# Explaining Group Recommendations via Counterfactuals

MARIA STRATIGI, Data Science Research Centre, Tampere University, Finland
NIKOS BIKAKIS, Hellenic Mediterranean University & Archimedes/Athena RC, Greece
KOSTAS STEFANIDIS, Data Science Research Centre, Tampere University, Finland

Group recommender systems help users make collective choices but often lack transparency, leaving group members uncertain about why items are suggested. Existing explanation methods focus on individuals, offering limited support for groups where multiple preferences interact. In this paper, we propose a framework for group counterfactual explanations, which reveal how removing specific past interactions would change a group recommendation. We formalize this concept, introduce utility and fairness measures tailored to groups, and design heuristic algorithms, such as Pareto-based filtering and grow-and-prune strategies, for efficient explanation discovery. Experiments on MovieLens and Amazon datasets show clear trade-offs: low-cost methods produce larger, less fair explanations, while other approaches yield concise and balanced results at higher cost. Furthermore, the Pareto-filtering heuristic demonstrates significant efficiency improvements in sparse settings.

CCS Concepts: • **Information systems** → **Recommender systems**.

Additional Key Words and Phrases: Explainable Recommender Systems, Explainable AI, Group Recommendations, Recommendation Fairness

## 1 Introduction

Recommender systems have become indispensable in a wide range of online platforms, from e-commerce and streaming services to tourism and education. While most research in recommender systems has traditionally focused on individuals, many real-world decision-making processes involve groups of users, e.g., families choosing a movie, friends selecting a restaurant, or colleagues planning a trip. Group recommender systems aim to support such scenarios by aggregating individual preferences into a joint recommendation. Despite their growing importance, they often remain opaque: users typically see the final output but have little insight into why certain items were recommended to the group. This lack of transparency undermines user trust and limits acceptance, particularly in settings where group members need to justify or negotiate decisions.

Explainability has thus emerged as a critical requirement for recommender systems [44]. Existing research on explainable AI has explored various forms of explanations, including feature-based justifications [27], rule-based reasoning [45], and, more recently, counterfactual explanations [38]. Counterfactuals answer "what if" questions by identifying the changes needed in the input that would alter a recommendation outcome. For instance, a system may explain the appearance of a movie in a user's list by stating: "If you had not rated item X, this recommendation would not appear." Counterfactual explanations are actionable, model-agnostic, and easily interpretable, making them especially suitable for increasing transparency in black-box systems. However, prior work has almost exclusively focused on individual users [13, 18], leaving group settings largely unexplored.

Explaining group recommendations introduces unique challenges. Unlike individuals, groups combine multiple preference profiles, often leading to complex dynamics of influence and compromise. A counterfactual explanation must therefore account not only for the system's internal logic but also for how group members' interactions collectively shape the outcome. Moreover, group explanations raise additional concerns of fairness: if an explanation suggests removing only one

member's interactions, that user may perceive the system as biased or exclusionary. Balancing interpretability, utility, and fairness is thus essential for explanations to be both informative and acceptable in group contexts [43].

One major advantage of counterfactual explanations in group recommendations is that they are constructed from data items with which the group has previously interacted. This ensures that explanations remain grounded in the system's internal knowledge and do not rely on external information, thereby preserving user trust. However, this very property also introduces a significant drawback.

The drawback arises from the process of generating a counterfactual explanation. To verify that a set of items constitutes a valid counterfactual explanation for a target item, the system must simulate the removal of these items from the group's interaction history, re-run the group recommender, and check whether the target item disappears from the new recommendation list. If it does, the removed items form a counterfactual explanation. Since a group typically contains multiple members, each with numerous interactions, the space of possible candidate items becomes extremely large. This challenge is further compounded by the need to consider not just individual items but also all possible combinations. As a result, exhaustive search becomes prohibitively costly.

To address this limitation, we propose heuristic methods designed to reduce the search space while also mitigating potential unfairness in the resulting explanations. To guide these heuristics, we introduce a set of metrics that capture different dimensions of item relevance: how extensively the group has interacted with an item, its popularity in the overall system, its influence on removing the target item from the recommendation list, and the group members' preferences for the target item. We also consider efficiency by measuring the number of times the group recommender must be invoked before a counterfactual explanation is identified.

Based on these considerations, we propose five heuristic methods for generating counterfactual explanations. First, the *Pareto-filtering* heuristic uses the Pareto notion to initially reduce the search space, requiring only a limited number of recommender calls. *GreedyGrow* sequentially adds items until a counterfactual is obtained. *ExpRebuild* and *Grow&Prune* refine *GreedyGrow*. In particular, *ExpRebuild* orders items according to their ability to explain the target item and then incrementally reconstructs the counterfactual. By contrast, *Grow&Prune* starts from an initial counterfactual and removes items step by step until a reduced-size explanation is reached. Finally, *FixedWindow* applies a sliding window over the candidate list to identify a subset containing a counterfactual explanation and then exhaustively searches within this restricted window to extract a more compact explanation. Together, these heuristics provide a spectrum of trade-offs between efficiency, minimality, and fairness, which we evaluate in our experiments.

Overall, in this paper, we introduce the first systematic framework for explaining group recommendations via counterfactuals. The main contributions are the following:

- **Modeling:** We formalize the concept of counterfactual explanations in group recommender systems, defining item-level metrics that capture recognition, rating, and influence both within and beyond the group.
- **Evaluation dimensions:** We introduce a set of metrics for assessing counterfactual explanations in groups, including minimality, interpretability, cost-efficiency, utility, and fairness.
- **Algorithms:** We propose a family of heuristic algorithms for efficiently discovering group counterfactual explanations.
- **Experiments:** We evaluate the algorithms across two real-world datasets (MovieLens and Amazon). Our results highlight the trade-offs between explanation size, computational cost, and fairness, and demonstrate the effectiveness of Pareto-based filtering in reducing search complexity.

A preliminary version of this work appears in [39]. In addition to refining the *FixedWindow* method, we now introduce a more robust set of metrics and three additional heuristic algorithms. We have also conducted experiments using a larger MovieLens dataset and evaluated the proposed heuristics on another dataset, Amazon.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces the item-level metrics, and Section 4 extends this with utility metrics defined at the explanation level. Next, Section 5 presents the proposed algorithms together with their pseudocode. Section 6 outlines the experimental setup and discusses the evaluation results. Finally, Section 7 concludes the paper.

## 2 Related Work

Recommender systems are a cornerstone of modern information access, assisting users in navigating large collections of items in domains such as e-commerce, media streaming, and online services. Early approaches were dominated by collaborative filtering and content-based techniques [1], which later evolved into hybrid and deep learning models to improve both scalability and accuracy. Beyond predictive performance, recent research has increasingly emphasized additional dimensions such as fairness, diversity, and explainability [26, 33, 55]. We next review existing work on group recommendations and counterfactual explanations.

### 2.1 Group Recommendations

Group recommendation has been widely studied in the literature [28]. A common strategy is to first apply a standard single-user recommendation model to each group member and then aggregate the resulting individual recommendation lists into a single group list, e.g., [2, 4, 32, 40].

Different approaches have been proposed for the aggregation stage. For instance, [54] considers the varying influence of individual group members when merging recommendations, while [8] employs attention networks and neural collaborative filtering to learn aggregation strategies directly from data. Along the same lines, [53] combines attention mechanisms with bipartite graph embeddings to capture member influence, whereas [37] leverages social networks enriched with user preferences and social interactions to determine group-level recommendations.

Group members interactions have also been explicitly modeled. For example, [46] simulates consensus-building as a series of voting processes and introduces a stacked social self-attention network to learn the underlying voting dynamics. To handle large groups, [34] proposes dividing them into subgroups with shared interests, identifying candidate media-user pairs for each subgroup, and then aggregating the resulting collaborative filtering lists. Another strategy is presented in [21], which introduces a two-phase recommender: the first phase seeks to maximize satisfaction at the group level, and the second phase refines recommendations to better satisfy individual preferences.

Utility-based approaches have also been explored. In [51], each member is assigned a utility score based on the relevance of items, and the system balances these utilities to generate a final group list. Similarly, [36] defines user utility as the similarity between individual and group-level recommendations and considers sets of Pareto-optimal items when constructing the list. Finally, [20] introduces the notion of rank-sensitive balance, where not only the top recommendation but also each successive prefix of the list must reflect a fair balance of member interests.

### 2.2 Counterfactual Explanations

Counterfactual explanations have emerged as a powerful approach to enhance the transparency and reliability of recommender systems. They clarify why certain items are recommended by identifying minimal changes in the input data that would alter the outcome. This form of explanation is intuitive, actionable, and model-agnostic, making it particularly valuable in contexts where

user trust and accountability are essential. While counterfactuals have primarily been studied in individual recommendation scenarios, their potential is especially relevant to group settings, where understanding how different members' preferences shape the joint outcome is critical for transparency and acceptance.

Recent research has explored counterfactual explanations across diverse recommendation tasks. For instance, CAVIAR [18] modifies image embeddings to reveal the influence of visual features on recommendations, while [50] introduces CFairER, a fairness-aware counterfactual framework addressing demographic biases such as age and gender. Graph-based methods have also been proposed, including GNNUERS [29], which explains fairness issues in GNN-based recommendation models. Beyond recommended items,[41] studies "why-not" questions in collaborative filtering, providing explanations for items that were absent from recommendations; an idea later extended to graph-based systems in [3]. Temporal dynamics have also been incorporated: the CETD method [16] generates counterfactual explanations for sequential recommendation by considering evolving user behavior.

Other works focus on general frameworks and model-agnostic solutions. A study in [42] proposes a framework for producing explainable counterfactual recommendations by showing how small changes lead to different results, while [52] presents a learning-based approach that adapts explanations to user interaction histories. MACER [49] leverages reinforcement learning to provide model-agnostic, item-based counterfactual explanations, particularly suitable in settings with multiple aggregation models, such as group recommendations. LXR [5] offers a post-hoc, self-supervised approach to identify critical user interactions for a recommendation, and [50] studies attribute-level counterfactuals in heterogeneous information networks, focusing on item exposure disparities and fairness.

A recent line of research has begun to explicitly consider group counterfactuals in relation to fairness. For example, CounterFair [24] proposes a framework that generates group counterfactuals for bias detection, fairness-aware recourse, and subgroup identification. By balancing recourse across sensitive groups, CounterFair demonstrates how counterfactual reasoning can serve not only as an explanatory tool but also as a means to mitigate algorithmic bias. This direction complements our focus by highlighting the potential of counterfactuals at the group level, though its emphasis lies in bias detection and subgroup discovery rather than in explaining group recommendations.

Together, these contributions have laid important foundations for counterfactual reasoning in recommender systems. However, most of this research remains centered on individual recommendations. In contrast, group recommendations introduce additional complexity, as they must reconcile diverse user preferences and balance fairness across members. To the best of our knowledge, counterfactual explanations tailored to group settings have not been systematically explored. Addressing this gap, we investigate how counterfactuals can be adapted to group recommendations, providing explanations that foster both trust and satisfaction among all group members.

## 3   Group Counterfactual Explanations Model

To enhance the transparency and interpretability of group recommendation systems, we introduce a model for generating counterfactual explanations tailored to groups of users. These explanations provide actionable insights by identifying which items would lead to different recommendation outcomes if removed from the group's interaction history. In this section, we present the foundational concepts of our model, formalize the definition of group counterfactual explanations, and introduce relevant item-level metrics to support the explanation generation process. Table 1 summarizes the notation used throughout the paper.

