A Graph Theoretic Approach for Multi-Objective Budget Constrained Capsule Wardrobe Recommendation

SHUBHAM PATIL, DEBOPRIYO BANERJEE, and SHAMIK SURAL, Indian Institute of Technology Kharagpur

Traditionally, capsule wardrobes are manually designed by expert fashionistas through their creativity and technical prowess. The goal is to curate minimal fashion items that can be assembled into several compatible and versatile outfits. It is usually a cost and time intensive process, and hence lacks scalability. Although there are a few approaches that attempt to automate the process, they tend to ignore the price of items or shopping budget. In this article, we formulate this task as a multi-objective budget constrained capsule wardrobe recommendation (*MOBCCWR*) problem. It is modeled as a bipartite graph having two disjoint vertex sets corresponding to top-wear and bottom-wear items, respectively. An edge represents compatibility between the corresponding item pairs. The objective is to find a 1-neighbor subset of fashion items as a capsule wardrobe that jointly maximize compatibility and versatility scores by considering corresponding user-specified preference weight coefficients and an overall shopping budget as a means of achieving personalization. We study the complexity class of *MOBCCWR*, show that it is NP-Complete, and propose a greedy algorithm for finding a near-optimal solution in real time. We also analyze the time complexity and approximation bound for our algorithm. Experimental results show the effectiveness of the proposed approach on both real and synthetic datasets.

CCS Concepts: • **Information systems** → **Recommender systems**; • **Applied computing** → **Online shopping**; • **Theory of computation** → **Graph algorithms analysis**; **Approximation algorithms analysis;**

Additional Key Words and Phrases: Capsule wardrobe, recommendation, bipartite graph, e-commerce, fashion, budget, outfit, compatibility, versatility

ACM Reference format:
Shubham Patil, Debopriyo Banerjee, and Shamik Sural. 2021. A Graph Theoretic Approach for Multi-Objective Budget Constrained Capsule Wardrobe Recommendation. *ACM Trans. Inf. Syst.* 40, 1, Article 1 (August 2021), 33 pages.
https://doi.org/10.1145/3457182

1 INTRODUCTION

Fashion is an impactful non-verbal language for expressing social status, occupation, role, self-confidence, intelligence, conformity, individuality, and other personality traits. With a rise in the

S. Patil and D. Banerjee contributed equally to this research.

Authors' addresses: S. Patil, Dept. of Electronics & Electrical Communication Engineering, Indian Institute of Technology Kharagpur, West Bengal, 721302, India; email: shubhampatil11698@iitkgp.ac.in; D. Banerjee and S. Sural, Dept. of Computer Science and Engineering, Indian Institute of Technology Kharagpur, West Bengal, 721302, India; emails: debopriyo@iitkgp.ac.in, shamik@cse.iitkgp.ac.in.

1046-8188/2021/08-ART1 $15.00
https://doi.org/10.1145/3457182

popularity of online shopping portals, vast collections of fashionable apparel are now easily available to millions of people around the world. However, finding relevant products from the right source is tedious work. As a result, there is a high demand for recommender systems [33, 39, 44] that ensure a satisfying shopping experience similar to retail stores. In the domain of fashion, existing work mainly focuses on the retrieval of similar or compatible items [18, 20, 23, 36, 39, 48–50], recommendation of items based on body shapes [24, 26], composition of outfits [8, 22, 38], semantic segmentation of fashion items on the human body [28–30, 51, 56], and virtual try-on frameworks [43, 45]. Recently, a new form of composite recommendation [55] called *recommendation of capsule wardrobes* [14, 25] was introduced, which promotes the concept of buying minimal fashion items that produce a maximum number of compatible and versatile outfit combinations.

For example, a capsule wardrobe may consist of three top-wear and two bottom-wear items. Presence of a top-bottom pair in a capsule wardrobe indicates that they are compatible and each has a versatility score, which contribute to the overall compatibility and versatility of the entire set. Generally, capsule wardrobes are created manually by fashion experts, which is costly, time consuming, and does not meet the current rising demands of the online shopping industry. Hsiao and Grauman [25] propose to create capsule wardrobes from fashion images by formulating the problem as a subset selection problem with sub-modular objective functions for handling compatibility, versatility, and user mentioned preferences. Dong et al. [14] introduce a combinatorial optimization based personalized capsule wardrobe creation framework, which jointly integrates user modeling and garment modeling. Both of these methods heavily depend on machine learning and deep learning techniques to model compatibility and versatility between items, user preferences, and body shapes. But none of them consider price of items or an overall shopping budget of the user.

In practice, users want the best possible set of fashion items within their budget, as it plays a critical role in influencing their purchase decision [7, 35, 61]. Moreover, there is a lack of complexity analysis of the optimization problem under consideration. This motivated us to address the shortcomings in the existing literature. In this article, we formally define a multi-objective budget constrained capsule wardrobe recommendation (*MOBCCWR*) problem as an optimization problem, where the objective is to find a subset of items as a capsule wardrobe from the set of all catalogued items of a shopping site that jointly maximize both compatibility and versatility scores, considering user mentioned preference weight coefficients for compatibility, versatility, and a maximum shopping budget.

Hsiao and Grauman [25] define the compatibility score of a set of items as the sum of the compatibility scores of all outfits in the set and the versatility score as the sum of the degree of different styles covered by the outfits in the set. We consider binary $\{0, 1\}$ scores for outfit compatibility and compute compatibility of a set as the sum of all compatible outfits with a score of 1 in the set. Now, an item may be used in various outfits portraying different styles. So, we consider item-specific versatility scores, e.g., a particular item that can be used in both formal and semi-formal outfits has a versatility score of 2.

Recommendation systems can be broadly categorized as content based, collaborative filtering based, and hybrid, i.e., an effective combination of both. Usually, items similar to a user's personal preferences are retrieved by content based recommendation systems. In this article, we mainly focus on content based recommendations and explore other dimensions of user preferences, namely compatibility of outfits and versatility of individual items in outfits for the recommendation of a capsule wardrobe, which are jointly maximized as an objective function. Personalization is achieved by using two user specified preference weight coefficients α_C and α_V that control the level of compatibility and versatility. Finally, a budget $\mathcal{K}$ sets an upper bound on the total price of distinct items in the recommended capsule wardrobe.

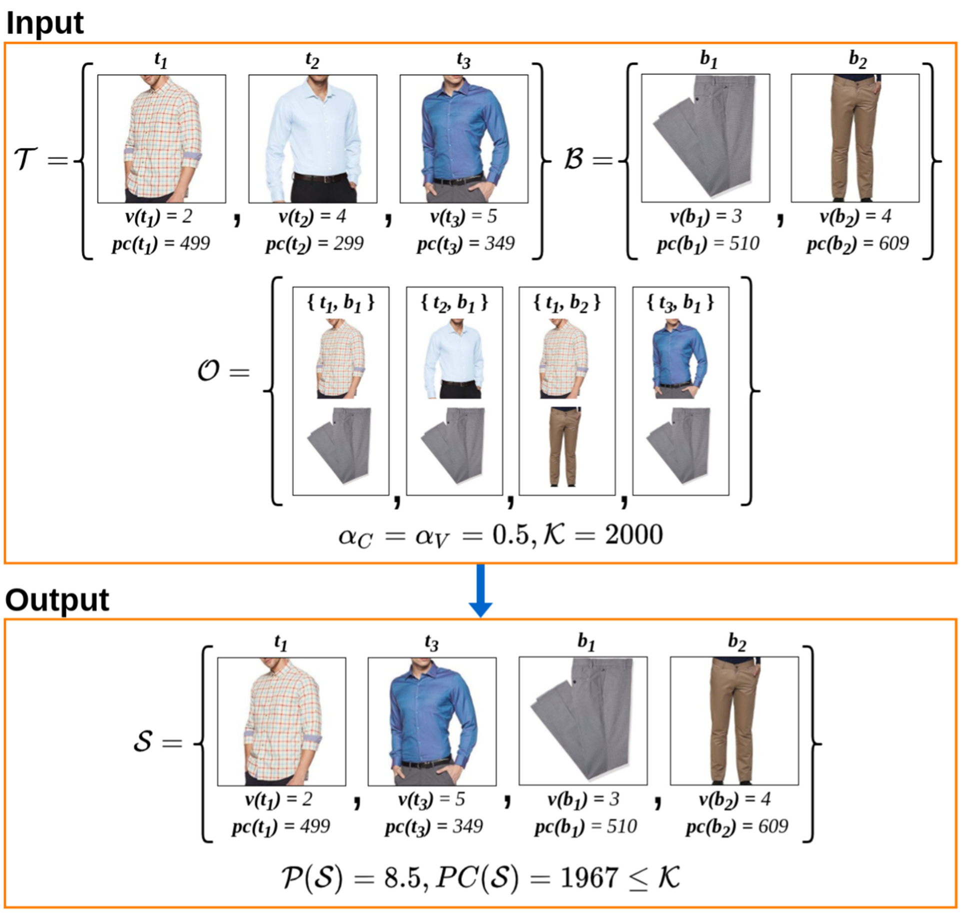


Fig. 1. Sample Input and Output of the *MOBCCWR* problem. Consider as Input a set of top-wear and bottom-wear items $\mathcal{T}=\{t_1,t_2,t_3\}$ and $\mathcal{B}=\{b_1,b2\}$, respectively, with price ($pc(x)$) and versatility score ($v(x)$) associated with each item $x\in\mathcal{T}\cup\mathcal{B}$, a collection of compatible outfit sets $\mathcal{O}=\{\{t_1,b_1\},\{t_2,b_1\},\{t_1,b_2\},\{t_3,b_1\}\}$, preference weight coefficients $\alpha_C=\alpha_V=0.5$ and budget $\mathcal{K}=2000$. The optimal Output is a subset $\mathcal{S}\subseteq\mathcal{T}\cup\mathcal{B}$, i.e., $\mathcal{S}=\{t_1,b_1,t_3,b_2\}$ having objective function value or total profit $\mathcal{P}(\mathcal{S})=8.5$ and total price $PC(\mathcal{S})=1967$.

Fashion outfits are mainly organized in two layers [34], commonly called *top-wear* and *bottom-wear*. Hence, we also consider tops and bottoms as the only components of an outfit. Figure 1 gives a high level pictorial view of the *MOBCCWR* problem with representative input and output. The input consists of a set of top-wears $\mathcal{T}=\{t_1,t_2,t_3\}$ and bottom-wears $\mathcal{B}=\{b_1,b_2\}$; a collection of compatible outfit sets $\mathcal{O}=\{\{t_1,b_1\},\{t_2,b_1\},\{t_1,b2\},\{t_3,b_1\}\}$; preference weights $\alpha_C=0.5$ and $\alpha_V=0.5$ corresponding to compatibility and versatility, respectively; and an overall shopping budget $\mathcal{K}=2000$. Each item $x\in\mathcal{T}\cup\mathcal{B}$ has an associated price and versatility score denoted by $pc(x)$ and $v(x)$, respectively. The objective is to find a subset $\mathcal{S}\in\mathcal{T}\cup\mathcal{B}$ such that total profit $\mathcal{P}(\mathcal{S})$ is maximized satisfying budget constraint $PC(\mathcal{S})\leq\mathcal{K}$. We provide detailed explanation of the notations in Section 3. As shown in Figure 1, the most profitable subset is $\mathcal{S}=\{t_1,t_3,b_1,b_2\}$ with the total profit $\mathcal{P}(\mathcal{S})$ and price as $PC(\mathcal{S})$ as 8.5 and 1967 ($\leq\mathcal{K}$), respectively.

The main contributions of our work can be summarized as follows:

- To the best of our knowledge, we are the first to formulate the multi-objective budget constrained capsule wardrobe recommendation—the *MOBCCWR* problem that considers user mentioned weighted preference for compatibility, versatility, and a maximum shopping budget for recommending a capsule wardrobe having a subset of compatible and versatile top-wear and bottom-wear items from a universal set of all top-wears and bottom-wears.
- We prove that the decision version of *MOBCCWR* is NP-Complete, propose a greedy heuristic algorithm based on graph theoretic approach, and analyze its time based complexity and approximation scheme. Our method efficiently finds a near-optimal solution for *MOBCCWR* in real time, which is otherwise intractable to solve.
- We employ both real and synthetic datasets for evaluating our approach and perform an in-depth analysis of the results.

The rest of the article is structured as follows. Section 2 briefly surveys related work. Section 3 introduces relevant notations and problem formulation. Section 4 presents formal definition of the *MOBCCWR* problem, followed by its complexity analysis. A greedy heuristic approach for solving the optimization problem of *MOBCCWR* is defined in Section 5. Experimental findings and discussions are presented in Sections 6 and 7, respectively. We conclude in Section 8 with supplemental tables and algorithms provided in Appendix A.

2 RELATED WORK

In recent years, recommender systems [5, 9–12] have become an essential tool in the e-commerce world that helps in providing the best shopping experience. A recommender system's primary task is to provide top-k recommendations from an extensive collection of items. Recommender systems are either content based or collaborative filtering based. Both have pros and cons. A hybrid alternative leverages the benefits of both versions. Now, collaborative filtering based methods are either user based or item based. The item based techniques are more efficient because modeling similarity among items is more intuitive than among users. The block-diagonal structure plays an essential role in top-N recommendations, capturing latent item groups or clusters, where the intra-cluster similarity among items is higher than the inter-cluster similarity. Chen et al. [12] propose a multi-task learning framework named *block-diagonal regularization* (BDR), where item clustering and item similarity are learned jointly to categorize items better.

Generally, recommender systems based on feature based collaborative filtering use Factorization Machines to effectively model feature interactions. Efficient **Feature Interaction Selection (FIS)** methods help filter out useless feature interactions, assuming that all users share the same feature interactions, which is not valid. Chen et al. [11] studied Personalized Feature Interaction Selection (P-FIS) for Factorization Machines and the propose Bayesian Personalized Feature Interaction (BP-FIS) mechanism by adopting the Bayesian Variable Selection theory to estimate the personalized interaction selection variables. BP-FIS replaces the widely used sparsity priors of Bayesian Variable Selection with hereditary spike and slab priors to assign non-zero probabilities to zero values for preserving hereditary relations.

An alternative way of incorporating feature interactions is through the use of **knowledge graphs (KGs)** [54] that make use of external information to provide superior recommendations. However, KG-based recommender systems mainly rely on **negative sampling (NS)** to optimize the KG embeddings and parameters of the recommender system. However, NS is not robust and fails to capture collaborative interactions among users, items, and entities. Chen et al. [5] propose a novel Jointly Non-Sampling learning model for Knowledge graph enhanced Recommendation (JNSKR), which employs an efficient (in terms of time complexity) joint learning framework that jointly models interactions (among users, items, and entities) and learns model parameters utilizing the entire training data without NS.

In the fashion domain, there are several interesting contributions on street to shop [33, 40], body shape [24, 26], and price and popularity [42] based recommender systems. The majority of the research is mainly focused on retrieval of similar or compatible items [9, 18, 23, 36, 39, 49, 50, 58].

Compatibility between item pairs depends on whether an item complements particular features of another item and vice versa. As a result, mutual attention between fashion image pairs [39] better models the matching relation between them. Another way of learning compatibility relationship between items in an outfit is by using a bi-directional LSTM that predicts the next item conditioned on the previous ones [22], where an outfit is considered as a sequence of items from top to bottom, followed by accessories, where each item acts as a timestep (similar to words in a sentence). This method is capable of recommending an item that matches existing components in a set to form a stylish outfit and generating an outfit with multi-modal (image/text) specification provided by a user. Some additional methods of learning compatibility between fashion items are found in other works [15, 21, 37, 57].

Li et al. [37] propose a new framework—Hierarchical Fashion Graph Network (HFGN)—that models a hierarchical structure consisting of three levels. The top level consists of users, the middle level is for outfits, and the bottom level comprises items. This kind of graph structure helps in capturing user-outfit interactions and outfit-item interactions following a message passing scheme of traditional graph neural networks [46]. Generally, outfit compatibility is determined using visual embeddings, which does not provide any explanation about the compatibility. Yang et al. [57] propose an Attribute-based Interpretable Compatibility that helps in reasoning compatibility between items in an outfit using fine-grain attributes. Attribute-based Interpretable Compatibility make use of attribute-based decision rule embedding and visual embeddings to produce the compatibility prediction score.

Existing methods try to model outfit compatibility via discrete item interactions without considering any try-on appearance of the items on a human body. Dong et al. [15] propose a Try-On-guided Compatibility Modeling scheme (TryOn-CM) that performs try-on appearance modeling and discrete item interaction modeling, which outperforms existing state-of-the-art baselines. Our work does not focus on prediction of compatibility between items. We assume that there is a set having all possible compatible outfits made from different top-wear and bottom-wear items.

In practice, a classical recommender system gives a list of recommendations where each recommendation is a single item [13, 19, 60]. For instance, Golubtsov et al. [19] conduct a survey to collect a user's preferred color type, age, price range, and preference of items (like/dislike). They use this data to propose a ladies-wear recommender system based on reinforcement learning that suggests top-k relevant single items to users.

Instead of single items, a recommender system can also suggest a package or a box full of items [55] based on user mentioned maximum total budget, where each item can have an associated value (rating) and a cost. In recent years, the approach of recommending items in bundles [2, 3, 17, 59, 62] has become very popular in the E-commerce industry. Ettl et al. [17] propose a framework that provides recommendations of personalized discounted product bundles to online shoppers according to user specified preferences. They consider a trade-off between profit maximization and efficient inventory management. Bai et al. [3] suggest a bundle generation framework that promotes both compatibility and diversity.

