CAPReS: Context Aware Persona Based Recommendation for Shoppers

Joydeep Banerjee
Arizona State University
jbanerje@asu.edu

Gurulingesh Raravi
Xerox Research Center India
gurulinges.raravi@xerox.com

Manoj Gupta
Xerox Research Center India
manoj.gupta@xerox.com

Sindhu K. Ernala
IIIT Hyderabad
eskiranmai94@gmail.com

Shruti Kunde and **Koustuv Dasgupta**
Xerox Research Center India
{firstname.lastname}@xerox.com

Abstract

Nowadays, brick-and-mortar stores are finding it extremely difficult to retain their customers due to the ever increasing competition from the online stores. One of the key reasons for this is the lack of personalized shopping experience offered by the brick-and-mortar stores. This work considers the problem of persona based shopping recommendation for such stores to maximize the value for money of the shoppers. For this problem, it proposes a non-polynomial time-complexity optimal dynamic program and a polynomial time-complexity non-optimal heuristic, for making top-k recommendations by taking into account shopper persona and her time and budget constraints. In our empirical evaluations with a mix of real-world data and simulated data, the performance of the heuristic in terms of the persona based recommendations (quantified by similarity scores and items recommended) closely matched (differed by only 8% each with) that of the dynamic program and at the same time heuristic ran at least twice faster compared to the dynamic program.

1 Introduction

Lack of solutions to provide a personalized shopping experience to shoppers is hurting the business of brick-and-mortar stores (aka *physical retail stores*) and they are losing their customer base to online stores who have successfully implemented such sophisticated solutions. A half baked effort in this direction by these physical retail stores via text messaging has hardly helped them as most of these messages that shoppers receive on their mobiles regarding products, offers and deals are mostly spam and does not cater to the needs of the shoppers, nor their context and not even remotely related to their shopping personas. Moreover, the combination of hectic lifestyles and the explosion of available products and stores has tremendously increased the struggle of shoppers in deciding what to buy and in which stores to buy and what route (say to minimize the commute time) to take to even reach those stores. Currently, no comprehensive recommendation system is available that tries to match the needs of the shoppers by taking into account their personas, needs and constraints to the products and deals offered by the physical retailers to maximize the *value for money* of shoppers.

Value for money (VFM) may have different implications for different shoppers depending on their persona; there are different categories of shopping personas, e.g., *luxuriant*, *infrequent*, etc. Informally, a luxuriant shopper is one who generally looks for products of high-end brands and that too may be in exclusive stores. Whereas an infrequent shopper is one who generally is a needy shopper and buys only those products that s/he is specifically looking for and may prefer to wait for one of the sale seasons to begin to get a good deal. Thus, a luxuriant and an infrequent shopper looking to buy a same product might have to be given different recommendations. This makes the designing of a persona based recommendation engine a challenging task.

Apart from shopping persona, the recommendation engine needs to take care of additional constraints that shopper might have. These additional constraints may be upper bounds on the money and the time that the shopper is willing to spend on products and on commuting, respectively.

Further, we believe that such recommendation systems can become effective only when they can make multiple recommendations where each recommendation is a set of stores along with the products to buy in each of those stores and the route to traverse to reach those stores. This is mainly because of the following reasons. First, having multiple options enables the user to evaluate pros and cons of the options and gives flexibility to choose one from the list. Second, the recommendation engine itself can continuously learn more about the persona of the shopper by observing the options selected by the user from the list of recommendations (the functionality for the latter is not in the scope of this work). Therefore, it is desirable to have a recommendation engine that gives shoppers multiple options to select from and hence we focus on developing such a recommendation solution.

Problem definition. In this work, we study the problem of persona based shopping recommendation in a physical retail setting where a recommendation system needs to make top-k recommendations to the shopper for maximizing her value for money considering the above mentioned challenges related to persona and shopper constraints.