Table 1. Summary of Common Notations

| Notation | Description |
|---|---|
| **Users and Items** | |
| $U, u$ | Set of all users, a user |
| $G$ | A group of users |
| $I, i$ | Set of all items, an item |
| $t$ | Target item, i.e., the item to be explained |
| $I_u$ | Items with which user $u$ has interacted |
| $I_G$ | Items with which any member of $G$ has interacted |
| $\rho_{i,u}$ | Rating assigned by user $u$ to item $i$ |
| **Recommendations** | |
| $\pi(I)$ | Group recommendation system given item set $I$ |
| $L_I$ | Group ranked recommendation list produced from $I$ |
| $\text{rank}(i, L_I)$ | Rank position of item $i$ in list $L_I$ |
| **Explanations and Metrics** | |
| $E$ | Counterfactual explanation |
| $\text{rc}(i, S)$ | Average recognition of item $i$ among users $S$ |
| $\text{rt}(i, S)$ | Average rating of item $i$ by users $S$ |
| $\text{expwr}(P, t, I)$ | Explanatory power of item subset $P$ w.r.t. target item $t$ and item set $I$ |
| $\text{infl}(i, t, I, G)$ | Influence of item $i$ on target item $t$ for group $G$ |
| $i.score$ | Total score of item $i$ |
| $\text{fair}(G, E)$ | Fairness of explanation $E$ with respect to group $G$ |
| $\min(E)$ | Minimality of explanation $E$ |
| $\text{interpr}(E)$ | Interpretability of explanation $E$ |
| $\text{cost}(E)$ | Computational cost of explanation $E$ |
| $\Omega(E)$ | Utility of explanation $E$ |

## 3.1 Basic Concepts

We begin by defining the core entities involved in our model, namely users, groups, their interactions with items, and how these interactions shape group recommendations.

**User, Group & Interacted Items.** Let $U$ denote the set of all *users* in the system. For each user $u \in U$, let $I_u$ represent the set of *items that the user has interacted with*, such as items they have ordered, rated, clicked, or liked. Let $G \subseteq U$ be a group of users, and let $I_G$ denote the set of *items that any group member has interacted with*, formally defined as $I_G = \bigcup_{\forall u \in G} I_u$. Furthermore, $\rho_{i,u} \in [0, 1]$ represents the *rating assigned by user $u$ to item $i$*. If a rating is not present, it is assumed to be zero. It is important to note that our approach does not require the presence of ratings for all users or items.

**Recommendations.** Let $\pi(I)$ denote a *group recommendation system*, which, given a set of interacted items $I$, produces an ordered *recommendation list* $L_I$, i.e., $\pi(I) = L_I$. For a given recommendation list $L_I$, let $\text{rank}(i, L_I) \in [1, |L_I|]$ denote the *position (rank) of item $i$ in $L_I$*. Generating a

recommendation list $L_I$ is associated with a *recommender call cost* $\phi$, such as system latency or API usage cost.

In this work, we treat the group recommender system as a *black box*, enabling general applicability of our approach regardless of the internal logic or type of recommendation algorithm employed.

## 3.2 Group Counterfactual Explanation

Having defined the basic entities and the recommendation mechanism, we now formalize the concept of counterfactual explanations in group settings. Inspired by prior work on counterfactuals in individual recommendations [19], we extend the concept to group scenarios, where explanations must account for collective interactions rather than those of a single user.

The essence of a counterfactual explanation lies in identifying the set of previously interacted items that, if removed, would prevent the appearance of a specific item (the *target item*) in the recommendation list. The problem can be described through the following question and its explanation.

> **Question**
> "*Why does the target item $t$ appear in the recommendation list*?"
>
> **Explanation**
> "*If you had not interacted with the items $X$, item $t$ would not have appeared in the recommendation list.*"

**Group Counterfactuals Explanation.** Formally, a *group counterfactuals explanation* is defined as follows. Given a group $G \subseteq U$ and a target item $t \in L_{I_G}$, a *group counterfactual explanation* $E \subset I_G$ is a set of interacted items such that if the group had not interacted with these items, the item $t$ would not have been recommended[1].

Considering the above, the follow holds. Let $\pi(I_G) = L_{I_G}$ with $t \in L_{I_G}$. Then, a set of items $E \subset I_G$ is a *counterfactual explanation* if $\pi(I_G \backslash E) = L_{I'_G}$ and $t \notin L_{I'_G}$.

**Factual and Counterfactual Scenario.** In this setting, the *factual scenario* is defined by the current interaction set $I_G$, where $\pi(I_G) = L_{I_G}$ and $t \in L_{I_G}$. The *counterfactual scenario* involves a modified interaction set $I'_G = I_G \backslash E$, such that $\pi(I'_G) = L_{I'_G}$ and $t \notin L_{I'_G}$. The difference $E = I_G \backslash I'_G$ thus constitutes the counterfactual explanation. This approach inherently assumes that the only *feasible action* is the *removal of items* from the group's interaction history.

**Explanation Process Example.** Figure 1 illustrates the overall explanation process. Suppose the recommendation list is $L_G$ and the *target item* is $i_9$, i.e., the item for which we aim to generate an explanation. The group consists of *three users*: $u_1$, $u_2$, and $u_3$, each associated with their respective *interaction lists*, denoted as $I_{u_1}$, $I_{u_2}$, and $I_{u_3}$. ① Initially, we remove a selected set of items from the users' interaction lists, i.e., $i_2$, $i_5$, and $i_8$. This set represents a candidate counterfactual explanation. ② Next, we input the modified interaction lists $I'_{u_1}$, $I'_{u_2}$, and $I'_{u_3}$ into the recommendation system. ③ The system returns a new group recommendation list $L'_G$. ④ Since the target item $i_9$ no longer appears in $L'_G$, the removed items $i_2$, $i_5$, and $i_8$ constitute a counterfactual explanation.

---

[1]For the remainder of this paper, *Counterfactual explanation* will be referred to interchangeably as *Counterfactual* or *Explanation*.
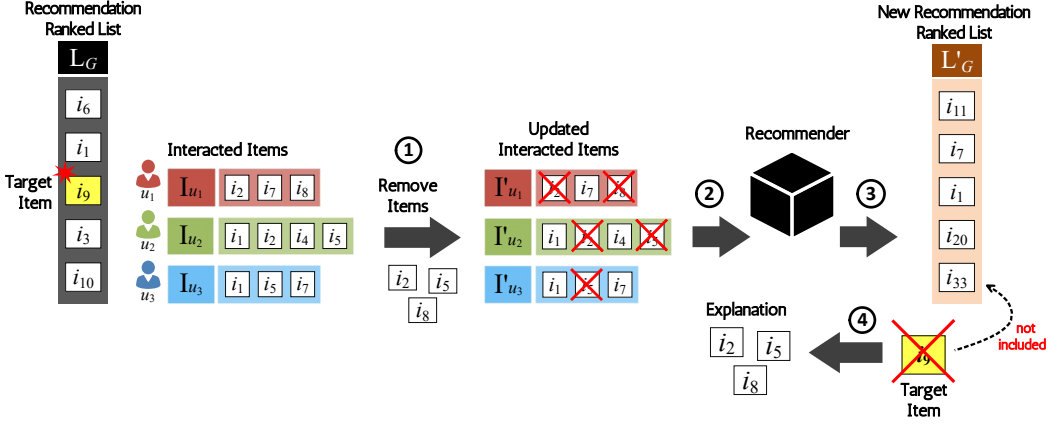
Fig. 1. Explanation Process: Finding Counterfactual Explanations

## 3.3 Item Metrics

To support the generation and evaluation of counterfactual explanations, we define a set of item-level metrics that quantify various aspects of item influence and group interaction patterns. These metrics guide heuristic algorithms in selecting candidate items for removal.

**Item Recognition.** We begin by defining how widely the users know or recognize an item. The *item recognition* metric captures the fraction of users in a given set who have interacted with the item.

Let $i$ be an item and $S \subseteq U$ a set of users. Then, item recognition $\mathrm{rc}(i, S) \in [0, 1]$ is defined as:

$$\mathrm{rc}(i, S) = \frac{\sum\limits_{\forall u \in S} \zeta(u, i)}{|S|} \tag{1}$$

where $\zeta(u, i) = 1$ if user $u$ has interacted with item $i$ (i.e., $i \in I_u$), and 0 otherwise.

We evaluate this item recognition in two contexts: (1) $\mathrm{rc}(i, G)$: among the group members (*group item recognition*); and (2) $\mathrm{rc}(i, U \backslash G)$: among users outside the group (*public item recognition*).

**Item Rating.** Next, we consider the average quality assessment of an item based on user feedback. Given an item $i$ and a user set $S \subseteq U$, the *item rating* is the average of ratings provided by $S$ users:

$$\mathrm{rt}(i, S) = \frac{\sum\limits_{\forall u \in S} \rho_{i,u}}{|S|} \tag{2}$$

Similar to recognition, item rating is computed in two contexts: (1) $\mathrm{rt}(i, G)$: among the group members (*group item rating*); and (2) $\mathrm{rt}(i, U \backslash G)$: among all users excluding the group (*public item rating*).

**Item Influence on the Target Item.** The *influence of an item on the target item* quantifies the extent to which a previously interacted item contributes to the recommendation score assigned to the target item $t$ by the recommender system, in a *model-agnostic* setting.

Intuitively, interacted items that lead to a higher recommendation score for $t$, resulting in $t$ being ranked higher, are considered to exert a stronger influence. These items are therefore more likely to appear in counterfactual explanations, as their removal can significantly alter the ranking of $t$.

To estimate this influence in a model-agnostic way, we treat each user in the group individually and evaluate the effect of their interactions on the recommendation score of $t$. Specifically, for each user $u \in G$, we call the group recommender using only their interaction history $I_u$ and retrieve the recommendation score for $t$.

Formally, let $G' = \{u \in G \mid i \in I_u\}$ be the set of group members who have interacted with item $i$. Then, the *influence* $\mathsf{infl}(i, t, I, G) \in [0, 1]$ of item $i$ on target item $t$ for group $G$ is computed as the average recommendation score of $t$ across $G'$ users:

$$\mathsf{infl}(i, t, I, G) = \sum_{\substack{\forall u \in G' \\ I_u \subseteq I}} \mathsf{recScore}(t, I_u) \Big/ |G'| \tag{3}$$

Here, $\mathsf{recScore}(t, I_u) \in [0, 1]$ denotes the recommendation score assigned to $t$ by the recommender system when invoked with only user $u$ interacted items $I_u$. This approach allows us to estimate item influence without requiring access to the internal workings of the recommender (i.e., in a *model-agnostic* way).

**Item Set Explanatory Power.** We introduce the *explanatory power* metric, which quantifies the potential of an item set to contribute to a counterfactual explanation and, consequently, its ability to explain the presence of a target item. The metric is computed in a *model-agnostic* manner, relying solely on observable changes in the recommendation output.

Let $I$ be the interacted items set, $L_I$ the resulting recommendation list with $|L_I| = m$, and $t$ the target item. Let $P \subset I$ be a subset of items considered for removal, and let $\pi(I \backslash P)$ be the new recommendation list.

The *explanatory power* of item set $P$, denoted $\mathsf{expwr}(P, t, I) \in [0, 1]$, is defined as:

$$\mathsf{expwr}(P, t, I) = \min \left\{ \frac{\mathsf{rank}(t, \pi(I \backslash P)) - 1}{m}, 1 \right\} \tag{4}$$

Explanatory power quantify the rank of $t$ when the item set $P$ is removed from interacted item set $I$. Higher values indicate that the items in $P$ are more likely to be part of an explanation.

Particularly, if $t$ is eliminated from the list ($t \notin \pi(I \backslash P)$), then $P$ forms a counterfactual and its explanatory power is set to one. On the other hand, if $P$ results in item $t$ being ranked top (i.e., top-1), explanatory power takes its lowest value of zero.

**Item Total Score.** While individual metrics (e.g., recognition, rating, influence) offer valuable insights, combining them into a single score enables more effective ranking and selection of items for explanation. To this end, we define the *item total score* as an aggregate of multiple item-based metrics.