In the context of fashion, package or composite recommendation corresponds to recommendation of capsule wardrobes [14, 25], which is an emerging field that has not been explored much. Capsule wardrobes are either constructed from scratch by including a set of inter-compatible items to provide maximal mix-and-match outfits [25] or an existing wardrobe of a user is updated by adding and deleting some items [14]. Hsiao and Grauman [25] propose an algorithm for capsule wardrobe creation using a subset selection heuristic. They devise a sub-modular objective function to efficiently capture compatibility and versatility. Their objective function assigns equal importance to compatibility and versatility scores. However, they do not consider price of items or shopping budget. Marketing research acknowledges that price or budget has an effect on the purchase decision [7, 35]. But surprisingly, very few recommendation systems consider price [6, 47, 53, 61] in their frameworks. We carefully address these issues and formulate *MOBCCWR*.

We present *MOBCCWR* as a knapsack problem on the bipartite graph with a budget constraint and tunable weight coefficients for compatibility and versatility, which are user specified and they help in incorporating personalization. Khuller et al. [32] propose a greedy heuristic for the neighbor constrained graphical knapsack problem with profit and weights on nodes only and do not capture edge profit. In a recommender system, the response time is also an important factor. Customers cannot be kept waiting indefinitely. We propose a greedy heuristic that solves *MOBCCWR*, overcoming all preceding drawbacks of existing approaches. Our approach is able to efficiently find the final recommendation set (which is near-optimal) in real time.

3 PROBLEM DEFINITION

We first introduce the set of notations for precise formulation of the problem being addressed and its solution as follows:

- $\mathcal{T}$: Set of all top-wear items. Each item in $\mathcal{T}$ is represented as t and the i^{th} item as t_i, $0 \leq i \leq |\mathcal{T}| - 1$.
- $\mathcal{B}$: Set of all bottom-wear items. Each item in $\mathcal{B}$ is represented as b and the j^{th} item as b_j, $0 \leq j \leq |\mathcal{B}| - 1$.
- $\mathcal{O}$: Collection of all compatible outfit sets, where each element $o_k \in \mathcal{O}$ $(0 \leq k \leq |\mathcal{O}| - 1)$ is defined as $o_k = \{t, b\}$, denoting that the top-wear t is compatible with bottom-wear b.
- x: In general, we denote a top-wear or a bottom-wear item as x.
- OC: Set of all occasions.
- $v(x)$: Versatility score of item x, where $v(x) \in \{1, 2, \ldots, |OC|\}$ denotes the number of suitable occasions for x.
- $pc(x)$: Price of item x, where $pc(x) \in \mathbb{R}_{>0}$.
- α_C, α_V: User-defined preference weight coefficients for compatibility and versatility scores, respectively, such that $\alpha_C + \alpha_V = 1$, where $\alpha_C, \alpha_V \in [0, 1]$.
- $\mathcal{K}$: User-defined maximum shopping budget, where $\mathcal{K} \in \mathbb{R}_{>0}$.
- $\mathcal{S}$: Recommended capsule wardrobe having a subset of top-wear and bottom-wear items from $\mathcal{T} \cup \mathcal{B}$, i.e., $\mathcal{S} \subseteq \mathcal{T} \cup \mathcal{B}$.
- Compatibility function $c : \mathcal{T} \times \mathcal{B} \rightarrow \{0, 1\}$ indicates the compatibility score between a pair of top-wear and bottom-wear, i.e., $c(t, b) = 1,$ if $\{t, b\} \in \mathcal{O}$ else 0.
- Compatibility score $C(S)$ of items in any set S: $C(S) = \sum_{t \in S \cap \mathcal{T}, b \in S \cap \mathcal{B}} c(t, b)$.
- Versatility score $V(S)$ of items in any set S: $V(S) = \sum_{x \in S} v(x)$.
- Total price $PC(S)$ of items in any set S: $PC(S) = \sum_{x \in S} pc(x)$.
- Total profit $\mathcal{P}(S)$ of items in any set S: $\mathcal{P}(S) = \alpha_C C(S) + \alpha_V V(S)$.
- V_G, E_G: Vertex and edge set of any graph G.
- $G[V]$: Graph induced by the the vertex set V.

Generally, top-wear and bottom-wear items constitute the most important elements in any outfit set. So, we mainly consider two-layered outfits, each having a top-wear and a bottom-wear item. Let $\mathcal{T} \cup \mathcal{B}$ be the set of all top-wear and bottom-wear items, and $\mathcal{O}$ be the collection of all compatible outfit sets. Now, recommendation of a capsule wardrobe requires finding a subset $\mathcal{S} \subseteq \mathcal{T} \cup \mathcal{B}$

containing top-wear and bottom-wear items such that the total price $PC(\mathcal{S}) \leq \mathcal{K}$ and the total profit $\mathcal{P}(\mathcal{S}) = \alpha_C C(\mathcal{S}) + \alpha_V V(\mathcal{S})$ is maximum, where $\mathcal{K}$, α_C and α_V are user specified input parameters.[1]

As mentioned earlier in the list of notations, compatibility score $C(S)$ measures the overall compatibility between each pair of top-wear and bottom-wear items in the set $\mathcal{S}$. The versatility score $v(x)$ of an item x captures the number of suitable occasions for x. For example, a shirt may be suitable for both formal and semi-formal occasions, which makes its versatility score 2. Next, $V(\mathcal{S})$ is the summation of the versatility scores of all items in the set $\mathcal{S}$. Total profit $\mathcal{P}(\mathcal{S})$ is the weighted sum of $C(\mathcal{S})$ and $V(\mathcal{S})$, where α_C and α_V are the user specified corresponding weight coefficients. When $\alpha_C > \alpha_V$ ($\alpha_C < \alpha_V$), then compatibility (versatility) is favored over versatility (compatibility). One can also assign equal importance to compatibility and versatility by setting $\alpha_C = \alpha_V = 0.5$. We name this problem as the **multi-objective budget constrained capsule wardrobe recommendation (*MOBCCWR*)** problem.

As mentioned previously, we have two disjoint sets $\mathcal{T}$ and $\mathcal{B}$ and a collection of sets $\mathcal{O}$, where each element $\{t, b\} \in \mathcal{O}$ denotes the association between elements of $\mathcal{T}$ and $\mathcal{B}$, thereby satisfying the properties of a bipartite graph. So, we formulate the *MOBCCWR* problem as a bipartite graph with the objective of finding the most profitable sub-graph, satisfying certain constraints. The vertices or nodes of the graph correspond to items, and edges denote compatibility between items.[2] We formally define the problem as follows.

Definition 3.1 (Multi-Objective Budget Constrained Capsule Wardrobe Recommendation (MOBCCWR) Problem). Given a bipartite graph $G(\mathcal{T} \cup \mathcal{B}, \mathcal{O})$ with two disjoint sets of vertices (or items) $\mathcal{T}$ and $\mathcal{B}$ (corresponding to top-wears and bottom-wears, respectively), edge set $\mathcal{O} \subseteq \mathcal{T} \times \mathcal{B}$ (corresponding to set of outfits), i.e., $\mathcal{O} = \{\{t, b\} : c(t, b) = 1, t \in \mathcal{T}, b \in \mathcal{B}\}$, where each item $x \in \mathcal{T} \cup \mathcal{B}$ is associated with a price $pc(x)$ and versatility score $v(x)$, preference weights for compatibility and versatility as α_C and α_V, respectively, and a maximum budget $\mathcal{K}$, find a 1-neighbor subset of items $\mathcal{S} \subseteq \mathcal{T} \cup \mathcal{B}$, such that the total price of all items in $\mathcal{S}$ is at most $\mathcal{K}$ and $\mathcal{P}(\mathcal{S})$ is maximum, i.e., to

$$\text{maximize } \mathcal{P}(\mathcal{S})$$

$$\text{s.t. } PC(\mathcal{S}) \leq \mathcal{K},$$

where each vertex x in G is either a top-wear ($\in \mathcal{T}$) or a bottom-wear ($\in \mathcal{B}$) item, $\mathcal{S}$ is a 1-neighbor set of vertices [4] in G, which are not isolated vertices. We assume that α_C and α_V are predefined such that $\alpha_C + \alpha_V = 1$, $pc(t') + pc(b') \leq \mathcal{K} \leq PC(\mathcal{T} \cup \mathcal{B})$ with $\{t', b'\} \in \mathcal{O}$, where $t' \in \mathcal{T}$ and $b' \in \mathcal{B}$ as the cheapest top-wear and bottom-wear items, respectively. The permitted range of $\mathcal{K}$ implies that the minimum budget can be no less than the price of the cheapest outfit in $\mathcal{O}$ and the maximum can be no more than the total price of all the items in $\mathcal{T} \cup \mathcal{B}$.

As an illustrative example of *MOBCCWR*, let us consider Figure 2. We construct a bipartite graph $G(\mathcal{T} \cup \mathcal{B}, \mathcal{O})$ with two disjoint sets of vertices $\mathcal{T}$ and $\mathcal{B}$ corresponding to top-wears and bottom-wears, respectively. Each edge $\{t, b\}$ in the edge set $\mathcal{O}$ corresponds to a compatible outfit. Here, the values of $\mathcal{T}$, $\mathcal{B}$, $\mathcal{O}$, α_C, α_V, and $\mathcal{K}$ are the same as in Figure 1. We formulate the objective of *MOBCCWR* as the task of finding a 1-neighbor [4] subset $\mathcal{S} \subseteq \mathcal{T} \cup \mathcal{B}$ of vertices (or items) in G that jointly maximize compatibility and versatility scores, i.e., the total profit $\mathcal{P}(\mathcal{S})$ by considering the user mentioned preference weights α_C, α_V (corresponding to compatibility and versatility, respectively) and satisfying the maximum shopping budget constraint $PC(\mathcal{S}) \leq \mathcal{K}$.

[1]Since, $\alpha_C + \alpha_V = 1$, a user needs to specify any one of them. For example, if α_V is known, then $\alpha_C = (1 - \alpha_V)$.
[2]We interchangeably use the terms *vertex*, *node*, and *item* for referring to a fashion item.

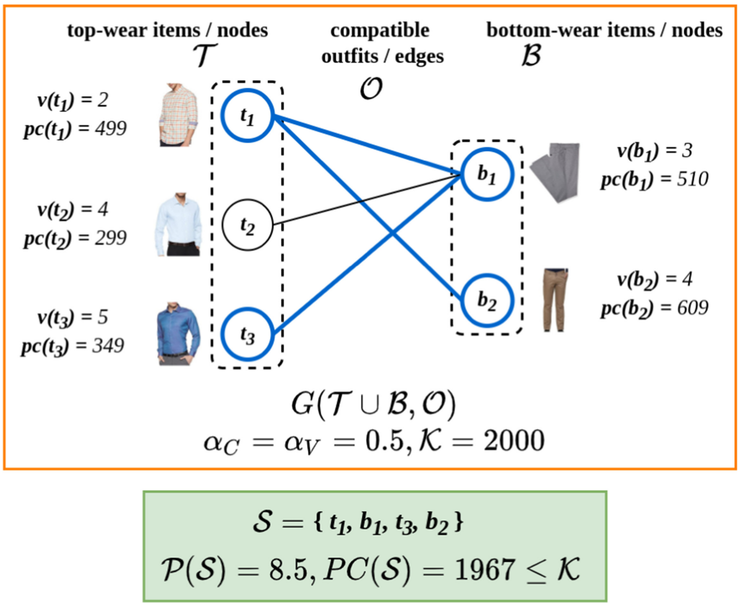

top-wear items / nodes
$\mathcal{T}$
compatible outfits / edges
$\mathcal{O}$
bottom-wear items / nodes
$\mathcal{B}$
v(t1) = 2
pc(t1) = 499
v(t2) = 4
pc(t2) = 299
v(t3) = 5
pc(t3) = 349
t1
t2
t3
b1
b2
v(b1) = 3
pc(b1) = 510
v(b2) = 4
pc(b2) = 609
$G(\mathcal{T} \cup \mathcal{B}, \mathcal{O})$
$\alpha_C = \alpha_V = 0.5, \mathcal{K} = 2000$
$\mathcal{S} = \{ t_1, b_1, t_3, b_2 \}$
$\mathcal{P}(\mathcal{S}) = 8.5, PC(\mathcal{S}) = 1967 \leq \mathcal{K}$

Fig. 2. Sample instance of the *MOBCCWR* problem, where the input is in the form of a bipartite graph $G(\mathcal{T} \cup \mathcal{B}, \mathcal{O})$ having two disjoint sets of vertices $\mathcal{T} = \{t_1, t_2, t_3\}$ (top-wears), $\mathcal{B} = \{b_1, b_2\}$ (bottom-wears) with price ($pc(x)$) and versatility score ($v(x)$) associated with each item $x \in \mathcal{T} \cup \mathcal{B}$, edge set $\mathcal{O} = \{\{t_1, b_1\}, \{t_2, b_1\}, \{t_1, b_2\}, \{t_3, b_1\}\}$ corresponding to compatible outfits, preference weights $\alpha_C = \alpha_V = 0.5$, and budget $\mathcal{K} = 2000$. The recommended 1-neighbor subset of items is $\mathcal{S} = \{t_1, b_1, t_3, b_2\}$ having total profit $\mathcal{P}(\mathcal{S})$ and price $PC(\mathcal{S})$ as 8.5 and 1967, respectively. The chosen items and corresponding compatible outfits are highlighted in blue.

Table 1. Details of All Possible Valid Capsule Wardrobes Considering $\mathcal{T}, \mathcal{B}, \mathcal{O}, \alpha_C, \alpha_V$, and $\mathcal{K}$ from Figure 2

$\mathcal{S}$	$PC(\mathcal{S})$	$C(\mathcal{S})$	$V(\mathcal{S})$	$\mathcal{P}(\mathcal{S}) = \alpha_C C(\mathcal{S}) + \alpha_V V(\mathcal{S})$ ($\alpha_C = \alpha_V = 0.5$)
$\{t_1, b_1\}$	1009	1	5	3.0
$\{t_1, b_2\}$	1108	1	6	3.5
$\{t_2, b_1\}$	809	1	7	4.0
$\{t_3, b_1\}$	859	1	8	4.5
$\{t_1, b_1, t_2\}$	1308	2	9	5.5
$\{t_1, b_1, t_3\}$	1358	2	10	6.0
$\{t_2, b_1, t_3\}$	1158	2	12	7.0
$\{t_1, b_1, t_2, b_2\}$	1917	3	13	8.0
$\{t_1, b_1, t_3, b_2\}$	1967	3	14	8.5
$\{t_1, b_1, t_2, t_3\}$	1657	3	14	8.5

Table 1 shows all possible valid capsule wardrobes with total price, versatility score, compatibility score, and objective function value for $\alpha_C = 0.5$, $\alpha_V = 0.5$, and budget $\mathcal{K} = 2000$. From Table 1, we observe that both sets $\{t_1, b_1, t_3, b_2\}$ and $\{t_1, b_1, t_2, t_3\}$ have the highest objective function value of 8.5. So, both are candidate capsule wardrobes for recommendation. We break the tie by recommending the set having greater price, i.e., $\{t_1, b_1, t_3, b_2\}$ with price 1967.

4 COMPLEXITY ANALYSIS

In this section, we prove that the *MOBCCWR* problem is NP-Complete by showing a decision version of *MOBCCWR* to be in NP and proving that it is NP-Hard by reducing a decision version of the 0–1 Knapsack (BKS) problem [31] in polynomial time to the decision version of the *MOBCCWR* problem. To develop the proof, we first formulate a decision version of *MOBCCWR*, then formally define the *BKS* problem and the decision version of the *BKS* (*D-BKS*) problem.

Definition 4.1 (Decision version of the MOBCCWR (D-MOBCCWR) Problem). Given a bipartite graph $G(\mathcal{T} \cup \mathcal{B}, \mathcal{O})$ with two disjoint sets of vertices (or items) $\mathcal{T}$ and $\mathcal{B}$ (corresponding to set of top-wear and bottom-wear items, respectively), edge set $O \subseteq \mathcal{T} \times \mathcal{B}$ (corresponding to set of outfits), i.e., $\mathcal{O} = \{(t_i, b_j) : c(t_i, b_j) = 1, t_i \in \mathcal{T}, b_j \in \mathcal{B}\}$, where each item $x \in \mathcal{T} \cup \mathcal{B}$ is associated with a price $pc(x)$ and versatility score $v(x)$, preference weights for compatibility and versatility as α_C and α_V, respectively, a maximum budget $\mathcal{K}$, and a positive integer $\mathcal{W}$, does there exist a 1-neighbor subset of items $\mathcal{S} \subseteq \mathcal{T} \cup \mathcal{B}$ such that

$$PC(\mathcal{S}) \leq \mathcal{K} \tag{1}$$

and

$$\mathcal{P}(\mathcal{S}) \geq \mathcal{W}? \tag{2}$$

Definition 4.2 (0–1 Knapsack (BKS) Problem). Given a finite set A having n items with $p(a_i)$ as the profit, $w(a_i)$ as the weight of each item a_i, and a knapsack with capacity C (where $p(a_i)$, $w(a_i)$, and C are positive integers), find a subset of items $J \subseteq A$ such that total weight of the selected items is at most C and total profit is maximum, i.e., to

$$\text{maximize} \sum_{a_j \in J} p(a_j)$$

$$\text{s.t.} \sum_{a_j \in J} w(a_j) \leq C.$$

Definition 4.3 (Decision version of the BKS (D-BKS) Problem). Given a finite set A having n items with $p(a_i)$ as the profit, $w(a_i)$ as the weight of each item a_i, and a knapsack with capacity C (where $p(a_i)$, $w(a_i)$, and C are positive integers) and a positive integer $\mathcal{M}$, does there exist a subset $J \subseteq A$ such that

$$\sum_{a_j \in J} w(a_j) \leq C \tag{3}$$

and

$$\sum_{a_j \in J} p(a_j) \geq \mathcal{M}? \tag{4}$$

Theorem 1. *D-MOBCCWR is NP-Complete.*

Proof. Given a bipartite graph $G(\mathcal{T} \cup \mathcal{B}, \mathcal{O})$, price $pc(\cdot)$ and versatility scores $v(\cdot)$ of each node in $\mathcal{T} \cup \mathcal{B}$, preference weights α_C and α_V, a 1-neighbor subset $\mathcal{S} \subseteq \mathcal{T} \cup \mathcal{B}$, and a positive integer $\mathcal{W}$, condition (1) can be verified in polynomial time by computing the total price $PC(\mathcal{S})$ of all the items in $\mathcal{S}$ and comparing it with $\mathcal{K}$. We can also verify condition (2) in polynomial time by computing the total profit $\mathcal{P}(\mathcal{S})$ and comparing it with $\mathcal{W}$. Hence, *D-MOBCCWR* is in NP.