Contributions of this work. This work makes the following contributions:

C1. It proposes a dynamic programming based optimal recommendation algorithm that makes top-k recommendations; the algorithm has a pseudo-polynomial time-complexity w.r.t the time and the budget constraints specified by the shopper and has an exponential time-complexity w.r.t. the number of products to be purchased.

C2. It then proposes a polynomial time-complexity heuristic that also makes top-k recommendations.

C3. Via rigorous experimental evaluations using a combination of real-world data and simulated data, it compares the efficacy of the proposed solutions.

Both the solutions are (i) persona aware; (ii) user budget constraint aware; and (iii) user time constraint aware.

In our experiments, heuristic's performance closely matched that of the dynamic program (it differed only by 8% in terms of both the persona based recommendation and the average items recommended) and ran at least twice faster.

Organization of the paper. Section 2 discusses the related work and Section 3 briefs the system model and notations. Section 4 describes a way to model the shopping persona using similarity scores. Section 5 formally defines the problem. Section 6 presents a dynamic programming based optimal algorithm. Since this algorithm has a non-polynomial time-complexity, Section 7 presents a polynomial time-complexity heuristic for the same problem. Section 8 discusses the experimental setup and evaluations of the proposed algorithms on a mix of real-world data and simulated data. Finally, Section 9 concludes.

2 Related Work

Modeling of user preference and personalized recommendation systems has been widely studied (Kim et al. 2003; Linden, Smith, and York 2003; Andersen et al. 2008; Hu and Pu 2011; Elahi et al. 2013; Goldberg 1990) across different domains but most extensively in e-commerce applications for product recommendations to customers. This is primarily due to abundant availability of user data on web usage, shopping cart details, wish list items, click-stream details, social media profiles, etc. However, there is no direct extension of the work done in online retail to the brick-and-mortar stores due to stark differences in shopping activity.

One domain which forms close relevance to physical retail is Location based recommendation systems using geo-spatial data. Authors in (Bao, Zheng, and Mokbel 2012) find a list of recommended venues for a given user within a given geo-spatial range by considering user preferences. Similarly, authors in (Yin et al. 2013) approach the same problem using topic modeling to capture user preferences. They look at overlapping topics between user interest and local preference of location to incorporate personalized recommendations. Our current work however goes a step further from giving top-k recommendations, to also give the items to purchase from each store and the order in which to visit these stores from source to destination, considering time and budget constraints. In (Muralidharan et al. 2014), a system that automatically ranks deals according to user preferences in terms of promotions and discounts and presents them to the user on their mobile device is presented. In extension, we propose a system that takes user preferences in terms of brands, popularity, budget along with deals — see Section 4.

With the rise of ubiquitous mobile computing, we see a number of works trying to bridge the gap between online and physical retail. Authors in (Decker, Kubach, and Beigl 2003) present a smart shelf technology based on interaction sensing which is able to track basic simple actions, such as take, return and remove, which are performed on items by the customers to model their behavior. Authors in (Van der Heijden, Kotsis, and Kronsteiner 2005) discuss key findings from behavioral decision theory to provide a set of functional requirements for decision making 'on the go' in mobile recommendation systems. We take inspiration from previous literature and present a system for persona based recommendation system in physical retail.

3 System Model

We consider the problem of persona based shopping recommendation to maximize the value for money of the shopper respecting the time and budget constraints of the shopper. Let $R = \{r_1, r_2, r_3, \ldots, r_n\}$ denote a set of retail shops where each retail shop $r_i \in R$ ($i \in \{1, 2, \ldots, n\}$) is characterized by a vector $P_i = \langle p_{i,1}, p_{i,2}, \ldots, p_{i,|P_i|} \rangle$ of products available in the shop and a vector $C_i = \langle c_{i,1}, c_{i,2}, \ldots, c_{i,|P_i|} \rangle$ of cost of the available products with $c_{i,j}$ representing the cost of product $p_{i,j}$ where $j \in \{1, 2, \ldots, |P_i|\}$. The universe of products is denoted by $U = \{p_1, p_2, \ldots, p_m\}$ implying that $\forall i, \forall j : p_{i,j} \in U$.