Given an item $i$ and a collection of $k$ metrics $(f_1, f_2, \ldots, f_k)$, we compute the total score as:

$$i.score = \sum_{j=1}^{k} f_j(i) \tag{5}$$

In this work, we use the following five metrics to calculate the total score:

- $\mathsf{rc}(i, G)$: group item recognition
- $\mathsf{rc}(i, U \backslash G)$: public item recognition

- $\mathrm{rt}(i, G)$: group item rating
- $\mathrm{rt}(i, U \backslash G)$: public item rating
- $\mathrm{infl}(i, t, I, G)$: item influence on the target item

Putting it all together, the final formula for the *item total score* is:

$$i.score = \mathrm{rc}(i, G) + \mathrm{rc}(i, U \backslash G) + \mathrm{rt}(i, G) + \mathrm{rt}(i, U \backslash G) + \mathrm{infl}(i, t, I, G) \tag{6}$$

This score serves as a unified measure of an item's importance, considering both its relevance within the group and its potential impact on the target recommendation outcome. It plays a central role in guiding the selection of "explanatory" items in our counterfactual framework.

## 4 Counterfactual Utility and Fairness

In this section, we present a set of evaluation dimensions for assessing the quality of counterfactual explanations in group recommender systems. Although prior research on counterfactual reasoning has largely centered on individual-level explanations, extending these approaches to group settings presents unique challenges and design considerations. Beyond generating explanations that are minimal and interpretable, we must also consider the computational cost of discovering them, the overall utility they offer to the group, and the fairness of how explanatory changes are distributed among group members.

We begin by formalizing utility-oriented metrics that capture properties such as explanation minimality, interpretability, and generation cost, and introduce a unified utility function that balances these aspects. We then define a notion of counterfactual fairness tailored to groups, ensuring that no single user is disproportionately affected by the changes recommended in the explanation. Collectively, these metrics enable a comprehensive assessment of group-level counterfactual explanations, emphasizing clarity, utility, and fairness.

### 4.1 Counterfactual Utility

In this section, we introduce a set of evaluation metrics to assess the quality and characteristics of group-level counterfactual explanations in recommender systems. These metrics capture different desirable properties of explanations, including clarity, interpretability, cost-efficiency, overall utility, and fairness.

**Counterfactual Explanation Minimality.** We use the notion of counterfactual *minimality* (also referred to as *sparsity*) to evaluate the *clarity* of an explanation; i.e., how easily it can be understood. Numerous studies in cognitive psychology and explainable AI have shown that explanations involving fewer changes are more understandable and cognitively efficient to process [12, 22, 30, 31, 35, 47, 48].

In our setting, the minimality of a counterfactual explanation $E$ is quantified by the number of items it contains. A smaller $|E|$ indicates a clearer explanation. This quantity corresponds to the number of user-item interactions removed to achieve the counterfactual.

We formally define the *counterfactual explanation minimality* as:

$$\min(E) = 1 - \frac{|E|}{|I_G|} \tag{7}$$

where $|I_G|$ is the total number of interacted items by the group. The value is normalized in $[0, 1]$ such that higher values indicate better minimality.

**Counterfactual Explanation Interpretability.** An explanation is more interpretable when the items it involves are familiar to users [9]. We measure interpretability through item recognition, i.e., how often an item appears in user interaction histories. This includes both (i) the recognition of an item within the group, and (ii) its recognition by the broader user base.

To quantify this, we define the interpretability of a counterfactual explanation $E$ as the average recognition of its items:

$$\text{interpr}(E) = \frac{1}{2|E|}\left(\sum_{\forall i \in E} \text{rc}(i, G) + \sum_{\forall i \in E} \text{rc}(i, U \setminus G)\right) \tag{8}$$

Here, $\text{rc}(i, G)$ denotes item $i$'s recognition in the group, and $\text{rc}(i, U \setminus G)$ represents its recognition among other users.

**Counterfactual Explanation Cost.** Producing a counterfactual explanation involves making multiple calls to the recommender system, each associated with a cost $\varphi$. The total cost of an explanation $E$ is thus defined as the number of calls made to discover it:

$$\text{cost}(E) = \sum_{i=1}^{n} \varphi \tag{9}$$

where $n$ is the number of recommender invocations required. For simplicity, we assume $\varphi = 1$ in our evaluation, which allows us to count the number of calls directly.

**Counterfactual Explanation Utility.** We define the *utility* of a counterfactual explanation $E$ as a combination of its clarity and interpretability. The utility function is defined as:

$$\Omega(E) = \psi\big(\min(E), \text{interpr}(E)\big) \tag{10}$$

where $\psi$ is an aggregation function that balances the two metrics, for instance via a weighted sum or product.

## Problem Definition
We formally define the *Group Counterfactual Explanation problem* as follows.

**Group Counterfactual Explanation Problem.** Given a *group $G$*; a *group interacted items $I_G$*; a *group recommended items list $L_{I_G}$*; a *target item $t$*; a *recommender call cost $\varphi$*; and a *budget $B$* in terms of recommender call cost; our goal is to find a *group counterfactual explanation $E^*$*, such that the *explanation utility* $\Omega(E^*)$ is maximized and the *counterfactual cost* $\text{cost}(E^*)$ is lower than the budget $B$.

$$E^* = \arg\max \Omega(E) \quad \text{s.t.} \quad \text{cost}(E) \leq B$$

**Computational Complexity.** Solving this problem involves evaluating all subsets of $I_G$, i.e., the power set excluding the empty set. Therefore, the number of candidates is $2^{|I_G|} - 1$. Since each candidate requires a recommender system call with cost $\varphi$, the computational complexity is $O(\varphi \cdot 2^{|I_G|})$.

## 4.2 Counterfactual Fairness
The fairness notion used in this work is inspired by *individual fairness* [7, 11, 25], which states that similar individuals should be treated similarly. We translate this to the group setting by requiring

Table 2. Summary of Time, Space and Recommender Calls Complexity

| Algorithm | Time Complexity | Space Complexity | Recommender Calls |
|---|---|---|---|
| **ParetoFiltering** (Sec. 5.1) | $O(|I||U|\phi + v \cdot (|I|\log^{d-2}|I| + \phi))$ | $O(|I|)$ | $O(|I||U| + v)$ |
| **FixedWindow** (Sec. 5.2) | $O(|I||U|\phi + |I|\log|I| + \frac{|I|^2}{w}\phi + 2^{|I|}\phi)$ | $O(|I|)$ | $O(|I||U| + \frac{|I|^2}{w} + 2^{|I|})$ |
| **GreedyGrow** (Sec. 5.3) | $O(|I||U|\phi + |I|\log|I|)$ | $O(|I|)$ | $O(|I||U|)$ |
| **Grow&Prune** (Sec. 5.4) | $O(|I||U|\phi + |I|\log|I|)$ | $O(|I|)$ | $O(|I||U|)$ |
| **ExpRebuild** (Sec. 5.5) | $O(|I||U|\phi + |I|\log|I|)$ | $O(|I|)$ | $O(|I||U|)$ |

that the burden of a counterfactual explanation (i.e., the items removed) be fairly distributed among group members.

Formally, we say that an explanation $E$ is fair if each group member contributes a similar number of interacted items to the explanation. For any user $u \in G$, let $\zeta(u, i)$ be an indicator function equal to 1 if user $u$ has interacted with item $i$, and 0 otherwise.

To assess fairness, we compute the standard deviation of the number of items from each user's profile that are included in $E$:

$$\text{fair}(G, E) = \frac{1}{\sigma\left(\sum_{\forall i \in E} \zeta(u_1, i), \ \sum_{\forall i \in E} \zeta(u_2, i), \ \ldots, \ \sum_{\forall i \in E} \zeta(u_{|G|}, i)\right)} \tag{11}$$

Lower standard deviation indicates more balanced participation, and therefore greater fairness. By taking the inverse, we ensure that higher values of $\text{fair}(G, E)$ correspond to fairer explanations. We define perfect fairness as the case where all users contribute equally (i.e., $\sigma = 0$).

## 5 Algorithmic Framework for Counterfactual Search

The search space for counterfactual explanations in group recommender systems is inherently combinatorial, making naive exploration intractable. To address this, we introduce an algorithmic framework that incrementally narrows the candidate space using metric-driven heuristics and Pareto-based filtering.

In the following sections we present five heuristic algorithms. The *Pareto-filtering* (Sec. 5.1) leverages Pareto dominance to prune the candidate item space early, requiring only a limited number of recommender calls. *FixedWindow* (Sec. 5.2) scans a sorted candidate list using a sliding window to locate a counterfactual, within which an exhaustive local search is performed to extract a more compact explanation. *GreedyGrow* (Sec. 5.3) follows a forward counterfactual construction strategy, progressively adding items until a counterfactual is found. Two methods build upon this procedure: *Grow&Prune* (Sec. 5.4), which starts from an initial counterfactual and iteratively eliminates items to reduce the explanation size; and *ExpRebuild* (Sec. 5.5), which ranks items by their ability to explain the target item and incrementally reconstructs the counterfactual.

Table 2 summarizes the worst-case time, space, and recommender call complexity of all proposed algorithms.

### 5.1 Search Space Reduction Using Pareto-based Filtering

To reduce the combinatorial search space, we first introduce a **ParetoFiltering** technique. The key goal of this method is to improve efficiency by pruning dominated items before counterfactual search, thereby reducing the number of recommender calls required. Hence, pareto filtering can be consider as a preprocessing step to effectively reduce the search space.

Specifically, we leverage the item metrics introduced in Section 4.1 to map each item into a multidimensional metric space, enabling dominance-based filtering to identify a "high-quality" counterfactual superset.

**Metric-based Representation.** Let $d$ denote the number of item-level metrics. Each item $i_k \in I_G$ is represented as a $d$-dimensional vector $o_k = (o_k^1, o_k^2, \ldots, o_k^d)$, where $o_k^j = f_j(i_k)$ is the value of metric $j$ for item $i_k$. The attribute space is defined over $\mathbb{R}^d$.

**Pareto Set.** An object $o_1$ *dominates* $o_2$, denoted $o_1 \succ o_2$, if: (1) $\forall w \in [1, d]$, $o_1^w \geq o_2^w$, and (2) $\exists j \in [1, d]$, $o_1^j > o_2^j$.
The *Pareto set*, or skyline [23], denoted $PS(O)$ for object set $O$, is defined as:

$$PS(O) = \{o_i \in O \mid \nexists o_k \in O : o_k \succ o_i\}.$$

Items in $PS(O)$ are referred to as Pareto-optimal items and represent a non-dominated frontier within the metric space.

**Threshold-based Generalization.** To flexibly control the cardinality of the result set, we generalize the Pareto set using a threshold vector $\tau = \langle \tau^1, \tau^2, \ldots, \tau^d \rangle$ where $\tau^j \in \mathbb{R}$ adjusts strictness per metric.
An object $o_1$ $\tau$-*dominates* $o_2$, written $o_1 \succ_\tau o_2$, if: (1) $\forall w \in [1, d]$, $o_1^w \geq o_2^w + \tau^w$, and (2) $\exists j \in [1, d]$, $o_1^j > o_2^j$. The $\tau$-*Pareto set* is then defined as:

$$PS_\tau(O) = \{o_i \in O \mid \nexists o_k \in O : o_k \succ_\tau o_i\}.$$

Setting $\tau = 0$ yields the standard Pareto set; $\tau < 0$ relaxes dominance criteria (superset), while $\tau > 0$ tightens them (subset). Efficient skyline algorithms like D&C [23], SFS [10], LESS [14], and SaLSa [6] can be adapted to compute $PS_\tau(O)$.

**Pareto-based Filtering Technique Overview.** To efficiently (i.e., using a small number of recommender calls) identify a counterfactual explanation set, we adopt an iterative filtering strategy based on $\tau$-Pareto dominance. Specifically, we remove from the group's interaction items all items belonging to the $\tau$-Pareto set, which represent a collection of "high-impact" items across the item-level metrics. The group recommender is then invoked to assess whether excluding these items causes the target item to disappear from the recommendation list. In such a case, the $\tau$-Pareto set is selected as the candidate counterfactual explanation.

