# Decomposition Strategies for Constructive Preference Elicitation

**Paolo Dragone***
University of Trento, Italy
TIM-SKIL, Trento, Italy
paolo.dragone@unitn.it

**Stefano Teso**†
KU Leuven, Belgium
stefano.teso@cs.kuleuven.be

**Mohit Kumar** †
KU Leuven, Belgium
mohit.kumar@cs.kuleuven.be

**Andrea Passerini**
University of Trento, Italy
andrea.passerini@unitn.it

## Abstract

We tackle the problem of constructive preference elicitation, that is the problem of learning user preferences over very large decision problems, involving a combinatorial space of possible outcomes. In this setting, the suggested configuration is synthesized on-the-fly by solving a constrained optimization problem, while the preferences are learned iteratively by interacting with the user. Previous work has shown that Coactive Learning is a suitable method for learning user preferences in constructive scenarios. In Coactive Learning the user provides feedback to the algorithm in the form of an improvement to a suggested configuration. When the problem involves many decision variables and constraints, this type of interaction poses a significant cognitive burden on the user. We propose a decomposition technique for large preference-based decision problems relying exclusively on inference and feedback over partial configurations. This has the clear advantage of drastically reducing the user cognitive load. Additionally, part-wise inference can be (up to exponentially) less computationally demanding than inference over full configurations. We discuss the theoretical implications of working with parts and present promising empirical results on one synthetic and two realistic constructive problems.

## Introduction

In constructive preference elicitation (CPE) the recommender aims at suggesting a *custom* or *novel* product to a customer (Teso, Passerini, and Viappiani 2016). The product is assembled on-the-fly from components or synthesized anew by solving a combinatorial optimization problem. The suggested products should of course satisfy the customer's preferences, which however are unobserved and must be learned *interactively* (Pigozzi, Tsoukiàs, and Viappiani 2016). Learning proceeds iteratively: the learner presents one or more candidate recommendations to the customer, and employs the obtained feedback to estimate the customer's preferences. Applications include recommending custom PCs or cars, suggesting touristic travel plans, designing room and building layouts, and producing recipe modifications, among others.

A major weakness of existing CPE methods (Teso, Passerini, and Viappiani 2016; Teso, Dragone, and Passerini 2017) is that they require the user to provide feedback on *complete* configurations. In real-world constructive problems such as trip planning and layout design, configurations can be large and complex. When asked to evaluate or manipulate a complex product, the user may become overwhelmed and confused, compromising the reliability of the obtained feedback (Mayer and Moreno 2003). Human decision makers can revert to a potentially uninformative prior when problem solving exceeds their available resources. This effect was observed in users tasked with solving simple SAT instances (three variables and eight clauses) (Ortega and Stocker 2016). In comparison, even simple constructive problems can involve tens of categorical variables and features, in addition to hard feasibility constraints. On the computational side, working with complete configurations poses scalability problems as well. The reason is that, in order to select recommendations and queries, constructive recommenders employ constraint optimization techniques. Clearly, optimization of complete configurations in large constructive problems can become computationally impractical as the problem size increases.

Here we propose to exploit factorized utility functions (Braziunas and Boutilier 2009), which occur very naturally in constructive problems, to work with *partial* configurations. In particular, we show how to generalize Coactive Learning (CL) (Shivaswamy and Joachims 2015) to part-wise inference and learning. CL is a simple, theoretically grounded algorithm for online learning and preference elicitation. It employs a very natural interaction protocol: at each iteration the user is presented with a single, appropriately chosen candidate configuration and asked to improve it (even slightly). In (Teso, Dragone, and Passerini 2017), it was shown that CL can be lifted to constructive problems by combining it with a constraint optimization solver to efficiently select the candidate recommendation. Notably, the theoretical guarantees of CL remain intact in the construc-

*PD is a fellow of TIM-SKIL Trento and is supported by a TIM scholarship.

†ST and MK are supported by the ERC Advanced Grant SYNTH – Synthesising inductive data models.

tive case.

Our part-wise generalization of CL, dubbed PCL, solves the two aforementioned problems in one go: (i) by presenting the user with partial configurations, it is possible to (substantially) lessen her cognitive load, improving the reliability of the feedback and enabling learning in larger constructive tasks; (ii) in combinatorial constructive problems, performing inference on partial configurations can be *exponentially* faster than on complete ones. Further, despite being limited to working with partial configurations, PCL can be shown to still provide *local optimality* guarantees in theory, and to perform well in practice.

This paper is structured as follows. In the next section we overview the relevant literature. We present PCL in the Method section, followed by a theoretical analysis. The performance of PCL are then illustrated empirically on one synthetic and two realistic constructive problems. We close the paper with some concluding remarks.

## Related Work

Generalized additive independent (GAI) utilities have been thoroughly explored in the decision making literature (Fishburn 1967). They define a clear factorization mechanism, and offer a good trade off between expressiveness and ease of elicitation (Chajewska, Koller, and Parr 2000; Gonzales and Perny 2004; Braziunas and Boutilier 2009). Most of the early work on GAI utility elicitation is based on graphical models, e.g. UCP and GAI networks (Gonzales and Perny 2004; Boutilier, Bacchus, and Brafman 2001). These approaches aim at eliciting the full utility function and rely on the comparison of full outcomes. Both of these are infeasible when the utility involves many attributes and features, as in realistic constructive problems.

Like our method, more recent alternatives (Braziunas and Boutilier 2005; 2007) handle both partial elicitation, i.e. the ability of providing recommendations without full utility information, and local queries, i.e. elicitation of preference information by comparing only (mostly) partial outcomes. There exist both Bayesian (Braziunas and Boutilier 2005) and regret-based (Braziunas and Boutilier 2007; Boutilier et al. 2006) approaches, which have different shortcomings. Bayesian methods do not scale to even small size constructive problems (Teso, Passerini, and Viappiani 2016), such as those occurring when reasoning over individual parts in constructive settings. On the other hand, regret-based methods require the user feedback to be strictly self-consistent, an unrealistic assumption when interacting with non-experts. Our approach instead is specifically designed to scale to larger constructive problems and, being derived from Coactive Learning, natively handles inconsistent feedback. Crucially, unlike PCL, these local elicitation methods also require to perform a number of queries over *complete* configurations to calibrate the learned utility function. In larger constructive domains this is both impractical (on the user side) and computationally infeasible (on the learner side).

