Preference Elicitation for Multi-objective Combinatorial Optimization with Active Learning and Maximum Likelihood Estimation

Marianne Defresne , **Jayanta Mandi** and **Tias Guns**

Department of Computer Science, KU Leuven, Leuven, Belgium

{marianne.defresne, tias.guns}@kuleuven.be

Abstract

Real-life combinatorial optimization problems often involve several conflicting objectives, such as price, product quality and sustainability. A computationally-efficient way to tackle multiple objectives is to aggregate them into a single-objective function, such as a linear combination. However, defining the weights of the linear combination upfront is hard; alternatively, the use of interactive learning methods that ask users to compare candidate solutions is highly promising. The key challenges are to generate candidates quickly, to learn an objective function that leads to high-quality solutions and to do so with few user interactions. We build upon the Constructive Preference Elicitation (CPE) framework and show how each of the three properties can be improved: to increase the interaction speed we investigate using pools of (relaxed) solutions, to improve the learning we adopt Maximum Likelihood Estimation of a Bradley-Terry preference model; and to reduce the number of user interactions, we select the pair of candidates to compare with an ensemble-based acquisition function inspired from Active Learning. Our careful experimentation demonstrates each of these improvements: on a PC configuration task and a realistic multi-instance routing problem, our method selects queries faster, needs fewer queries and synthesizes higher-quality combinatorial solutions than previous CPE methods.

1 Introduction

Combinatorial optimization (CO) problems are omnipresent in real-life decision-making, such as scheduling and routing. They often require balancing multiple objectives. For instance, for a routing problem, a Decision Maker (DM) may wish to balance duration, fuel consumption and driver familiarity. One computationally-efficient way to tackle such a multi-objective CO problem is to build an approximate single-objective function aggregating individual (sub)objectives. The simplest and most common aggregation is a linear combination of sub-objectives [Aneja and Nair, 1978; Halffmann *et al.*, 2022]. The main challenge lies in defining a set of weights leading to desirable solutions. For a DM, explicitly stating how sub-objectives should be balanced is difficult. However, comparing two solutions is an easier task, and a choice for one over the other implicitly informs about underlying preferences.

We aim to learn a linear objective function from such pairwise comparisons, where that linear function can then be used to synthesize new desirable solutions by calling a CO solver. The goal fits within the general preference elicitation framework [Guo and Sanner, 2010]. Preferences are captured in a utility function, often a weighted average over attributes of an object. In the general preference elicitation framework, the learning task is to estimate weights such that after learning, the DM's choice from a pool of alternative objects has the highest estimated utility.

In CO, the utility is the single-objective function, an attribute is a sub-objective and an object is a solution. In this case, there is no explicit pool of objects. Indeed, feasible solutions are implicitly defined by the set of constraints and there can be exponentially many solutions. The learning task is then to estimate weights such that the resulting utility function leads to desirable solutions when used as an objective function for the multi-objective CO problem. Constructive Preference Elicitation (CPE) [Dragone *et al.*, 2018a] was introduced for this setting. As in regular preference elicitation, weights are estimated in a two-step loop: first, a query, *e.g.* objects to compare, is selected and the DM is asked to express a preference. Second, the weight estimates are updated accordingly. What makes this setting *constructive* is that the learned utility function is then used to synthesize the best possible solution by calling a CO solver.

Such preference elicitation involves interacting with a human DM. Among the target properties listed by Guo and Sanner [2010], the one that comes first is the ability to propose queries in *real-time*. Yet, existing CPE methods select a query by using a CO solver to generate the objects [Teso *et al.*, 2016; Dragone *et al.*, 2018b]. When learning preferences over NP-hard problems, this can become prohibitively time intensive. The other target properties listed are 2) *multi-attribute* in utility function, 3) inducing a *low cognitive load* as difficult queries have noisy answers, 4) *robust* to noise in the responses and 5) requiring a minimal number of queries. For CO problems, we add that the method should be 6) *constructive* and 7) *contextual* to adapt to multiple instances of

the same CO. Indeed, the DM's preferences depend on the CO problem but not on a specific instance. For instance, a DM is likely to balance total distance and fuel consumption similarly for different sets of stops.

We propose a new CPE approach that follows these properties. Our main insight is that *real-time* interaction on NP-hard problems requires to bypass the need for a solver during query selection. For this we propose using a pool of precomputed (relaxed) solutions. After learning, a CO solver is still required to synthesize a new desirable solution by optimizing the learned objective function over the feasible space; thus, the method is *constructive*. Queries prompt the user to compare two solutions; pairwise comparisons keep the *cognitive load* low [Conitzer, 2007]. For *robust* learning, both with respect to solution quality and *number of queries*, we adopt a Maximum Likelihood Estimation (MLE) of the Bradley-Terry preference model [Bradley and Terry, 1952]. Finally, our method learns preferences over multiple instances of similar CO problem instances, making it *contextual*. Our main contributions are:

1. We propose to select queries from a precomputed pool of (relaxed) solutions; it bypasses the need of solving CO problems during training;
2. For query selection, we adapt the ensemble-based Upper Confidence Bound acquisition function from the active learning literature, a non-linear function that would be difficult for a CO solver to optimize directly;
3. Inspired by Deep Reinforcement Learning, we learn the weights by maximizing the likelihood of a Bradley-Terry model. We draw connections with Noise Contrastive Estimation (NCE) and the previously-used Structured Perceptron, which shows that MLE can be interpreted as a smooth version thereof;
4. We assess our method by extensive experiments on two multi-objective CO problems: a previously-used PC configuration problem and a multi-instance Prize-Collecting Traveling Salesperson Problem (PC-TSP). On both tasks, our method is orders of magnitude faster, requires fewer queries and synthesizes higher-quality solutions than existing CPE approaches.