A city is represented as a graph $G = (V, E)$ where $V = \{V_1 \cup V_2\}$ is a set of nodes with each node representing either a retail shop (consisting of vertices in V_2) or some other location/landmark in the city (such as an apartment, an office space, a tech park, etc consisting of vertices in V_1) and E is a set of weighted directed edges with each edge $e_{x,y}$ representing the road connectivity between node v_x and node v_y and weight of this directed edge $t_{x,y}$ represents the time to travel from node v_x to node v_y considering the real-time traffic information.

A shopper is denoted by u. Set $\pi^u = \{\pi_1, \pi_2, \ldots, \pi_{|\pi^u|}\} \subseteq U$ denotes a set of products that the shopper u is looking to purchase with each product $\pi_k \in \pi^u$ where $k \in \{1, 2, \ldots, |\pi^u|\}$. The source and the destination locations of shopper u in the graph is denoted by $v_s^u \in V$ and $v_d^u \in V$ respectively. The budget and the time constraint of the shopper is denoted by b^u and t^u respectively. Persona of a shopper u is modeled using similarity scores as follows. Each retail shop r_i is characterized by a vector $S_i^u = \left\langle s_{i,\pi_1}^u, s_{i,\pi_2}^u, \ldots, s_{i,|\pi_{|\pi^u|}|}^u \right\rangle$ of similarity score of the available products for a shopper u with $s_{i,j}^u$ representing the similarity score of the product π_j for shopper u in retail shop r_i. Section 3 discusses one of the ways to determine these scores to model a shopper's persona. The recommendations of our solutions depend on the values of these scores, however working of the solutions is independent of the way these scores are computed.

A recommendation to the shopper contains path to be taken by the shopper from source to destination, a set of retail shops that the shopper must visit, a set of products suggested for purchasing in each of these shops, the amount that the shopper would be spending to purchase the suggested products upon choosing this recommendation and the time that the shopper will take to cover the suggested path.

4 Persona Modeling via Similarity Scores

In this section, we discuss an approach to model a shopper's persona using similarity scoring. For a shopper, for each product listed (for purchasing) in her query, a score is computed. The score of a product will generally be different for shoppers with different personas. For example, the score say for a high-end wrist watch in a store will be higher for a luxuriant shopper compared to that of an infrequent shopper.

The way we have computed the similarity scores to model the shopper's persona is as follows. Each user u is characterized with a set of features $F = \{x_1, x_2 ..., x_{|F|}\}$. The feature identification was done using demographic studies for 60 different human subjects followed by quantification of the the feature values. The data is used to classify between *luxuriant* and *infrequent* shoppers. The classification results were used to generate weights $W = \{w_{1,i}, w_{2,i}, w_{3,i}, w_{4,i}\}$ for four different identified features namely *brand consciousness, product popularity, media influence, promotions* respectively pertaining to the two different persona class (denoted by i where $i = 1$ for *luxuriant* persona and $i = 2$ for *infrequent* persona). The weight signifies the importance of a feature to a specific persona label. For any given queried product π_i the matched products are quantified for the four features discussed earlier. Let M denote the set of all matched products. With feature values being $\{fv_1, ..., fv_4\}$ the corresponding quantified value for the product $m \in M$ is given as $\{C_{fv_1,m}, ..., C_{fv_4,m}\}$. With these definitions, the similarity score s^u_{j,π_i} for the product π_i at a retail store r_j for a user u with persona p is given by:

$$s^u_{j,\pi_i} = \max_{\forall m \in M} \left\{ \sum_{j=1}^{4} w_{j,p} * \frac{1}{1 + \left|C_{fv_j,m} - fv_j\right|^2} \right\}$$

The details of feature quantification and classification approach are excluded due to space limitation. We are aware that there may be better ways to compute these similarity scores (to model the persona). Moreover, the categories of persona's are not restricted to luxuriant and infrequent. Hence, we would like to clarify that although the proposed recommendation solutions are dependent on the similarity scores, the working of our solutions is independent of it. Any persona modeling which can generate such similarity scores can be incorporated into the proposed system.