In cases where the target item persists in the recommendation list, the examined $\tau$-Pareto set is not a counterfactual. To explore a broader set of items, we incrementally relax the dominance criteria by reducing the threshold vector $\tau$, and recompute the $\tau$-Pareto set. This threshold-based relaxation enables a controlled expansion of the candidate item space. The process is repeated until a $\tau$-Pareto set corresponds to a counterfactual explanation.

**Algorithm Description.** Algorithm 1 takes as input a interacted item set $I$ and a target item $t$, and returns a counterfactual explanation $CS$. Items are first mapped to $d$-dimensional metric vectors based on items metrics values (*lines 2–4*). The main loop (*lines 7–14*) iteratively relaxes the Pareto dominance criteria. At each iteration, a threshold vector $\tau$ is computed based on the standard deviation of each metric dimension (*line 9*), and the $\tau$-Pareto set is computed (*line 10*). The items that correspond to the $\tau$-Pareto set objects are inserted into the item set $PI$ (*line 11*).

---

**Algorithm 1.** ParetoFiltering ( $I$, $t$ )

---

**Input:** $I$: interacted items; $t$: target item
**Output:** $CS$: counterfactual explanation;
**Parameters:** $\pi$: group recommender; $f_1, f_2, \ldots, f_d$: item metrics functions
**Variables:** $i$: item; $O$: $d$-dimensional objects; $it$: iteration counter; $\tau$: attributes thresholds;
$\quad\quad\quad$ $PS$: $\tau$-pareto set; $PI$: items of $\tau$-pareto set; $cfFound$: boolean variable

1  $O \leftarrow \varnothing$
2  **foreach** $i \in I$ **do**                                                            //transform items to $d$-dimensional objects
3  $\quad$ $o_i \leftarrow (f_1(i), f_2(i), \ldots, f_d(i))$                                      //compute the $d$ metrics for item $i$
4  $\quad$ $O \leftarrow O \cup o_i$

5  $PI \leftarrow \varnothing$
6  $it \leftarrow 0$
7  **do**
8  $\quad$ **if** $PI = I$ **then break**                                                      // all items examined – counterfactual not exist
9  $\quad$ $\tau \leftarrow -\left\langle it \cdot \sigma(\bigcup_{\forall o_i \in O} o_i^1), it \cdot \sigma(\bigcup_{\forall o_i \in O} o_i^2), \ldots, it \cdot \sigma(\bigcup_{\forall o_i \in O} o_i^d) \right\rangle$  //compute attributes' thresholds values as attributes
$\quad\quad$ std – assign to $\tau$ the negative values
10 $\quad$ $PS \leftarrow$ find $\tau$-pareto set for objects $O$
11 $\quad$ $PI \leftarrow$ get the items of $I$ that correspond to $\tau$-pareto set $PS$ objects  //items set to be checked as counterfactual
12 $\quad$ $cfFound \leftarrow$ isCF$(I, PI, t)$                                                //call recommender system
13 $\quad$ $it \leftarrow it + 1$
14 **while** ($cfFound = $ false)

15 **if** $cfFound = $ true **then**
16 $\quad$ $CS \leftarrow PI$
17 **else**                                                                                 //counterfactual not exist
18 $\quad$ $CS \leftarrow \varnothing$

19 **return** $CS$

---

---

**Procedure 1:** isCF$(I, S, t)$

---

**Input:** $I$: interacted items; $S$: interacted items to be checked as counterfactual; $t$: target item;
**Output:** boolean value:(if $S$ is counterfactual)
**Parameters:** $\pi$: group recommender
**Variables:** $L$: recommendation list generated considering $I \backslash S$ as interacted

1  $L \leftarrow \pi(I \backslash S)$                                                          //call recommender system

2  **if** $t \notin L$ **then**
3  $\quad$ **return** true
4  **else**
5  $\quad$ **return** false

---

The procedure isCF (Proc. 1) is then invoked to determine if $PI$ corresponds to a counterfactual explanation (*line* 12). Threshold relaxation continues until a counterfactual expiation is found.

**Computational Analysis.** In Algorithm 1, *line* 3, considering the item metrics used in our implementation, the worst-case computational cost for computing the *total item score* (Eq. 6) is analyzed as follows.

The cost of computing the two item *recognition metrics*, $\text{rc}(i, G)$ and $\text{rc}(i, U \backslash G)$ (Eq. 1), is $O(max(|U|, |I|))$ per item (if hash-based indexes are used). For simplicity, let us assume that $|U| > |I|$, so the complexity is $O(|U|)$. The same holds for the *rating metrics*, $\text{rt}(i, G)$ and $\text{rt}(i, U \backslash G)$ (Eq. 2); both require $O(|U|)$ per item. In order to compute the *influence metric*, $\text{infl}(i, t, I, G)$ (Eq. 3), for each user in $G$, we need to invoke the recommender. Hence, the cost of computing influence metric is $O(|U|\phi)$, where $\phi$ is the recommender invocation cost.

Therefore, the total cost for computing all metrics for a single item is $O(|U|\phi)$, and for all items $I$ is:

$$O(|I||U|\phi)$$

In the *second loop* (*line* 7), the standard deviation computation in *line* 9 costs $O(d|I|)$. In *line* 10 the $\tau$-pareto set computation has a worst-case complexity of $O(|I| \log^{d-2} |I|)$ using partition-based algorithms [23], or $O(d|I|^2)$ for practical algorithms such as SFS [10], LESS [14], or SaLSa [6].

The procedure isCF in *line* 13 incurs a cost of $O(\phi)$. The loop (*line* 7) is executed up to a small constant $v$ times, as the thresholds $\tau$ quickly converge to include all items in $I$. Therefore, the second loop (*line* 7) has a worst-case complexity of:

$$O\left(v \cdot (d|I| + |I| \log^{d-2} |I| + \phi)\right)$$

Combining both loops, the overall *worst-case time complexity* of the *Pareto-filtering* algorithm is:

$$O\left(|I||U|\phi + v \cdot (|I| \log^{d-2} |I| + \phi)\right)$$

*Space Complexity.* The *space complexity* of the algorithm (excluding input data) is $O(|I|)$, accounting for internal structures $O$, $PS$, $PI$, and $CS$.

*Recommender Calls.* The *number of recommender calls* in the worst case is $O(|I||U| + v)$.

### 5.2 FixedWindow Search

The **FixedWindow** search technique serves as a local search strategy that incrementally examines contiguous subsets of items. The algorithm operates on a ranked item list and iteratively explores fixed-size windows, followed by an internal refinement phase to extract a small subset that corresponds to a counterfactual.

The interactive items are sorted by their total score, and a sliding window of fixed size is applied to the ordered list. The window starts at the beginning of the list and moves one position at a time, covering consecutive items in each step. At every position, the algorithm simulates the removal of the window items from the group's interaction items and then re-invokes the group recommender system. If the target item is no longer recommended to the group, the current window corresponds to an explanation. If the window traverses the entire list without yielding a counterfactual explanation, the window size is increased, and the list is scanned again from the beginning.

When the items within a window form a counterfactual explanation, we apply a *refinement phase* to improve minimality (i.e., reduce the explanation size). Specifically, we examine all possible subsets (i.e., the powerset) of the window items in ascending order of cardinality. The first subset that corresponds to a counterfactual, is selected as the final explanation, ensuring conciseness and interpretability. Note that, since this refinement phase has exponential complexity, we keep the window size relatively small to balance explanation quality with computational efficiency.

---

**Algorithm 2.** FixedWindow($I$, $t$, $w$)

---

**Input:** $I$: interacted items;   $t$: target item;   $w$: initial window size
**Output:** $CF$: counterfactual explanation
**Parameters:** $\pi$: group recommender; $f_1, f_2, \ldots, f_d$: item metric functions
**Variables:** $H$: ranked item list; $W$: current window; $S$: subset of window;
           $w_{cur}$: current window size

1  $H \leftarrow \varnothing$
2  **foreach** $i \in CS$ **do**                                                 // compute items scores & initialize items list
3      $i.score \leftarrow \sum_{g=1}^{d} f_g(i)$                  // compute item score by aggregating item metrics
4      insert $i$ into $H$

5  sort $H$ in descending order based on items scores

6  $w_{cur} \leftarrow w$                                                         // start with initial window size
7  **while** $w_{cur} \leq |H|$ **do**
8      **for** $i \leftarrow 1$ **to** $|H| - w_{cur} + 1$ **do**   // sliding window search
9         $W \leftarrow H[i : i + w_{cur} - 1]$      // window of $w_{cur}$ items
10        **if** isCF$(I, W, t)$ **then**            // check if window is a counterfactual
11           **foreach** $S \in$ PowerSet$(W)$ *in order of increasing* $|S|$ **do**   // **refinement phase**: check subsets
12              **if** isCF$(I, S, t)$ **then**
13                 $CF \leftarrow S$
14                 **break**

15           **if** $CF \neq \varnothing$ **then return** $CF$

16     $w_{cur} \leftarrow w_{cur} + w$                             // increase window size by initial $w$

17 **return** $\varnothing$

---

**Algorithm Description.** Algorithm 2 receives as input an interactive items set $I$, a target item $t$, and an initial window size $w$, and returns a counterfactual explanation $CF$. Initially, the total item score is computed for each item in $I$ (*lines* 2–4), and the items are sorted in descending order to form the ranked list $H$ (*line* 5). Then, the current window size $w_{cur}$ is initialized to the predefined input value $w$ (*line* 6).

For each window size, the algorithm slides a contiguous window $W$ over $H$ (*line* 9) and checks whether the items $W$ correspond to a counterfactual explanation by invoking the isCF procedure (which invokes the recommender) (*line* 10).

When a window contains a counterfactual explanation, the algorithm enters the refinement phase (*lines* 11–14), where all non-empty subsets of $W$ are examined in increasing order of cardinality. The first subset $S$ that corresponds to a counterfactual is returned (*line* 12–15).

If no counterfactual window is found, the window size is increased by $w$ and the search is repeated (*line* 16). The algorithm terminates when either an explanation is identified (*line* 15) or all window sizes are examined (i.e., $w_{cur} = |H|$).

**Complexity Analysis.** In Algorithm 2, *line* 3, each item $i \in I$ is assigned a score by aggregating the values of $d$ metric functions $f_1, f_2, \ldots, f_d$. As shown in Section 5.1 the complexity to compute the metric functions for an item is $O(|U|\phi)$. Therefore, the cost of computing all scores is:

$$O\left(|I||U|\phi\right)$$

Sorting the ranked list $H$ (*line* 5) employing a linearithmic sorting algorithm (e.g., mergesort) costs $O(|I| \log |I|)$. In main loop (*lines* 7–16) the number of distinct window sizes considered in the worst case is $\lceil |I|/w \rceil$. For each window size $w_{cur}$ the algorithm checks $O(|I| - w_{cur} + 1)$ windows. Hence, the total number of windows examined is $\sum_{k=1}^{\lceil \frac{|I|}{w} \rceil} (|I| - kw + 1)$, which corresponds to:

$$O \left( \frac{|I|^2}{w} \right)$$

For each window, the algorithm invokes the isCF procedure once (*line* 10), incurring a cost of $O(\phi)$. In *line* 11, the worst case, all $2^{w_{cur}} - 1$ subsets are evaluated, each requiring a call to isCF with cost $O(\phi + m)$ (*line* 12). Since $w_{cur}$ can grow up to $|I|$, the refinement phase may require up to $2^{|I|}$ checks in the worst case. Thus, the computational cost of the main loop (*lines* 7–16) is:

$$O \left( \frac{|I|^2}{w} \phi + 2^{|I|} \phi \right)$$

Combining all parts, the overall *worst-case time complexity* of the *FixedWindow* algorithm is:

$$O \left( |I||U|\phi + |I| \log |I| + \frac{|I|^2}{w} \phi + 2^{|I|} \phi \right)$$

*Space Complexity.* The *space complexity* is $O(|I|)$, accounting for the sorted list $H$ and the temporary subset storage during refinement.