2 Related Work

Preference Elicitation. It aims to estimate the parameters of a utility function representing preferences. It relies on an incremental interactive approach of selecting a query and updating the parameter estimate based on the feedback received [Pigozzi *et al.*, 2016]. Most approaches aim to select the best choice out of an explicit finite set of alternatives. As such, they are not *constructive* as we consider here. Examples include multi-attribute utility theory [Braziunas and Boutilier, 2007] and multi-criteria decision-making [Martyn and Kadziński, 2023; Guo *et al.*, 2021]. Moreover, these approaches are either Bayesian and thus computationally expensive to train [Guo and Sanner, 2010], or exact regret-based and not robust to inconsistent answers [Boutilier, 2013; Toffano *et al.*, 2022]. All these works are non-contextual as they consider single-instance problems. Some non-constructive approaches use more sophisticated aggregation function than

a linear utility, such as Choquet integral [Herin *et al.*, 2024; Vernerey *et al.*, 2024].

Preference elicitation for CO. The dominating approach to elicit preference weights in the context of multi-objective CO is the polyhedral method [Toubia *et al.*, 2004; Benabbou and Perny, 2018; Bourdache and Perny, 2019; Bourdache *et al.*, 2020]. Each query reduces uncertainty over the weight space, until the remaining sets of weights lead to the same solution. This weight update is based on minimax regret, which is computationally expensive (thus limiting *real-time* interactions), and it assumes no noise in the DM's answer as that would make the polyhedron empty through inconsistent constraints. Additionally, the number of sub-objectives that can be considered is limited. Therefore, this approach does not fulfill the *real-time*, *robustness* and *contextual* properties and is restricted in the *multi-attribute* property. We chose to build upon the CPE framework [Dragone *et al.*, 2018a] because it is *constructive*, *contextual*, *robust* and is has been extended to *multi-attribute* [Dragone *et al.*, 2018b]. However, it is not *real-time* as query selection relies on solving CO problems, potentially NP-hard. Previous CPE approaches have considered the learning task as structured-output prediction, the output being a CO feasible solution. Weights are updated with the simple Structured Perceptron [Collins, 2002], adapted for preference data [Shivaswamy and Joachims, 2015].

Active and Preference Learning. Our investigation adapts ideas from the active learning community to a constructive setting. First, queries are selected from a precomputed pool of solutions using UCB (upper confidence bound) [Cox and John, 1992; Archetti and Candelieri, 2019] as an acquisition function. UCB has strong theoretical results in the context of dueling bandits [Srinivas *et al.*, 2010] and has been used for active learning [Pandi *et al.*, 2022]. It quantifies both the quality and the uncertainty of each solution. To estimate uncertainty, we use an ensemble of weights, as also used in query-by-committee in active learning [Seung *et al.*, 1992]. Second, we revisit the weight update using Maximum Likelihood Estimation of a Bradley-Terry preference model [Bradley and Terry, 1952]. It was first proposed for Deep Reinforcement Learning for preference learning [Christiano *et al.*, 2017] and it is the objective used to align Large Language Models [Rafailov *et al.*, 2024]. To the best of our knowledge, it is the first time this objective is used in the context of CPE.

Learning an objective function. As we represent preferences as a weighted average of sub-objectives, learning preferences is equivalent to learning the parameters of an objective function. In that respect, it is related to Decision-Focused Learning (DFL) [Mandi *et al.*, 2024] and inverse optimization [Chan *et al.*, 2023]. Instead of preference data (in our case, comparing a pair of solutions), these approaches learn from near-optimal solutions and/or historic parameters in a given dataset. In DFL, features are provided to estimate the parameters from [Sadana *et al.*, 2024]. Data-driven inverse optimization uses the same notion of context as we do: parameters are estimated for multiple instances of the same CO problem. However it learns from (near) optimal solutions rather than querying a user as in our case.

Algorithm 1 Template for CPE

Initial weight estimate w^0
for $t < T$ **do**
    Observe problem instance p^t
    $(y_1, y_2) \leftarrow$ SELECT QUERY(p^t, w^t)
    DM chooses y_+ from (y_1, y_2)
    $w^{t+1} \leftarrow$ WEIGHT UPDATE(w^t, y_1, y_2)
end for
return $\hat{y} = \text{argmax}_{y \in \mathcal{F}} \langle w^T, \phi(y) \rangle$

---

3 Problem Statement

We consider a CO problem defined over a set of variables $\{y_1, \ldots, y_l\}$, where each variable y_i takes values from its corresponding domain D_i. The Cartesian product of the domains, $\mathcal{Y} = \prod_{i=1}^{l} D_i$, represents the space of all possible assignments. This space is restricted by a set of constraints, defining a feasible region $\mathcal{F} \subseteq \mathcal{Y}$. The quality of an assignment $y \in \mathcal{Y}$ (whether feasible or not) is evaluated using n sub-objectives, represented by a function $\phi : \mathcal{Y} \to \mathbb{R}^n$. As common in utility theory [Keeney, 1993], we assume sub-objectives can be aggregated into a utility function u that captures the DM's preferences over assignments. An assignment y_+ is preferred to y_-, denoted by $y_+ \succ y_-$, iff $u(y_+) > u(y_-)$. We further assume the utility to be linear in the sub-objectives ϕ:

$$u(y) = \langle w, \phi(y) \rangle = \sum_{i=1}^{n} w_i \phi_i(y)$$

with w_i the weighting of the i-th sub-objective. The learning task is to estimate the weights $w \in \mathbb{R}^n$ such that the synthesized solution aligns with the DM's preferences. Weights are estimated through interaction with the DM. We follow the standard preference elicitation process [Guo and Sanner, 2010]: at each iteration, a query is selected and the weights are updated based on the DM feedback.

Because the utility function u is linear, it can easily be used as an objective function to the CO problem, to synthesize a desirable solution:

$$\hat{y} = \underset{y \in \mathcal{F}}{\text{argmax}} \, u(y)$$

To limit the cognitive burden for the DM [Conitzer, 2007], we restrict queries to be comparisons between pairs of assignments. We allow the DM to indicate indifference (for example because the assignments are equally preferred or considered incomparable). Furthermore, we take into account that a DM can make mistakes or provide inconsistent answers leading to noisy observations.