Now, we prove that *D-BKS* $\leq_p$ *D-MOBCCWR*, i.e., there is a polynomial time reduction from any instance of *D-BKS* to some instance of *D-MOBCCWR*. It suffices to show that there exists a polynomial time reduction $\mathcal{F}(\cdot)$ such that $\mathcal{F}(Y)$ is a "Yes" instance to the *D-MOBCCWR* problem if and only if Y is a "Yes"instance to the *D-BKS* problem. Suppose we are given a set $A = \{a_1, a_2, \ldots, a_n\}$, with $p(a_i) \in \mathbb{Z}_{>0}$ and $w(a_i) \in \mathbb{Z}_{>0}$ as the profit and weight, respectively, of each item $a_j \in A$, knapsack capacity $C \in \mathbb{Z}_{>0}$, and integer $\mathcal{M} \in \mathbb{Z}_{>0}$ for any instance Y of the *D-BKS* problem. From the given instance Y of *D-BKS*, an instance $\mathcal{F}(Y) = <G(\mathcal{T} \cup \mathcal{B}, \mathcal{O}), \alpha_C, \alpha_V, \mathcal{K}, \mathcal{W} >$ of *D-MOBCCWR* can be constructed in polynomial time, as follows: for each item $a_i \in A$, there is an outfit $o_i = \{t_i, b_i\}$ in $\mathcal{O}$ with $v(t_i) = \epsilon_i$, $v(b_i) = p(a_i) - \epsilon_i$ and $pc(t_i) = \eta_i$, $pc(b_i) = w(a_i) - \eta_i$, where $0 < \epsilon_i < p(a_i)$, $0 < \eta_i < w(a_i)$ and $|\mathcal{T}| = |\mathcal{B}| = |\mathcal{O}| = |A| = n$. Any two outfits $o_i, o_j \in \mathcal{O}$ are disjoint, i.e., $o_i \cap o_j = \phi$. Finally, we set $\mathcal{K} = C, \mathcal{W} = \mathcal{M}, \alpha_C = 0, \alpha_V = 1$. Now, if the answer for Y is "Yes"for the *D-BKS* problem, i.e., there exists a subset $J \subseteq A$ such that $\sum_{a_j \in J} w(a_j) \leq C$ and $\sum_{a_j \in J} p(a_j) \geq \mathcal{M}$, then for $\mathcal{F}(Y)$, we take $\mathcal{S} = J$. Thus, we have the following:

(1) $PC(\mathcal{S}) = PC(\mathcal{S} \cap \mathcal{T}) + PC(\mathcal{S} \cap \mathcal{B}) = \sum_{a_j \in J} w(a_j) \leq C = \mathcal{K}$,
(2) $\mathcal{P}(\mathcal{S}) = \alpha_C C(\mathcal{S}) + \alpha_V V(\mathcal{S}) = V(S) = V(\mathcal{S} \cap \mathcal{T}) + V(\mathcal{S} \cap \mathcal{B}) = \sum_{a_j \in J} p(a_j) \geq \mathcal{M} = \mathcal{W}$.

This shows that the answer for $\mathcal{F}(Y)$ is also "Yes."

Conversely, suppose that the answer for $\mathcal{F}(Y)$ is "Yes"for the *D-MOBCCWR* problem, i.e., there exists a subset $\mathcal{S} \subseteq \mathcal{T} \cup \mathcal{B}$, satisfying conditions (1) and (2). In case of Y, from conditions (1) and (2), we have $\sum_{a_j \in J} w(a_j) = PC(\mathcal{S} \cap \mathcal{T}) + PC(\mathcal{S} \cap \mathcal{B}) = PC(\mathcal{S}) \leq \mathcal{K} = C$ and $\sum_{a_j \in J} p(a_j) = V(\mathcal{S} \cap \mathcal{T}) = V(\mathcal{S} \cap \mathcal{T}) + V(\mathcal{S} \cap \mathcal{B}) = V(\mathcal{S}) = \alpha_C C(\mathcal{S}) + \alpha_V V(\mathcal{S}) \geq \mathcal{W} = \mathcal{M}$. Thus, Y is the "Yes"instance of the *D-BKS* problem. Hence, *D-MOBCCWR* is NP-Hard. Since *D-MOBCCWR* is both in NP and is NP-Hard, it is NP-Complete. □

5 HEURISTIC FOR SOLVING *MOBCCWR*

As shown in Section 4, *MOBCCWR* is an NP-Complete problem. Therefore, it is not likely that *MOBCCWR* can be solved deterministically in polynomial time. So, we propose a **Greedy Bipartite Knapsack Graph (GREEDY-BKG)** algorithm, inspired by the greedy 1-neighbor algorithm [4] to solve *MOBCCWR*. Given a bipartite graph $G(\mathcal{T} \cup \mathcal{B}, \mathcal{O})$ with two disjoint sets of vertices (or items) $\mathcal{T}$ and $\mathcal{B}$, edge set $\mathcal{O} = \{\{t, b\} : c(t, b) = 1, t \in \mathcal{T}, b \in \mathcal{B}\}$, where each item $x \in \mathcal{T} \cup \mathcal{B}$ is associated with a price $pc(x)$ and versatility score $v(x)$, preference weights as α_C for compatibility, α_V for versatility, and a maximum budget $\mathcal{K}$, GREEDY-BKG returns a 1-neighbor subset $\mathcal{S} \subseteq \mathcal{T} \cup \mathcal{B}$ such that $PC(\mathcal{S}) \leq \mathcal{K}$ and $\mathcal{P}(\mathcal{S})$ is maximum.

5.1 Greedy Solution

Algorithm 1 illustrates the GREEDY-BKG heuristic for solving *MOBCCWR*. At first, we make a copy G' and K of the input graph G and budget $\mathcal{K}$ (Line 2), respectively, then initialize each of Z and $\mathcal{S}$ with empty sets, and i (iteration number) with 1 (Line 3). We denote the vertex and edge set of the graph G' as $V_{G'}$ and $E_{G'}$, respectively. Next, we set *profit*(x) of each item $x \in \mathcal{T} \cup \mathcal{B}$ by multiplying preference weight α_V with the versatility score $v(x)$ (Lines 4 and 5). The main computation is done in four steps between Lines 6 and 27, where Algorithms 2 and 3 are used as sub-routines:

Step 1: At the start of each iteration (Line 7), we compute the best ratio edge in G' and retrieve the corresponding items S_i and associated profit P_i using Algorithm 2.
Step 2: At Line 8, we check whether there exists an item $x \in Z$ with price within K or not. If the condition at Line 8 is satisfied, we retrieve an item z from Z that has the maximum profit to price ratio such that price $pc(z) \leq K$ (Line 9). Now, if S_i is non-empty (Line 10) and the profit to price ratio of z is greater than the profit to price ratio of the items present in S_i (Line 11) or if S_i is empty (Line 13), then we replace all the items in S_i with z and update P_i with the profit of z (Lines 12 and 14).
Step 3: In this step, we first update G' by removing vertices from $V_{G'}$ corresponding to items in S_i (Line 15). When a particular vertex is removed from $V_{G'}$, the corresponding incident

ALGORITHM 1: Greedy Bipartite Knapsack Graph

```
1:  procedure GREEDY-BKG($G(\mathcal{T} \cup \mathcal{B}, \mathcal{O}), \alpha_C, \alpha_V, \mathcal{K}$)
2:      $G' \leftarrow G, K \leftarrow \mathcal{K}$    ▷ making a copy of the input graph $G$ and budget $\mathcal{K}$, respectively
3:      $Z \leftarrow \phi, \mathcal{S} \leftarrow \phi, i \leftarrow 1$
4:      for each item $x \in \mathcal{T} \cup \mathcal{B}$ do    ▷ iterating over all items
5:          $\mathit{profit}(x) \leftarrow \alpha_V v(x)$    ▷ setting profit of each item
6:      while TRUE do
7:          $\{S_i, P_i\} \leftarrow$ BEST-RATIO-EDGE($G', \alpha_C, \alpha_V, K$) ▷ getting best ratio items, profit from $G'$
8:          if $\exists x \in Z$ such that $pc(x) \leq K$ then
9:              $z \leftarrow \underset{x \in Z \wedge pc(x) \leq K}{\arg\max} \frac{\mathit{profit}(x)}{pc(x)}$    ▷ retrieving item with best ratio in $Z$
10:             if $S_i \neq \phi$ then
11:                 if $\frac{\mathit{profit}(z)}{pc(z)} > \frac{P_i}{PC(S_i)}$ then  ▷ comparing profit to price ratio of $z$ with items in $S_i$
12:                     $S_i \leftarrow \{z\}, P_i \leftarrow \mathit{profit}(z)$    ▷ updating $S_i$ and $P_i$
13:             else
14:                 $S_i \leftarrow \{z\}, P_i \leftarrow \mathit{profit}(z)$    ▷ updating $S_i$ and $P_i$
15:         $G' \leftarrow G'[V_{G'} \backslash S_i]$    ▷ updating $G'$ by removing vertices and edges
16:         $\mathcal{S} \leftarrow \mathcal{S} \cup S_i$    ▷ adding items of $S_i$ in $\mathcal{S}$
17:         $K \leftarrow K - PC(S_i)$    ▷ decrementing $K$ by the total price of items in $S_i$
18:         for each node $x \in S_i$ do
19:             for each node $y \in N_G(x) \cap V_{G'}$ do
20:                 $\mathit{profit}(y) \leftarrow \mathit{profit}(y) + \alpha_C$    ▷ updating profit of items that are not yet selected
21:         $Z \leftarrow N_G^-(\mathcal{S})$    ▷ updating $Z$ with set of items not in $\mathcal{S}$ but that have a neighbor in $\mathcal{S}$
22:         if $S_i = \phi$ then
23:             break    ▷ breaking out of the loop when $S_i$ is empty
24:         $i \leftarrow i + 1$
25:     $S_e \leftarrow$ BEST-PSEUDO-PROFIT-NODES($G, \alpha_C, \alpha_V, \mathcal{K}$)
26:     if $\mathcal{P}(S_e) > \mathcal{P}(\mathcal{S})$ then
27:         $\mathcal{S} \leftarrow S_e$
28:     return $\mathcal{S}$
```

edge is also dropped from $E_{G'}$. Then we add items of S_i in $\mathcal{S}$ (Line 16) and decrement K by the total price of items in S_i (Line 17). Now, all the edges that are incident on the items of S_i in the original graph G are dropped from G' at Line 15. So, for each item x in S_i (Line 18), we iterate over all the neighboring items y belonging to $N_G(x) \cap V_{G'}$, where $N_G(x) = \{y : \{x, y\} \in \mathcal{O}\}$ (Line 19). We update the corresponding profit of y by adding α_C times the unit compatibility score associated with each edge present in the original input graph G (Line 20). At Line 21, we update Z with the set of items in $N_G^-(\mathcal{S}) = \{x : \{x, y\} \in \mathcal{O}, x \notin \mathcal{S} \wedge y \in \mathcal{S}\}$ and break out of the outer while loop (Line 23) when S_i is empty (Line 22). We keep track of the iteration number by incrementing i at the end of the while loop (Line 24).

Initially, the graph G' is a 1-neighbor undirected graph. But after updation at Line 15, it may not remain so, i.e., there may exist some vertices (or items) in G' with degree zero. We call such items as *singleton* items, which are stored in Z. However, these singleton items have a potential of improving the final solution when added to the set $\mathcal{S}$. Therefore, at every iteration, we try to add items from Z in $\mathcal{S}$.

Step 4: In case the budget is low, the items present in $\mathcal{S}$ at the end of the while loop do not produce good results. So, we use Algorithm 3 (Line 25) to retrieve the most

ALGORITHM 2: Best Profit to Price Ratio Edge

```
procedure BEST-RATIO-EDGE($G', \alpha_C, \alpha_V, K$)
    $Q \leftarrow$ queue containing all the edges of $G'$ sorted in descending order based on $\frac{profit}{price}$ ratio
    $S^* \leftarrow \phi, P^* \leftarrow 0$
    while $Q \neq \phi$ do                                  ▹ iterating until $Q$ becomes empty
        $\{t, b\} \leftarrow dequeue(Q)$                     ▹ obtaining an edge $\{t, b\}$ from $Q$
        if $PC(\{t, b\}) \leq K$ then   ▹ checking whether total price of $t$ and $b$ is less than $K$ or not
            $S^* \leftarrow \{t, b\}$                        ▹ updating $S^*$ with items in best ratio edge
            $P^* \leftarrow \mathcal{P}(S^*)$                ▹ updating $P^*$ with the total profit of $S^*$
            break                                           ▹ breaking out of the loop
    return $\{S^*, P^*\}$   ▹ returning items present in best ratio edge along with associated profit
```

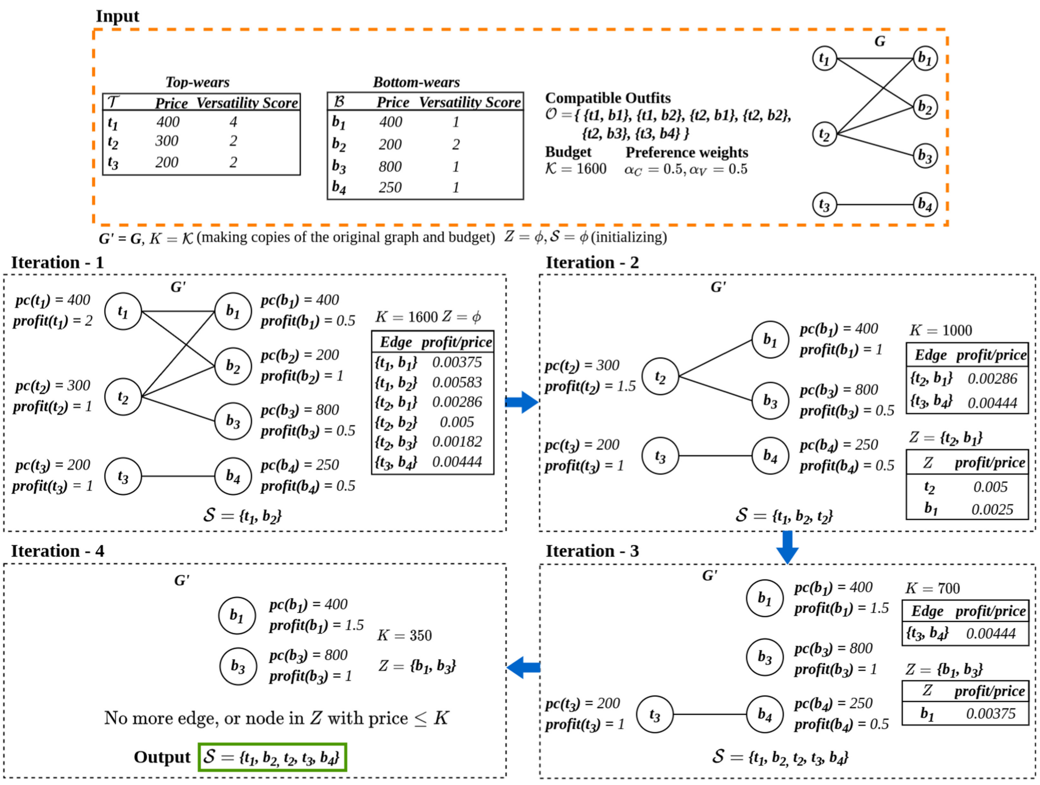
ALGORITHM 3: Best Pseudo Profit Nodes

```
procedure BEST-PSEUDO-PROFIT-NODES($G, \alpha_C, \alpha_V, K$)
    $S_e \leftarrow \phi, P_e \leftarrow 0$
    for each edge $\{t, b\}$ in $\mathcal{O}$ do              ▹ iterating over all the edges in the graph $G$
        if $pc(t) + pc(b) > K$ then               ▹ comparing total price of $t$ and $b$ with budget $K$
            continue
        $S^* \leftarrow \{t, b\}$                            ▹ initializing $S^*$ with the edge $\{t, b\}$
        $P^* \leftarrow \mathcal{P}(S^*)$                    ▹ initializing $P^*$ with the total profit of $S^*$
        $Q \leftarrow$ queue of all 1-neighbor vertices of $t$ and $b$, sorted in ascending order of price
        while $Q \neq \phi$ do                              ▹ iterating until $Q$ becomes empty
            $x \leftarrow dequeue(Q)$                        ▹ obtaining an item $x$ from Q
            if $PC(S^* \cup \{x\}) > K$ then     ▹ comparing total price of items in $S^*$ and $x$ with $K$
                break
            $S^* \leftarrow S^* \cup \{x\}, P^* \leftarrow P^* + \alpha_C$   ▹ adding item $x$ to $S^*$ and incrementing $P^*$ with $\alpha_C$
        if $P_e < P^*$ then                                 ▹ comparing $P_e$ with $P^*$
            $S_e \leftarrow S^*, P_e \leftarrow P^*$         ▹ updating $S_e$ with $S^*$ and $P_e$ with $P^*$
    return $S_e$
```

profitable smaller set of items S_e and compare the total profit of the items in S_e with that of $\mathcal{S}$ (Line 26). If the condition at Line 26 is satisfied, we update $\mathcal{S}$ with S_e (Line 27) and finally return the 1-neighbor subset $\mathcal{S} \subseteq \mathcal{T} \cup \mathcal{B}$ as the output (Line 28).