*Recommender Calls.* The *number of recommender calls* in the worst case is:

$$O \left( |I||U| + \frac{|I|^2}{w} + 2^{|I|} \right)$$

Note that, in practice, the exponential term is rarely reached due to computational cut-offs, and $w$ is typically chosen as a small constant, which keeps the quadratic term as the dominant factor.

## 5.3    GreedyGrow Search

The **GreedyGrow** search is a method that progressively expands the search window until a counterfactual explanation is found. Unlike the fixed-size sliding window approach, this technique selects the top-ranked items in the list and adds one item at a time in sequential order. At each iteration, the item set, consisting of the top-ranked items is checked to see if it corresponds to a counterfactual explanation.

This method is particularly effective at rapidly identifying explanations with strong influence; however, it has a notable limitation: the resulting explanations are not guaranteed to be minimal. Specifically, although the approach identifies a subset that successfully alters the recommendation outcome, it may include extraneous items that are not necessary for forming a counterfactual. Consequently, the final explanation may be "unnecessarily" large, which, as discussed in Section 4, can reduce interpretability and increase cognitive load for end users.

**Algorithm Description.** Algorithm 3 implements the *GreedyGrow* search strategy, takes as input the interacted item set $I$ and a target item $t$, and returns the explanation $CF$. The algorithm first computes the item total score for each interacted item and stores the items in a ranked list $H$ (*lines* 2–5). It then initializes an empty candidate set $S$ (*line* 6).

During each iteration of the main loop (*lines* 7–10), the next highest-ranked item from $H$ is added to $S$, and the algorithm checks whether $S$ constitutes a counterfactual explanation by

---

**Algorithm 3.** GreedyGrow $(I, t)$

---

**Input:** $I$: interacted items;  $t$: target item;
**Output:** $CF$: counterfactual explanation;
**Parameters:** $\pi$: group recommender;  $f_1, f_2, \ldots, f_d$: item metrics functions;
**Variables:** $H$: items list;  $i$: item;  $S$: selected items set;  *cfFound*: boolean variable;

1   $H \leftarrow \varnothing$
2   **foreach** $i \in I$ **do**          //compute items scores & initialize items list
3      $i.score \leftarrow \sum_{g=1}^{d} f_g(i)$       //compute item score by aggregating item metrics
4      insert $i$ into $H$

5   sort $H$ in descending order based on items scores

6   $S \leftarrow \varnothing$
7   **for** $j \leftarrow 1$ **to** $|H|$ **do**            //select top items
8      $S \leftarrow S \cup H[j]$       //items set to be checked as counterfactual
9      $cfFound \leftarrow \text{isCF}(I, S, t)$       //call recommender system
10     **if** $cfFound$ = true **then  break**

11   **if** $cfFound$ = true **then**
12     $CF \leftarrow S$
13   **else**            //counterfactual not exist
14     $CF \leftarrow \varnothing$

15   **return** $CF$

---

invoking isCF (Procedure 1). The process terminates as soon as the target item is removed from the recommendation list.

If a counterfactual explanation is found, the current set $S$ is returned as the final explanation (*line* 12). Otherwise, if all items are exhausted without success, the algorithm returns the empty set (*line* 14).

**Complexity Analysis.** In *line* 3, each item $i \in I$ is assigned a score by aggregating the values of $d$ metric functions $f_1, f_2, \ldots, f_d$. As shown in Section 5.1, the complexity to compute the metric functions for an item is $O(|U|\phi)$. Thus, computing all item scores has a total cost:

$$O\left(|I||U|\phi\right)$$

Sorting the scored list $H$ (*line* 6) takes $O(|I| \log |I|)$. Then, in the main loop (*lines* 7–10), items from $H$ are incrementally added to the set $S$. After each addition, the isCF function is called (*line* 9) to check whether $S$ constitutes a valid counterfactual explanation. In the worst case, this process continues until all $|I|$ items have been added. Each call to isCF has cost $O(\phi)$. Thus, in the worst case, the total cost of the calls to isCF is:

$$O\left(|I|\phi\right)$$

From the above analysis, the overall *worst-case time complexity* of the *GreedyGrow* algorithm is:

$$O\left(|I||U|\phi + |I| \log |I|\right)$$

*Space Complexity.* The *space complexity* is $O(|I|)$, due to the storage of the list $H$ and the growing set $S$.

*Recommender Calls.* The *number of recommender calls* in the worst case is $O(|I||U| + |I|) = O(|I||U|)$.

## 5.4 Grow&Prune Search

To mitigate the verbosity of the *GreedyGrow* approach, we introduce the **Grow&Prune** search method, which first identifies an initial explanation and then refines it by improving minimality through backward elimination.

The *GreedyGrow* approach initially applies the *GreedyGrow* technique to generate an initial counterfactual explanation. Once this initial explanation is identified, we enter a *reduction phase that evaluates the necessity of each item within the explanation.* Each item in the initial counterfactual explanation is examined individually by temporarily excluding it from the set. For each exclusion, the group recommender is evaluated on the modified explanation to verify whether it continues to correspond to a counterfactual. If the exclusion results in a non-counterfactual explanation, the item is classified as essential and reinserted; otherwise, the exclusion is preserved, yielding a more concise explanation.

**Algorithm Description.** Algorithm 4 implements the *Grow&Prune* strategy. The algorithm first executes a grow phase (*lines* 1–10) identical to *GreedyGrow* algorithm, incrementally adding items until a counterfactual explanation is identified. If no such explanation is found, the algorithm terminates and returns the empty set (*line* 20).

Once a counterfactual explanation is obtained, the items in the candidate set $S$ are sorted in ascending order of score (*line* 12), and the algorithm enters the prune phase. In this phase, each item $i \in S$ is temporarily removed from the explanation (*line* 15), and the algorithm checks whether the reduced set still constitutes a valid counterfactual by invoking isCF (Proc. 1) (*line* 16).

If the counterfactual condition is preserved, the removal is retained (*line* 17); otherwise, the item is restored. This process continues until no further items can be removed, and the resulting set is returned as the final counterfactual explanation $CF$ (*line* 18).

**Complexity Analysis.** In *line* 3, each item $i \in I$ is assigned a score via the aggregation of $d$ metric functions $f_1, \ldots, f_d$. As shown in Section 5.1, the complexity to compute the metric functions for an item is $O(|U|\phi)$. Thus, computing all item scores has a total cost:

$$O\left(|I||U|\phi\right)$$

Sorting the scored list $H$ (*line* 6) takes $O(|I| \log |I|)$. Then, during the *grow phase* (*lines* 7–10), the algorithm incrementally builds a candidate set $S$ by scanning items in $H$ in descending order. After each addition, the isCF function is invoked. In the worst case, this loop iterates $|I|$ times, leading to:

$$O\left(|I| \cdot \phi\right)$$

In *prune phase* (*lines* 11–20), the items in $S$ (of size at most $|I|$) are sorted in descending score order, costing at most $O(|I| \log |I|)$. Then, for each item $i \in S$, the algorithm checks whether removing $i$ still yields a valid counterfactual by calling isCF. In the worst case, this requires $O(|I|)$ additional calls, resulting in:

$$O\left(|I| \log |I| + |I|\phi\right)$$

Combining all parts, the overall *worst-case time complexity* of the *Grow&Prune* algorithm is:

$$O\left(|I||U|\phi + |I| \log |I| + |I|\phi + |I| \log |I| + |I|\phi\right) = O\left(|I||U|\phi + |I| \log |I|\right)$$

---

**Algorithm 4.** Grow&Prune $(I, t)$

---

**Input:** $I$: interacted items;   $t$: target item
**Output:** $CF$: counterfactual explanation
**Parameters:** $\pi$: group recommender;   $f_1, f_2, \ldots, f_d$: item metrics functions
**Variables:** $H$: items list;   $i$: item;   $S$: selected items set;   $cfFound$: boolean variable

1  $H \leftarrow \varnothing$
2  **foreach** $i \in I$ **do**                                           // compute items scores & initialize items list
3  |   $i.score \leftarrow \sum_{g=1}^{d} f_g(i)$                          // compute item score by aggregating item metrics
4  |   insert $i$ into $H$
5  sort $H$ in descending order based on items scores
6  $S \leftarrow \varnothing$
7  **for** $j \leftarrow 1$ **to** $|H|$ **do**                            // grow phase
8  |   $S \leftarrow S \cup H[j]$
9  |   $cfFound \leftarrow \text{isCF}(I, S, t)$
10  |   **if** $cfFound = $ true **then  break**
11  **if** $cfFound = $ true **then**
12  |   sort $S$ in ascending order based on item scores
13  |   $CF \leftarrow S$
14  |   **foreach** $i \in S$ **do**                                       // prune phase
15  |   |   $CF' \leftarrow CF \setminus \{i\}$
16  |   |   **if** $\text{isCF}(I, CF', t)$ **then**                       // check if $P'$ still a counterfactual
17  |   |   |   $CF \leftarrow CF'$
18  |   **return** $CF$
19  **else**                                                              // counterfactual not found
20  |   **return** $\varnothing$

---

*Space Complexity.* The *space complexity* is $O(|I|)$, due to the storage of lists $H$, $S$, and temporary counterfactual sets.

*Recommender Calls.* The *total number of recommender calls* in the worst case is $O(|I||U|)$.

### 5.5  ExpRebuild Search

The **ExpRebuild** search approach extends *GreedyGrow* by incorporating *items explanatory power* (Eq. 4) to guide items selections. Specifically, exploiting explanatory power, it prioritizes items whose removal is expected to most strongly degrade the rank of the target item, or eliminate it entirely from the recommendation list.

Initially the method applies the *GreedyGrow* procedure to efficiently find an initial counterfactual explanation. During this process, the algorithm computes the exploration power (Equation 4) of each item.

After an initial explanation is identified, the items are sorted in descending order of explanatory power, placing the most impactful items first. The algorithm then constructs a new explanation incrementally: starting from an empty set, items are added one by one in order of decreasing explanatory power. After each addition, the recommender is invoked to check whether the growing set constitutes a counterfactual explanation.

---

**Algorithm 5.** ExpRebuild $(I, t)$

---

**Input:** $I$: interacted items;   $t$: target item;
**Output:** $CF$: counterfactual explanation;
**Parameters:** $\pi$: group recommender;   $f_1, f_2, \ldots, f_d$: item metrics functions
**Variables:** $H$: items list;   $i$: item;   $S$: selected items set;   $Q$: items set;   $inflScore$: influence score ;
$\qquad\qquad$ $cfFound$: boolean variable;   $P$: selected items from $S$

1   $H \leftarrow \varnothing$
2   **foreach** $i \in I$ **do**                                                //compute items scores & initialize items list
3       $i.score \leftarrow \sum_{g=1}^{d} f_g(i)$                              //compute item score by aggregating item metrics
4       insert $i$ into $H$

5   sort $H$ in descending order based on items scores

6   $S \leftarrow \varnothing$
7   **for** $j \leftarrow 1$ **to** $|H|$ **do**                                              //select top items
8       $Q \leftarrow S$     //If a counterfactual is found, $Q$ will contains the items excluding the last selected items (i.e., the most recently inserted
          item into $S$ that led to the counterfactual)
9       $S \leftarrow S \cup H[j]$                                      //items set to be checked as counterfactual
10      $cfFound, inflScore \leftarrow$ checkForCFandComputeExplPwr$(I, t, S)$  //check if $S$ is counterfactual and compute $H[j]$
          influence score
11      $H[j].pow \leftarrow explPower/|S|$                           //set explanatory power score to item $H[j]$
12      **if** $cfFound$ = true **then  break**

13 **if** $cfFound$ = true **then**                                     //rebuild based on explanatory power
14      sort $S$ in descending order based on the explanatory power score
15      $P \leftarrow \varnothing$
16      **for** $j \leftarrow 1$ **to** $|S|$ **do**
17         $P \leftarrow P \cup S[j]$                                 //items set to be checked as counterfactual
18         **if** $P \nsubseteq Q$ **then**
19            $cfFound \leftarrow$ isCF$(I, P, t)$
20         **if** $cfFound$ = true **then return** $P$
21      **return** S;
22 **else**                                                     //counterfactual not exist
23      **return** $\varnothing$

---

This approach offers a compromise between minimality and interpretability, ensuring that the explanation is both compact and grounded in a transparent attribution of item effects.