All problem instances are assessed with the same n sub-objectives, and we assume that the DM balances the sub-objective in the same manner for all instances. We call a specific instance p the *context*. When needed, we will make the use of context explicit, *e.g.* $u(y, p) = \langle w, \phi(y, p) \rangle$. Note the preference weights w are agnostic to the specific instance p, but the sub-objectives ϕ are a function of p. In a routing problem, p may represent the distance between stops, and thus be used to compute the total distance travelled of any route y.

Algorithm 2 Proposed method

Input: number of steps T, number of sub-objectives n, number of models m, number of clusters
Output: synthesized solution $\hat{y}$
1: **for** $i \in \{1, \ldots, m\}$ **do**   ▷ Initial weight estimate
2:    $W_i^0 \leftarrow \mathcal{N}_n(1, \mathbb{I}_n)$
3: **end for**
4: Dataset $\mathcal{D} \leftarrow \emptyset$
5: Generate a pool of random solutions Y
6: Cluster Y with k-means
7: **for** $t \in \{0, \ldots, T\}$ **do**
8:    Observe problem instance p^t
9:    **for** $l \in \{1, 2\}$ **do**   ▷ Query selection
10:        Select a cluster C at random
11:        $y_l \leftarrow \text{argmax}_{y \in C} \text{UCB}(W^t, y)$
12:    **end for**
13:    Ask DM for preference label a
14:    $\mathcal{D} \leftarrow (p^t, (y_1, y_2), a)$
15:    **for** $i \in \{1, \ldots, m\}$ **do**   ▷ Weight update
16:        $W_i^{t+1} \leftarrow \text{MLE}(\mathcal{D}, W_i^0)$
17:    **end for**
18: **end for**
19: $w \leftarrow \text{mean}_i(\{W_i^t\}_{i \in \{1, \ldots, m\}})$
20: **return** $\hat{y} = \text{argmax}_{y \in \mathcal{F}} \langle w, \phi(y) \rangle$

Formally, the set of preference observations to learn from will be $\{(p, (y_1, y_2), a))\}$ with y_1 and y_2 two assignments of the instance p. The preference label is $a = 1$ if $y_1 \succ y_2$, $a = -1$ if $y_2 \succ y_1$ and $a = 0$ if the DM is indifferent. The data is collected from the DM one data point at a time, during the preference elicitation process. We aim to improve the overall interaction by selecting queries in real-time and minimizing the number of queries required for effective learning.

4 CPE with Active Learning & Likelihood

We instantiate the CPE framework [Dragone *et al.*, 2018a], summarized in Algorithm 1. We propose improvements to its two main steps, query selection and weight update. Algorithm 2 presents the pseudo-code of our proposed approach.

4.1 Active Learning-based Query Selection

A key step in our CPE approach is selecting a query (y_1, y_2) given a context p and ask for feedback. Existing CPE methods call a solver twice to select two feasible solutions. Eliciting preferences this way for NP-hard problems can be computationally expensive, leading to waiting times for the DM and limiting the number of interactions possible. Instead, we build a pool of precomputed solutions, from which two solutions are selected. This bypasses the need for solver calls during training, enabling *real-time* query selection. After training, we will still call the solver to synthesize a new solution (line 20 in Algorithm 2).

Pool generation.   A large pool of solutions is built before training (line 5). We assume that either a large number of feasible solutions can be enumerated or sampled [Pesant *et al.*, 2022], or that relaxed solutions can be efficiently generated. Relaxed solutions are typically problem-specific and

such that it is still meaningful for a DM to express preferences over. For instance, in a PC-TSP where each city offers a prize but only a subset can be visited, a feasible solution is a valid (sub)circuit collecting a minimal reward, while a relaxed solution is simply any (sub)circuit. A non-meaningful relaxation would be the generation of disconnected routes. Problem-specific relaxations, if applicable, are typically very efficient to generate; for example, for TSP it amounts to generating permutations of subsets of stops.

Acquisition function. Not having to call a solver to generate a solution during query selection has a second major advantage: we are not limited to acquisition functions that the solver can efficiently optimize over. For instance, CPE method Choice Perceptron [Dragone *et al.*, 2018b] selects a query (y_1, y_2) by solving the following two CO problems:

$$y_1 = \underset{y \in \mathcal{F}}{\operatorname{argmax}} \, u(y)$$
$$y_2 = \underset{y \in \mathcal{F}}{\operatorname{argmax}} (1 - \gamma) u(y) + \gamma \|\phi(y_1) - \phi(y)\|_1$$

where y_1 uses utility as acquisition function and y_2 a linear combination of utility and L1 distance to y_1's sub-objectives.

Instead, we take inspiration from active learning [Balcan *et al.*, 2010], where the acquisition strategy [Seung *et al.*, 1992] is often based on selecting the most uncertain sample. To estimate uncertainty, techniques from preference Reinforcement Learning [Christiano *et al.*, 2017; Marta *et al.*, 2023] commonly train an ensemble of models, each one giving one estimation of the utility function. The most uncertain assignment is then the one on which the ensemble disagree the most, *e.g.* as measured by the variance of the ensemble predictions.

In CPE, both uncertainty and solution quality matter. However, variance is a non-linear function, making it difficult for a solver to optimize. Identifying the solution with the highest variance in the pool is much simpler. Thus, the pre-computed pool enables us to propose an ensemble-based acquisition function that balances both uncertainty and solution quality. We propose to train an ensemble $W^t = \{W_1^t, \ldots, W_m^t\}$ composed of m weight vectors $W_i^t \in \mathbb{R}^n$, all initialized differently following a normal distribution of mean 1 and variance 1 (line 2 in Algorithm 2). Uncertainty is measured by the variance σ_{W^t} of the ensemble W^t (at time step t) and solution quality by its mean μ_{W^t}. It corresponds to the UCB [Archetti and Candelieri, 2019]:

$$\text{UCB}(W^t, y) = (1 - \gamma) \mu_{W^t}(y) + \gamma \sigma_{W^t}(y)$$

with $\gamma \in [0, 1]$ controlling the exploration/exploitation trade-off. As in [Dragone *et al.*, 2018b], we favour exploration at early training stages by setting $\gamma = 1/t$. We will use UCB twice as the acquisition function, to select two distinct solutions from the solution pool as queries.