5.1.1 Retrieval of Items from Best Ratio Edges. Algorithm 2 shows the steps for retrieving items present in the edge that have maximum profit to price ratio. At first, Q (FIFO queue) is initialized with all the edges of input graph G', after sorting in descending order based on the profit to price ratio of the edges (Line 2). Profit of an edge $\{t, b\}$ is computed as the sum of the individual profits of t, b (as shown in Line 5, Algorithm 1) and α_C times the unit compatibility score of the edge $\{t, b\}$. At Line 3, we initialize S^* and P^* with an empty set and zero, respectively. Now, for every edge $\{t, b\}$ fetched from Q (Line 5), we check whether the total price of t and b is within K or not (Line 6). If the condition at Line 6 is satisfied, we update S^* with the items of the best ratio edge $\{t, b\}$ (Line 7) and also update P^* with the total profit of the edge $\{t, b\}$ (Line 8). At Line 9, we break out of the loop and return S^* and P^* as the output (Line 10).

5.1.2 Retrieval of Items from Best Pseudo Profit Nodes. In case of a low shopping budget, the 1-neighbor subset set $\mathcal{S}$ computed after execution of the while loop in Algorithm 1 may not contain


Input
G
Compatible Outfits
$\mathcal{O}$ = { {t1, b1}, {t1, b2}, {t2, b1}, {t2, b2}, {t2, b3}, {t3, b4} }
Budget
$\mathcal{K}$ = 1600
Preference weights
$\alpha_C = 0.5, \alpha_V = 0.5$
$G' = G$, $K = \mathcal{K}$ (making copies of the original graph and budget) $Z = \phi, \mathcal{S} = \phi$ (initializing)
Iteration - 1
G'
pc(t1) = 400 profit(t1) = 2
pc(t2) = 300 profit(t2) = 1
pc(t3) = 200 profit(t3) = 1
pc(b1) = 400 profit(b1) = 0.5
pc(b2) = 200 profit(b2) = 1
pc(b3) = 800 profit(b3) = 0.5
pc(b4) = 250 profit(b4) = 0.5
K = 1600 Z = ϕ
$\mathcal{S}$ = {t1, b2}
Iteration - 2
G'
pc(t2) = 300 profit(t2) = 1.5
pc(t3) = 200 profit(t3) = 1
pc(b1) = 400 profit(b1) = 1
pc(b3) = 800 profit(b3) = 0.5
pc(b4) = 250 profit(b4) = 0.5
K = 1000
Z = {t2, b1}
$\mathcal{S}$ = {t1, b2, t2}
Iteration - 3
G'
pc(b1) = 400 profit(b1) = 1.5
pc(b3) = 800 profit(b3) = 1
pc(t3) = 200 profit(t3) = 1
pc(b4) = 250 profit(b4) = 0.5
K = 700
Z = {b1, b3}
$\mathcal{S}$ = {t1, b2, t2, t3, b4}
Iteration - 4
G'
pc(b1) = 400 profit(b1) = 1.5
pc(b3) = 800 profit(b3) = 1
K = 350
Z = {b1, b3}
No more edge, or node in Z with price ≤ K
Output $\mathcal{S}$ = {t1, b2, t2, t3, b4}

Top-wears

$\mathcal{T}$	Price	Versatility Score
t_1	400	4
t_2	300	2
t_3	200	2

Bottom-wears

$\mathcal{B}$	Price	Versatility Score
b_1	400	1
b_2	200	2
b_3	800	1
b_4	250	1

Iteration - 1

Edge	profit/price
{t1, b1}	0.00375
{t1, b2}	0.00583
{t2, b1}	0.00286
{t2, b2}	0.005
{t2, b3}	0.00182
{t3, b4}	0.00444

Iteration - 2

Edge	profit/price
{t2, b1}	0.00286
{t3, b4}	0.00444

Z	profit/price
t2	0.005
b1	0.0025

Iteration - 3

Edge	profit/price
{t3, b4}	0.00444

Z	profit/price
b1	0.00375

Fig. 3. Example showing the working of the GREEDY-BKG algorithm using an instance of *MOBCCWR*.

the most profitable items. So, we refine the results using Algorithm 3. As shown in Algorithm 3, we first initialize S_e and P_e with empty set and zero, respectively (Line 2). Now, for each edge $\{t, b\} \in \mathcal{O}$ (Line 3), we compare the total price of t and b with budget $\mathcal{K}$ (Line 4). If the condition at Line 4 is not satisfied, we initialize S^* with the edge $\{t, b\}$ (Line 6) and P^* with the total profit of S^*. Profit of the edge $\{t, b\}$ is computed in the same way as shown in Line 8 of Algorithm 2. At Line 8, Q (FIFO queue) is initialized with the set of all 1-neighbor vertices of both t and b, sorted in ascending order of price. Next, we iterate over all the items in Q (Line 9). For each item x fetched from Q, if the total price of the items in S^* and x is within $\mathcal{K}$, we add item x to S^* and increment P^* by α_C (Line 13). Finally, we update S_e with S^* and P_e with P^* (Line 15) if the condition at Line 14 is satisfied and return S_e at Line 16.

Example 5.1. Consider the example shown in Figure 3. We have top-wears $\mathcal{T} = \{t_1, t_2, t_3\}$, bottom-wears $\mathcal{B} = \{b_1, b_2, b_3, b_4\}$, and compatible outfits $\mathcal{O} = \{\{t_1, b_1\}, \{t_1, b_2\}, \{t_2, b_1\}, \{t_2, b_2\}, \{t_2, b_3\}, \{t_3, b_4\}\}$. Price and versatility score of items are $pc(t_1) = 400$, $v(t_1) = 4$, $pc(t_2) = 300$, $v(t_2) = 2$, $pc(t_3) = 200$, $v(t_3) = 2$, $pc(b_1) = 400$, $v(b_1) = 1$, $pc(b_2) = 200$, $v(b_2) = 2$, $pc(b_3) = 800$, $v(b_3) = 1$, and $pc(b_4) = 250$, $v(b_4) = 1$. The input bipartite graph is $G(\mathcal{T} \cup \mathcal{B}, \mathcal{O})$. The objective is to find a 1-neighbor subset of vertices (or items) $\mathcal{S} \subseteq \mathcal{T} \cup \mathcal{B}$, satisfying user mentioned preference weights $\alpha_C = \alpha_V = 0.5$ and budget $\mathcal{K} = 1600$. G' and K are copies of the input graph G and the budget $\mathcal{K}$. Z, $\mathcal{S}$, and i are variables that are suitably initialized.

Iteration-1. In the first iteration, BEST-RATIO-EDGE returns $S_1 = \{t_1, b_2\}$ and $P_1 = \frac{2+1+0.5}{400+200} = 0.00583$ as the edge $\{t_1, b_2\}$ has highest profit to price ratio among all edges whose total price is less than K. Since Z and $\mathcal{S}$ are empty, $\mathcal{S} = S_1$ and $K = 1600 - (400 + 200) = 1000$. Now we remove vertices t_1, b_2 and edge $\{t_1, b_2\}$ from the graph G'. Next, we update profit of remaining items, i.e., t_2, b_1 that are 1-neighbor of the items in S_1, but not yet added to $\mathcal{S}$ to 1.5 and 1, respectively. At the end of first iteration, $Z = \{t_2, b_1\}$ and $i = 2$.

Iteration-2. BEST-RATIO-EDGE returns the edge $\{t_3, b_4\}$ whose profit to price ratio is 0.00444. But the item $t_2 \in Z$ has a higher profit to price ratio of 0.005. So, we add $\{t_2\}$ to S_2, assign 0.005 to P_2, and remove t_2 from G'. Now, we are left with $K = 1000 - 300 = 700$ and add items of S_2 to $\mathcal{S}$, i.e., $\mathcal{S} = \{t_1, b_2, t_2\}$. Before the end of the second iteration, we update profit of b_1 and b_3 to 1.5 and 1, respectively; currently, $Z = \{b_1, b_3\}$ and $i = 3$.

Iteration-3. Now, BEST-RATIO-EDGE returns $S_3 = \{t_3, b_4\}$. In the set Z, only b_1 has price less than $K = 700$, but has a lesser profit to price ratio than the edge $\{t_3, b_4\}$. So, we add t_3 and b_4 in $\mathcal{S}$ and delete them from G' along with the corresponding edge. At the end of the third iteration, $K = 350$, $Z = \{b_1, b_3\}$ and $i = 4$.

Iteration-4. The graph does not have any edges, so BEST-RATIO-EDGE returns $S_4 = \phi$, and every item in set Z has a price greater than $K = 350$. Thus, the while loop ends here. Now, the candidate solution is $\mathcal{S} = \{t_1, b_2, t_2, t_3, b_4\}$.

After execution of the loop, the BEST-PSEUDO-PROFIT-NODES sub-routine retrieves edge $\{t_1, b_1\}$ from the input graph G. Currently, $S^* = \{t_1, b_1\}$ and $P^* = profit(t_1) + profit(b_1) + \alpha_C = 2 + 0.5 + 0.5 = 3$. Now, t_2 and b_2 are the 1-neighbor vertices of t_1 and b_1. Q is assigned with b_2, followed by t_2 (as $pc(b_2) < pc(t_2)$). At first, b_2 is fetched from Q. Now, price $PC(S^* \cup \{b_2\}) = pc(t_1) + pc(b_1) + pc(b_2) = 400 + 400 + 200 = 1000 \leq \mathcal{K}$. So, we update $S^* = \{t_1, b_1, b_2\}$ and $P^* = 3 + 0.5 = 3.5$. Now, t_2 is fetched from Q and added to S^* and P^* is updated to 4. Similarly, we retrieve all the remaining edges of G and compute corresponding S^* and P^*. After comparing all P^* corresponding to each edge, we find $S^* = \{t_1, b_1, t_2, b_2\}$ having the maximum P^* value. So, we update S^e with $\{t_1, b_1, t_2, b_2\}$ and return it as the output. But, $\mathcal{P}(S_e) = 6.5 < \mathcal{P}(\mathcal{S}) = 7$. Thus, we do not update our candidate output set $\mathcal{S}$ with S_e. Finally, GREEDY BKG outputs $\mathcal{S} = \{t_1, b_2, t_2, t_3, b_4\}$, where the total price of all the items $PC(\mathcal{S}) = 1350$, which is within the budget $\mathcal{K} = 1600$.

5.2 Time Complexity and Approximation Bound of GREEDY-BKG

The time complexity of Algorithm 1 mainly depends upon BEST-RATIO-EDGE and BEST-PSEUDO-PROFIT-NODES sub-routines. BEST-RATIO-EDGE computes the best profit to price ratio edge in $O(|\mathcal{O}|)$. At each iteration in Algorithm 1, at most two items are selected. In the best case, only one iteration is performed, whereas in the worst case, $O(|\mathcal{T} \cup \mathcal{B}|)$ iterations are performed. BEST-PSEUDO-PROFIT-NODES iterates over each edge, and for each edge it iterates over all neighboring items. It runs in $O(\sum_{\{t,b\} \in \mathcal{O}} (d_t + d_b)) = O(\sum_{x \in V_G} d_x * d_x) = O(|\mathcal{O}|^2)$ time, where d_u is the degree of item u in graph G. In the best case, Algorithm 1 takes $O(|\mathcal{O}| + |\mathcal{O}|^2) = O(|\mathcal{O}|^2)$, whereas in the worst case, it takes $O(|\mathcal{T} \cup \mathcal{B}| \times |\mathcal{O}| + |\mathcal{O}|^2) = O(|\mathcal{O}|^2)$ time (since G is 1-neighbor graph $|\mathcal{T} \cup \mathcal{B}| \leq 2 \times |\mathcal{O}|$). Thus, the running time of Algorithm 1 is $O(|\mathcal{O}|^2)$.

Let OPT be the set of items in an optimal solution. If $\mathcal{K} \geq \sum_{x \in \mathcal{T} \cup \mathcal{B}} pc(x)$, $OPT = \mathcal{T} \cup \mathcal{B}$, and Algorithm 1 will eventually end up by considering all items in $\mathcal{T} \cup \mathcal{B}$, i.e., $\mathcal{S} = OPT$. If $\mathcal{K} < \sum_{x \in \mathcal{T} \cup \mathcal{B}} pc(x)$, let $l+1$ be the first iteration in which S_i is empty. Let $\mathcal{S}_i = \cup_{j=0}^{i} S_j$ correspond to $\mathcal{S}$ after the first i iterations, where $\mathcal{S}_0 = \phi$. Since S_{l+1} is empty, that means every item in Z or edges in G' has price exceeding K. For convenience, let $S_i^* = S_i$ for $i = 1, \ldots, l$. Let S_{l+1}^* be the highest profit to price ratio of items in Z, or edge in G' at iteration $l+1$. Let $\mathcal{S}_{l+1} = \mathcal{S}_l \cup S_{l+1}^*$.

Notice that $\mathcal{S}_l$ is a feasible solution to our problem but $\mathcal{S}_{l+1}$ is not, as it contains S^*_{l+1}, which has price exceeding K. We analyze our algorithm with respect to $\mathcal{S}_{l+1}$.

Let d_{OPT} be the maximum degree of item in $G[OPT]$, i.e., $d_{OPT} = max_{u \in V_{G[OPT]}} d_u$, where d_u is the degree of item u in $G[OPT]$. Let $\mathcal{P}_i(S)$ be the total profit of set of items S at iteration i, i.e., $\mathcal{P}_i(S) = \mathcal{P}(\mathcal{S}_{i-1} \cup S) - \mathcal{P}(\mathcal{S}_{i-1}) = \sum_{x \in S} profit_{i-1}(x) + \alpha_C C(S)$, where $profit_{i-1}(x)$ is the profit of item x at the end of iteration $(i-1)$.

Lemma 1. *For each iteration $i = 1, \ldots, l+1$, the following holds:*

$$\mathcal{P}_i(S^*_i) \geq \frac{PC(S^*_i)}{d_{OPT}\mathcal{K}}(\mathcal{P}(OPT) - \mathcal{P}(\mathcal{S}_{i-1})). \tag{5}$$

Proof. For fix iteration i, let I be the graph induced by $OPT \backslash \mathcal{S}_{i-1}$, i.e., $I = G[OPT \backslash \mathcal{S}_{i-1}]$. OPT and $\mathcal{S}_{i-1}$ are 1-neighbor sets. However, I may not be 1-neighbor graph. I may contain *singleton* items, which are stored in Z. Thus, S^*_i is either an item in Z or an edge in I. So, we have

$$\frac{\mathcal{P}_i(S^*_i)}{PC(S^*_i)} \geq \frac{\mathcal{P}_i(e)}{PC(e)}, \forall e \in E_I \tag{6}$$

and

$$\frac{\mathcal{P}_i(S^*_i)}{PC(S^*_i)} \geq \frac{\mathcal{P}_i(x)}{pc(x)}, \forall x \in Z. \tag{7}$$

It follows that for $i = 1, \ldots, l+1$:

$$\mathcal{P}(OPT) - \mathcal{P}(\mathcal{S}_{i-1}) \leq \sum_{x \in V_I} \mathcal{P}_i(x) + |E_I|\alpha_C, \tag{8}$$

$$\leq \sum_{e \in E_I} \mathcal{P}_i(e) + \sum_{x \in Z} \mathcal{P}_i(x), \tag{9}$$

$$\leq \frac{\mathcal{P}_i(S^*_i)}{PC(S^*_i)} \left(\sum_{e \in E_I} PC(e) + \sum_{x \in Z} pc(x) \right), \text{by Equations (6) and (7)}, \tag{10}$$

$$\leq \frac{\mathcal{P}_i(S^*_i)}{PC(S^*_i)} \left(\sum_{\{t,b\} \in E_I} (pc(t) + pc(b)) + \sum_{x \in Z} pc(x) \right), \tag{11}$$

$$\leq \frac{\mathcal{P}_i(S^*_i)}{PC(S^*_i)} \left(d_{OPT} \sum_{x \in V_I} pc(x) \right), \tag{12}$$

$$\leq \frac{\mathcal{P}_i(S^*_i)}{PC(S^*_i)} (d_{OPT}\mathcal{K}), \text{since } I \text{ is a subgraph of } G[OPT]. \tag{13}$$

Rearranging Equation (13), we get Lemma 1. □

Lemma 2. *For $i = 1, \ldots, l+1$, the following holds:*

$$\mathcal{P}(\mathcal{S}_i) \geq \left[1 - \prod_{j=1}^{i} \left(1 - \frac{PC(S^*_j)}{d_{OPT}\mathcal{K}} \right) \right] \mathcal{P}(OPT). \tag{14}$$

Proof. We prove this lemma by induction on i. For $i = 1$, we need to show that

$$\mathcal{P}(\mathcal{S}_1) \geq \frac{PC(S^*_1)}{d_{OPT}\mathcal{K}} \mathcal{P}(OPT). \tag{15}$$

Since $\mathcal{S}_0 = \phi$, and hence, $\mathcal{P}(\mathcal{S}_0) = 0$, and $S_1 = S_1^* = \mathcal{S}_1$. This result immediately follows from Lemma 1. Suppose Equation (14) holds for iterations $1, \ldots, i-1$. Then, using Lemma 1 and the inductive hypothesis, we can show that Equation (14) also holds for iteration i.

$$\mathcal{P}(\mathcal{S}_{i-1}) \geq \mathcal{P}(OPT) - d_{OPT}\mathcal{K}\frac{\mathcal{P}_i(S_i^*)}{PC(S_i^*)}, \text{Rearranging Lemma 1} \tag{16}$$

As $\mathcal{S}_i = \mathcal{S}_{i-1} \cup S_i^*$, and $\mathcal{P}(\mathcal{S}_i) = \mathcal{P}(\mathcal{S}_{i-1}) + \mathcal{P}_i(S_i^*)$, and substituting it in Equation (16) and rearranging it, we get

$$\mathcal{P}(\mathcal{S}_i) \geq \frac{PC(S_i^*)}{d_{OPT}\mathcal{K}}\mathcal{P}(OPT) + \left(1 - \frac{PC(S_i^*)}{d_{OPT}\mathcal{K}}\right)\mathcal{P}(\mathcal{S}_{i-1}), \tag{17}$$

$$\mathcal{P}(\mathcal{S}_{i-1}) \geq \left[1 - \prod_{j=1}^{i-1}\left(1 - \frac{PC(S_j^*)}{d_{OPT}\mathcal{K}}\right)\right]\mathcal{P}(OPT), \text{Induction on } i. \tag{18}$$

Substituting Equation (18) in Equation (17), we get

$$\mathcal{P}(\mathcal{S}_i) \geq [1 - \prod_{j=1}^{i}(1 - \frac{PC(S_j^*)}{d_{OPT}\mathcal{K}})]\mathcal{P}(OPT). \tag{19}$$

□

Theorem 2. *Algorithm 1 returns a solution whose total profit is at least* $\frac{1}{2}(1 - \frac{1}{e^{1/d_{OPT}}})\mathcal{P}(OPT)$ *within budget* $\mathcal{K}$.