5 Problem Formulation

In this section, we formally present the problem using the notations discussed in Section 3. First, we describe the formulation to obtain a single optimal recommendation and term it as Optimal Shopper Persona Based Recommendation (OSPBR) problem. Then, we describe the formulation to obtain top-k recommendations and term it as Shopper Persona Based Recommendation problem (SPBR).

5.1 Optimal shopper persona based recommendation problem formulation

The OSPBR problem is defined as follows: *Given* (g1) a set π^u of products that a shopper u is looking to purchase; (g2) a time constraint t^u and a budget constraint b^u specified by the shopper; (g3) a source $v^u_s \in V$ and a destination $v^u_d \in V$ location of the shopper; (g4) a set R of retail shops and inventory of each retail shop. Each queried product π_i in the inventory of retail shop r_j is represented by a tuple $\left\langle c_{\pi_i}, s^u_{j,\pi_i} \right\rangle$ where c_{π_i} represents the cost of the product (0 if not available) and s^u_{j,π_i} represents the score of the product for shopper u; and (g5) a road network represented by a directed graph $G = (V, E)$ ($V = \{V_1, V_2\}$ as discussed earlier) with edge weights t_{xy} representing the time it takes to travel from location v_x to location v_y, *we need to find* a route (or walk) from v^u_s to v^u_d along with a mapping of products recommended to be purchased in each of the retail shop along the route with the objective of maximizing the scores of products picked *such that* the following constraints are satisfied: (c1) sum of edge weights in the route is no greater than t^u; (c2) each product $\pi_i \in \pi^u$ is picked up at most once; (c3) sum of costs of the products picked up is no greater than b^u; and (c4) any edge $(x, y) \in E$ is traversed at most once.

It can be shown that the decision version of this problem is NP-Complete by reduction from Minimum Steiner Tree problem. The proof is skipped here due to space constraint.

5.2 Shopper persona based recommendation problem formulation

The recommendation obtained from OSPBR problem is optimal with respect to the sum of scores of recommended products. Similarly, we can obtain second best, third best and so on till k best recommendations (referred to as top-k recommendations). The problem of determining top-k recommendations to the OSPBR problem is termed as the Shopper Persona Based Recommendation problem (SPBR).

For a given problem instance, it may happen that there may not be k recommendations (respecting all the constraints) with each recommendation recommending all the products from the user query to buy. In that case, a few recommendations will only suggest a subset of products to buy from the list of products given in the user query. In such a case, we order those recommendations as follows. A recommendation that suggests a higher number of products compared to the other is listed before the latter. Further, if the number of products suggested are the same then the recommendation with the highest score is listed before the other.

6 An Optimal Recommendation Algorithm

In this section, we propose an optimal algorithm using dynamic programming to obtain top-k recommendations for the SPBR problem. We do this in two steps. First, we describe the dynamic program for the OPSBR problem (for getting a single optimal recommendation) and then extend it to the SPBR problem to obtain top-k recommendations.

6.1 Dynamic program for OSPBR problem

If we knew the order/sequence in which the products queried by the user need to be picked up then we would directly invoke the dynamic program (described later in this section) with this sequence. Since there is no pre-specified sequence, we need to generate all possible permutations of the queried products and for each permutation, we need to invoke the dynamic program and chose the best output. The basic assumption behind the feasibility (w.r.t. time and space complexity) of the dynamic program is that in real world scenario, the number of queried products are expected to be relatively small for which the number of permutations are well under system requirements. For example, with the products queried as $\{\pi_1, \pi_2, \pi_3\}$, the permutations are $\{\pi_1, \pi_2, \pi_3\}$, $\{\pi_1, \pi_3, \pi_2\}$, $\{\pi_2, \pi_1, \pi_3\}$, $\{\pi_2, \pi_3, \pi_1\}$, $\{\pi_3, \pi_1, \pi_2\}$ and $\{\pi_3, \pi_2, \pi_1\}$. For a given permutation $\{\pi_3, \pi_2, \pi_1\}$ denoted by $\mathcal{P}$ and a walk from v_s^u to v_d^u, the product π_2 can be picked up from a store if and only if π_3 has already been picked up. The same holds for product π_1 with respect to product π_2.