Clustering. The two highest-scoring UCB solutions may be highly similar or even identical in sub-objective values. A DM is likely to be indifferent to such a query, which would not provide a training signal. To ensure more diversity, previous work in preference-based Reinforcement Learning [Marta *et al.*, 2023] proposed to cluster the samples in a (learned) latent representation space. In the case of
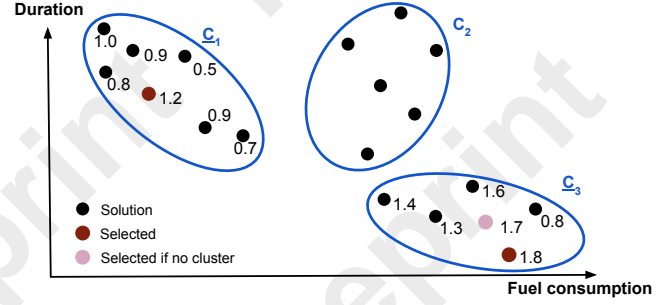

Figure 1: Query selection on a routing problem with 2 sub-objectives (duration and fuel consumption) : 1) the pool of solution is clustered based on the sub-objectives (k=3), 2) two clusters (underlined) are chosen at random, 3) the UCB values in those clusters are computed and 4) the solution with the highest UCB value in each cluster is selected (in red).

CO, the sub-objectives $\phi : Y \rightarrow \mathbb{R}^n$ can be interpreted as a latent representation of an assignment y. Therefore, we cluster the solution pool based on the sub-objective values, using k-means++. At each iteration, two clusters are randomly chosen and the solution with the highest UCB is selected in each cluster (line 11 in Algorithm 2). The resulting pairwise query is both informative (thanks to UCB) and diverse (due to clustering). Our query selection is illustrated in Figure 1.

Complexity. The time complexity of a single query selection is the complexity of sorting the points of two clusters by UCB value, $O(2 * q \log(q))$ with q the size of the largest cluster. It does not depend on the complexity of the CO problem considered, resulting in high-efficiency even for NP-hard and large-scale problems. The computation time of the solutions is amortized in the pre-computation of a large pool of feasible or relaxed solutions, which can easily be parallelized.

4.2 Weight Update with Maximum Likelihood

At each iteration, weight estimates are updated according to the received feedback on the selected query. If the DM is not indifferent, they select one preferred solution, denoted y_+ (the other one is denoted y_-). In case of indifference, a new query is selected. Existing CPE approaches [Dragone *et al.*, 2018a; Dragone *et al.*, 2018b] update preference weights using the Preference Perceptron (PP) [Shivaswamy and Joachims, 2015], a variant of the Structured Perceptron (SP) [Collins, 2002] for preference data. The weight update rule of PP is based on how the two selected solutions differ in sub-objectives, measured by $\Delta = \phi(y_+, p) - \phi(y_-, p)$. In the following, we omit the context p to simplify notation. The PP weight update rule is

$$w_{\text{PP}}^{t+1} = w_{\text{PP}}^t + \eta . \Delta$$

with η the learning rate. The original SP is updated only in case of misprediction. As with preference data, the prediction is incorrect if $u^t(y_+) < u^t(y_-)$, the SP-based update rule is:

$$w_{\text{SP}}^{t+1} = w_{\text{SP}}^t + \eta . \mathbb{1}_{u^t(y_+) < u^t(y_-)} \Delta$$

Maximum Likelihood

Instead of considering the learning task as structured-output prediction, we formulate it with Maximum Likelihood Es-
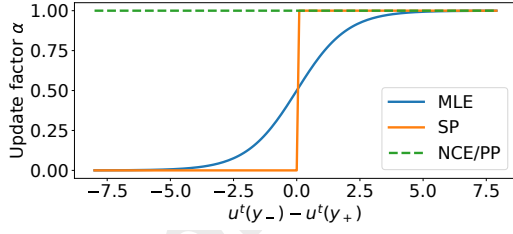
Figure 2: Comparing the update factor of MLE, SP and PP/NCE.

timation (MLE). Under the well-established Bradley-Terry model [Bradley and Terry, 1952], the probability of y_+ being preferred over y_- is

$$P(y_+ \succ y_-) = \frac{e^{u^*(y_+)}}{e^{u^*(y_+)} + e^{u^*(y_-)}}$$

with u^* the true (unknown) DM's utility. Training with MLE aims to maximize this probability. As usual, MLE is re-written as a loss L minimizing the negative log-likelihood:

$$\begin{aligned}
L &= -\log \frac{e^{\sum_i w_i \phi_i(y_+)}}{e^{\sum_i w_i \phi_i(y_+)} + e^{\sum_i w_i \phi_i(y_-)}} \\
&= -\log \frac{1}{1 + e^{-\sum_i w_i [\phi_i(y_+) - \phi_i(y_-)]}} \\
&= -\log \mathcal{S}(\sum_i w_i \Delta_i)
\end{aligned}$$

With $\mathcal{S}$ the sigmoid function and $\Delta_i = \phi_i(y_+) - \phi_i(y_-)$. This loss is the standard ML objective to fit a Bradley-Terry model [Rafailov et al., 2024], but has not been used for CPE so far.