**Algorithm Description.** Algorithm 5 implements the *ExpRebuild* strategy. The algorithm begins by computing the total score for all items and sorting them into a ranked list $H$ (*lines* 1–5). It then incrementally builds a candidate set $S$ (*lines* 6–12), invoking the procedure checkForCFandComputeExplPwr (Proc. 2) at each step to determine whether a counterfactual explanation has been found and to compute the explanatory power of the current set.

Once a counterfactual explanation is identified, the items in $S$ are ordered according to their explanatory power (*line* 14). The algorithm then reconstructs the explanation by incrementally

---

**Procedure 2:** checkForCFandComputeExplPwr($I$, $t$, $S$)

---

**Input:** $I$: interacted items;  $t$: target item;  $S$: interacted items to be checked as counterfactual;
**Output:** $isCF$: boolean value (if $S$ is a counterfactual);  $explPower$: explanatory power score of item $i$
**Parameters:** $\pi$: group recommender
**Variables:** $L$: recommendation list generated considering $I \setminus S$ as interacted

1  $L \leftarrow \pi(I \setminus S)$                                                    // call recommender system
2  $explPower \leftarrow \min \left\{ \frac{\text{rank}(t,L)-1}{|L|}, 1 \right\}$        // compute $S$ explanatory power (Eq. 4)
3  **if** $t \notin L$ **then**
4  |   $isCF \leftarrow$ true
5  **else**
6  |   $isCF \leftarrow$ false
7  **return** $isCF$,  $explPower$

---

adding items in descending explanatory power order (*lines 16–20*), checking after each addition whether the counterfactual condition holds.

The first subset that satisfies the counterfactual condition is returned as the final explanation. If no such subset is found, the full set $S$ is returned (*line 21*). If no counterfactual explanation is identified during the initial phase, the algorithm returns the empty set (*line 23*).

**Complexity Analysis.** In *line 3* item scores are computed for all items $i \in I$ using $d$ metric functions $f_1, \ldots, f_d$ As shown in Section 5.1, the complexity to compute the metric functions for an item is $O(|U|\phi)$. Thus, computing all item scores has a total cost:

$$O\left(|I||U|\phi\right)$$

Sorting the scored list $H$ costs $O(|I| \log |I|)$. The main loop (*lines 7–12*) adds items one-by-one to the candidate set $S$ and invokes the checkForCFandComputeInfluence procedure at each step. In the worst case, the loop is performed $|I|$ times.

The checkForCFandComputeExplPwr procedure calls the recommender system once (*line 1*), and computes the item exploratory power score (Eq. 4), which can be done in $O(1)$ (if a hash-based indexing is used over $L$). Hence, each invocation of checkForCFandComputeExplPwr costs:

$$O(\phi)$$

Therefore, the total cost of the main loop (*lines 7–12*) is:

$$O(|I|\phi)$$

Next, if a counterfactual explanation is found (*line 13*), the algorithm sorts $S$ based on item influence power scores, requiring $O(|I| \log |I|)$. Then, in the loop (*lines 16–20*), attempts to rebuild the minimal explanation set $P \subseteq S$ by adding one item at a time and calling isCF. In the worst case, $|S| = |I|$, so this phase invokes the recommender system at most $|I|$ times, resulting to $O(|I|\phi)$. Summing up all parts, the overall *worst-case time complexity* of the *ExpRebuild* algorithm is:

$$O\left(|I||U|\phi + |I| \log |I| + |I|\phi + |I| \log |I| + |I|\phi\right) = O\left(|I||U|\phi + |I| \log |I|\right)$$

*Space Complexity.* The *space complexity* is $O(|I|)$, for storing lists $H$, $S$, and $P$.

*Recommender Calls.* In the worst case, the number of recommender calls for metrics computations is $O(|I||U|)$, during the grow phase (via checkForCFandComputeExplPwr) is $O(|I|)$, and $O(|I|)$ calls

during the rebuild phase (via isCF). Therefore, in the worst case, the *total number of recommender calls* is $O(|I||U|)$.

Table 3. Number of Users, Items, and Ratings

| Dataset | Users | Items | Ratings |
|---------|-------|-------|---------|
| **MovieLens** | 69,878 | 10,677 | 10,000,054 |
| **Amazon** | 344,747 | 373,665 | 5,573,065 |

## 6 Experimental Evaluation

### 6.1 Experimental Setting

Our experimental design aims to demonstrate the effectiveness and reliability of the proposed heuristic methods for group counterfactual explanations across various recommendation contexts.

**Datasets.** We employed two benchmark datasets, *MovieLens* 10M [15] and *Amazon* [17], which differ considerably in their characteristics. While MovieLens represents a dense and well-studied domain of movie ratings, the Amazon dataset is much sparser and reflects a more heterogeneous item space. Table 3 presents the characteristics of each dataset.

**Group Formation.** To represent realistic group recommendation scenarios, we simulated groups of two sizes: five members (*Group 5*), reflecting smaller and more common decision-making units, and ten members (*Group 10*), representing larger and more diverse groups. We limited our analysis to users with a minimum of 50 ratings to ensure sufficient preference information and minimize noise in the recommendation process. This choice also increases the size of the candidate item pool, creating a more challenging environment for identifying group counterfactual explanations.

**Recommender System.** For constructing group recommendation lists, we followed a two-step process. Individual recommendations were first generated for each group member using user-based collaborative filtering, after which group-level rankings were obtained through the average aggregation strategy. This setup offers a balanced and transparent baseline against which the performance of our counterfactual explanation methods can be assessed.

**Evaluation Scenario.** We evaluate our proposed methods using the two datasets, MovieLens and Amazon, reporting average results over 20 randomly generated groups for each group size (5 and 10). Each method is evaluated both with and without a Pareto-filtering phase. When a Pareto-filtering phase is applied, we denote the corresponding variants as Pareto-filtering. In the absence of Pareto-filtering, the methods are denoted as Sorted List, reflecting the fact that each algorithm begins by sorting the interaction item list.

We impose a maximum budget of 1000 calls to the group recommender system. The group recommendation list length is fixed at 10. For consistency and to minimize randomness, the target item for which we seek a counterfactual explanation is always the top-1 recommendation in the group list. We consider it as no longer recommended if it disappears entirely from the top-10 recommendation list. For the *FixedWindow* method, the initial window size $w$ is set to 15.
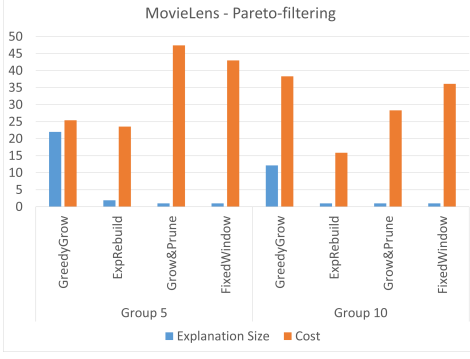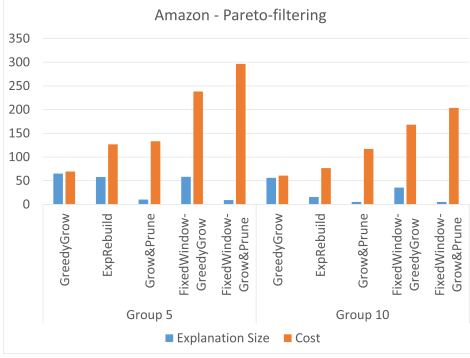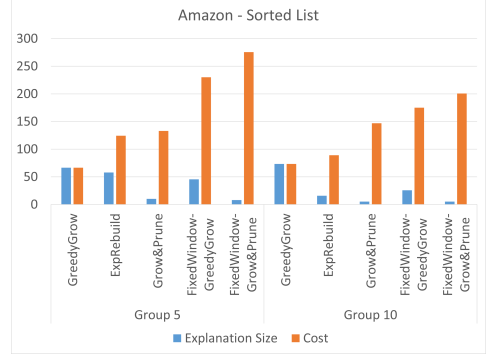
(a) Pareto-filtering
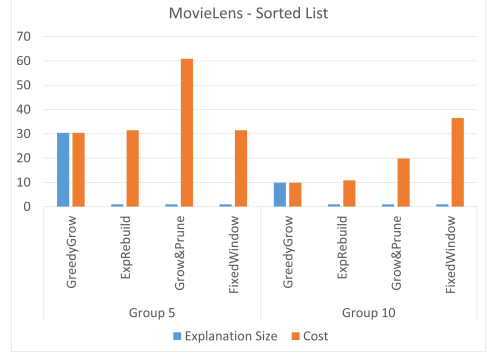
(b) Sorted List

Fig. 2. Explanation Size and Cost on the MovieLens dataset
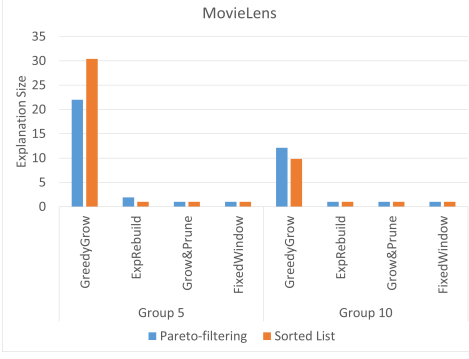


(a) Pareto-filtering

(b) Sorted List

Fig. 3. Explanation Size and Cost on the Amazon dataset

## 6.2 Explanation Size and Cost Evaluation

**MovieLens.** Figure 2 reports results for the MovieLens dataset, comparing explanation size and cost with and without Pareto-filtering. The *GreedyGrow* heuristic consistently yields the lowest cost but produces the largest explanations among all methods. By contrast, the *FixedWindow* approach often identifies minimal explanations but at a substantially higher cost. The *Grow&Prune* method also achieves minimal explanations, though for smaller group sizes it incurs the highest costs. For larger groups, *FixedWindow* struggles to identify a suitable search window that contains a valid counterfactual explanation, which further increases the cost.

**Amazon.** Figure 3 presents results for the Amazon dataset. Due to the dataset's higher sparsity compared to MovieLens, the *FixedWindow* approach fails in its standard form. The required window size becomes excessively large when attempting to locate the initial counterfactual window, forcing exhaustive search over a large number of possible combinations. This quickly reaches the available budget. To address this, we introduce two hybrid variants of *FixedWindow*: *FixedWindow-GreedyGrow*, where the exhaustive search is replaced by *GreedyGrow*, and *FixedWindow-Grow&Prune*, where it is
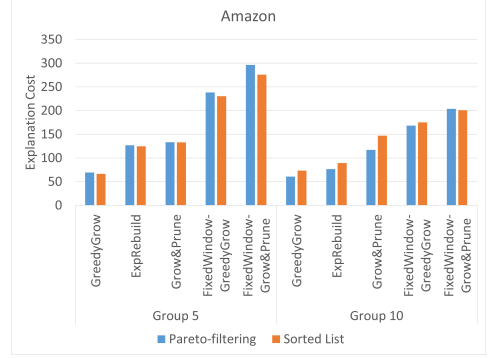
(a) MovieLens dataset
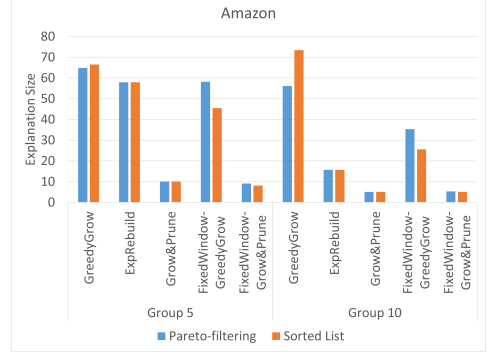
(b) Amazon dataset

Fig. 4. Pareto-filtering vs. Sorted List: Explanation Size
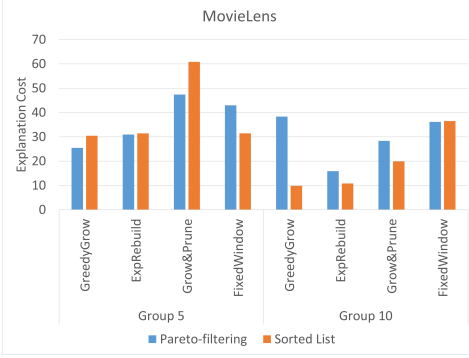


(a) MovieLens dataset

(b) Amazon dataset

Fig. 5. Pareto-filtering vs. Sorted List: Explanation Cost

replaced by *Grow&Prune*. Although these variants execute successfully within the budget limit, the additional overhead incurred when first identifying an initial counterfactual explanation leads to the highest overall cost. This highlights a key limitation of *FixedWindow*: when the items forming a counterfactual explanation are not close in ranking, the window size becomes impractically large.