With that, we now describe the working of the proposed dynamic program. For a given permutation $\mathcal{P}$, a four dimensional table $\mathcal{D}_{\mathcal{P}}$ is constructed:

D1. The first dimension captures the time constraint and is denoted by t with $t \in [0, t^u + 1]$.

D2. The second dimension consists of the vertices $V_1 \cup V_2$ in the road network. We will abuse the notation and refer to $u \in V_1 \cup V_2$ as a real number. So, $u \in [0, |V_1 \cup V_2| - 1]$.

D3. The number of products picked up is captured by the third dimension. Similar to dimension 1, it is represented by a variable i with $i \in [0, |\pi^u|]$.

D4. The fourth dimension represents the cost/budget constraint and is denoted by b with $b \in [0, b^u + 1]$.

An entry in the table $\mathcal{D}_{\mathcal{P}}$ is represented by $\mathcal{D}_{\mathcal{P}}[t'][v][i][b]$. A given entry stores a score for a given time t, vertex v, number of products i picked and cost b incurred so far based on a substructure (to be discussed later in this subsection).

The initial condition for the substructure is $\mathcal{D}_{\mathcal{P}}[t'][v][i][b] = 0$ for $t' = 0, 0 \leq v \leq |V_1 \cup V_2| - 1, 0 \leq i \leq |\pi^u|, 0 \leq b \leq b^u$. Additionally a list $L_{t',v,i,b}$ is maintained for each table entry $\mathcal{D}_{\mathcal{P}}[t'][v][i][b]$. For $t' = 0$, v as the index of v_s^u, $0 \leq i \leq |\pi^u|, 0 \leq b \leq b^u$, $L_{t',v,i,b}$ (defined next) is initialized to v_s^u. With same values of t', i and b the list is initialized to an empty set for all other vertex index v. This list maintains the walk from v_s^u to the vertex in index v corresponding to the value at any table entry t', v, i, b. A list $\mathcal{L}$ is defined for all given table entries. When transiting from a node u to node v the list maintains the products that can be picked up from vertex v minus the products which are already picked up in the path that ended in vertex u. The substructure is now formulated as follows:

$$\mathcal{D}_{\mathcal{P}}[t'][v][i][b] = \Big(\mathcal{D}_{\mathcal{P}}[t'-1][v][i][b], \max_{u,i',b'} \Big(\lambda\big(\sum_{j \in \mathcal{L}} s^u_{v,\pi_j} + \mathcal{D}[t' - t_{u,t'}][u][i'][b']\big)\Big), \max_{u,i',b'} (\gamma(\mathcal{D}[t' - t_{u,t'}][u][i'][b']))\Big)$$

The expression $\max_{u,i',b'} \Big(\lambda\big(\sum_{j \in \mathcal{L}} s^u_{v,\pi_j} + \mathcal{D}[t' - t_{u,t'}][u][i'][b']\big)\Big)$ is explained as follows:

1. All $u \in V_1 \cup V_2$ with $(u, v) \in E$.
2. Correspondingly $t' - t_{u,t'} > 0$ which implies that we reached vertex u at time $t' - t_{u,t'}$.
3. $i' \leq i$ and gives the number of products already picked by the path that ended at vertex u.
4. Similarly $b' \leq b$ denotes the cost at vertex u.

The value of λ is set to 1 when all of the following conditions are satisfied:

1. The list $L_{t'-t_{u,t'},u,i',b'}$ does not include the traversal of edge (u, v).
2. v contains products indexed $i' + 1$ to i in permutation $\mathcal{P}$.
3. The cost of such products from (2) (if satisfied) added with b' is $\leq b$.