Using gradient descent, the update rule is: $w_{t+1} = w_t - \eta \nabla L$. The partial gradients w.r.t. to a given weight w_i are:

$$\frac{\partial L}{\partial w_i} = -\Delta_i \frac{e^{-\sum_j w_j \Delta_j}}{1 + e^{-\sum_j w_j \Delta_j}} = -\Delta_i \frac{1}{1 + e^{\sum_j w_j \Delta_j}} = -\alpha \Delta_i$$

With $\alpha = \mathcal{S}(-\sum_j w_j \Delta_j) = \mathcal{S}(u(y_-) - u(y_+))$. Note that α is independent of i and is therefore the same for all weights w_i. Therefore, the MLE update rule is

$$w_{\text{MLE}}^{t+1} = w_{\text{MLE}}^t + \eta.\mathcal{S}(u^t(y_-) - u^t(y_+))\Delta$$

When an ensemble is used for query selection, this update is applied to each weight vector of the ensemble.

A unified view

The PP update can be interpreted in the context of Noise Contrastive Estimation (NCE). As in [Mulamba et al., 2021], we write the probability of a solution y as $P(y|w) = e^{u(y,w)}$. Then, the infoNCE loss [Oord et al., 2018] can be used to increase the separation between the preferred solution y_+ (positive sample) and the other solution y_- (negative sample):
$L^{\text{NCE}} := -\log \frac{P(y_+|w)}{P(y_-|w)} = -\log \frac{e^{u(y_+,w)}}{e^{u(y_-,w)}} = -\sum_i w_i \Delta_i$.
The partial gradients $\frac{\partial L}{\partial w_i} = -\Delta_i$ lead to the same gradient update as PP: $w_{NCE}^{t+1} = w_{NCE}^t + \Delta$.

All three weights updates – SP, PP/NCE and MLE – can be expressed using an update factor α and a learning rate η:

$$w^{t+1} = w^t + \eta.\alpha^t \Delta$$

There is $\alpha^t = \mathbb{1}_{u^t(y_+) < u^t(y_-)}$ for SP, $\alpha^t = 1$ for PP/NCE and $\alpha^t = \mathcal{S}(u^t(y_-) - u^t(y_+))$ for MLE. The update factor α^t ranges between 0 and 1 and controls the amplitude of the gradient step amplitude. As plotted in Figure 2, MLE can be interpreted as a smoothed version of SP. When the difference in predicted utility between y_+ and y_- is large, i.e., a clear preference is predicted, the update factor tends to 0 if the prediction is correct and is maximum (1) otherwise. When no clear preference is predicted , the MLE update factor is a function of the utility difference.

Batch learning. Previous CPE approaches update weights online, i.e., after each received feedback. We instead propose to train MLE with batch training. We create a feedback dataset and retrain the whole model at each iteration (line 16 in Algorithm 2). Since the model is a single linear vector – the estimated weights w – fully retraining it is still fast.

5 Experiments
5.1 Experimental Setting
Baselines. We re-implemented the two CPE methods applicable for pairwise queries. The first, Set-wise Max-margin (SetMargin) [Teso et al., 2016] does not follow the preference elicitation steps but rather turns each data point into a constraint, and the weight vector is optimized to have the largest separation margin. This method is limited to Boolean sub-objectives. The second, Choice Perceptron [Dragone et al., 2018b], was proposed to overcome this limitation and handles both Boolean and numerical sub-objectives. With the Choice Perceptron, weights are updated according to the PP weight update, and queries (y_1, y_2) are selected by solving as explained in Section 4.1. It uses $\gamma = 1/t$ with t an iteration counter. It also includes costly learning-rate tuning between iterations that we omit for more equal comparison.

Tasks. We first consider the single-instance task of PC Configuration used by both baselines as a constructive variant of PC recommendation [Guo and Sanner, 2010]. Each configuration is described by 7 components – such as storage and CPU model – and the price. Sub-objectives are Boolean (77 in total) indicating if a component is selected, plus the normalized price. We also consider the more realistic multi-instance task of Prize-Collecting Traveling Salesperson Problem (PC-TSP), previously used for structured prediction [Véjar et al., 2024]. Each node represents a city, which offers a prize if visited and costs a penalty ρ if not. Each edge (i, j) has $k = 4$ values v_k^{ij} that can be interpreted as distances, duration, fuel consumption and driver familiarity. The total penalty is a 5th sub-objective. Mathematically, the CO problem objective is

$$\min_{y \in \mathcal{F}} \sum_{l=1}^k w_l \sum_{(i,j) \in E} v_l^{ij} y_{ij} + w_{k+1} \sum_{i \in V} \rho^i (1 - \sigma_i)$$

where Boolean variables $y_{i,j}$ (resp. σ_i) indicate whether the edge (i, j) (resp. city i) is part of the solution. The constraints state that the chosen edges form a Hamiltonion circuit. All sub-objectives are normalized by dividing the columns of the distance matrices by its largest value, ensuring all entries lies

Query	SetMargin	ChoicePerc	r-pool (ours)
Configuration	1.9	0.54	0.004
PC-TSP 10	*NA*	0.24	0.001
PC-TSP 20	*NA*	52.2	0.004
PC-TSP 100	*NA*	*Timeout*	0.050

Table 1: Average time (s) for selecting a single query single. Set-Margin is not applicable to the PC-TSP (indicated by *NA*).

in $[0, 10]$. We consider problems with 10, 20 and 100 nodes. These problems are multi-instance: we train on 50 instances and test (*i.e.,* synthesize solutions) on 10 unseen instances.

User simulation. We replicate the same experimental protocol as baselines. Each experiment is repeated 20 times, each with a different DM whose ground-truth utility is obtained by randomly sampling each weight. For the configuration task, DM weights were sampled from a normal distribution $\mathcal{N}(25, 25/3)$ [Teso *et al.*, 2016] then 80% were randomly set to 0 to create sparse preferences. For the PC-TSP, DM weights were sampled from a Dirichlet's distribution of concentration parameter 100 [Véjar *et al.*, 2024].