The other techniques behave similarly to the MovieLens case: *GreedyGrow* yields the largest explanations at the lowest cost, whereas *Grow&Prune* produces the smallest explanations at a higher cost.

### 6.3 Pareto-filtering Method Evaluation

Figures 4 and 5 provide a direct comparison of results with and without Pareto-filtering. Figure 4 illustrates explanation sizes. In the MovieLens dataset, Pareto-filtering slightly reduces explanation size for *GreedyGrow*, particularly with smaller groups. In the Amazon dataset, the impact is more pronounced: Pareto-filtering consistently reduces explanation sizes, confirming its utility when explanations are more difficult to locate.

For explanation cost (Figure 5), Pareto-filtering reduces costs across most heuristics in the MovieLens dataset, with the exception of *FixedWindow*. This is because items with high total scores
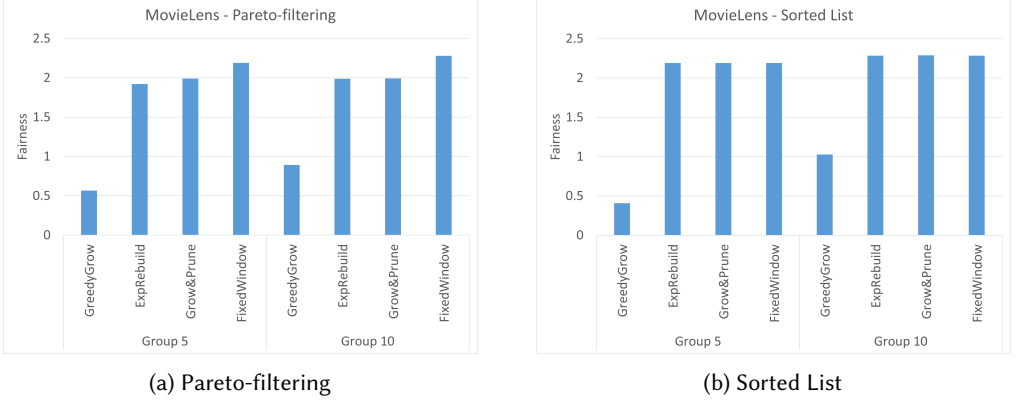
(a) Pareto-filtering

(b) Sorted List

Fig. 6. Explanation Fairness on the MovieLens dataset
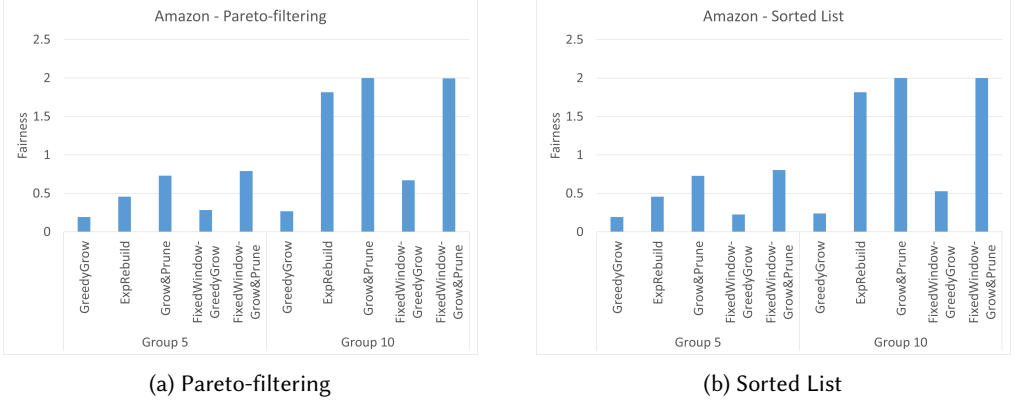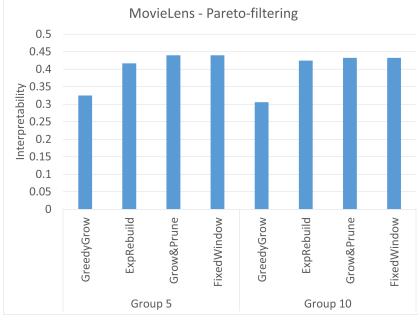


(a) Pareto-filtering

(b) Sorted List

Fig. 7. Explanation Fairness on the Amazon dataset

are more likely to influence the group recommendation outcome, making them more likely to appear in early windows. In such cases, the overhead of computing the Pareto skyline outweighs its benefits. In contrast, in the sparser Amazon dataset and for larger group sizes, Pareto-filtering substantially reduces explanation costs, demonstrating its value in more challenging scenarios.

## 6.4 Explanation Fairness Evaluation

Figures 6 and 7 present the fairness scores (Eq. 11) for the MovieLens and Amazon datasets, respectively. For the MovieLens dataset, group size does not substantially affect fairness, as the scores remain consistent across all heuristics. In contrast, the Amazon dataset shows a clearer trend: fairness increases with larger group sizes. This effect is primarily explained by the difference in explanation sizes between small and large groups. For smaller groups, heuristics tend to produce larger explanations on average (Figure 4). As the explanation size grows, the items it contains are less evenly shared across group members. This uneven distribution leads to greater variability in individual contributions, reflected in a higher standard deviation, and consequently results in lower fairness.
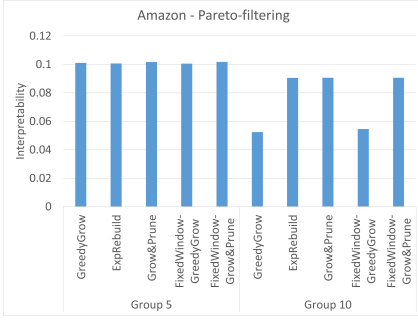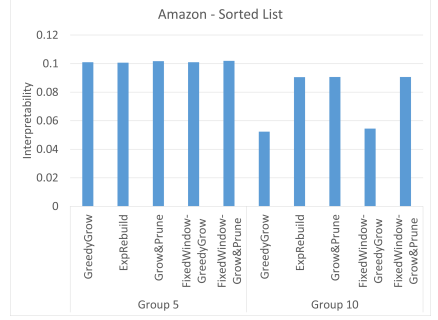
(a) Pareto-filtering
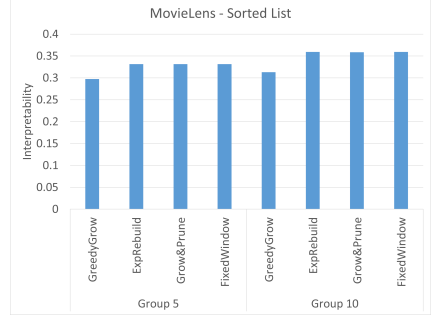
(b) Sorted List

Fig. 8. Explanation Interpretability on the MovieLens dataset



(a) Pareto-filtering

(b) Sorted List

Fig. 9. Explanation Interpretability on the Amazon dataset

Across heuristics, those that yield smaller explanations also achieve higher fairness, consistent with the above relationship. Specifically, in the MovieLens dataset, *ExpRebuild*, *Grow&Prune*, and *FixedWindow* achieve the highest fairness, whereas *GreedyGrow* produces the lowest. The same pattern is observed in the Amazon dataset. This highlights a trade-off: heuristics optimized for low cost (such as *GreedyGrow*) may do so at the expense of fairness, whereas heuristics that prioritize minimality (e.g., *Grow&Prune*) tend to yield explanations that are more balanced across group members.

Taken together, these results underline the importance of considering fairness alongside cost and minimality when evaluating counterfactual explanation methods, especially in sparse datasets where the distribution of interactions is more uneven.

## 6.5 Explanation Interpretability Evaluation

We evaluate explanation interpretability by measuring the recognition of the items included in a counterfactual explanation (Eq. 8). Recognition is considered both within the group and across all users in the system.

As shown in Figures 8 and 9, for the MovieLens and Amazon datasets respectively, interpretability decreases slightly as group size increases. This effect arises because with larger groups it becomes less likely that all members have interacted with the items in the explanation.
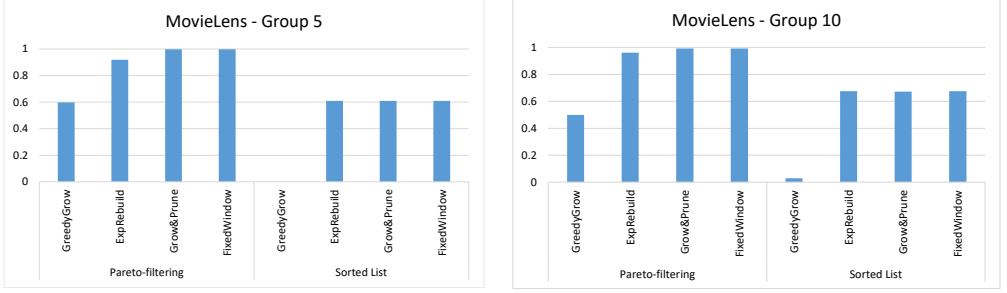
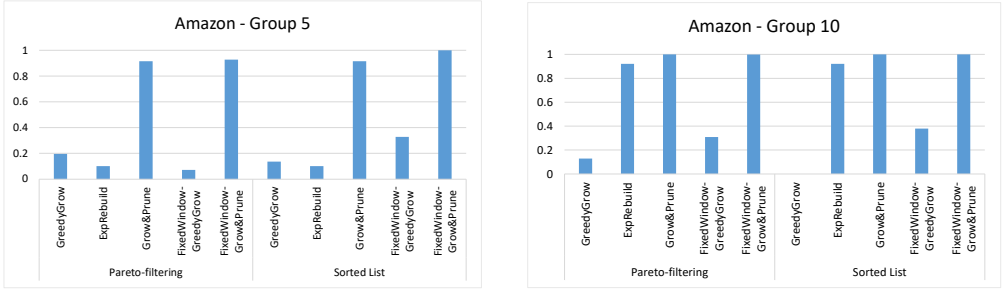Fig. 10. Explanation Utility on the MovieLens dataset



Fig. 11. Explanation Utility on the Amazon dataset

For the MovieLens dataset (Figure 8), Pareto-filtering has a strong positive effect on interpretability. This is expected since two of the dimensions considered in the Pareto skyline are precisely the components of interpretability: item recognition within the group and item recognition across the system. By promoting items along these dimensions, Pareto-filtering systematically increases the interpretability of the resulting explanations.

In contrast, for the Amazon dataset (Figure 9), counterfactual explanations are harder to identify due to the dataset's sparsity. In this case, Pareto-filtering has little effect, as the main challenge lies in locating any valid explanation at all rather than in refining the recognition characteristics of the items it contains.

Overall, these results highlight that interpretability is strongly influenced by dataset density. Pareto-filtering is especially beneficial in denser domains such as MovieLens, while in sparse domains like Amazon, its impact is limited.

### 6.6 Explanation Utility Evaluation

Figures 10 and 11 report the utility scores (Eq. 10). Recall that utility is an aggregation of explanation minimality and interpretability. We normalize each metric using min–max normalization and compute the utility as an equally weighted sum, ensuring that both metrics contribute comparably despite differing dynamic ranges. These results reflect trends under the aggregated utility metric; detailed trade-offs among individual criteria are analyzed in previous sections.