Our work is based on Coactive Learning (CL) (Shivaswamy and Joachims 2015), a framework for learning utility functions over structured domains, which has been successfully applied to CPE (Teso, Dragone, and Passerini 2017; Dragone et al. 2016). When applied to constructive problems, a crucial limitation of CL is that the learner and the user interact by exchanging *complete* configurations. Alas, inferring a full configuration in a constructive problem can be computationally demanding, thus preventing the elicitation procedure from being real-time. This can be partially addressed by performing approximate inference, as in (Raman, Shivaswamy, and Joachims 2012), at the cost of weaker learning guarantees. A different approach has been taken in (Goetschalckx, Fern, and Tadepalli 2014), where the exchanged (complete) configurations are only required to be locally optimal, for improved efficiency. Like PCL, this method guarantees the local optimality of the recommended configuration. All of the previous approaches, however, require the user to improve a potentially large *complete* configuration. This is a cognitively demanding task which can become prohibitive in large constructive problems, even for domain experts, thus hindering feedback quality and effective elicitation. By dealing with parts only, PCL avoids this issue entirely.

## Method

**Notation.** We use rather standard notation: scalars $x$ are written in italics and column vectors $\boldsymbol{x}$ in bold. The inner product of two vectors is indicated as $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_i a_i b_i$, the Euclidean norm as $\|\boldsymbol{a}\| = \sqrt{\langle \boldsymbol{a}, \boldsymbol{a} \rangle}$ and the max-norm as $\|\boldsymbol{a}\|_\infty = \max_i a_i$. We abbreviate $\{1, \dots, n\}$ to $[n]$, and indicate the complement of $I \subseteq [n]$ as $-I := [n] \setminus I$.

**Setting.** Let $\mathcal{X}$ be the set of candidate structured products. Contrarily to what happens in standard preference elicitation, in the constructive case $\mathcal{X}$ is defined by a set of hard constraints rather than explicitly enumerated[1]. Products are represented by a function $\boldsymbol{\phi}(\cdot)$ that maps them to an $m$-dimensional feature space. While the feature map can be arbitrary, in practice we will stick to features that can be encoded as constraints in a mixed-integer linear programming problem, for efficiency; see the Empirical Analysis section for details. We only assume that the features are bounded, i.e. $\forall x \in \mathcal{X}$ it holds that $\|\boldsymbol{\phi}(x)\|_\infty \leq D$ for some fixed $D$.

As is common in multi-attribute decision theory (Keeney and Raiffa 1976), we assume the desirability of a product $x$ to be given by a utility function $u^* : \mathcal{X} \to \mathbb{R}$ that is linear *in the features*, i.e., $u^*(x) := \langle \boldsymbol{w}^*, \boldsymbol{\phi}(x) \rangle$. Here the weights $\boldsymbol{w}^* \in \mathbb{R}^m$ encode the true user preferences, and may be positive, negative, or zero (which means that the corresponding feature is irrelevant for the user). Utilities of this kind can naturally express rich constructive problems (Teso, Passerini, and Viappiani 2016; Teso, Dragone, and Passerini 2017).

**Parts.** Here we formalize what parts and partial configurations are, and how they can be manipulated. We assume to

[1] In this paper, "hard" constraints refer to the constraints delimiting the space of feasible configurations, as opposed to "soft" constraints, which determine preferences over feasible configurations (Meseguer, Rossi, and Schiex 2006).

be given a set of $n$ *basic parts* $p \in \mathcal{P}$. A *part* is any subset $P \subseteq \mathcal{P}$ of the set of basic parts. Given a part $P$ and an object $x$, $x_P \in \mathcal{X}_P$ indicates the *partial configuration* corresponding to $P$. We require that the union of the basic parts reconstructs the whole object, i.e. $x_{\mathcal{P}} = x$ for all $x \in \mathcal{X}$. The proper semantics of the decomposition into basic parts is task-specific. For instance, in a scheduling problem a month may be decomposed into days, while in interior design a house may be decomposed into rooms. Analogously, the non-basic parts could then be weeks or floors, respectively. In general, any combination of basic parts is allowed. We capture the notion of combination of partial configurations with the *part combination* operator $\circ : \mathcal{X}_P \times \mathcal{X}_Q \to \mathcal{X}_{P \cup Q}$, so that $x_P \circ x_Q = x_{P \cup Q}$. We denote the complement of part $P$ as $\overline{P} = \mathcal{P} \setminus P$, which satisfies $x = x_P \circ x_{\overline{P}}$ for all $x \in \mathcal{X}$.

Each basic part $p \in \mathcal{P}$ is associated to a *feature subset* $I_p \subseteq [m]$, which contains all those features that depend on $p$ (and only those). In general, the sets $I_p$ may overlap, but we do require each basic part $p$ to be associated to some features that do not depend on any other basic part $q$, i.e. that $I_p \setminus \left(\bigcup_{q \neq p} I_q\right) \neq \varnothing$ for all $p \in \mathcal{P}$. The features associated to a part $P \subseteq \mathcal{P}$ are defined as $\bigcup_{p \in P} I_p$. Since the union of the basic parts makes up the full object, we also have that $\bigcup_{p \in \mathcal{P}} I_p = [m]$.

**GAI utility decomposition.** In the previous section we introduced a decomposition of configurations into parts. In order to elicit the user preferences via part-wise interaction, which is our ultimate goal, we need to decompose the utility function as well. Given a part $P$ and its feature subset $I_P$, let its *partial utility* be:

$$u[I_P](x) := \langle \boldsymbol{w}_{I_P}, \boldsymbol{\phi}_{I_P}(x) \rangle = \textstyle\sum_{i \in I_P} w_i \phi_i(x) \qquad (1)$$

If the basic parts have no shared features, the utility function is *additive*: it is easy to verify that $u(x) = \sum_{p \in \mathcal{P}} u[I_p](x)$. In this case, each part can be managed independently of the others, and the overall configuration maximizing the utility can be obtained by separately maximizing each partial utility and combining the resulting part-wise configurations.

However, in many applications of interest the feature subsets *do* overlap. In a travel plan, for instance, one can be interested in alternating cultural and leisure activities in consecutive days, in order to make the experience more diverse and enjoyable. In this case, the above decomposition does not apply anymore as the basic parts may depend on each other through the shared features. Nonetheless, it can be shown that our utility function is *generalized additive independent* (GAI) over the feature subsets $I_p$ of the basic parts. Formally, a utility $u(x)$ is GAI if and only if, given $n$ feature subsets $I_1, \ldots, I_n \subseteq [m]$, it can be decomposed into $n$ *independent* subutilities (Braziunas and Boutilier 2005):

$$u(x) = u_{I_1}(x) + \cdots + u_{I_n}(x) \qquad (2)$$

where each subutility $u_{I_k}$ can only depend on the features in $I_k$ (but does not need to depend on *all* of them). This decomposition enables applying ideas from the GAI literature

**Algorithm 1** An example of ordering selection procedure using a GAI network (Gonzales and Perny 2004).

**procedure** SELECTORDERING($\mathcal{P}$)
  Build a GAI network $\mathcal{G}$ from $I_p$, $p \in \mathcal{P}$
  Produce sequence $p_1, \ldots, p_n$ by sorting the nodes in $\mathcal{G}$ in ascending order of node degree
  **return** $p_1, \ldots, p_n$

to produce a well-defined part-wise elicitation protocol. Intuitively, we will assign features to subutilities so that whenever a feature is shared by multiple parts, only the subutility corresponding to one of them will depend on that feature.