Proof. From Lemma 2, we have

$$\mathcal{P}(\mathcal{S}_{l+1}) \geq \left[1 - \prod_{j=1}^{l+1}\left(1 - \frac{PC(S_j^*)}{d_{OPT}\mathcal{K}}\right)\right]\mathcal{P}(OPT), \tag{20}$$

$$\geq \left[1 - \prod_{j=1}^{l+1}\left(1 - \frac{PC(S_j^*)}{d_{OPT}PC(\mathcal{S}_{l+1})}\right)\right]\mathcal{P}(OPT), \text{Since } PC(\mathcal{S}_{l+1}) > \mathcal{K}, \tag{21}$$

$$\geq \left[1 - \left(1 - \frac{1/d_{OPT}}{l+1}\right)^{l+1}\right]\mathcal{P}(OPT), \tag{22}$$

$$\geq \left(1 - \frac{1}{e^{1/d_{OPT}}}\right)\mathcal{P}(OPT). \tag{23}$$

Since $PC(\mathcal{S}_{l+1}) > \mathcal{K}$ and $PC(\mathcal{S}_l) \leq \mathcal{K}$, now we try to find result on $\mathcal{P}(\mathcal{S}_l)$. We know that $\mathcal{S}_{l+1} = \mathcal{S}_l \cup S_{l+1}^*$, and hence, $\mathcal{P}(\mathcal{S}_{l+1}) = \mathcal{P}(\mathcal{S}_l) + \mathcal{P}_{l+1}(S_{l+1}^*)$. S_e is the solution returned by BEST-PSEUDO-PROFIT-NODES, and therefore $\mathcal{P}(S_e) \geq \mathcal{P}_{l+1}(S_{l+1}^*)$. By combining equations, we get

$$\mathcal{P}(\mathcal{S}_l) + \mathcal{P}(S_e) \geq \mathcal{P}(\mathcal{S}_l) + \mathcal{P}_{l+1}(S_{l+1}^*), \tag{24}$$

$$\geq \mathcal{P}(\mathcal{S}_{l+1}), \tag{25}$$

$$\geq \left(1 - \frac{1}{e^{1/d_{OPT}}}\right)\mathcal{P}(OPT), \tag{26}$$

$$max(\mathcal{P}(\mathcal{S}_l), \mathcal{P}(S_e)) \geq \frac{1}{2}\left(1 - \frac{1}{e^{1/d_{OPT}}}\right)\mathcal{P}(OPT). \tag{27}$$

□

6 EXPERIMENTAL EVALUATION

In this section, we first explain the dataset creation, followed by implementation details, and finally elaborate on the evaluation of our proposed GREEDY-BKG heuristic algorithm with baseline methods across both real and synthetic datasets.

6.1 Datasets

We make use of three real world E-commerce datasets, where the first is based on fashion items collected from Amazon-India[3] (Amazon-1), where the price of the items is expressed in **Indian Rupees (INR)**. The second and third are subsets of the Amazon Clothing dataset [41] (Amazon-2) and the Polyvore dataset [22], respectively, where the price of each item is in **US Dollars (USD)**. So, we convert them to INR[4] and consider items with price ≤2500. We also create and perform experiments on three types of synthetically generated datasets for evaluating the proposed GREEDY-BKG heuristic solution.

Amazon-1 dataset. We use the Product Advertising API of Amazon and collect two types of male clothing catalog items: top-wear and bottom-wear from the Amazon India website. We create 7046 distinct combinations by randomly combining 1457 top-wears with 227 bottom-wears and manually annotate them using our own online interface, where combinations are randomly shown to the annotators. We present six different occasions: formal, casual, semi-formal, semi-casual, party, and sports. For each combination, the annotators must verify whether it is compatible or not and identify one or more suitable occasions. We assign the task of outfit annotation to two male and one female fashionistas and ensure that each combination is annotated by all three of them. Items in an outfit are regarded as compatible if at least two annotators consider them as compatible, and otherwise they are incompatible. Finally, we have 5022 positive compatible outfit samples, consisting of 1419 top-wears and 209 bottom-wears, which we use to conduct our experiments. In addition, we assign an overall rating between 1 and 5 stars to each compatible outfit combination.

Amazon-2 dataset. McAuley et al. [41] introduced the massive Amazon Product dataset[5] that comprises data from various domains, e.g., books, electronics, movies, clothing, shoes, and jewelry. We consider the clothing, shoes, and jewelry dataset and use keywords related to tops and bottoms in item titles for filtering items. Next, we use *also_bought, bought_together*, and *buy_after_viewing* relations associated with each top (or bottom) item and extract corresponding bottom (or top) item. This resulted in 14370 top-bottom pairs having several repeated pairs. Moreover, some items have keywords related to tops (bottoms) in item titles that are not top-wears (bottom-wears). So, we manually checked whether a top and a bottom are correctly categorized in each of the chosen top-bottom pairs. Finally, we obtained 3388 distinct top-bottom pairs (or compatible outfits) that covers 1070 top-wears and 1178 bottom-wears. The metadata contains the price of each item, but occasionally tags/labels are missing. So, we tag one or more (up to six) occasions to each top-bottom pair in this dataset and assign an overall rating similar to the Amazon-1 dataset.

Polyvore dataset. The Polyvore dataset [22] consists of 21889 compatible outfits, where each outfit set has a maximum of eight fashion items from different categories. Since we require only top-bottom pairs for our experiments, we extract 1686 compatible outfit sets (each having a top-wear

[3]https://www.amazon.in/.
[4]We considered 1 USD = 74.13 INR at the time of data pre-processing.
[5]http://jmcauley.ucsd.edu/data/amazon/.

and a bottom-wear) covering 3015 top-wears and 2014 bottom-wears. Similar to the Amazon-1 dataset, we assign occasion tags/labels and an overall rating to each top-bottom pair.

The density of graph $G = (V, E)$ is defined as $dens(G) = |E|/|V|$ [16]. The density of the Amazon-1 dataset is 3.08, the Amazon-2 dataset is 1.51, and the Polyvore dataset is 0.33. This data was computed for meaningful comparative study of the runtime of the algorithm on these datasets.

Synthetic datasets. The time complexity of GREEDY-BKG (Algorithm 1) is $O(|\mathcal{O}|^2)$, i.e., it mainly depends upon the number of compatible outfits (top-bottom pairs). But there are limited number of compatible outfits in the Amazon-1, Amazon-2, and Polyvore datasets. Since we are only concerned about the price of items, followed by compatibility and versatility scores of item pairs and individual items, respectively, and do not require item image/text features, we can generate synthetic datasets similar to the real world data and evaluate how GREEDY-BKG performs on very dense input bipartite graphs. This motivated us to generate synthetic datasets with a higher number of compatible outfits, where prices and versatility scores of tops and bottoms follow the same distribution of real world data. We consider the distribution of price and versatility score of the Amazon-1 dataset for generating different synthetic datasets.

As shown in Table 2, we group the price of top-wear and bottom-wear items separately in the Amazon dataset into intervals, each having size 100, and denote each interval as the price range. For each price range, we count the number of items and compute the percentage of items in it. Then, we compute the mean and standard deviation of items in the price range. Similarly, we compute the mean and standard deviation of versatility score of items in each price range. Without loss of generality, we consider $|\mathcal{T}| = |\mathcal{B}| = N$. We generate the price of each item x corresponding to the j^{th} price range as a random variable with a normal distribution, i.e., $pc(x) \sim \mathcal{N}(\mu_{pc}^j, \sigma_{pc}^j)$, where μ_{pc}^j and σ_{pc}^j are the mean and standard deviation of item prices in the j^{th} price range. Similarly, we generate the versatility score of each item x in the j^{th} price range as a random variable with a normal distribution, i.e., $v(x) \sim \mathcal{N}(\mu_v^j, \sigma_v^j)$. We use separate distributions for top-wear and bottom-wear items.

Next, we construct a complete bipartite graph with a set of top-wears and bottom-wears as the two disjoint vertex set and then delete each edge with probability p. We also delete singleton vertices (or items) from the graph to ensure 1-neighbor constraint. The remaining edges after deletion form the compatible outfit set. In this way, we create different synthetic datasets denoted as D_p^N, where $N \in \{10, 1000, 3000\}$ and $p = \{0.25, 0.50, 0.75\}$.

6.2 Implementation Details

The hardware platform used for running the experiments is an Intel i7 CPU with clock speed of 2.70 GHz and 8 GB of RAM. We implement our proposed GREEDY-BKG algorithm in the C++ programming language. The execution environment is a 64-bit Windows 10 operating system. The execution time of GREEDY-BKG does not include the time taken for data loading and preprocessing.

6.3 Evaluation of GREEDY-BKG

We conduct several experiments on the Amazon dataset and different synthetic datasets for evaluating the performance of our proposed GREEDY-BKG algorithm for solving the *MOBCCWR* problem.

Results of total profit, compatibility, and versatility scores. Figure 4 shows different results on the Amazon-1, Amazon-2, and Polyvore datasets. From Figure 4(a) through (c), we observe that the

Table 2. Statistics Showing Count (#) and Percentage (%) of Items, Followed by the Distribution of Price and Versatility Score (V. Score) of Items in the Amazon Dataset Across Different Price Ranges Corresponding to Top-wears and Bottom-wears

Item Type	**Price Range**	**# (Items)**	**% (Items)**	**Mean (Price)**	**Std. Dev. (Price)**	**Mean (V. Score)**	**Std. Dev. (V. Score)**
Top-wear	100–199	4	0.28	182.75	17.09	1.00	0.00
	200–299	44	3.10	270.70	25.16	1.16	0.56
	300–399	100	7.05	366.04	28.89	1.33	0.68
	400–499	236	16.63	473.21	28.16	1.57	0.74
	500–599	121	8.53	572.33	28.30	1.28	0.56
	600–699	99	6.98	674.13	27.17	1.40	0.65
	700–799	104	7.33	774.11	27.61	1.44	0.65
	800–899	65	4.58	876.57	28.44	1.48	0.61
	900–999	83	5.85	976.05	30.34	1.51	0.70
	1000–1099	48	3.38	1057.58	30.25	1.60	0.76
	1100–1199	53	3.74	1167.18	33.29	1.38	0.59
	1200–1299	48	3.38	1271.62	27.50	1.33	0.69
	1300–1399	35	2.47	1376.49	25.31	1.43	0.60
	1400–1499	46	3.24	1484.63	28.79	1.43	0.61
	1500–1599	36	2.54	1576.56	33.58	1.42	0.55
	1600–1699	33	2.33	1679.25	28.47	1.42	0.55
	1700–1799	42	2.96	1790.62	20.20	1.31	0.46
	1800–1899	28	1.97	1888.57	21.88	1.39	0.49
	1900–1999	40	2.82	1994.83	13.69	1.62	0.62
	2000–2099	42	2.96	2087.75	27.40	1.40	0.54
	2100–2199	21	1.48	2183.76	25.63	1.62	0.58
	2200–2299	25	1.76	2294.24	13.23	1.52	0.64
	2300–2399	33	2.33	2391.94	22.68	1.36	0.54
	2400–2499	33	2.33	2496.97	8.66	1.67	0.59
Bottom-wear	500–599	25	11.96	554.76	29.57	2.12	0.99
	600–699	18	8.61	667.84	22.76	2.28	1.19
	700–799	29	13.88	755.83	32.58	3.14	1.04
	800–899	27	12.92	857.78	34.38	3.30	0.90
	900–999	29	13.88	964.22	33.44	2.86	0.90
	1000–1099	17	8.13	1065.49	29.61	2.18	1.04
	1100–1199	22	10.53	1175.36	29.78	2.14	0.81
	1200–1299	9	4.31	1244.05	28.10	3.00	1.05
	1300–1399	13	6.22	1368.54	23.27	2.00	1.11
	1400–1499	18	8.61	1472.69	30.11	1.83	0.83
	1700–1799	1	0.48	1799.00	0.00	2.00	0.00
	2400–2499	1	0.48	2499.29	0.00	2.00	0.00

total profit or objective value of the recommended set of items $\mathcal{S}$, i.e., $\mathcal{P}(\mathcal{S})$, increases with increase in budget $\mathcal{K}$, which is intuitive because with a higher budget, more items are chosen. This in turn increases the overall profit. It is also observed that for a fixed budget, $\mathcal{P}(\mathcal{S})$ is higher when $\alpha_C < \alpha_V$ than when $\alpha_C > \alpha_V$, whereas the profit corresponding to $\alpha_C = \alpha_V$ lies midway between them (Figure 4(a) and (c)). This is because in the case of Amazon-1 (Figure 4(d) and (g)) and Polyvore

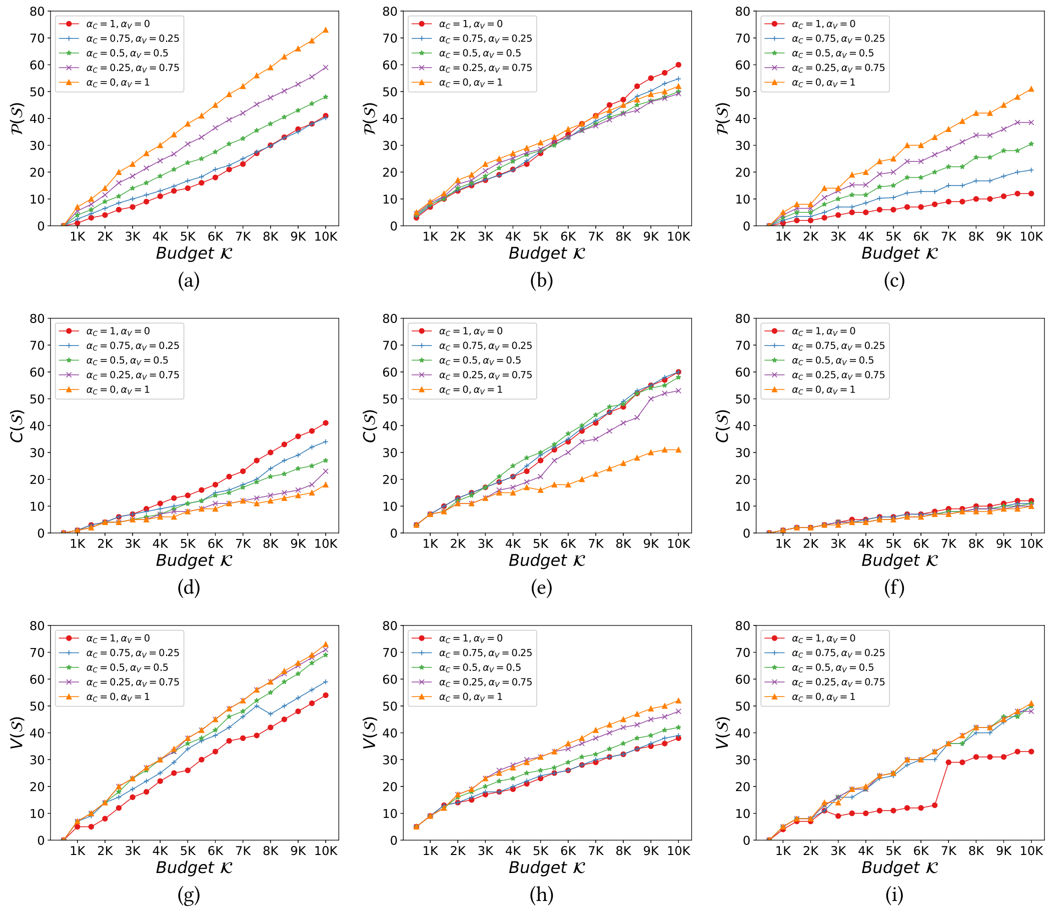


Fig. 4. Performance of GREEDY-BKG on Amazon-1 (a, d, g), Amazon-2 (b, e, h), and Polyvore (c, f, i) datasets. Variation of (a) through (c) total profit (or objective function value) - $\mathcal{P}(\mathcal{S})$, (d) through (f) total compatibility score - $C(\mathcal{S})$, and (g) through (i) total versatility score - $V(\mathcal{S})$ of the recommended set $\mathcal{S}$ across different budgets ($\mathcal{K}$) with corresponding variations in α_C and α_V.

(Figure 4(f) and (i)) datasets, the versatility score - $V(\mathcal{S})$ dominates over the compatibility score - $C(\mathcal{S})$ in the overall profit $\mathcal{P}(\mathcal{S})$.

