i.e., indifferent) UI configurations; there is only a single feasible trade-off between conflicting objectives or no conflicts exist at all.
- **Linear**: The Pareto frontier reflects a perfect trade-off between conflicting objectives where improving one objective by one unit anywhere on the Pareto frontier necessarily worsens the other objective(s) by one unit and vice versa; there is a range of feasible trade-offs, but a weighted sum may return either a single-objective optimum (depending on the specific weights) or any of these trade-offs at random (if the weights perfectly match the slope of the Pareto frontier).
- **Convex**: The Pareto frontier has regions where moving from one objective's global optimum towards a compromise solution by steadily increasing the cost for that objective implicates a decelerating increase in cost for at least one other objective; an equally-weighted sum has its optimum at a compromise solution.
- **Concave**: The Pareto frontier has regions where moving from one objective's global optimum towards a compromise solution by steadily increasing the cost for that objective implicates an accelerating increase in cost for at least one other objective; an equally-weighted sum has its optimum at one of the individual objective's global optima.

Out of the three non-convex shapes, the evaluation presented in the body of this paper focuses on concavity as the non-convexity condition, since a linear Pareto frontier constitutes a special case of conflicting objectives while a point shape denies such conflicts altogether. We acknowledge, however, that point shapes and — in very rare cases — even linear Pareto frontiers may still be possible in practical adaptive UI applications.

A.6 Further Results

Further details of the evaluation results are displayed in Table 2 on the next page.

Table 2: Maximum Utility and Runtime by Optimization Method, Maximum Number of Proposals, and Scenario

		$complete \wedge convex$		$complete \wedge \overline{convex}$		$\overline{complete} \wedge convex$		$\overline{complete} \wedge \overline{convex}$	
Optim	No. Props	$\overline{U_{max}}$ (SD)	$\overline{t}$ (in s; SD)	$\overline{U_{max}}$ (SD)	$\overline{t}$ (in s; SD)	$\overline{U_{max}}$ (SD)	$\overline{t}$ (in s; SD)	$\overline{U_{max}}$ (SD)	$\overline{t}$ (in s; SD)
WS	1	0.90 (0.03)	1.64 (0.02)	0.91 (0.06)	1.54 (0.04)	0.64 (0.14)	1.53 (0.02)	0.80 (0.09)	1.48 (0.02)
	10	0.90 (0.03)	15.20 (0.08)	0.91 (0.04)	14.95 (0.05)	0.64 (0.14)	15.66 (0.31)	0.80 (0.06)	14.78 (0.06)
Ours	1	0.89 (0.00)	2.30 (0.18)	0.89 (0.01)	2.06 (0.03)	0.70 (0.11)	2.13 (0.03)	0.72 (0.09)	2.04 (0.02)
	10	0.93 (0.03)	2.15 (0.04)	0.93 (0.03)	2.05 (0.02)	0.79 (0.06)	2.11 (0.02)	0.79 (0.06)	2.06 (0.03)

Table 3: Maximum Utility and Runtime by Optimization Method, Maximum Number of Proposals, and Scenario (Cont.)

		$\overline{complete_{sem}} \wedge \overline{convex}$	
Optim	No. Props	$\overline{U_{max}}$ (SD)	$\overline{t}$ (in s; SD)
WS	1	0.70 (0.07)	1.51 (0.02)
	10	0.73 (0.06)	14.96 (0.06)
Ours	1	0.67 (0.07)	2.04 (0.03)
	10	0.72 (0.06)	2.04 (0.03)