We will now construct a suitable decomposition of $u(x)$ into $n$ independent subutilities. Fix some order of the basic parts $p_1, \ldots, p_n$, and let:

$$J_k := I_k \setminus \left(\textstyle\bigcup_{j=k+1}^{n} I_j\right) \qquad (3)$$

for all $k \in [n]$. We define the subutilities as $u_{I_k}(x) := u[J_k](x)$ for all $k \in [n]$. By summing up the subutilities for all parts, we obtain a utility where each feature is computed exactly once, thus recovering the full utility $u(x)$:

$$\begin{aligned} u(x) &= \textstyle\sum_{k=1}^{n} u_{I_k}(x) = \sum_{k=1}^{n} u[I_k \setminus \bigcup_{j=k+1}^{n} I_j](x) \\ &= \textstyle\sum_{i \in I_{\mathcal{P}}} w_i \phi_i(x) \end{aligned}$$

The GAI decomposition allows to elicit each subutility $u_{I_k}$ separately **Stefano: @Paolo: explain why, see reviewer 2**. By doing so, however, we end up ignoring some of the dependencies between parts, namely the features in $I_k \setminus J_k$. This is the price to pay in order to achieve decomposition and partwise elicitation, and it may lead to suboptimal solutions if too many dependencies are ignored. It is therefore important to minimize the broken dependencies by an appropriate ordering of the parts. Going back to the travel planning with diversifying features example, consider a multi-day trip. Here the parts may refer to individual days, and $I_k$ includes *all* features of day $k$, including the features relating it to the other days. Note that the $I_k$'s overlap. On the other hand, the $J_k$'s are subset of features chosen so that every feature only appears only once. A diversifying feature relating days 3 and 4 of the trip is either assigned to $J_3$ or $J_4$, but not both.

One way to control the ignored dependencies is by leveraging GAI networks (Gonzales and Perny 2004). A GAI network is a graph whose nodes represent the subsets $I_k$ and whose edges connect nodes sharing at least one feature. Algorithm 1 presents a simple and effective solution to provide an ordering. It builds a GAI network from $\mathcal{P}$ and sorts the basic parts in ascending order of node degree (number of incoming and outgoing edges). By ordering last the subsets having intersections with many other parts, this ordering attempts to minimize the lost dependencies in the above decomposition (Eq. 3). This is one possible way to order the parts, which we use as an example; more informed or task-specific approaches could be devised.

**The PCL algorithm.** The pseudocode of our algorithm, PCL, is listed in Algorithm 2. PCL starts off by sorting the

**Algorithm 2** The PCL algorithm.

**procedure** PCL($\mathcal{P}, T$)
  $p_1, \ldots, p_n \leftarrow$ SELECTORDERING($\mathcal{P}$)
  $\boldsymbol{w}^1 \leftarrow 0,\ x^1 \leftarrow$ initial configuration
  **for** $t = 1, \ldots, T$ **do**
    $p^t \leftarrow$ SELECTPART($\mathcal{P}$)
    $x^t_{p^t} \leftarrow \operatorname{argmax}_{x_{p^t} \in \mathcal{X}_{p^t}} u^t[J^t](x_{p^t} \circ x^t_{\overline{p}^t})$
    $x^t_{\overline{p}^t} \leftarrow x^{t-1}_{\overline{p}^t}$
    User provides improvement $\hat{x}^t_{p^t}$ of $x^t_{p^t}$, keeping $x^t_{\overline{p}^t}$ fixed
    **if** $\left(u^t[I^t](\hat{x}^t_{p^t} \circ x^t_{\overline{p}^t}) - u^t[I^t](x^t_{p^t} \circ x^t_{\overline{p}^t}) \leq 0\right)$ **then** $Q^t \leftarrow I^t$ **else** $Q^t \leftarrow J^t$ ;
    $\boldsymbol{w}^{t+1}_{-Q^t} \leftarrow \boldsymbol{w}^t_{-Q^t}$
    $\boldsymbol{w}^{t+1}_{Q^t} \leftarrow \boldsymbol{w}^t_{Q^t} + \boldsymbol{\phi}_{Q^t}(\hat{x}^t_{p^t} \circ x^t_{\overline{p}^t}) - \boldsymbol{\phi}_{Q^t}(x^t_{p^t} \circ x^t_{\overline{p}^t}))$

basic parts, producing an ordering $p_1, \ldots, p_n$. Algorithm 1 could be employed or any other (e.g. task-specific) sorting solution. Then it loops for $T$ iterations, maintaining an estimate $\boldsymbol{w}^t$ of the user weights as well as a complete configuration $x^t$. The starting configuration $x^1$ should be a reasonable initial guess, depending on the task. At each iteration $t \in [T]$, the algorithm selects a part $p^t$ using the procedure SELECTPART (see below). Then it updates the object $x^t$ by inferring a new partial configuration $x^t_{p^t}$ while keeping the rest of $x^t$ fixed, that is $x^t_{\overline{p}^t} = x^{t-1}_{\overline{p}^t}$. The inferred partial configuration $x^t_{p^t}$ is optimal with respect to the local subutility $u^t_{I^t}(\cdot)$ given $x^{t-1}_{\overline{p}^t}$. Note that inference is over the partial configuration $x^t_{p^t}$ only, and therefore can be *exponentially faster* than inference over full configurations.

Next, the algorithm presents the inferred partial configuration $x^t_{p^t}$ as well as some contextual information (see below). The user is asked to produce an improved partial configuration $\hat{x}^t_{p^t}$ according to the her own preferences, while the rest of the object $x^t_{\overline{p}^t}$ is kept fixed. We assume that a user is satisfied with a partial configuration $x^t_{p^t}$ if she cannot improve it further, or equivalently when the object $x^t$ is *conditionally optimal* with respect to part $p^t$ given the rest of the object $x^t_{\overline{p}^t}$ (the formal definition of conditional optimality is given in the Analysis section). When a user is satisfied with a partial configuration, she returns $\hat{x}^t_{p^t} = x^t_{p^t}$, thereby implying no change in the weights $\boldsymbol{w}^{t+1}$.