The rate of increase in $C(\mathcal{S})$ and $V(\mathcal{S})$ with the budget is relatively less in the case of the Polyvore dataset than the Amazon-1 dataset. Moreover, Figure 4(f) and (i) show that $C(\mathcal{S})$ and $V(\mathcal{S})$ do not vary with changing α_C and α_V values in the Polyvore dataset. This is because the density of the bipartite graph corresponding to tops and bottoms of the Polyvore dataset is relatively low. However, there is one exception in the case of $V(\mathcal{S})$ for $\alpha_C = 1, \alpha_V = 0$, where $V(\mathcal{S})$ is relatively low (Figure 4(i)). In this case, when budget $\mathcal{K}$ changes from 6500 to 7000, $C(\mathcal{S})$ increases by only 1 from 8 to 9 (Figure 4(f)) by including more items. There is only one top and eight bottoms in the case of $\mathcal{K} = 6500$ (where all the bottoms are compatible with the top) and seven tops and bottoms, respectively, in case of $\mathcal{K} = 7000$ (where nine top-bottom pairs are compatible with each other). So, this leads to a sudden rise in $V(\mathcal{S})$ with more items in the recommendation set.

In Figure 4(b), we observe that the profit $\mathcal{P}(\mathcal{S})$ for different α_C, α_V values are overlapping with each other across different budgets. So, from Figure 4(b), we cannot infer about the causal effect

Table 3. Comparison Between Total Profit $\mathcal{P}(\mathcal{S})$ of the Recommended Set $\mathcal{S}$ Returned by Two Methods: GREEDY-BKG with and Without BEST-PSEUDO-PROFIT-NODES (Denoted as M_1 and M_2, Respectively) for Five Equally Spaced Budgets ($\mathcal{K}$) in the Interval [1000, 2000] Across Three Different Synthetic Datasets: $D_{0.75}^{1000}$ ($N = 1000$), $D_{0.75}^{2000}$ ($N = 2000$), and $D_{0.75}^{3000}$ ($N = 3000$), Considering $p = 0.75$ for Each Dataset

Dataset	$\mathcal{K}$	**Total Profit $\mathcal{P}(\mathcal{S})$**									
		$\alpha_C = 1.00$ $\alpha_V = 0.00$		$\alpha_C = 0.75$ $\alpha_V = 0.25$		$\alpha_C = 0.50$ $\alpha_V = 0.50$		$\alpha_C = 0.25$ $\alpha_V = 0.75$		$\alpha_C = 0.00$ $\alpha_V = 1.00$	
		M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2
$D_{0.25}^{3000}$	1000	2	2	4.25	4.25	6.5	6.5	8.75	8.75	11	11
	1500	5	5	6.75	6.75	9.5	9.5	12.25	12.25	15	15
	2000	9	9	10.25	10.25	13.5	13.5	17.25	17.25	21	21
	2500	14	14	15.5	15.5	18.5	18.5	22.25	22.25	28	28
	3000	20	20	21.75	21.75	22.5	22.5	27	27	32	32
$D_{0.50}^{3000}$	1000	2	2	4.25	4.25	6	6	8	8	10	10
	1500	**5**	**4.5**	**7.75**	**7**	11	11	14.5	14.5	18	18
	2000	9	9	11.5	11.5	15	15	19.5	19.5	24	24
	2500	14	14	15.75	15.75	19	19	24.5	24.5	30	30
	3000	20	20	20.75	20.75	24	24	29.25	29.25	36	36
$D_{0.75}^{3000}$	1000	2.00	2.00	4.00	4.00	**6.00**	**5.00**	**8.00**	**6.25**	8.00	8.00
	1500	**5.00**	**4.00**	**6.75**	**6.00**	**9.50**	**9.00**	**12.25**	**11.50**	15.00	15.00
	2000	8.00	8.00	**9.50**	**9.00**	12.50	12.50	16.25	16.25	20.00	20.00
	2500	12	12	13.5	13.5	16.5	16.5	21.25	21.25	26	26
	3000	15	15	17	17	20.5	20.5	25.75	25.75	31	31

The impact of BEST-PSEUDO-PROFIT-NODES is highlighted in bold font.

of α_C, α_V on $\mathcal{P}(\mathcal{S})$. Figure 4(e) and (h) suggests that compatibility dominates over versatility in maximizing $\mathcal{P}(\mathcal{S})$.

Results of total profit using different methods. Table 3 demonstrates the importance of using BEST-PSEUDO-PROFIT-NODES (Algorithm 3) in GREEDY-BKG (Algorithm 1). We denote GREEDY-BKG with and without BEST-PSEUDO-PROFIT-NODES as M_1 and M_2, respectively. We observe that both M_1 and M_2 work equally well, producing almost similar profits in the majority of the cases. However, M_1 equipped with BEST-PSEUDO-PROFIT-EDGE outperforms M_2 when budget is low, α_C is high, and there are less compatible outfits in the dataset. For example, we observe that M_1 performs better than M_2 mostly up to budget value of 2000 and $\alpha_C \neq 0$ in the dataset $D_{0.75}^{3000}$, which has fewer compatible outfits.

Greedy vs optimal results. To verify that the solution produced by GREEDY-BKG better approximates the optimal solution, we make use of three synthetic datasets: $D_{0.25}^{10}$, $D_{0.50}^{10}$, and $D_{0.75}^{10}$ with 25, 50, and 75 compatible outfits, respectively. We compute the ratio of the greedy objective function value to the optimal one, across different α_C, α_V, and budget ($\mathcal{K}$) values. The scale of this experiment is limited only by the need to compute the true optimal solution. Table 4 shows that on average, the greedy solution achieves ~97% of the optimal objective function value.

Runtime results of GREEDY-BKG. We run our experiments with different values of α_C, α_V, and budget $\mathcal{K}$.

Table 5 shows the variation of runtime of GREEDY-BKG with increasing budget values and four different combinations of α_C and α_V on the Amazon dataset. We observe that GREEDY-BKG

Table 4. Comparison Between the Greedy and Optimal Solution for the *MOBCCWR* Problem, Using Five Combinations of α_C, α_V and 10 Equally Spaced Budget Values Ranging from 1000 (1K) to 10000 (10K), Across Different Synthetic Datasets: $D^{10}_{0.25}$ ($p = 0.25$), $D^{10}_{0.50}$ ($p = 0.50$), and $D^{10}_{0.75}$ ($p = 0.75$), Where $N = 10$ for Each Dataset

Dataset	$\|\mathcal{T}\| = 10$ $\|\mathcal{B}\| = 10$	α_C	α_V	**Obj./Optimal Obj.** (for different budget values)									
	$\|\mathcal{O}\|$			**1K**	**2K**	**3K**	**4K**	**5K**	**6K**	**7K**	**8K**	**9K**	**10K**
$D^{10}_{0.25}$	25	1.00	0.00	1.00	1.00	0.98	0.96	0.98	0.97	0.98	0.98	0.98	0.98
		0.75	0.25	1.00	0.99	0.97	0.97	0.97	0.97	0.98	0.98	0.98	0.98
		0.50	0.50	1.00	0.97	0.99	0.97	0.98	0.98	0.98	0.99	0.98	0.99
		0.25	0.75	1.00	0.98	0.98	0.97	0.98	0.98	0.99	0.99	0.98	0.99
		0.00	1.00	1.00	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	1.00
$D^{10}_{0.50}$	50	1.00	0.00	1.00	0.99	0.95	0.97	0.96	0.97	0.96	0.98	0.98	0.98
		0.75	0.25	1.00	0.99	0.96	0.96	0.97	0.97	0.97	0.98	0.98	0.98
		0.50	0.50	1.00	0.98	0.97	0.97	0.97	0.98	0.98	0.98	0.99	0.99
		0.25	0.75	1.00	0.98	0.97	0.98	0.98	0.98	0.98	0.99	0.98	0.98
		0.00	1.00	1.00	0.98	0.98	0.99	0.99	1.00	0.99	0.99	0.99	1.00
$D^{10}_{0.75}$	75	1.00	0.00	1.00	0.99	0.97	0.95	0.97	0.97	0.98	0.98	0.98	0.98
		0.75	0.25	1.00	0.98	0.96	0.97	0.97	0.98	0.98	0.97	0.97	0.98
		0.50	0.50	1.00	0.96	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98
		0.25	0.75	1.00	0.97	0.98	0.99	0.98	0.98	0.98	0.99	0.98	0.98
		0.00	1.00	1.00	0.99	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00

Table 5. Variation of Runtime (Expressed in Seconds) with Budget ($\mathcal{K}$) of GREEDY-BKG Tested on Different Datasets (Amazon-1, Amazon-2, and Polyvore) for Different Values of α_C and α_V

Dataset	α_C	α_V	**Runtime (in seconds)**									
			1K	**2K**	**3K**	**4K**	**5K**	**6K**	**7K**	**8K**	**9K**	**10K**
Amazon-1	1.00	0.00	0.001	0.004	0.006	0.010	0.011	0.015	0.015	0.015	0.018	0.019
	0.75	0.25	0.001	0.003	0.006	0.009	0.010	0.012	0.016	0.014	0.018	0.019
	0.50	0.50	0.001	0.003	0.008	0.008	0.013	0.013	0.012	0.016	0.018	0.020
	0.25	0.75	0.001	0.004	0.006	0.011	0.011	0.014	0.013	0.016	0.017	0.016
	0.00	1.00	0.001	0.003	0.006	0.012	0.010	0.014	0.015	0.014	0.018	0.018
Amazon-2	1.00	0.00	0.001	0.002	0.003	0.005	0.006	0.007	0.008	0.011	0.013	0.020
	0.75	0.25	0.001	0.002	0.004	0.005	0.008	0.007	0.008	0.008	0.009	0.014
	0.50	0.50	0.001	0.003	0.004	0.009	0.013	0.011	0.013	0.008	0.014	0.012
	0.25	0.75	0.001	0.002	0.003	0.007	0.006	0.007	0.010	0.013	0.015	0.017
	0.00	1.00	0.001	0.002	0.003	0.011	0.010	0.013	0.013	0.015	0.009	0.012
Polyvore	1.00	0.00	0.001	0.001	0.001	0.002	0.003	0.003	0.003	0.006	0.004	0.004
	0.75	0.25	0.001	0.001	0.002	0.005	0.003	0.003	0.003	0.005	0.004	0.004
	0.50	0.50	0.001	0.001	0.002	0.002	0.003	0.003	0.004	0.004	0.004	0.005
	0.25	0.75	0.001	0.001	0.002	0.002	0.003	0.003	0.003	0.004	0.004	0.012
	0.00	1.00	0.001	0.002	0.002	0.002	0.003	0.007	0.003	0.003	0.004	0.004

computes the solution in the order of milliseconds. The runtime increases with an increase in budget, but the maximum runtime is less than 0.5 seconds to compute the recommended set of items. Compared to GREEDY-BKG, the brute force approach is intractable under similar conditions.

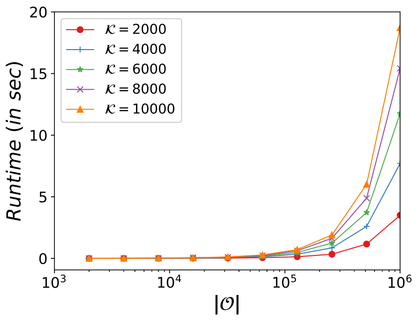


Fig. 5. Variation of runtime with the number of compatible outfits ($|\mathcal{O}|$) of GREEDY-BKG on different synthetic datasets (with $\mathcal{T} = \mathcal{B} = 1000$ and $\alpha_C = \alpha_V = 0.50$) for different budget ($\mathcal{K}$) values.

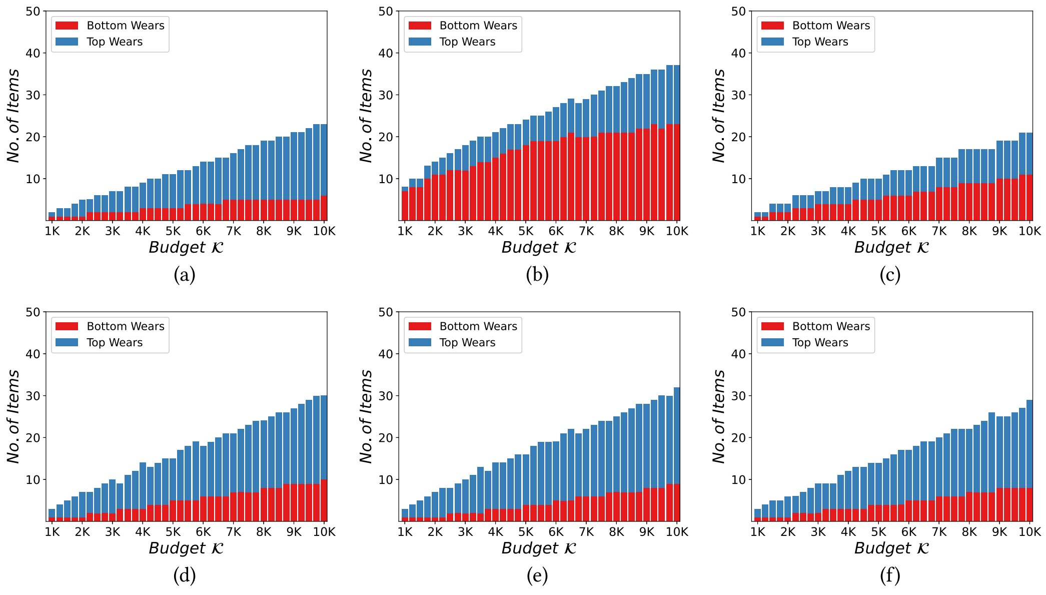


Fig. 6. Distribution of top-wears and bottom-wears chosen by GREEDY-BKG in the recommended set $\mathcal{S}$ for different budget ($\mathcal{K}$) values across six datasets: three real (a) Amazon-1, (b) Amazon-2, and (c) Polyvore, and three synthetic (d) $D^{3000}_{0.25}$, (e) $D^{3000}_{0.50}$, and (f) $D^{3000}_{0.75}$ (where $\alpha_C = \alpha_V = 0.50$).

To study the variation of runtime of GREEDY-BKG with the number of compatible outfits in datasets, we construct several synthetic datasets with 1000 top-wears and bottom-wears, respectively, where the set of compatible outfits, i.e., $|\mathcal{O}|$, ranges from 2×10^3 to 10^6. We generate around 50 datasets corresponding to each $|\mathcal{O}|$ and consider the average runtime for evaluation. Preference weights $\alpha_C, \alpha_V \in [0, 1]$ such that $\alpha_C + \alpha_V = 1$. We fix $\alpha_C = \alpha_V = 0.5$ and consider five budget ($\mathcal{K}$) values. As shown in Figure 5, runtime increases with an increase of $|\mathcal{O}|$ because the time complexity of GREEDY-BKG is $O(|\mathcal{O}|^2)$. We also observe that for fixed $|\mathcal{O}|$, runtime increases with an increase in $\mathcal{K}$. This is mainly because GREEDY-BKG tries to add items to the recommended set until the budget is exhausted.

Distribution of top-wears and bottom-wears in the recommended Set. Figure 6 illustrates the results conducted on six datasets covering Amazon-1, Amazon-2, Polyvore, and three synthetic datasets,

Table 6. Different Combinations of Objective(s) and Constraint(s) Using Budget, Compatibility, and Versatility Scores (Where $\mathcal{C}, \mathcal{V}, \mathcal{K}$ are Non-Negative Integers, Down (↓) and Up (↑) Arrows Denote Minimization and Maximization of the Preceding Quantities, Respectively)

Objective(s)	**Constraint(s)**
budget (↓), versatility (↑)	compatibility $\geq \mathcal{C}$
budget (↓), compatibility (↑)	versatility $\geq \mathcal{V}$
budget (↓)	compatibility $\geq \mathcal{C}$, versatility $\geq \mathcal{V}$
compatibility (↑)	budget $\leq \mathcal{K}$, versatility $\geq \mathcal{V}$
versatility (↑)	budget $\leq \mathcal{K}$, compatibility $\geq \mathcal{C}$
compatibility (↑), versatility (↑)	**budget $\leq \mathcal{K}$**

$D^{3000}_{0.25}$, $D^{3000}_{0.50}$, and $D^{3000}_{0.75}$, that show the distribution of top-wears and bottom-wears chosen by GREEDY-BKG in the recommended set $\mathcal{S}$ for different values of $\mathcal{K}$, where $\alpha_C = \alpha_V = 0.5$.

In Figure 6(a) and (d) through (f), we observe that the number of chosen top-wears are more than the bottom-wears. As mentioned in Section 6.1, the price and versatility score of items in the synthetic dataset is in accordance with the Amazon-1 dataset. If we observe the price distribution in the Amazon-1 dataset (Table 2), 50% of the top-wears has price in between 100 and 799, whereas 50% of the bottom-wears has price in between 500 and 899. So, the majority of the top-wears are cheaper than the bottom-wears. Moreover, the total number of bottom-wears is only 209 compared to 1419 top-wears. Hence, our observation regarding the distribution of top-wears and bottom-wears in the recommended set is justified.

As shown in Figure 6(b), the number of items in the recommended set is relatively high for every budget value, when GREEDY-BKG is applied on the Amazon-2 dataset, because items are cheaper compared to other datasets. Now, due to the presence of more bottom-wears than top-wears and the majority of bottom-wears having price less than top-wears, the number of bottoms is more than tops in the recommended sets. Table 8 shows detailed statistics of the Amazon-2 dataset in Appendix A.

The distribution of tops and bottoms (Polyvore dataset) in Figure 6(c) shows that for every budget, the number of tops and bottoms in the recommended set is almost equal and the number of recommended items is relatively less than other datasets. As mentioned before, density of the bipartite graph corresponding to the Polyvore dataset is relatively low. So, GREEDY-BKG is forced to select isolated top-bottom pairs. Table 9 shows detailed statistics of the Polyvore dataset in Appendix A.