The DM's response to a query is simulated based on the Bradley-Terry model extended to indifference, to create noisy responses including indifference [Guo and Sanner, 2010]. For both tasks, the quality of synthesized solutions is assessed using relative regret: $\frac{u^*(y^*)-u^*(\hat{y})}{u^*(y^*)}$, with $\hat{y}$ the synthesized solution and y^* the true optimal solution. At test time, we measure how many of the 20 DMs are satisfied, meaning that they are indifferent between $\hat{y}$ and y^*. We also measure the number of queries needed to reach an average relative regret below 10%. This threshold is considered an 'acceptable' solution by Teso et al [2016].

Implementation. The experiments are implemented using CPMpy [Guns, 2019] and GurobiPy [Gurobi Optimization, LLC, 2024]. MLE was trained both online and with batch training, the latter for 4 epochs and a batch size of 4. Parameters were tuned based on a separate set of 10 DMs. After tuning, the learning rate for the configuration task is 2; for PC-TSP it is 0.5 for SP-online and MLE-online, 0.1 for PP-online and 1 for MLE-batch. We applied 100 training steps like [Teso *et al.*, 2016; Dragone *et al.*, 2018b]. For the ensemble-based acquisition functions, an ensemble of $m = 25$ models is used. Since the sub-objectives are Boolean for the Configuration problem, all configurations with the same number of identical components are equidistant, so no clustering is applied. For PC-TSP, the number of clusters used was 5. For each problem, a pool of $10,000$ different random relaxed solutions was generated, by randomly selecting attributes (configuration task) or by randomly sampling node permutations (PC-TSP). For PC-TSP, varying-size circuits were generated (the circuit is completed when the warehouse node is reached). The code is available at: https://github.com/mdefresne/CPE_pool.

5.2 Query Selection Time
Table 1 compares the average time to select a single query for the SetMargin method, the Choice Perceptron method, and our method with a solution pool of $10,000$ relaxed solutions

(r-pool). Due to its Boolean attribute restriction, SetMargin can only be run on the Configuration problem, and we can see that it has the slowest query selection here. For the Choice Perceptron, we can see that the runtime is heavily influenced by the difficulty of the CO problem, even timing out (300s) for PC-TSP problems of size 100. Our query selection using the solution pool takes marginal compute time.

Most of our computation time is offline, before interacting with the DM. The generation of the relaxed solution pool takes fewer than 20 minutes for the largest considered problem (PC-TSP with 100 nodes). Note that in the Choice Perceptron, one could also replace the solver call with an argmax call over our solution pool to achieve similar speed-ups.

5.3 Impact of Query Selection and Weight Update
We now compare the baseline query selection technique of the Choice Perceptron, with our proposed UCB selection over a pool of relaxed solutions. For each, we also compare the three weight updates discussed (the standard Choice Perceptron uses 'PP'), plus batched MLE. Results for Configuration and for the Prize-collecting TSP are shown in Table 2.

Selecting queries with UCB on a relaxed pool outperforms the Choice Perceptron query selection on both tasks and for each weight update except PP + ChoicePerc on Configuration. On the other cases, using UCB leads synthesized solutions of higher quality – both in regret and % DM – and up to half the number of queries are required. The results are especially pronounced on the multi-instance PC-TSP problem, where using the UCB query selection leads to DMs being satisfied across all the weight updates, and with considerably fewer queries needed to reach 10% regret (#Q).

Looking closer at the different weight updates, SP performs the worst overall. The smooth variant MLE-online improves all metrics on both problems and both query selections, showing that the smooth weight update is beneficial. Moving from online to batch re-training (MLE-batch) provides further improvements, in particular in the required number of queries. Overall, the choice of the UCB query selection with the MLE-batch update rule leads to higher-quality solutions by a wide margin, while requiring fewer queries.

5.4 Ablations
Acquisition function. We assess how restricting the acquisition function to either uncertainty or solution quality affects performances in Table 3. Focusing only on solution quality is achieved by setting γ to zero; it is close to CO-based query selection schemes. Using only uncertainty restricts UCB to the variance ($\gamma = 1$) and this is how ensemble-based Active Learning is usually used for single samples. In both cases, the synthesized solution quality drops while requiring more queries. Using only the mean results in 3 times lower regret than with variance only, confirming the importance of solution quality in a constructive setting.

We also verify the effect of using clustering to increase diversity (only applicable to PC-TSP). We here describe the results of that experiment: when no clustering was used, 98.3% of selected pairs were similar, *i.e.,* they would have belonged to the same cluster. This lack of diversity led to more indifferent answers: 8.0%, vs 0.7% with clustering. The quality

| Query | Configuration | | | | | | Prize-Collecting TSP | | | | | |
| | ChoicePerc | | | UCB, rpool (ours) | | | ChoicePerc | | | UCB, rpool (ours) | | |
Update	Regret (%)	#Q	% DM	Regret (%)	#Q	% DM	Regret (%)	#Q	% DM	Regret (%)	#Q	% DM
PP-online	6.6 ± 0.5	51	55	7.7 ± 0.5	67	45	2.7 ± 0.2	66	75	1.1 ± 0.05	29	**100**
SP-online	14.6 ± 0.7	-	20	8.9 ± 0.5	69	55	12.1 ± 0.7	-	40	1.2 ± 0.06	22	**100**
MLE-online	8.2 ± 0.6	90	65	4.9 ± 0.6	70	60	1.7 ± 0.1	39	90	0.6 ± 0.03	24	**100**
MLE-batch	8.6 ± 0.6	**47**	50	2.2 ± 0.2	50	**80**	1.4 ± 0.1	25	**100**	0.5 ± 0.02	20	**100**

Table 2: PC Configuration and Prize-Collecting TSP (10 stops) for changing query selection (columns) and weight update (rows).
#Q ($\downarrow$) is the number of queries to reach 10% regret, % DM ($\uparrow$) is the percentage of satisfied DMs. For regret ($\downarrow$), standard errors are given.