After receiving an improvement, if the user is not satisfied, the weights are updated through a perceptron step. The subset $Q^t$ of weights that are actually updated depends on whether $u^t[I^t](\hat{x}^t_{p^t} \circ x^t_{\overline{p}^t}) - u^t[I^t](x^t_{p^t} \circ x^t_{\overline{p}^t})$ is negative or (strictly) positive. Since we perform inference on $u^t[J^t](\cdot)$, we have that $u^t[J^t](\overline{x}^t_{p^t} \circ x^t_{\overline{p}^t}) - u^t[J^t](\overline{x}^t_{p^t} \circ x^t_{\overline{p}^t}) \leq 0$. The user improvement can, however, potentially change all the features in $I^t$. Intuitively, the weights associated to a subset of features should change only if the utility computed on this subset ranks $\hat{x}^t_{p^t}$ lower than $x^t_{p^t}$. The algorithm therefore checks whether $u^t[I^t](\overline{x}^t_{p^t} \circ x^t_{\overline{p}^t}) \leq u^t[I^t](x^t_{p^t} \circ x^t_{\overline{p}^t})$, in which case the weights associated to the whole subset $I^t$ should be updated. If this condition is not met, instead, the algorithm can only safely update the weights associated to $J^t$, which, as said, meet this condition by construction.

As for the SELECTPART procedure, we experimented with several alternative implementations, including prioritizing parts with a large feature overlap ($|I_k \setminus J_k|$) and bandit-based strategies aimed at predicting a surrogate of the utility gain (namely, a variant of the UCB1 algorithm (Auer, Cesa-Bianchi, and Fischer 2002)). Preliminary experiments have shown that informed strategies do not yield a significant performance improvement over the random selection stategy; hence we stick with the latter in all our experiments.

The algorithm stops either when the maximum number of iterations $T$ is met or when a "local optimum" has been found. For ease of exposition we left out the latter case from Algorithm 2, but we explain what a local optimum is in the following Analysis section; the stopping criterion will follow directly from Proposition 1.

**Interacting through parts.** In order for the user to judge the quality of a suggested partial configuration $x_p$, some contextual information may have to be provided. The reason is that, if $p$ depends on other parts via shared features, these have to be somehow communicated to the user, otherwise his/her improvement will not be sufficiently informed.

We distinguish two cases, depending on whether the features of $p$ are local or global. Local features only depend on small, localized portions of $x$. This is for instance the case for features that measure the diversity of consecutive activities in a touristic trip plan, which depend on consecutive time slots or days only. Here the context amounts to those other portions of $x$ that share local features with $p$. For instance, the user may interact over individual days only. If the features are local, the context is simply the time slots before and after the selected day. The user is free to modify the activities scheduled that day based on the context, which is kept fixed.

On the other hand, global features depend on all of $x$ (or large chunks of it). For instance, in house furnishing one may have features that measure whether the total cost of the furniture is within a given budget, or how much the cost surpasses the budget. A naive solution would be that of showing the user the whole furniture arrangement $x$, which can

be troublesome when $x$ is large. A better alternative is to present the user a *summary* of the global features, in this case the percentage of the used budget. Such a summary would be sufficient for producing an informed improvement, independently from the actual size of $x$.

Of course, the best choice of context format is application specific. We only note that, while crucial, the context only provides auxiliary information *to the user*, and does not affect the learning algorithm directly.

## Analysis

In preference elicitation, it is common to measure the quality of a recommended (full) configuration in terms of the *regret*:

$$\text{REG}(x) := \max_{\hat{x}\in\mathcal{X}} u^*(\hat{x}) - u^*(x) = \langle \boldsymbol{w}^*, \boldsymbol{\phi}(x^*) - \boldsymbol{\phi}(x)\rangle$$

where $u^*(\cdot)$ is the true, unobserved user utility and $x^* = \text{argmax}_{x\in\mathcal{X}}\, u^*(x)$ is a truly optimal configuration. In PCL, interaction with the user occurs via partial configurations, namely $x^t_{p^t}$ and $\hat{x}^t_{p^t}$. Since the regret is defined in terms of complete configurations, it is difficult to analyze it directly based on information about the partial configurations alone, making it hard to prove convergence to globally optimal recommendations.

The aim of this analysis is, however, to show that our algorithm converges to a locally optimal configuration, which is in line with guarantees offered by other Coactive Learning variants (Goetschalckx, Fern, and Tadepalli 2014); the latter, however still rely on interaction via complete configurations. Here a configuration $x$ is a local optimum for $u^*(\cdot)$ if no part-wise modification can improve $x$ with respect to $u^*(\cdot)$. Formally, $x$ is a local optimum for $u^*(\cdot)$ if and only if:

$$\forall\, p\in\mathcal{P} \nexists\, x'_p\in\mathcal{X}_p \quad u^*(x'_p \circ x_{\overline{p}}) > u^*(x_p \circ x_{\overline{p}}) \qquad (4)$$

To measure local quality of a configuration $x$ with respect to a part $p$, we introduce the concept of *conditional regret* of the partial configuration $x_p$ given the rest of the object $x_{\overline{p}}$:

$$\text{CREG}_p(x) := u^*(x^*_p \circ x_{\overline{p}}) - u^*(x_p \circ x_{\overline{p}})$$

where $x^*_p = \text{argmax}_{\hat{x}_p\in\mathcal{X}_p}\, u^*(\hat{x}_p \circ x_{\overline{p}})$. Notice that:

$$\text{CREG}_p(x) = u^*[I_p](x^*_p \circ x_{\overline{p}}) - u^*[I_p](x_p \circ x_{\overline{p}})$$

since $u^*[-I_p](x^*_p \circ x_{\overline{p}}) - u^*[-I_p](x_p \circ x_{\overline{p}}) = 0$.

We say that a partial configuration $x$ is *conditionally optimal* with respect to part $p$ if $\text{CREG}_p(x) = 0$. The following lemma gives sufficient and necessary conditions for local optimality of a configuration $x$.