7 DISCUSSIONS

In this section, we discuss the different combinations of objective(s) and constraint(s), probabilistic measure of compatibility scores, comparison of GREEDY-BKG with different baselines, and other E-commerce applications of GREEDY-BKG.

7.1 Objectives and Constraints

We present different combinations of objectives and constraints in Table 6. In this article, we jointly consider compatibility and versatility as objective and budget as constraint (highlighted by bold text in Table 6).

When budget is part of the objective function, we assume that users want it to be minimized. However, when compatibility and/or versatility are included in the objective function, both are expected to be maximized. In case of constraints, the budget should have a maximum upper bound

Table 7. Variation in Runtime (Expressed in Seconds) with Budget ($\mathcal{K}$) of GREEDY-BKG When Tested on Amazon-1 (Amz-1), Amazon-2 (Amz-2), and Polyvore (Poly) Datasets with $\alpha_C = \alpha_V = 0.5$ for Binary and Probabilistic Compatibility Scores (Cmp-Sc.)

Dataset	**Range of Cmp-Sc.**	**Runtime (in seconds)**									
		1K	**2K**	**3K**	**4K**	**5K**	**6K**	**7K**	**8K**	**9K**	**10K**
Amz-1	$\{0, 1\}$	0.001	0.003	0.008	0.008	0.013	0.013	0.012	0.016	0.018	0.020
	$(0, 1]$	0.008	0.459	0.724	0.837	0.818	0.827	0.828	0.829	0.827	0.831
Amz-2	$\{0, 1\}$	0.001	0.003	0.004	0.009	0.013	0.011	0.013	0.008	0.014	0.012
	$(0, 1]$	0.038	0.591	0.961	0.946	0.944	0.968	1.000	1.057	1.037	1.019
Poly	$\{0, 1\}$	0.001	0.001	0.002	0.002	0.003	0.003	0.004	0.004	0.004	0.005
	$(0, 1]$	0.001	0.012	0.003	0.004	0.006	0.007	0.007	0.008	0.009	0.009

Range of Cmp-Sc. is expressed either as $\{0, 1\}$ (Binary) or $(0, 1]$ (Probabilistic).

($\leq\mathcal{K}$), whereas compatibility and versatility scores should have lower bounds, such as $\geq\mathcal{C}$ and $\geq\mathcal{V}$, respectively.

Although all the objective-constraint combinations mentioned in Table 6 are valid, our proposed GREEDY-BKG heuristic approach cannot be directly applied to the ones having budget as part of the objective. However, by incorporating some modifications in GREEDY-BKG, we can handle compatibility (versatility) as objective and budget followed by versatility (compatibility) as constraints—for example, when the objective is maximization of the compatibility score with budget ≤5000 and versatility score ≥10 as constraints.

7.2 Compatibility Score in Terms of Probability

In this article, we consider compatibility as a binary value, i.e., either 0 or 1. A top-bottom pair may be compatible with each other, but a binary compatibility score does not quantify the extent to which they are compatible. This motivates us to consider compatibility as a probabilistic measure with values in the range $(0, 1]$. We also normalize the versatility score $v(x)$ of each item $x \in \mathcal{T} \cup \mathcal{B}$ by the maximum number of occasions $|OC|$. But GREEDY-BKG cannot handle real valued compatibility scores in its original form and needs to be modified.

Let $c(t, b) \in (0, 1]$ be the degree of compatibility between top-wear t and bottom-wear b. So, the collection of all outfit sets $\mathcal{O}$ should be redefined as $\mathcal{O} = \{\{t, b\} : 0 < c(t, b) \leq 1, t \in \mathcal{T}, b \in \mathcal{B}\}$. In Algorithm 1, Line 20 should be modified as $profit(y) = profit(y) + \alpha_C c(x, y)$. In case of binary compatibility scores, the presence of an edge between a top-bottom pair denotes compatibility score as 1 and absence of an edge denotes compatbility score as 0. So, all the edges are having unit weights. But when the compatibility score is real valued, the edge weights are not the same and vary between $(0, 1]$. We solve the latter version using FPTAS [27, 52] of a traditional 0−1 Knapsack problem and modify Lines 8 through 13 of Algorithm 3. The corresponding updated version is Algorithm 4 in Appendix A. Lines 8 through 17 illustrate the modified portion.

Since FPTAS of the 0−1 Knapsack problem (Algorithm 5 in Appendix A) runs in $O(n^3/\epsilon)$ (where n is the number of items in a Knapsack) for any $\epsilon > 0$, the complexity of Algorithm 4 is higher than Algorithm 3, which in turn increases the overall complexity of Algorithm 3. Table 7 shows the variation in runtime of GREEDY-BKG when top-bottom compatibility is expressed in binary $\{0, 1\}$ vs probabilistic real valued $[0, 1]$ for different real world datasets. We observe that the average execution time of the original GREEDY-BKG algorithm with binary compatibility scores is 62, 97, and two times faster than the modified version with probabilistic compatibility scores when tested on Amazon-1, Amazon-2, and Polyvore datasets, respectively. As mentioned in Section 6.1, density of the Polyvore dataset is relatively less compared to the Amazon-1 and Amazon-2 datasets. So,

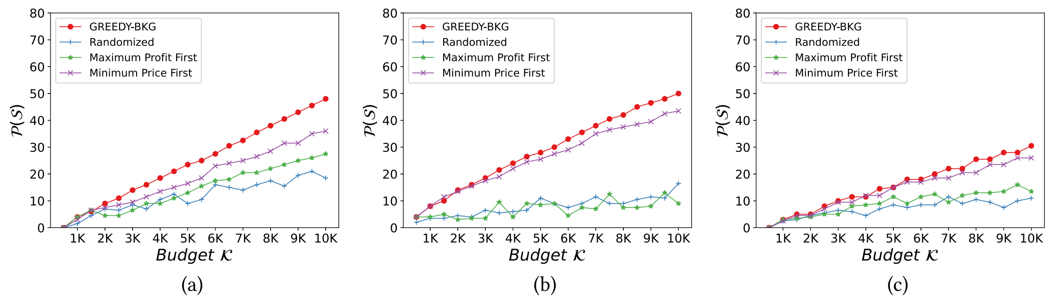


Fig. 7. Comparison between the total profit -$\mathcal{P}(\mathcal{S})$ of the recommended set $\mathcal{S}$ returned by GREEDY-BKG, *Randomized*, *Maximum Profit First*, and *Minimum Price First* on three datasets: (a) Amazon-1, (b) Amazon-2, and (c) Polyvore across different budgets - $\mathcal{K}$ (where $\alpha_C = \alpha_V = 0.5$).

the additional complexity of the modified GREEDY-BKG has a minimal impact on the execution time.

7.3 Baselines

We consider the same input bipartite graph $G(\mathcal{T} \cup \mathcal{B}, \mathcal{O})$, preference weight coefficients α_v, α_C, and budget $\mathcal{K}$ as mentioned in Section 5 to compare the performance of the GREEDY-BKG algorithm with the following three baselines:

- *Randomized:* At every iteration, this method randomly selects an edge from the set of all edges in the input bipartite graph such that the price of corresponding items incident on the edge is less than the current remaining budget. The chosen edge is removed, and the budget is updated by subtracting the price of the edge items. This is continued until no more edge items can be bought with the remaining budget. During the random edge selection step, a specific item may be incident on more than one edges. So, the price of every item is considered only once when computing the total price of the recommendation set.
- *Maximum Profit First*: In this method, at every iteration, we select the edge with the maximum profit among all edges whose price is less than the current remaining budget. Let us consider the bipartite graph G shown in Figure 2. Initially, the profit of each edge $\{t_i, b_i\}$ is set as $\mathcal{P}(\{t_i, b_i\}) = \alpha_V v(t_i) + \alpha_V v(b_i) + \alpha_C$. Now, $\{t_3, b_1\}$ is the edge having maximum profit that gets selected in the first iteration. Since $\{t_3, b_1\}$ is already chosen, we update $\mathcal{P}(\{t_1, b_1\}) = \alpha_V v(t_1) + \alpha_C$, $\mathcal{P}(\{t_2, b_1\}) = \alpha_V v(t_2) + \alpha_C$ and $\mathcal{P}(\{t_1, b_2\}) = \alpha_V v(b_2) + \alpha_C$. This ensures that we consider the $\alpha_V v(x)$ term of each item x only once while computing the profit of an edge. The budget is updated similar to the *Randomized* method at the end of each iteration.
- *Minimum Price First*: This method is similar to the *Randomized* method, except it selects the edge $\{t_i, b_i\}$ with the lowest total price $PC(\{t_i, b_i\})$ at each iteration among all edges, whose price is less than the current remaining budget.

From Figure 7, our observations regarding the performance of GREEDY-BKG with respect to the baseline methods are as follows.

- GREEDY-BKG outperforms all the baselines when tested on different real world datasets (Figure 7(a) through (c)). The success of GREEDY-BKG lies in the fact that it promotes selection of tops and bottoms that have maximum overlap between them. The closest contender of GREEDY-BKG is *Minimum Price First*.

- The *Minimum Price First* approach greedily selects the top-bottom pairs with minimum price at each iteration. So, this method inherently favors compatibility and does not give importance to versatility. As a result, it performs better than the remaining baselines but fails to outperform GREEDY-BKG. When the input bipartite graph is sparse, and average versatility score of the items is low (Figure 7(c)), then this method tends to perform similar to GREEDY-BKG.
- *Maximum Profit First* performs poorly because at each iteration it is mainly concerned about selecting the most profitable top-bottom pair irrespective of the price. In case profit is higher for costly top-bottom pairs than the cheaper pairs by a small margin, this method exhausts the total shopping budget by selecting the costly pairs first, leading to a relatively less number of top-bottom pairs in the recommendation set, which results in a low overall profit.
- The *Randomized* approach has the worst performance with the lowest profit across different budget values compared to all other baselines.

7.4 Other E-commerce Applications

In culinary practices, the ingredients play an important role in food sensation [1]. We can divide these ingredients mainly into two categories: (a) ingredients that are used to add flavors in different cuisines and (b) ingredients that are used to add mechanical stability, colors, and so forth. We can create a bipartite graph with each node as the ingredient, and an edge connecting two ingredients (one from category 1 and other from category 2) represents compatibility between them, i.e., whether these two ingredients are used together in culinary practices for some dish. The versatility score of each ingredient is the number of different cuisines (North American, Western European, Latin American, Southern European, and East Asian) in which the ingredient can be used. The price of each ingredient is the cost of the purchase. Similar to *MOBCCWR*, we can define the objective as joint maximization of compatibility and versatility with personalization in the form of the user mentioned corresponding weight coefficients and a maximum budget.

8 CONCLUSION AND FUTURE WORK

In this article, we addressed the task of automating the recommendation of capsule wardrobes by considering item prices and user specified preference weights for compatibility, versatility, and a maximum shopping budget. We formally defined this optimization problem as an *MOBCCWR* problem that is modeled as a bipartite graph with a pair of disjoint vertex sets (corresponding to top-wears and bottom-wears) and an edge set (corresponding to compatible outfits). The objective is to find a 1-neighbor subset of fashion items that maximize both compatibility and versatility scores by considering user mentioned weight coefficients for compatibility, versatility, and a maximum shopping budget that captures personalized preferences. We made an in-depth complexity analysis of *MOBCCWR*, proved it to be NP-Complete, and proposed a greedy heuristic algorithm (GREEDY-BKG) to solve real instances of *MOBCCWR*, producing near-optimal solution for each instance in less than a second that are otherwise intractable. We analyzed the time complexity and approximation bound of GREEDY-BKG and evaluated the usefulness of our proposed algorithm on both real and synthetic datasets. In the future, we plan to extend from the current two-layered to a more generalized n-layered fashion outfit recommendation framework, followed by use of collaborative filtering and deep learning techniques to incorporate user preferences, item preferences, and body shapes. The code and dataset can be found at https://github.com/debobanerjee/MOBCCWR.

APPENDIX

A SUPPLEMENTAL ALGORITHMS AND TABLES

ALGORITHM 4: Modified Best Pseudo Profit Nodes

```
1: procedure BEST-PSEUDO-PROFIT-NODES($G, \alpha_C, \alpha_V, K$)
2:     $S_e \leftarrow \phi, P_e \leftarrow 0$
3:     for each edge $\{t, b\}$ in $\mathcal{O}$ do        ▷ iterating over all the edges in the graph $G$
4:         if $pc(t) + pc(b) > K$ then        ▷ comparing total price of $t$ and $b$ with budget $K$
5:             continue
6:         $S^* \leftarrow \{t, b\}$        ▷ initializing $S^*$ with the edge $\{t, b\}$
7:         $P^* \leftarrow \mathcal{P}(S^*)$        ▷ initializing $P^*$ with the total profit of $S^*$
8:         $KP \leftarrow \{\}$        ▷ initializing with an empty set
9:         for each item $x \in N_G(t) \setminus \{b\}$ do ▷ iterating over all the neighboring items of $t$, except $b$
10:            $I.profit = \alpha_C c(x, t), I.weight = pc(x)$        ▷ creating knapsack item $I$
11:            Add item $I$ in the Knapsack $KP$
12:        for each item $x \in N_G(b) \setminus \{t\}$ do        ▷ iterating over all the neighboring items of $b$, except $t$
13:            $I.profit = \alpha_C c(x, b), I.weight = pc(x)$        ▷ creating knapsack item $I$
14:            Add item $I$ in the Knapsack $KP$
15:        $\{S_{KP}, P_{KP}\} \leftarrow$ FPTAS-Knapsack($KP, K - pc(t) - pc(b)$)▷ 0–1 Knapsack solution for $KP$
16:        $S^* \leftarrow S^* \cup S_{KP}$        ▷ updating $S^*$ with $S_{KP}$
17:        $P^* \leftarrow P^* + P_{KP}$        ▷ updating $P^*$ with $P_{KP}$
18:        if $P_e < P^*$ then        ▷ comparing $P_e$ with $P^*$
19:            $S_e \leftarrow S^*, P_e \leftarrow P^*$        ▷ updating $S_e$ with $S^*$ and $P_e$ with $P^*$
20:    return $S_e$
```

ALGORITHM 5: FPTAS for Knapsack

```
1: procedure FPTAS-KNAPSACK($KP, K$)
2:     $P \leftarrow$ profit of the item having maximum profit in $KP$
3:     $n \leftarrow$ number of items in $KP$
4:     Given $\epsilon > 0$, let $H = \frac{\epsilon P}{n}$
5:     For each item $I$ in $KP$, define $I.profit^* = \lfloor \frac{I.profit}{H} \rfloor$
6:     With these as profits of knapsack items, using the dynamic programming algorithm, find the most profitable set within capacity $K$, say $S^*$ with profit $P^*$.
7:     return $\{S^*, P^*\}$
```

Table 8. Statistics Showing Count (#) and Percentage (%) of Items, followed by Distribution of Price and Versatility Score (V. Score) of Items in the Amazon-2 Dataset Across Different Price Ranges Corresponding to Top-wears and Bottom-wears

Item Type	Price Range	# (Items)	% (Items)	Mean (Price)	Std. Dev. (Price)	Mean (V. Score)	Std. Dev. (V. Score)
Top-wear	200–299	17	1.59	265.52	32.82	1.06	0.24
	300–399	21	1.96	360.45	24.28	1.24	0.53
	400–499	50	4.67	452.12	25.01	1.18	0.38
	500–599	48	4.49	542.51	34.19	1.23	0.42
	600–699	40	3.74	649.21	19.70	1.05	0.22
	700–799	62	5.79	747.16	22.59	1.18	0.38
	800–899	56	5.23	857.75	32.23	1.34	0.61
	900–999	46	4.30	954.25	15.89	1.24	0.47
	1000–1099	57	5.33	1035.88	19.71	1.25	0.43
	1100–1199	65	6.07	1155.73	33.61	1.26	0.47
	1200–1299	35	3.27	1259.11	19.45	1.26	0.50
	1300–1399	50	4.67	1335.84	12.72	1.22	0.46
	1400–1499	96	8.97	1470.99	25.33	1.26	0.51
	1500–1599	15	1.40	1553.12	24.20	1.13	0.34
	1600–1699	30	2.80	1640.74	23.11	1.17	0.58
	1700–1799	44	4.11	1730.50	34.31	1.18	0.39
	1800–1899	85	7.94	1847.21	15.19	1.28	0.54
	1900–1999	30	2.80	1926.24	1.18	1.37	0.55
	2000–2099	66	6.17	2055.09	32.55	1.42	0.60
	2100–2199	15	1.40	2149.72	11.02	1.27	0.57
	2200–2299	93	8.69	2224.19	8.21	1.26	0.55
	2300–2399	17	1.59	2360.43	16.38	1.47	0.85
	2400–2499	32	2.99	2445.50	11.13	1.38	0.48
Bottom-wear	0–99	9	0.76	62.68	10.55	1.00	0.00
	100–199	1	0.08	141.59	0.00	1.00	0.00
	200–299	26	2.21	267.35	28.51	1.35	0.48
	300–399	42	3.57	362.60	23.03	1.07	0.26
	400–499	71	6.03	449.12	26.70	1.03	0.17
	500–599	62	5.26	555.60	35.35	1.06	0.25
	600–699	45	3.82	649.36	27.19	1.16	0.36
	700–799	97	8.23	741.41	14.02	1.07	0.26
	800–899	81	6.88	850.55	35.82	1.12	0.33
	900–999	75	6.37	955.38	15.53	1.23	0.42
	1000–1099	46	3.90	1042.64	25.18	1.11	0.31
	1100–1199	73	6.20	1135.14	33.60	1.11	0.31
	1200–1299	52	4.41	1249.57	16.44	1.21	0.41
	1300–1399	49	4.16	1336.61	16.65	1.12	0.33
	1400–1499	142	12.05	1459.76	32.93	1.30	0.46
	1500–1599	8	0.68	1550.34	15.36	1.25	0.66
	1600–1699	38	3.23	1630.65	10.92	1.53	0.68
	1700–1799	43	3.65	1734.44	36.39	1.28	0.66
	1800–1899	68	5.77	1850.18	17.07	1.40	0.52
	1900–1999	25	2.12	1930.82	14.62	1.16	0.61
	2000–2099	28	2.38	2037.83	36.30	1.39	0.77
	2100–2199	19	1.61	2156.37	17.64	1.05	0.22
	2200–2299	51	4.33	2224.19	8.26	1.43	0.66
	2300–2399	21	1.78	2360.90	21.54	1.24	0.61
	2400–2499	6	0.51	2442.58	4.22	2.00	0.58