If all the three conditions are satisfied then the list $\mathcal{L}$ of products is set to the products in the permutation $\mathcal{P}$ from $(i'+1)^{th}$ index to i^{th} and the sum $\sum_{j \in \mathcal{L}} s^u_{v,\pi_j}$ is computed and added to $\mathcal{D}[t' - t_{u,t'}][u][i'][b'])$. λ is set to 0 otherwise. This sum signifies the increase in the similarity score due to addition of new products in the current node u.

In case λ is set to 0 because one of the three conditions is not satisfied (for all range of u, i', b'), then it means that we cannot pick any products from v. The next expression γ implies that we have taken a route from a neighbor u of v but did not pick any product from v. For the expression $max_{u,i',b'}(\gamma(\mathcal{D}[t' - t_{u,t'}][u][i'][b']))$, the value of γ is set to 1 if $\lambda = 0$ and $b' \leq b$ and $i' \leq i$.

In qualitative terms, the values of $\mathcal{D}_{\mathcal{P}}[t'][v][i][b]$ is computed as the maximum of

a. value of path that ended at v at its previous time index,

b. maximum of all feasible paths that can end at u that is a neighbor of v, then took edge (u, v) and some products are picked from v,

c. maximum of all feasible path that can end at u that is a neighbor of v, then took edge (u, v) and no product is picked from v,

If the value at table location $\mathcal{D}_{\mathcal{P}}[t'][v][i][b]$ is due to (a) or (b) the index of the vertex v' is appended to $L_{t,v,i,b}$. It can be shown that the value at $\mathcal{D}_{\mathcal{P}}[t'][v][i][b]$ is optimum for any feasible value of t', v, i, b (aka *substructure optimality proof*); the proof is omitted here due to space limitation.

Observe that the time-complexity of the dynamic program is pseudo polynomial w.r.t. the time t^u and the budget constraint b^u and is exponential w.r.t. the number of products queried $|\pi^u|$. In the final recommendation, order in which the products are actually picked up is stored. So for all possible permutations $\mathcal{P}$, the value $max(\mathcal{D}_{\mathcal{P}}[t^u][v_d^u][|\pi^u|][b^u])$ gives the recommendation to OSPBR problem.

6.2 Optimal Solution to the SPBR problem

We now discuss how to obtain top-k recommendations for the SPBR problem using the dynamic program discussed in the previous section. We need to consider two cases. For a given problem instance, (i) there are $\geq k$ ways to collect *all* the products queried by the shopper (referred to as *top-k feasible problem*) and (ii) there are $< k$ ways to collect *all* the products queried by the shopper (referred to as *top-k infeasible problem*). These cases are described separately.

Case 1. top-k feasible problem To obtain the top-k optimal recommendations from the dynamic program described earlier, we maintain a list $\mathcal{O}^{\mathcal{P}}_{t',v,i,b}$ at each entry of the table $\mathcal{D}_{\mathcal{P}}[t'][v][i][b]$. The list has the following properties — (i) it can hold a maximum of k elements, (ii) each element consists of the accumulated similarity score and the path in the graph G to obtain the score, (iii) the list is sorted