**Lemma 1.** *A configuration $x$ is locally optimal with respect to $u^*(\cdot)$ if and only if $x$ is conditionally optimal for $u^*(\cdot)$ with respect to all basic parts $p \in \mathcal{P}$.*

*Proof.* By contradiction. (i) Assume that $x$ is locally optimal but not conditionally optimal with respect to $p \in \mathcal{P}$. Then $\text{CREG}_p(x) > 0$, and thus there exists a partial configuration $x'_p$ such that $u^*(x'_p \circ x_{\overline{p}}) > u^*(x_p \circ x_{x_{\overline{p}}})$. This violates the local optimality of $x$ (Eq. 4). (ii) Assume that all partial configurations $x_p\ \forall p \in \mathcal{P}$ are conditionally optimal but $x$ is not locally optimal. Then there exists a part $q \in \mathcal{P}$ and a partial configuration $x'_q$ such that $u^*(x'_q \circ x_{\bar{q}}) > u^*(x_q \circ x_{\bar{q}})$. This in turn means that $\text{CREG}_q(x) > 0$. This violates the conditional optimality of $x$ with respect to $q$. □

The above lemma gives us a part-wise measurable criterion to determine if a configuration $x$ is a local optimum through the conditional regret of $x$ for all the provided parts.

The rest of the analysis is devoted to derive an upper bound on the conditional regret incurred by the algorithm and to prove that PCL eventually reaches a local optimum.

In order to derive the bound, we rely on the concept of $\alpha$-informativeness from (Shivaswamy and Joachims 2015), adapting it to part-wise interaction[2]. A user is *conditionally $\alpha$-informative* if, when presented with a partial configuration $x^t_{p^t}$, he/she provides a partial configuration $\hat{x}^t_{p^t}$ that is at least some fraction $\alpha \in (0, 1]$ better than $x^t_{p^t}$ in terms of conditional regret, or more formally:

$$u^*[I^t](\hat{x}^t_{p^t} \circ x^t_{\overline{p}^t}) - u^*[I^t](x^t_{p^t} \circ x^t_{\overline{p}^t}) \geq \alpha\left(u^*[I^t](x^*_{p^t} \circ x^t_{\overline{p}^t}) - u^*[I^t](x^t_{p^t} \circ x^t_{\overline{p}^t})\right) \qquad (5)$$

In the rest of the paper we will use the notation $u[I^t](\hat{x}^t)$ meaning $u[I^t](\hat{x}^t_{p^t} \circ x^t_{\overline{p}^t})$, i.e. drop the complement, when no ambiguity can arise.

At all iterations $t$, the algorithm updates the weights specified by $Q^t$, producing a new estimate $\boldsymbol{w}^{t+1}$ of $\boldsymbol{w}^*$. The actual indices $Q^t$ depend on the condition at line 9 of Algorithm 2: at some iterations $Q^t$ includes all of $I^t$, while at others $Q^t$ is restricted to $J^t$. We distinguish between these cases by:

$$\mathcal{I} = \{t \in [T] : u^t[I^t](\hat{x}^t) - u^t[I^t](x^t) \leq 0\}$$
$$\mathcal{J} = \{t \in [T] : u^t[I^t](\hat{x}^t) - u^t[I^t](x^t) > 0\}$$

so that if $t \in \mathcal{I}$ then $Q^t = I^t$, and $Q^t = J^t$ if $t \in \mathcal{J}$. For all $t \in [T]$, the quality of $\boldsymbol{w}^{t+1}$ is:

$$\begin{aligned}
\langle \boldsymbol{w}^*, \boldsymbol{w}^{t+1}\rangle &= \langle \boldsymbol{w}^*_{-Q^t}, \boldsymbol{w}^{t+1}_{-Q^t}\rangle + \langle \boldsymbol{w}^*_{Q^t}, \boldsymbol{w}^{t+1}_{Q^t}\rangle \\
&= \langle \boldsymbol{w}^*_{-Q^t}, \boldsymbol{w}^{t}_{-Q^t}\rangle + \langle \boldsymbol{w}^*_{Q^t}, \boldsymbol{w}^{t}_{Q^t}\rangle \\
&\quad + \langle \boldsymbol{w}^*_{Q^t}, \boldsymbol{\phi}_{Q^t}(\hat{x}^t) - \boldsymbol{\phi}_{Q^t}(x^t)\rangle \\
&= \langle \boldsymbol{w}^*, \boldsymbol{w}^{t}\rangle + \langle \boldsymbol{w}^*_{Q^t}, \boldsymbol{\phi}_{Q^t}(\hat{x}^t) - \boldsymbol{\phi}_{Q^t}(x^t)\rangle \\
&= \langle \boldsymbol{w}^*, \boldsymbol{w}^{t}\rangle + u^*[Q^t](\hat{x}^t) - u^*[Q^t](x^t)
\end{aligned}$$

Therefore if the second summand in the last equation, the *utility gain* $u^*[Q^t](\hat{x}^t) - u^*[Q^t](x^t)$, is positive, the update produces a better weight estimate $\boldsymbol{w}^{t+1}$.

Since the user is conditionally $\alpha$-informative, the improvement $\hat{x}^t$ always satisfies $u^*[I^t](\hat{x}^t) - u^*[I^t](x^t) \geq 0$. When $t \in \mathcal{I}$, we have $Q^t = I^t$, and thus the utility gain is guaranteed to be positive. On the other hand, when $t \in \mathcal{J}$ we have $Q^t = J^t$ and the utility gain reduces to $u^*[J^t](\hat{x}^t) - u^*[J^t](x^t)$. In this case the update

[2]Here we adopted the definition of *strict* $\alpha$-informativeness for simplicity. Our results can be directly extended to the more general notions of informativeness described in (Shivaswamy and Joachims 2015).

ignores the weights in $I^t \setminus J^t$, "missing out" a factor of $u^*[I^t \setminus J^t](\hat{x}^t) - u^*[I^t \setminus J^t](x^t)$.

We compactly quantify the missing utility gain as:

$$\zeta^t = \begin{cases} 0 & \text{if } t \in \mathcal{I} \\ u^*[I^t \setminus J^t](\hat{x}^t) - u^*[I^t \setminus J^t](x^t) & \text{if } t \in \mathcal{J} \end{cases}$$

Note that $\zeta^t$ can be positive, null or negative for $t \in \mathcal{J}$. When $\zeta^t$ is negative, making the update on $J^t$ only actually avoids a loss in utility gain.

We now prove that PCL minimizes the average conditional regret $\text{CREG}_T := \frac{1}{T}\sum_{t=1}^{T} \text{CREG}_{p^t}(x^t)$ as $T \to \infty$ for conditionally $\alpha$-informative users.