Table 9. Statistics Showing Count (#) and Percentage (%) of Items, Followed by Distribution of Price and Versatility Score (V. Score) of Items in the Polyvore Dataset Across Different Price Ranges Corresponding to Top-wears and Bottom-wears

Item Type	**Price Range**	# **(Items)**	% **(Items)**	**Mean (Price)**	**Std. Dev. (Price)**	**Mean (V. Score)**	**Std. Dev. (V. Score)**
Top-wear	0–99	6	0.20	62.39	38.74	1.33	0.47
	100–199	15	0.50	176.97	18.04	1.47	0.62
	200–299	53	1.76	270.27	20.83	1.43	0.57
	300–399	78	2.59	360.26	19.44	1.56	0.76
	400–499	95	3.15	450.59	15.24	1.49	0.65
	500–599	141	4.68	557.24	29.61	1.50	0.63
	600–699	132	4.38	649.35	23.07	1.55	0.63
	700–799	205	6.80	737.43	5.57	1.43	0.61
	800–899	322	10.68	864.93	34.92	1.44	0.63
	900–999	117	3.88	963.69	0.00	1.39	0.60
	1000–1099	145	4.81	1037.82	0.00	1.50	0.64
	1100–1199	336	11.14	1149.90	37.05	1.37	0.57
	1200–1299	89	2.95	1260.21	0.00	1.40	0.59
	1300–1399	121	4.01	1334.34	0.00	1.40	0.55
	1400–1499	326	10.81	1447.35	37.02	1.45	0.64
	1500–1599	72	2.39	1556.73	0.00	1.29	0.48
	1600–1699	113	3.75	1630.86	0.00	1.34	0.56
	1700–1799	105	3.48	1748.76	36.45	1.36	0.59
	1800–1899	129	4.28	1853.25	0.00	1.51	0.67
	1900–1999	51	1.69	1927.38	0.00	1.49	0.57
	2000–2099	141	4.68	2038.31	37.06	1.44	0.56
	2100–2199	45	1.49	2149.77	0.00	1.44	0.50
	2200–2299	130	4.31	2248.42	34.88	1.37	0.51
	2300–2399	35	1.16	2372.16	0.00	1.40	0.60
	2400–2499	13	0.43	2446.29	0.00	1.77	0.58
Bottom-wear	100–199	2	0.10	139.36	1.48	1.00	0.00
	200–299	15	0.74	263.90	25.55	1.67	0.87
	300–399	18	0.89	366.86	13.89	1.67	0.67
	400–499	31	1.54	457.26	8.86	1.68	0.64
	500–599	42	2.09	553.56	28.30	1.45	0.70
	600–699	38	1.89	655.13	14.24	1.45	0.64
	700–799	93	4.62	736.62	6.62	1.41	0.68
	800–899	160	7.94	871.03	32.10	1.41	0.61
	900–999	49	2.43	963.69	0.00	1.55	0.73
	1000–1099	75	3.72	1037.82	0.00	1.55	0.77
	1100–1199	214	10.63	1149.01	37.06	1.38	0.60
	1200–1299	62	3.08	1260.21	0.00	1.44	0.59
	1300–1399	68	3.38	1334.34	0.00	1.40	0.57
	1400–1499	249	12.36	1449.26	36.88	1.43	0.62
	1500–1599	57	2.83	1556.73	0.00	1.44	0.59
	1600–1699	93	4.62	1630.86	0.00	1.37	0.56
	1700–1799	114	5.66	1744.66	36.97	1.25	0.46
	1800–1899	156	7.75	1853.25	0.00	1.39	0.58
	1900–1999	45	2.23	1927.38	0.00	1.40	0.61
	2000–2099	113	5.61	2027.75	35.45	1.35	0.59
	2100–2199	59	2.93	2149.77	0.00	1.42	0.56
	2200–2299	189	9.38	2251.75	35.90	1.45	0.61
	2300–2399	39	1.94	2372.16	0.00	1.41	0.54
	2400–2499	33	1.64	2446.29	0.00	1.39	0.55

REFERENCES

[1] Yong-Yeol Ahn, Sebastian E. Ahnert, James P. Bagrow, and Albert-Laszlo Barabasi. 2011. Flavor network and the principles of food pairing. *Journal of Scientific Reports* 1, 196 (2011), 1–7.

[2] Sihem Amer-Yahia, Francesco Bonchi, Carlos Castillo, Esteban Feuerstein, Isabel Mendez-Diaz, and Paula Zabala. 2014. Composite retrieval of diverse and complementary bundles. *IEEE Transactions on Knowledge and Data Engineering* 26, 11 (2014), 2662–2675.

[3] Jinze Bai, Chang Zhou, Junshuai Song, Xiaoru Qu, Weiting An, Zhao Li, and Jun Gao. 2019. Personalized bundle list recommendation. In *Proceedings of the 28th International Conference on World Wide Web (WWW'19)*. 60–71.

[4] Glencora Borradaile, Brent Heeringa, and Gordon Wilfong. 2012. The knapsack problem with neighbour constraints. *Journal of Discrete Algorithms* 16 (2012), 224–235.

[5] Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Jointly non-sampling learning for knowledge graph enhanced recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 189–198.

[6] Jia Chen, Qin Jin, Shiwan Zhao, Shenghua Bao, Li Zhang, Zhong Su, and Yong Yu. 2014. Does product recommendation meet its Waterloo in unexplored categories? No, price comes to help. In *Proceedings of the 37th ACM International Conference on Research and Development in Information Retrieval (SIGIR'14)*. 667–676.

[7] Shih-Fen S. Chen, Kent B. Monroe, and Yung-Chien Lou. 1998. The effects of framing price promotion messages on consumers' perceptions and purchase intentions. *Journal of Retailing* 74, 3 (1998), 353–372.

[8] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: Personalized outfit generation for fashion recommendation at Alibaba iFashion. In *Proceedings of the 2019 ACM Conference on Knowledge Discovery and Data Mining (KDD'19)*. 2662–2670.

[9] Xu Chen, Hanxiong Chen, Hongteng Xu, Yongfeng Zhang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2019. Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 765–774.

[10] Xu Chen, Kun Xiong, Yongfeng Zhang, Long Xia, Dawei Yin, and Jimmy Xiangji Huang. 2020. Neural feature-aware recommendation with signed hypergraph convolutional network. *ACM Transactions Information Systems* 39, 1 (2020), Article 8, 22 pages.

[11] Yifan Chen, Pengjie Ren, Yang Wang, and Maarten de Rijke. 2019. Bayesian personalized feature interaction selection for factorization machines. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 665–674.

[12] Yifan Chen, Yang Wang, Xiang Zhao, Jie Zou, and Maarten De Rijke. 2020. Block-aware item similarity models for top-n recommendation. *ACM Transactions on Information Systems* 38, 4 (2020), Article 42, 26 pages.

[13] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. 2016. A unified point-of-interest recommendation framework in location-based social networks. *ACM Transactions on Intelligent Systems and Technology* 8, 1 (2016), Article 10, 21 pages.

[14] Xue Dong, Xuemeng Song, Fuli Feng, Peiguang Jing, Xin-Shun Xu, and Liqiang Nie. 2019. Personalized capsule wardrobe creation with garment and user modeling. In *Proceedings of the 2019 ACM International Conference on Multimedia (MM'19)*. 302–310.

[15] Xue Dong, Jianlong Wu, Xuemeng Song, Hongjun Dai, and Liqiang Nie. 2020. Fashion compatibility modeling through a multi-modal try-on-guided scheme. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 771–780.

[16] Esther Galbrun, Aristides Gionis, and Nikolaj Tatti. 2016. Top-k overlapping densest subgraphs. *Journal of Data Mining and Knowledge Discovery* 30, 5 (2016), 1134–1165.

[17] Markus Ettl, Pavithra Harsha, Anna Papush, and Georgia Perakis. 2018. A data-driven approach to personalized bundle pricing and recommendation. *Journal of Manufacturing & Service Operations Management* 22, 3 (2018), 1–41.

[18] Dehong Gao, Linbo Jin, Ben Chen, Minghui Qiu, Yi Wei, Yi Hu, and Hong-Ming Wang. 2020. FashionBERT: Text and image matching with adaptive loss for cross-modal retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 2251–2260.

[19] Nikita Golubtsov, Daniel Galper, and Andrey Filchenkov. 2016. Active adaptation of expert-based suggestions in ladieswear recommender system LookBooksClub via reinforcement learning. In *Proceedings of the 2016 Biologically Inspired Cognitive Architectures for Young Scientists (BICA'16)*. 61–69.

[20] Xianjing Han, Xuemeng Song, Yiyang Yao, Xin-Shun Xu, and Liqiang Nie. 2020. Neural compatibility modeling with probabilistic knowledge distillation. *IEEE Transactions on Image Processing* 29 (2020), 871–882.

[21] Xianjing Han, Xuemeng Song, Jianhua Yin, Yinglong Wang, and Liqiang Nie. 2019. Prototype-guided attribute-wise interpretable scheme for clothing matching. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 785–794.

[22] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S. Davis. 2017. Learning fashion compatibility with bidirectional LSTMs. In *Proceedings of the 2017 ACM International Conference on Multimedia (MM'17)*. 1078–1086.

[23] Tong He and Yang Hu. 2018. FashionNet: Personalized outfit recommendation with deep neural network. arXiv:1810.02443.

[24] Shintami Chusnul Hidayati, Cheng-Chun Hsu, Yu-Ting Chang, Kai-Lung Hua, Jianlong Fu, and Wen-Huang Cheng. 2018. What dress fits me best? Fashion recommendation on the clothing style for personal body shape. In *Proceedings of the 2018 ACM International Conference on Multimedia (MM'18)*. 438–446.

[25] Wei-Lin Hsiao and Kristen Grauman. 2018. Creating capsule wardrobes from fashion images. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 7161–7170.

[26] Wei-Lin Hsiao and Kristen Grauman. 2019. ViBE: Dressing for diverse body shapes. arXiv:1912.06697.

[27] Oscar H. Ibarra and Chul E. Kim. 1975. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM* 22, 4 (1975), 463–468.

[28] Wei Ji, Xi Li, Fei Wu, Zhijie Pan, and Yueting Zhuang. 2019. Human-centric clothing segmentation via deformable semantic locality-preserving network. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 12 (2019), 1–12.

[29] Wei Ji, Xi Li, Yueting Zhuang, Omar El Farouk Bourahla, Yixin Ji, Shihao Li, and Jiabao Cui. 2018. Semantic locality-aware deformable network for clothing segmentation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 764–770.

[30] Yannis Kalantidis, Lyndon Kennedy, and Li-Jia Li. 2013. Getting the look: Clothing recognition and segmentation for automatic product suggestions in everyday photos. In *Proceedings of the 3rd ACM International Conference on Multimedia Retrieval (ICMR'13)*. 105–112.

[31] Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2004. Introduction to NP-completeness of knapsack problems. In *Knapsack Problems*. Springer, Berlin, Germany, 483–493.

[32] Samir Khuller, Anna Moss, and Joseph (Seffi) Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters* 70, 1 (1999), 39–45.

[33] M. Hadi Kiapour, Xufeng Han, Svetlana Lazebnik, Alexander C. Berg, and Tamara L. Berg. 2015. Where to buy it: Matching street clothing photos in online shops. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV'15)*. 3343–3351.

[34] Duje Kodzoman. 2019. The psychology of clothing: Meaning of colors, body image and gender expression in fashion. *Textile & Leather Review* 2, 2 (2019), 90–103.

[35] Lakshman Krishnamurthi, Tridib Mazumdar, and S. P. Raj. 1992. Asymmetric response to price in consumer brand choice and purchase quantity decisions. *Journal of Consumer Research* 19, 3 (1992), 387–400.

[36] Kedan Li, Chen Liu, Ranjitha Kumar, and David Forsyth. 2019. Using discriminative methods to learn fashion compatibility across datasets. arXiv:1906.07273.

[37] Xingchen Li, Xiang Wang, Xiangnan He, Long Chen, Jun Xiao, and Tat-Seng Chua. 2020. Hierarchical fashion graph network for personalized outfit recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 1–10.

[38] Yuncheng Li, Liang Liang Cao, Jiang Zhu, and Jiebo Luo. 2017. Mining fashion outfit composition using an end-to-end deep learning approach on set data. *IEEE Transactions on Multimedia* 19, 8 (2017), 1946–1955.

[39] Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2020. Explainable outfit recommendation with joint outfit matching and comment generation. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2020), 1502–1516.

[40] Si Liu, Zheng Song, Guangcan Liu, Changsheng Xu, Hanqing Lu, and Shuicheng Yan. 2012. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12)*. 3330–3337.

[41] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International Conference on Research and Development in Information Retrieval (SIGIR'15)*. 43–52.

[42] Hai Thanh Nguyen, Thomas Almenningen, Martin Havig, Herman Schistad, Anders Kofod-Petersen, Helge Langseth, and Heri Ramampiaro. 2014. Learning to rank for personalised fashion recommender systems via implicit feedback. In *Proceedings of the 2014 International Conference on Mining Intelligence and Knowledge Exploration (MIKE'14)*. 51–61.

[43] Nilesh Pandey and Andreas Savakis. 2019. Poly-GAN: Multi-conditioned GAN for fashion synthesis. arXiv:1909.02165.

[44] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting cross-session information for session-based recommendation with graph neural networks. *ACM Transactions on Information Systems* 38, 3 (2020), Article 22, 23 pages.

[45] Debapriya Roy, Sanchayan Santra, and Bhabatosh Chanda. 2020. LGVTON: A landmark guided approach to virtual try-on. arXiv:2004.00562.

[46] Franco Scarselli, Marco Gori, Ah Chung Tsoia, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80.

[47] J. Ben Schafer, Joseph Konstan, and John Riedl. 1999. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce (EC'99).* 158–166.

[48] Xuemeng Song, Fuli Feng, Jinhuan Liu, Zekun Li, Liqiang Nie, and Jun Ma. 2017. NeuroStylist: Neural compatibility modeling for clothing matching. In *Proceedings of the 25th ACM International Conference on Multimedia (MM'17).* 753–761.

[49] Xuemeng Song, Liqiang Nie, Yinglong Wang, and Gary Marchionini. 2019. *Compatibility Modeling: Data and Knowledge Applications for Clothing Matching*. Morgan & Claypool.

[50] Reuben Tan, Mariya I. Vasileva, Kate Saenko, and Bryan A. Plummer. 2019. Learning similarity conditions without explicit supervision. In *Proceedings of the 2019 IEEE International Conference on Computer Vision (ICCV'19).* 10373–10382.

[51] Pongsate Tangseng, Zhipeng Wu, and Kota Yamaguchi. 2017. Looking at outfit to parse clothing. arXiv:1703.01386.

[52] Vijay Vazirani. 2003. *Approximation Algorithms*. Springer-Verlag, Berlin, Germany.

[53] Mengting Wan, Di Wang, Matt Goldman, Matt Taddy, Justin Rao, Jie Liu, Dimitrios Lymberopoulos, and Julian McAuley. 2017. Modeling consumer preferences and price sensitivities from large-scale grocery shopping transaction logs. In *Proceedings of the 26th International Conference on World Wide Web (WWW'17).* 1103–1112.

[54] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[55] Min Xie, Laks V. S. Lakshmanan, and Peter T. Wood. 2010. Breaking out of the box of recommendations: From items to packages. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys'10).* ACM, New York, NY, 151–158.

[56] Wei Yang, Ping Luo, and Liang Lin. 2014. Clothing co-parsing by joint image segmentation and labeling. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14).* 3182–3189.

[57] Xun Yang, Xiangnan He, Xiang Wang, Yunshan Ma, Fuli Feng, Meng Wang, and Tat-Seng Chua. 2019. Interpretable fashion matching with rich attributes. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19).* 775–784.

[58] Xin Yang, Xuemeng Song, Xianjing Han, Haokun Wen, Jie Nie, and Liqiang Nie. 2020. Generative attribute manipulation scheme for flexible fashion search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20).* 941–950.

[59] Wenli Yu, Li Li Xiaofei Xu, Dengbao Wang, Jingyuan Wang, and Shiping Chen. 2017. ProductRec: Product bundle recommendation based on user's sequential patterns in social networking service environment. In *Proceedings of the 2017 IEEE International Conference on Web Services (ICWS'17).* 301–308.

[60] Guoshuai Zhao, Hao Fu, Ruihua Song, Tetsuya Sakai, Zhongxia Chen, Xing Xie, and Xueming Qian. 2019. Personalized reason generation for explainable song recommendation. *ACM Transactions on Intelligent Systems and Technology* 10, 4 (2019), Article 41, 21 pages.

[61] Yu Zheng, Chen Gao, Xiangnan He, Yong Li, and Depeng Jin. 2020. Price-aware recommendation with graph convolutional networks. In *Proceedings of the 36th IEEE International Conference on Data Engineering (ICDE'20).* 133–144.

[62] Tao Zhu, Patrick Harrington, Junjun Li, and Lei Tang. 2014. Bundle recommendation in eCommerce. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'14).* 657–666.

Received July 2021; revised February 2021; accepted March 2021