Additionally, we evaluate the sensitivity of the utility score to different aggregation weights between minimality and interpretability. In particular, we consider weight configurations of 0.3–0.7 and 0.7–0.3. The variations in aggregation weights does not alter the relative ranking of methods, with *Grow&Prune* remaining dominant across all settings. Differences primarily affect weaker

heuristics: increasing the weight on the minimality metric improves the utility of *GreedyGrow* in MovieLens, whereas emphasizing the interpretability metric degrades *GreedyGrow* but slightly benefits *ExpRebuild* in Amazon. The resulting plots are omitted for brevity.

## 7 Conclusions

In this paper, we introduced the first systematic framework for generating counterfactual explanations in group recommender systems. By formalizing group-level counterfactuals, we highlighted how explanations can reveal the influence of individual interactions on collective outcomes, while also addressing important dimensions such as minimality, interpretability, utility, and fairness. To cope with the inherent combinatorial complexity of explanation discovery, we proposed a family of heuristic algorithms, including *GreedyGrow*, *Grow&Prune*, *ExpRebuild*, and *FixedWindow*, as well as a *Pareto-based filtering* strategy. Our experimental evaluation on the MovieLens and Amazon datasets demonstrated clear trade-offs between efficiency, explanation size, and fairness, and showed the benefits of Pareto-filtering in sparse settings. Looking ahead, several research directions remain open. A promising avenue is to extend the framework to dynamic and sequential group recommendations, where both group composition and preferences evolve over time.

## References

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. Knowl. Data Eng.* 17, 6 (2005).

[2] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawlat, Gautam Das, and Cong Yu. [n. d.]. Group Recommendation: Semantics and Efficiency. *VLDB Endowment (PVLDB)* ([n. d.]).

[3] Hervé Attolou, Katerina Tzompanaki, Kostas Stefanidis, and Dimitris Kotzinos. 2024. Why-Not Explainable Graph Recommender. In *IEEE International Conference on Data Engineering (ICDE)*.

[4] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. 2010. Group recommendations with rank aggregation and collaborative filtering. In *ACM International Conference on Recommender Systems (RecSys)*.

[5] Oren Barkan, Veronika Bogina, Liya Gurevitch, Yuval Asher, and Noam Koenigstein. 2024. A Counterfactual Framework for Learning and Evaluating Explanations for Recommender Systems. In *ACM Web Conference (WWW)*.

[6] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. 2008. Efficient sort-based skyline evaluation. *ACM Trans. Database Syst.* 33, 4 (2008).

[7] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. 2018. Equity of Attention: Amortizing Individual Fairness in Rankings. In *ACM International Conference on Research and Development in Information Retrieval (SIGIR)*.

[8] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive Group Recommendation. In *ACM International Conference on Research and Development in Information Retrieval (SIGIR)*.

[9] Lenart Celar and Ruth M. J. Byrne. 2023. How people reason with counterfactual and causal explanations for Artificial Intelligence decisions in familiar and unfamiliar domains. *Memory & Cognition* 51 (2023).

[10] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. 2003. Skyline with Presorting. In *IEEE International Conference on Data Engineering (ICDE)*.

[11] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. 2012. Fairness through awareness. In *Innovations in Theoretical Computer Science*.

[12] Azin Ghazimatin, Oana Balalau, Rishiraj Saha Roy, and Gerhard Weikum. 2020. PRINCE: Provider-side Interpretability with Counterfactual Explanations in Recommender Systems. In *ACM Intrnational Conference on Web Search and Data Mining (WSDM)*.

[13] Azin Ghazimatin, Oana Denisa Balalau, Rishiraj Saha Roy, and Gerhard Weikum. 2020. PRINCE: Provider-side Interpretability with Counterfactual Explanations in Recommender Systems. In *ACM Intrnational Conference on Web Search and Data Mining (WSDM)*.

[14] Parke Godfrey, Ryan Shipley, and Jarek Gryz. 2007. Algorithms and analyses for maximal vector computation. *Journal on Very Large Data Bases (VLDBJ)* 16, 1 (2007).

[15] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2015).

[16] Miao He, Bo An, Jun Wang, and Hao Wen. 2023. CETD: Counterfactual Explanations by Considering Temporal Dependencies in Sequential Recommendation. *Applied Sciences* 13, 20 (2023).

[17] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *World Wide Web Conference (WWW)*.

[18] Neham Jain, Vibhhu Sharma, and Gaurav Sinha. 2024. Counterfactual Explanations for Visual Recommender Systems. In *ACM Web Conference*.

[19] Vassilis Kaffes, Dimitris Sacharidis, and Giorgos Giannopoulos. 2021. Model-Agnostic Counterfactual Explanations of Recommendations. In *ACM Intrnational Conference on User Modeling, Adaptation and Personalization (UMAP)*.

[20] Mesut Kaya, Derek Bridge, and Nava Tintarev. 2020. Ensuring Fairness in Group Recommendations by Rank-Sensitive Balancing of Relevance. In *ACM International Conference on Recommender Systems (RecSys)*.

[21] Jae Kyeong Kim, Hyea Kyeong Kim, Hee Young Oh, and Young U. Ryu. 2010. A group recommendation system for online communities. *International Journal of Information Management* (2010).

[22] Tomás Kliegr, Stepán Bahník, and Johannes Fürnkranz. 2021. A review of possible effects of cognitive biases on interpretation of rule-based machine learning models. *Artif. Intell.* 295 (2021).

[23] H. T. Kung, Fabrizio Luccio, and Franco P. Preparata. 1975. On Finding the Maxima of a Set of Vectors. *J. ACM* 22, 4 (1975).

[24] Alejandro Kuratomi, Zed Lee, Panayiotis Tsaparas, Guilherme Dinis Junior, Evaggelia Pitoura, Tony Lindgren, and Panagiotis Papapetrou. 2024. CounterFair: Group Counterfactuals for Bias Detection, Mitigation and Subgroup Identification. In *IEEE International Conference on Data Mining (ICDM)*.

[25] Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual Fairness. In *Advances in Neural Information Processing Systems 30 (NIPS)*.

[26] Emilia Lenzi and Kostas Stefanidis. 2025. ADAPT: Fairness & diversity for sequential group recommendations. *Inf. Syst.* 133 (2025).

[27] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30 (NIPS)*.

[28] Judith Masthoff. 2015. Group Recommender Systems: Aggregation, Satisfaction and Group Attributes. In *Recommender Systems Handbook*. Springer.

[29] Giacomo Medda, Francesco Fabbri, Mirko Marras, Ludovico Boratto, and Gianni Fenu. 2024. GNNUERS: Fairness Explanation in GNNs for Recommendation via Counterfactual Reasoning. *ACM Transactions on Intelligent Systems and Technology* (2024).

[30] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267 (2019).

[31] Menaka Narayanan, Emily Chen, Jeffrey He, Been Kim, Sam Gershman, and Finale Doshi-Velez. 2018. How do Humans Understand Explanations from Machine Learning Systems? An Evaluation of the Human-Interpretability of Explanation. *CoRR* abs/1802.00682 (2018).

[32] Eirini Ntoutsi, Kostas Stefanidis, Kjetil Nørvåg, and Hans-Peter Kriegel. 2012. Fast Group Recommendations by Applying User Clustering. In *Conceptual Modeling International Conference (ER)*.

[33] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. 2022. Fairness in rankings and recommendations: an overview. *Journal on Very Large Data Bases (VLDBJ)* 31, 3 (2022).

[34] D. Qin, X. Zhou, L. Chen, G. Huang, and Y. Zhang. 2018. Dynamic Connection-based Social Group Recommendation. *IEEE Trans. Knowl. Data Eng.* (2018).

[35] Raymond Reiter. 1987. A Theory of Diagnosis from First Principles. *Artif. Intell.* 32, 1 (1987).

[36] Dimitris Sacharidis. 2019. Top-N Group Recommendations with Fairness. In *ACM Symposium On Applied Computing (SAC)*.

[37] Amirali Salehi-Abari and Craig Boutilier. 2015. Preference-Oriented Social Networks: Group Recommendation and Inference. In *ACM International Conference on Recommender Systems (RecSys)*.

[38] Kacper Sokol and Peter A. Flach. 2019. Counterfactual Explanations of Machine Learning Predictions: Opportunities and Challenges for AI Safety. In *Workshop on Artificial Intelligence Safety*.

[39] Maria Stratigi, Nikos Bikakis, and Kostas Stefanidis. 2025. Counterfactual Explanations for Group Recommendations. In *International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP)*.

[40] Maria Stratigi, Evaggelia Pitoura, Jyrki Nummenmaa, and Kostas Stefanidis. 2022. Sequential group recommendations based on satisfaction and disagreement scores. *J. Intell. Inf. Syst.* 58, 2 (2022), 227–254.

[41] Maria Stratigi, Katerina Tzompanaki, and Kostas Stefanidis. 2020. Why-Not Questions & Explanations for Collaborative Filtering. In *Web Information Systems Engineering (WISE)*.

[42] Juntao Tan, Shuyuan Xu, Yingqiang Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. 2021. Counterfactual Explainable Recommendation. In *ACM International Conference on Information & Knowledge Management (CIKM)*.

[43] Nava Tintarev. 2007. Explanations of recommendations. In *ACM International Conference on Recommender Systems (RecSys)*.

[44] Nava Tintarev and Judith Masthoff. 2007. A Survey of Explanations in Recommender Systems. In *IEEE International Conference on Data Engineering (ICDE)*.

[45] Giulia Vilone, Lucas Rizzo, and Luca Longo. 2020. A comparative analysis of rule-based, model-agnostic methods for explainable artificial intelligence. In *Irish Conference on Artificial Intelligence and Cognitive Science (AICS)*.

[46] Lucas Vinh Tran, Tuan-Anh Nguyen Pham, Yi Tay, Yiding Liu, Gao Cong, and Xiaoli Li. 2019. Interact and Decide: Medley of Sub-Attention Networks for Effective Group Recommendation. In *ACM International Conference on Research and Development in Information Retrieval (SIGIR)*.

[47] Sandra Wachter, Brent D. Mittelstadt, and Chris Russell. 2018. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *Harvard Journal of Law & Technology* 31, 2 (2018).

[48] Danding Wang, Qian Yang, Ashraf M. Abdul, and Brian Y. Lim. 2019. Designing Theory-Driven User-Centric Explainable AI. In *ACM International Conference on Human Factors in Computing Systems (CHI)*.

[49] QingXian Wang. 2023. Reinforced Model-Agnostic Counterfactual Explanations for Recommender Systems. In *Proceedings of SPIE*, Vol. 12700.

[50] Xiangmeng Wang, Qian Li, Dianer Yu, Qing Li, and Guandong Xu. 2024. Counterfactual Explanation for Fairness in Recommendation. *ACM Transactions on Information Systems* 42, 4 (2024).

[51] Lin Xiao, Zhang Min, Zhang Yongfeng, Gu Zhaoquan, Liu Yiqun, and Ma Shaoping. 2017. Fairness-Aware Group Recommendation with Pareto-Efficiency. In *ACM International Conference on Recommender Systems (RecSys)*.

[52] Yuanshun Yao, Chong Wang, and Hang Li. 2022. Learning to Counterfactually Explain Recommendations. *ArXiv* abs/2211.09752 (2022).

[53] H. Yin, Q. Wang, K. Zheng, Z. Li, J. Yang, and X. Zhou. 2019. Social Influence-Based Group Representation Learning for Group Recommendation. In *IEEE International Conference on Data Engineering (ICDE)*.

[54] Quan Yuan, Gao Cong, and Chin-Yew Lin. 2014. COM: A Generative Model for Group Recommendation. In *ACM International Conference on Knowledge Discovery & Data Mining (KDD)*.

[55] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Found. Trends Inf. Retr.* 14, 1 (2020).