**Theorem 1.** *For a conditionally $\alpha$-informative user, the average conditional regret of* PCL *after $T$ iterations is upper bounded by:*

$$\frac{1}{T}\sum_{t=1}^{T} \textit{CREG}_{p^t}\left(x^t\right) \leq \frac{2DS\|\boldsymbol{w}^*\|}{\alpha\sqrt{T}} + \frac{1}{\alpha T}\sum_{t=1}^{T}\zeta^t$$

*Proof.* The following is a sketch[3]. We start by splitting the iterations into the $\mathcal{I}$ and $\mathcal{J}$ sets defined above, and bound the norm $\|\boldsymbol{w}^{T+1}\|^2$. In both cases we find that $\|\boldsymbol{w}^{T+1}\|^2 \leq 4D^2S^2T$. We then expand the term $\langle \boldsymbol{w}^*, \boldsymbol{w}^{T+1}\rangle$ for iterations in both $\mathcal{I}$ and $\mathcal{J}$, obtaining:

$$\begin{aligned}\langle \boldsymbol{w}^*, \boldsymbol{w}^{T+1}\rangle = &\sum_{t\in\mathcal{I}}\langle \boldsymbol{w}^*_{I^t}, \boldsymbol{\phi}_{I^t}(\hat{x}^t) - \boldsymbol{\phi}_{I^t}(x^t)\rangle \\ &+ \sum_{t\in\mathcal{J}}\langle \boldsymbol{w}^*_{J^t}, \boldsymbol{\phi}_{J^t}(\hat{x}^t) - \boldsymbol{\phi}_{J^t}(x^t)\rangle\end{aligned}$$

With few algebraic manipulations we obtain:

$$\begin{aligned}&\langle \boldsymbol{w}^*, \boldsymbol{w}^{T+1}\rangle \\ &= \sum_{t=1}^{T}\langle \boldsymbol{w}^*_{I^t}, \boldsymbol{\phi}_{I^t}(\hat{x}^t) - \boldsymbol{\phi}_{I^t}(x^t)\rangle - \sum_{t=1}^{T}\zeta^t\end{aligned}$$

Which we then bound using the Cauchy-Schwarz inequality:

$$\begin{aligned}&\|\boldsymbol{w}^*\|\sqrt{4D^2S^2T} \\ &\geq \|\boldsymbol{w}^*\|\|\boldsymbol{w}^{T+1}\| \\ &\geq \langle \boldsymbol{w}^*, \boldsymbol{w}^{T+1}\rangle \\ &= \sum_{t=1}^{T}\langle \boldsymbol{w}^*_{I^t}, \boldsymbol{\phi}_{I^t}(\hat{x}^t) - \boldsymbol{\phi}_{I^t}(x^t)\rangle - \sum_{t=1}^{T}\zeta^t\end{aligned}$$

Applying the conditional $\alpha$-informative feedback (Eq. 5) and rearranging proves the claim. □

Theorem 1 ensures that the average conditional regret suffered by our algorithm decreases as $\mathcal{O}(1/\sqrt{T})$. This alone, however, does not prove that the algorithm will eventually arrive at a local optimum, even if $\sum_{t=\tau_0}^{T}\text{CREG}_{p^t}(x^t) = 0$, for some $\tau_0 \in [T]$. This is due to the fact that partial inference is performed keeping the rest of the object $x^t_{\overline{p}^t}$ fixed. Between iterations an inferred part may change as a result of a change of the other parts in previous iterations. The object could, in principle, keep changing at every iterations, even if $\text{CREG}_{p^t}(x^t)$ is always equal to 0. The next proposition, however, shows that this is not the case thanks to the utility decomposition we employ.

**Proposition 1.** *Let $\tau_1 < \ldots < \tau_n < \hat{\tau}_1 < \ldots < \hat{\tau}_n \leq T$ such that $p^{\tau_k} = p^{\hat{\tau}_k} = p_k$ and $\textit{CREG}_{p^t}(x^t) = 0$ for all $t \geq \tau_1$. The configuration $x^T$ is a local optimum.*

*Proof.* Sketch. The proof procedes by strong induction. We first show that $x^t_{p_1} = x^{\tau_1}_{p_1}$ for all $t > \tau_1$, as $u^t_{I_1}(\cdot)$ only depends on the features in $J_1$ and by assumption $\text{CREG}_{p^t}(x^t) = 0$ for all $t \geq \tau_1$. By strong induction, assuming that $x^t_{p_j} = x^{\tau_j}_{p_j}$ for all $j = 1, \ldots, k-1$ and all $t > \tau_{k-1}$, we can easily show that $x^t_{p_k} = x^{\tau_k}_{p_k}$ as well.

Now, $x^t_{p_k} = x^{\tau_k}_{p_k}$ for all $t > \tau_n$, therefore $x^{\hat{\tau}_k} = x^{\tau_n}$ for all $k \in [n]$. Since $\text{CREG}_{p_k}\left(x^{\hat{\tau}_k}\right) = 0$ for all $k \in [n]$ by assumption, then $x^T = x^{\hat{\tau}_n} = x^{\tau_n}$ is a local optimum and will not change for all $t > \tau_n$.

□

The algorithm actually reaches a local optimum at $t = \tau_n$, but it needs to double check all the parts in order to be sure that the configuration is actually a local optimum. This justifies a termination criterion that we use in practice: if the algorithm completes two full runs over all the parts, and the user can never improve any of the recommended partial configurations, then the full configuration is guaranteed to be a local optimum, and the algorithm can stop. As mentioned, we employ this criterion in our implementation but we left it out from Algorithm 2 for simplicity.

To recap, Theorem 1 guarantees that $\text{CREG}_T \to 0$ as $t \to \infty$, therefore $\text{CREG}_{p_t}(x^t)$ approaches 0 as well. Combining this fact with Proposition 1, we proved that our algorithm approaches a local optimum for $T \to \infty$.

## Empirical Analysis

We ran PCL on three constructive preference elicitation tasks of increasing complexity, comparing different degrees of user informativeness. According to our experiments, informativeness is the most critical factor. The three problems involve rather large configurations, which can not be handled by coactive interaction via complete configurations. For instance, in (Ortega and Stocker 2016) the user is tasked to solve relatively simple SAT instances over three variables and (at most) eight clauses; in some cases users were observed to show signs of cognitive overload. In comparison, our simplest realistic problem involve 35 categorical variables (with 8 possible values) and 74 features, plus additional hard constraints. As a consequence, Coactive Learning can not be applied as-is, and part-wise interaction is necessary.

In all of these settings, part-wise inference is cast as a mixed integer linear problem (MILP), and solved

[3]The complete proof can be found in the ArXiV version of the paper at XXX

with Gecode[4]. Despite being NP-hard in general, MILP solvers can be very efficient on practical instances. Efficiency is further improved by inferring only partial configurations. Our experimental setup is available at `https://github.com/unitn-sml/pcl`.