	Regret (%)	#Q	% DM	PoolGen (s)
Variance, **r**-pool	8.0 ± 0.5	78	45	0.3
Mean, **r**-pool	2.7 ± 0.2	55	60	0.3
UCB, **r**-pool	2.2 ± 0.2	50	80	0.3
UCB, **f**-pool	1.9 ± 0.2	54	85	9.5

Table 3: PC Configuration, varying acquisition functions, MLE-batch weight update. PoolGen is upfront time for pool generation.

	Regret (%)	% DM	Train (s)
Choice Perc, r-pool	2.9 ± 0.30	85	0.44
UCB, r-pool	$\mathbf{1.4 \pm 0.06}$	**95**	0.93

Table 5: PC-TSP with 20 nodes. Average training time per iteration.

of synthesized solutions also dropped, with a regret of 13.5% (vs 0.5%) and only 40% (vs 100%) of DMs were satisfied.

Feasible vs relaxed solutions. We compare training on feasible solutions or generating relaxed solutions. On the configuration task, all the feasible solutions ($\sim 60,000$) can be enumerated and the results are shown in Table 3. Generating the feasible pool takes about 30 times longer than randomly generating a pool of $10,000$ relaxed solutions. Training on a pool of relaxed solutions (**r**-pool) instead of feasible ones (**f**-pool) only marginally degrades metrics. Therefore, a random pool offers a much better efficiency/quality trade-off.

5.5 Comparison to State-of-the-art

We compare our complete proposed method, combining our query selection (UCB, r-pool) with the MLE-batch weight update, to previous CPE baselines, SetMargin [Teso *et al.*, 2016] and Choice Perceptron [Dragone *et al.*, 2018b]. On the configuration task (Table 4), our method outperforms both baselines in terms of quality of synthesized solutions while training an order of magnitude faster.

On the PC-TSP with 10 nodes (Table 2), SetMargin is not applicable due to numerical sub-objectives. Our method outperforms the Choice Perceptron (with its native PP-online weight update) by reaching twice lower regret, requiring half the queries and satisfying 100% of DMs (vs 75%). We also want to test the approaches on larger instances; however query selection time becomes prohibitive for Choice Perceptron. To enable a comparison, we run both our method and the Choice Perceptron on our randomized pool of solutions

	Regret (%)	#Q	% DM	Train (s)
SetMargin	4.9 ± 0.7	62	75	348
Choice Perceptron	6.6 ± 0.5	51	55	64
UCB, r-pool (ours)	$\mathbf{2.2 \pm 0.2}$	**50**	**80**	**4**

Table 4: Our method and the baselines on the configuration task.

for PC-TSP with 20 nodes in Table 5. Training on the precomputed pool is fast and leads to high-quality solution with both methods. Our UCB-based acquisition function with the MLE-batch weight update results in half the regret and 10% more satisfied DMs than (relaxed) Choice Perceptron. Training time per iteration is longer due to batch training, but remains under one second.

6 Conclusion

We revisited CPE to learn preferences over the sub-objectives of a multi-objective CO problem. We targeted desirable properties for preference elicitation systems: real-time, robust to noisy answers and a minimal number of queries while synthesizing high-quality solutions. Previous solutions repeatedly called a CO solver during query selection; we instead select from a pool of relaxed solutions, ensuring *real-time* query selection even for NP-hard problems, which in turn enabled us to use ensemble-based activation functions. We also proposed the smoothed MLE weight update, and a batch retraining strategy. In the experiments, our method synthesized higher-quality solutions, both in terms of regret and DM satisfaction, and required fewer queries than the baselines.

Multiple avenues for further methodological developments are possible. First, we sampled solution pools using problem-specific relaxations. Automating the relaxation for any CO problem would broaden the applicability of this approach; as would techniques for sampling feasible solutions. The challenge there is to obtain a diverse and unbiased sampling in an efficient way [Pesant *et al.*, 2022]. We ran all experiments for 100 iterations as in previous work; developing an instance-dependent stopping criterion would be desirable as it could restrict the number of needed DM interactions or continue to obtain even better performance. An acquisition function defined over pairs directly could potentially further boost efficiency. And while we used simple linear models, the weight updates can be used in gradient descent for arbitrary neural networks; meaning DM-specific features could also be taken into account. Finally, by improving on the desirable properties, we hope our method opens the door to real-life testing on rich multi-objective optimization problems.

Acknowledgments

References

[Aneja and Nair, 1978] Yash P Aneja and Kunhiraman PK Nair. The constrained shortest path problem. *Naval Research Logistics Quarterly*, 25(3):549–555, 1978.

[Archetti and Candelieri, 2019] Francesco Archetti and Antonio Candelieri. *Bayesian optimization and data science*, volume 849. Springer, 2019.

[Balcan *et al.*, 2010] Maria-Florina Balcan, Steve Hanneke, and Jennifer Wortman Vaughan. The true sample complexity of active learning. *Machine learning*, 80:111–139, 2010.

[Benabbou and Perny, 2018] Nawal Benabbou and Patrice Perny. Interactive resolution of multiobjective combinatorial optimization problems by incremental elicitation of criteria weights. *EURO journal on decision processes*, 6(3):283–319, 2018.

[Bourdache and Perny, 2019] Nadjet Bourdache and Patrice Perny. Active preference learning based on generalized gini functions: Application to the multiagent knapsack problem. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7741–7748, 2019.

[Bourdache *et al.*, 2020] Nadjet Bourdache, Patrice Perny, and Olivier Spanjaard. Bayesian preference elicitation for multiobjective combinatorial optimization. *arXiv preprint arXiv:2007.14778*, 2020.

[Boutilier, 2013] Craig Boutilier. Computational decision support: Regret-based models for optimization and preference elicitation, 2013.

[Bradley and Terry, 1952] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[Braziunas and Boutilier, 2007] Darius Braziunas and Craig Boutilier. Minimax regret based elicitation of generalized additive utilities. In *UAI*, volume 7, pages 25–32, 2007.

[Chan *et al.*, 2023] Timothy CY Chan, Rafid Mahmood, and Ian Yihang Zhu. Inverse optimization: Theory and applications. *Operations Research*, 2023.