Algorithm 1: Substructure to obtain the list $\mathcal{O}^{\mathcal{P}}_{t',v,i,b}$ of top-k recommendations in the dynamic table

```
Data: An entry $\mathcal{D}_{\mathcal{P}}[t'][v][i][b]$ in the dynamic program and
      list $\mathcal{O}^{\mathcal{P}}_{t',v',i',b'}$ with $t' \leq t-1$, $v' \leq v$, $i' \leq i$ and
      $b' \leq b$.
Result: The list $\mathcal{O}^{\mathcal{P}}_{t',v,i,b}$
1  covering all the products from the user query) begin
2      Initialize $\mathcal{O}^{\mathcal{P}}_{t',v,i,b}$ to $\{\}$;
3      for $\forall$ elements in list $\mathcal{O}^{\mathcal{P}}_{t-1,v,i,b}$ do
4          Include the element in $\mathcal{O}^{\mathcal{P}}_{t,v,i,b}$ to $\{\}$ such that the
           elements maintain a non decreasing order w.r.t. score
           and no element in the existing list has a perfect
           matching between the products collected and the
           retail store from which it is collected (we name the
           latter condition as $\mathcal{C}$);
5      for $\forall$ feasible transitions given by the equation
       $\lambda(\sum_{i \in \mathcal{L}} s^u_{v,\pi_j} + \mathcal{D}[t'-t_{u,v}][u][i_p][b_p])$ of the
       substructure in Section 6.1 do
6          for $\forall$ elements in such a feasible transition do
7              Include the element in the list $\mathcal{O}^{\mathcal{P}}_{t',v,i,b}$ if: (1) the
               number of elements in the list is $k$ and the
               element has a higher score than the last element
               in the list $\mathcal{O}^{\mathcal{P}}_{t',v,i,b}$ and (2) the number of
               elements in the list is less than $k$. For both (1)
               and (2), the condition $\mathcal{C}$ should be satisfied;
8              Maintain the non increasing order w.r.t. score;
9      for $\forall$ feasible transitions given by the equation
       $\gamma(\mathcal{D}[t'-t_{u,v}][u][i][b_p])$ of the substructure in
       Section 6.1 do
10         Perform Steps 6–8;
11 return $\mathcal{O}^{\mathcal{P}}_{t',v,i,b}$ ;
```

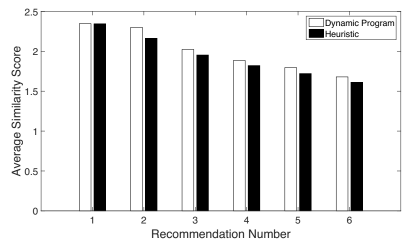
in a non-increasing order of the score. The initial conditions are set as follows: (a) for $t' = 0$, $v =$ index of v^u_s, $i \leq |\pi^u|$ and $0 \leq b \leq b^u$, the list consists of one element with 0 as the score and v^u_s as the path, (b) for $t' = 0, v =$ index of $V_1 \cup V_2 \setminus v^u_s, i \leq |\pi^u|, 0 \leq b \leq b^u$, an empty list is initialized. After this initialization, using Algorithm 1, top-k recommendations can be determined and the list $\mathcal{O}^{\mathcal{P}}_{t',v,i,b}$ can be filled with these recommendations.

Observe that with $k = 1$ the algorithm uses the same substructure as that in the previous subsection. For $k > 1$, it maintains a list of l (with $l \leq k$) elements at each position which essentially are the first l optimal recommendations (with each recommendation covering all the products from the query). It is also evident that $|\mathcal{O}^{\mathcal{P}}_{t',v,i,b}| \leq |\mathcal{O}^{\mathcal{P}}_{t'',v',i',b'}|$ with $t'' \leq t', v' = v, i' = i, b' \leq b$. Hence the number of elements in the list $\mathcal{O}^{\mathcal{P}}_{t^u, \text{index of } v^u_d, |\pi^u|, b^u}$ are the possible $\leq k$ recommendations for the given permutation $\mathcal{P}$. Hence $l \leq k$ optimal recommendations (with each recommendation covering all the products from the query) to SPBR problem can be obtained by choosing the best l recommendations from the list $\mathcal{O}^{\mathcal{P}}_{t^u, \text{index of } v^u_d, |\pi^u|, b^u}$ for all possible permutations $\mathcal{P}$.