We employed a user simulation protocol similar to that of (Teso, Dragone, and Passerini 2017). First, for each problem, we sampled 20 vectors $\boldsymbol{w}^*$ at random from a standard normal distribution. Then, upon receiving a recommendation $x^t_{p^t}$, an improvement $\hat{x}^t_{p^t}$ is generated by solving the following problem:

$$\begin{aligned}\operatorname*{argmin}_{x_{p^t} \in \mathcal{X}_{p^t}} \quad & u^*[I^t](x_{p^t} \circ x^t_{\overline{p}^t}) \\ \text{s.t.} \quad & u^*[I^t](x_{p^t} \circ x^t_{\overline{p}^t}) - u^*[I^t](x^t_{p^t} \circ x^t_{\overline{p}^t}) \\ & \geq \alpha(u^*[I^t](x^*_{p^t} \circ x^t_{\overline{p}^t}) - u^*[I^t](x^t_{p^t} \circ x^t_{\overline{p}^t}))\end{aligned}$$

This formulation clearly satisfies the conditional $\alpha$-informativeness assumption (Eq. 5).

**Synthetic setting.** We designed a simple synthetic problem inspired by spin glass models, see Figure 1 for a depiction. In this setting, a configuration $x$ consists of a $4 \times 4$ grid. Each node in the grid is a binary 0-1 variable. Adjacent nodes are connected by an edge, and each edge is associated to an indicator feature that evaluates to 1 if the incident nodes have different values (green in the figure), and to $-1$ otherwise (red in the figure). The utility of a configuration is simply the weighted sum of the values of all features (edges). The basic parts $p$ consist of all the non-overlapping $2 \times 2$ sub-grids of $x$, for a total of 4 basic parts (indicated by dotted lines in the figure).

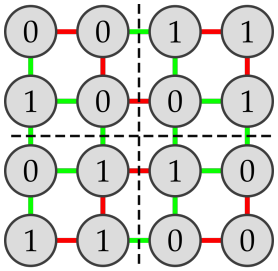


Figure 1: Example grid configuration.

Since the problem is small enough for inference of complete configurations to be practical, we compared PCL to standard Coactive Learning, using the implementation of (Teso, Dragone, and Passerini 2017). In order to keep the comparison as fair as possible, the improvements fed to CL were chosen to match the utility gain obtained by PCL. We further report the performance of three alternative part selection strategies: random, smallest (most independent) part first, and UCB1.

The results can be found in the first column of Figure 2. We report both the regret (over *complete* configurations) and the cumulative runtime of all algorithms, averaged over all users, as well as their standard deviation. The regret plot shows that, despite being restricted to work with $2 \times 2$ configurations, PCL does recommend *complete* configurations of quality comparable to CL after enough queries are made. Out of the three part selection strategies, random performs best, with the other two more informed alternatives (especially smallest first) quite close. The runtime gap between full and part-wise inference is already clear in this small synthetic problem; complete inference quickly becomes impractical as the problem size increases.

**Training planning.** Generating personalized training plans based on performance and health monitoring has received a lot of attention recently in sport analytics (see e.g. (Fister et al. 2015)). Here we consider the problem of synthesizing a week-long training plan $x$ from information about the target athlete. Each day includes 5 time slots (two for the morning, two for the afternoon, one for the evening), for 35 slots total. We assume to be given a fixed number of training activities (7 in our experiments: walking, running, swimming, weight lifting, pushups, squats, abs), as well as knowledge of the slots in which the athlete is available. The training plan $x$ associates an activity to each slot where the athlete is available. Our formulation tracks the amount of improvement (e.g. power increase) and fatigue over five different body parts (arms, torso, back, legs, and heart) induced by performing an activity for one time slot. Each day defines a basic part.

The mapping between training activity and improvement/fatigue over each body part is assumed to be provided externally. It can be provided by the athlete or medical personnel monitoring his/her status. The features of $x$ include, for each body part, the total performance gain and fatigue, computed over the recommended training plan according to the aforementioned mapping. We further include inter-part features to capture activity diversity in consecutive days. The fatigue accumulated in 3 consecutive time slots in any body parts does not exceed a given threshold, to prevent injuries.

In this setting, CL is impractical from both the cognitive and computational points of view. We ran PCL and evaluated the impact of user informativeness by progressively increasing $\alpha$ from 0.1, to 0.3, to 0.5. The results can be seen in Figure 2. The plots show clearly that, despite the complexity of the configuration and constraints, PCL can still produce very low-regret configurations after about 50 iterations or less.

Understandably, the degree of improvement $\alpha$ plays an important role in the performance of PCL and, consequently, in its runtime (users at convergence do not contribute to the runtime), at least up to $\alpha = 0.5$. Recall, however, that the improvements are part-wise, and hence $\alpha$ quantifies the degree of *local* improvement: part improvements may be very informative on their own, but only give a modest amount of information about the full configuration. However, it is not unreasonable to expect that users to be very informative when presented with reasonably sized (and simple) parts. Crucially PCL allows the system designer to define the parts appropriately depending on the application.