[Christiano *et al.*, 2017] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[Collins, 2002] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP 2002)*, pages 1–8, 2002.

[Conitzer, 2007] Vincent Conitzer. Eliciting single-peaked preferences using comparison queries. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, 2007.

[Cox and John, 1992] Dennis D Cox and Susan John. A statistical method for global optimization. In *[Proceedings] 1992 IEEE international conference on systems, man, and cybernetics*, pages 1241–1246. IEEE, 1992.

[Dragone *et al.*, 2018a] Paolo Dragone, Stefano Teso, and Andrea Passerini. Constructive preference elicitation. *Frontiers in Robotics and AI*, 4:71, 2018.

[Dragone *et al.*, 2018b] Paolo Dragone, Stefano Teso, and Andrea Passerini. Constructive preference elicitation over hybrid combinatorial spaces. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Guns, 2019] Tias Guns. Increasing modeling language convenience with a universal n-dimensional array, cppy as python-embedded example. In *Proceedings of the 18th workshop on Constraint Modelling and Reformulation at CP (Modref 2019)*, volume 19, 2019.

[Guo and Sanner, 2010] Shengbo Guo and Scott Sanner. Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 289–296. JMLR Workshop and Conference Proceedings, 2010.

[Guo *et al.*, 2021] Mengzhuo Guo, Qingpeng Zhang, Xiuwu Liao, Frank Youhua Chen, and Daniel Dajun Zeng. A hybrid machine learning framework for analyzing human decision-making through learning preferences. *Omega*, 101:102263, 2021.

[Gurobi Optimization, LLC, 2024] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[Halffmann *et al.*, 2022] Pascal Halffmann, Luca E Schäfer, Kerstin Dächert, Kathrin Klamroth, and Stefan Ruzika. Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. *Journal of Multi-Criteria Decision Analysis*, 29(5-6):341–363, 2022.

[Herin *et al.*, 2024] Margot Herin, Patrice Perny, and Nataliya Sokolovska. Online learning of capacity-based preference models. In *International Joint Conference on Artificial Intelligence (IJCAI) 2024*, 2024.

[Keeney, 1993] Ralph L Keeney. *Decisions with multiple objectives: Preferences and value tradeoffs*. Cambridge university press, 1993.

[Mandi *et al.*, 2024] Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, , and Ferdinando Fioretto. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 80:1623–1701, 2024.

[Marta *et al.*, 2023] Daniel Marta, Simon Holk, Christian Pek, Jana Tumova, and Iolanda Leite. Variquery: Vae segment-based active learning for query selection

in preference-based reinforcement learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7878–7885. IEEE, 2023.

[Martyn and Kadziński, 2023] Krzysztof Martyn and Miłosz Kadziński. Deep preference learning for multiple criteria decision analysis. *European Journal of Operational Research*, 305(2):781–805, 2023.

[Mulamba *et al.*, 2021] Maxime Mulamba, Jayanta Mandi, Michelangelo Diligenti, Michele Lombardi, Victor Bucarey, Tias Guns, et al. Contrastive losses and solution caching for predict-and-optimize. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 2833–2840. ijcai. org, 2021.

[Oord *et al.*, 2018] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[Pandi *et al.*, 2022] Amir Pandi, Christoph Diehl, Ali Yazdizadeh Kharrazi, Scott A Scholz, Elizaveta Bobkova, Léon Faure, Maren Nattermann, David Adam, Nils Chapin, Yeganeh Foroughijabbari, et al. A versatile active learning workflow for optimization of genetic and metabolic networks. *Nature Communications*, 13(1):3876, 2022.

[Pesant *et al.*, 2022] Gilles Pesant, Claude-Guy Quimper, and Hélène Verhaeghe. Practically uniform solution sampling in constraint programming. In Pierre Schaus, editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 335–344, Cham, 2022. Springer International Publishing.

[Pigozzi *et al.*, 2016] Gabriella Pigozzi, Alexis Tsoukias, and Paolo Viappiani. Preferences in artificial intelligence. *Annals of Mathematics and Artificial Intelligence*, 77:361–401, 2016.

[Rafailov *et al.*, 2024] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[Sadana *et al.*, 2024] Utsav Sadana, Abhilash Chenreddy, Erick Delage, Alexandre Forel, Emma Frejinger, and Thibaut Vidal. A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, 2024.

[Seung *et al.*, 1992] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[Shivaswamy and Joachims, 2015] Pannaga Shivaswamy and Thorsten Joachims. Coactive learning. *Journal of Artificial Intelligence Research*, 53:1–40, 2015.

[Srinivas *et al.*, 2010] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1015–1022. Omnipress, 2010.

[Teso *et al.*, 2016] Stefano Teso, Andrea Passerini, and Paolo Viappiani. Constructive preference elicitation by setwise max-margin learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2067–2073, 2016.

[Toffano *et al.*, 2022] Federico Toffano, Michele Garraffa, Yiqing Lin, Steven Prestwich, Helmut Simonis, and Nic Wilson. A multi-objective supplier selection framework based on user-preferences. *Annals of Operations Research*, pages 1–32, 2022.

[Toubia *et al.*, 2004] Olivier Toubia, John R Hauser, and Duncan I Simester. Polyhedral methods for adaptive choice-based conjoint analysis. *Journal of Marketing Research*, 41(1):116–131, 2004.

[Véjar *et al.*, 2024] Bastián Véjar, Gaël Aglin, Ali İrfan Mahmutoğulları, Siegfried Nijssen, Pierre Schaus, and Tias Guns. An efficient structured perceptron for np-hard combinatorial optimization problems. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 253–262. Springer, 2024.

[Vernerey *et al.*, 2024] Charles Vernerey, Noureddine Aribi, Samir Loudni, Yahia Lebbah, and Nassim Belmecheri. Learning to rank based on choquet integral: Application to association rules. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 313–326. Springer, 2024.