[4] `http://www.gecode.org/`

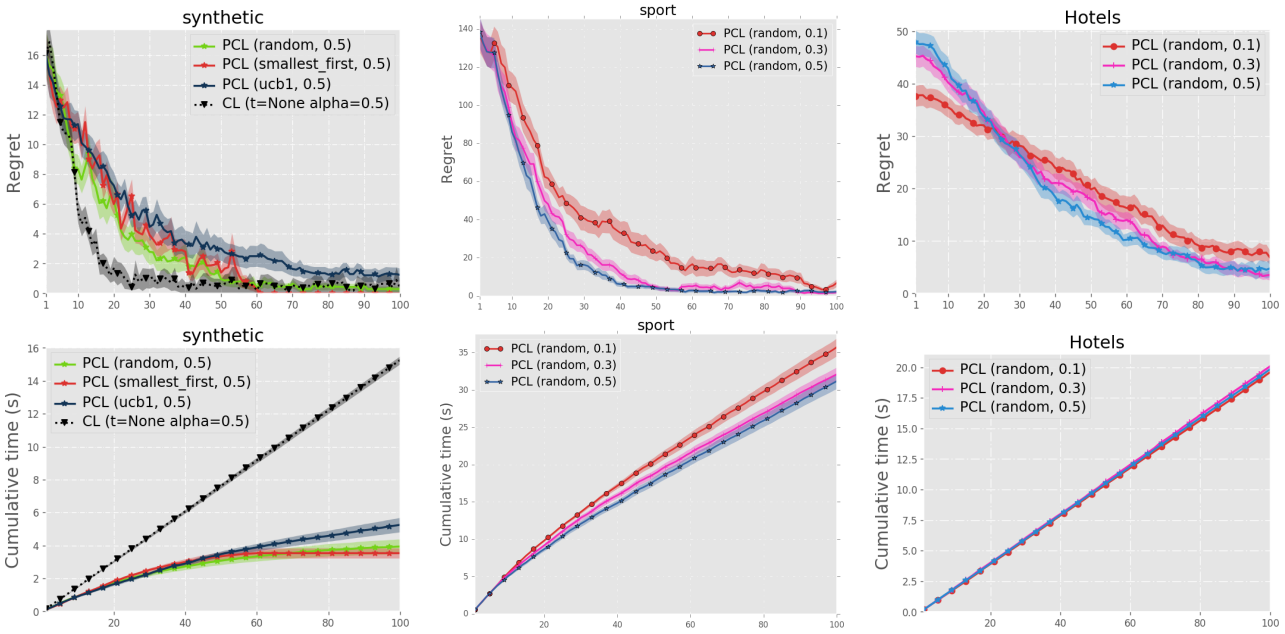


Figure 2: Regret over *complete* configurations (top) and cumulative runtime (bottom) of PCL and CL on our three constructive problems: synthetic (left), training planning (middle), and hotel planning (right). The $x$-axis is the number of iterations, while the shaded areas represent the standard deviation. Best viewed in color.

**Hotel planning.** Finally, we considered a complex furniture allocation problem: furnishing an entire hotel. The problem is encoded as follows. The hotel is represented by a graph: nodes are rooms and edges indicate which rooms are adjacent. Rooms can be of three types: normal rooms, suites, and dorms. Each room can hold a maximum number of furniture pieces, each associated to a cost. Additional, fixed nodes represent bathrooms and bars. The type of a room is decided dynamically based on its position and furniture. For instance, a normal room must contain at most three single or double beds, no bunk beds, and a table, and must be close to a bathroom. A suite must contain one bed, a table and a sofa, and must be close to a bathroom and a bar. Each room is a basic part, and there are 15 rooms to be allocated.

The feature vector contains 20 global features, e.g. the total number of possible guests and the total cost of the furniture, plus 8 local features per room. These include features shared by adjacent rooms, e.g. whether two rooms have the same type. These can encode preferences like "suites and dorms should not be too close". **Stefano: @Paolo: please add more.** Given the graph structure, room capacities, and total budget, the goal is to furnish all rooms according to the user's preferences.

This problem is hard to solve to optimality with current solvers; part-based inference alleviates this issue by focusing on individual rooms. There are 15 rooms in the hotel, so that at each iteration only 1/15 of the configuration is affected. Furthermore, the presence of the global features implies dependences between all rooms. Nonetheless, the algorithm manages to reduce the regret by an order of magnitude in around a 100 iterations, starting from a completely uninformed prior. Note also that as for the training planning scenario, an alpha of 0.3 achieves basically the same results as those for alpha equal to 0.5.

## Conclusion

In this work we presented an approach to constructive preference elicitation able to tackle large constructive domains, beyond the reach of previous approaches. It is based on Coactive Learning (Shivaswamy and Joachims 2015), but only requires inference of partial configurations and partial improvement feedback, thereby significantly reducing the cognitive load of the user. We presented an extensive theoretical analysis demonstrating that, despite working only with partial configurations, the algorithm converges to a locally optimal solution. The algorithm has been evaluated empirically on three constructive scenarios of increasing complexity, and shown to perform well in practice.

Possible future work includes improving part-based interaction by exchanging additional contextual information (e.g. features (Teso, Dragone, and Passerini 2017) or explanations) with the user, and applying PCL to large layout synthesis problems (Dragone et al. 2016).

## References

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47(2-3):235–256.

Boutilier, C.; Bacchus, F.; and Brafman, R. I. 2001. Ucp-networks: A directed graphical representation of conditional utilities. In *Proceedings of the Seventeenth conference on*

*Uncertainty in artificial intelligence*, 56–64. Morgan Kaufmann Publishers Inc.

Boutilier, C.; Patrascu, R.; Poupart, P.; and Schuurmans, D. 2006. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence* 170(8-9):686–713.

Braziunas, D., and Boutilier, C. 2005. Local utility elicitation in GAI models. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 42–49. AUAI Press.

Braziunas, D., and Boutilier, C. 2007. Minimax regret based elicitation of generalized additive utilities. In *UAI*, 25–32.

Braziunas, D., and Boutilier, C. 2009. Elicitation of factored utilities. *AI Magazine* 29(4):79.

Chajewska, U.; Koller, D.; and Parr, R. 2000. Making rational decisions using adaptive utility elicitation. In *AAAI/IAAI*, 363–369.

Dragone, P.; Erculiani, L.; Chietera, M. T.; Teso, S.; and Passerini, A. 2016. Constructive layout synthesis via coactive learning. In *Constructive Machine Learning workshop, NIPS*.

Fishburn, P. C. 1967. Interdependence and additivity in multivariate, unidimensional expected utility theory. *International Economic Review* 8(3):335–342.

Fister, I.; Rauter, S.; Yang, X.-S.; and Ljubič, K. 2015. Planning the sports training sessions with the bat algorithm. *Neurocomputing* 149:993–1002.

Goetschalckx, R.; Fern, A.; and Tadepalli, P. 2014. Coactive learning for locally optimal problem solving. In *Proceedings of AAAI*.

Gonzales, C., and Perny, P. 2004. GAI networks for utility elicitation. *KR* 4:224–234.

Keeney, R. L., and Raiffa, H. 1976. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*.

Mayer, R. E., and Moreno, R. 2003. Nine ways to reduce cognitive load in multimedia learning. *Educational psychologist* 38(1):43–52.

Meseguer, P.; Rossi, F.; and Schiex, T. 2006. Soft constraints. *Foundations of Artificial Intelligence* 2:281–328.

Ortega, P. A., and Stocker, A. A. 2016. Human decision-making under limited time. In *Advances in Neural Information Processing Systems*, 100–108.

Pigozzi, G.; Tsoukiàs, A.; and Viappiani, P. 2016. Preferences in artificial intelligence. *Ann. Math. Artif. Intell.* 77(3-4):361–401.

Raman, K.; Shivaswamy, P.; and Joachims, T. 2012. Online learning to diversify from implicit feedback. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 705–713. ACM.

Shivaswamy, P., and Joachims, T. 2015. Coactive learning. *JAIR* 53:1–40.

Teso, S.; Dragone, P.; and Passerini, A. 2017. Coactive critiquing: Elicitation of preferences and features. In *AAAI*.

Teso, S.; Passerini, A.; and Viappiani, P. 2016. Constructive preference elicitation by setwise max-margin learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2067–2073. AAAI Press.