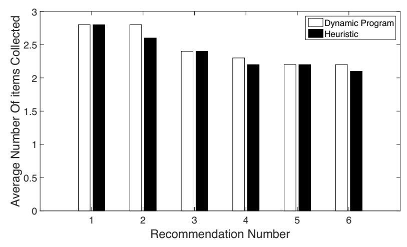
Case 2. top-k infeasible problem The Optimal Solution to SPBR problem for the case where there are $< k$ different recommendations such that each recommendation can cover all the products queried by the shopper can be directly obtained from the approach described in the previous subsection. The $l \leq k$ optimal recommendations can be obtained by choosing the best l recommendations satisfying the order described in Section 5 from the list $\mathcal{O}^{\mathcal{P}}_{t^u, \text{index of } v^u_d, |\pi^u|, b^u}$ for all possible permutations $\mathcal{P}$ and $1 \leq i_p \leq |\pi^u|$.

Algorithm 2: Heuristic Solution to the SPBR Problem

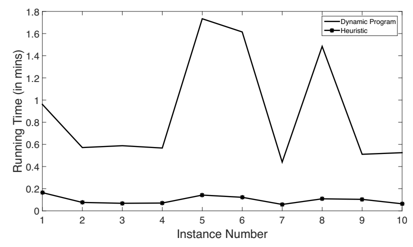
```
Data: (1) A road network $G = (V_1 \cup V_2, E)$ with edge
      weights $e_{xy}$ on each edge $(x, y) \in E$, (2) Two vertices
      $v_s, v_d \in V'_1 \cup V'_2$, (3) Two positive integers $t^u$ and $b^u$
      representing time and cost constraint respectively, (4) A
      list of items $\pi^u$, (5) For each $v \in V'_2$ a sublist of items
      $\pi^u_v \subseteq \pi^u$, (6) For each item $p$ in set $\pi^u_v$ a tuple of form
      $< i_{pv}, s_{pv}, c_{pv} >$, (7) a positive integer $K$, (8) A pre
      determined value P.
Result: A structure $\mathcal{X}$. Each element of the structure consist
      of two parts (1) a directed path from $v_{start}$ to $v_{end}$,
      (2) Let $V' \subseteq V$ and $V'$ be the set of vertices in a
      given path. The second part of the structure holds a
      list of items $i_{pv}$ with $v \in V'$. Additionally the data
      structure maintains the ordering defined in Section 5.2
      and has a maximum capacity of $K$. Insertion of
      elements in $\mathcal{X}$ satisfies the condition that for any two
      pair of elements there exists no perfect matching
      between the products collected and the retail store
      from which it is collected.
1  begin
2      Initialize the data structure $\mathcal{X}$ to NULL;
3      Create a heap data structure to retrieve the shortest paths
       from $v_s$ to $v_d$ using Eppsteins P-shortest path algorithm
       (Eppstein 1999);
4      Set currentPath = first shortest path;
5      while length of currentPath $\leq t^u$ do
6          Create a set $\pi'^u \subseteq \pi^u$ which contains the union of all
           items that are available at the retail shops contained
           in the path ;
7          For each item $\pi_i \in \pi'^u$ create a set of doubles
           $\{(s_{i1}, c_{i1}), (s_{i2}, c_{i12}), \ldots, (s_{il_i}, c_{il_i})\}$ denoting all
           of its occurrences in the path. Here $s_{ij}$ denotes the
           similarity score and $c_{ij}$ denotes cost of $j^{th}$
           occurrence of the item $\pi_i \in \pi'^u$ ;
8          Choose the occurrence for each item $\pi_i$ with
           minimum cost ;
9          Sort the occurrences in non decreasing order of cost
           and choose the maximum number of occurrences that
           can be included with budget $b^u$. Let the
           corresponding items be denoted as $\pi''^u \subseteq \mathcal{I}'$.
           Attempt to include the solution in $\mathcal{X}$;
10         The greedy approach of MCKP (Pisinger 1995) is
           employed here. An iterative approach is used to reach
           a local optima based on improvement of a metric.
           The improvement metric is taken as the change in
           similarity score, when occurrence of one item is
           replaced by same item with a better similarity score.
           The initial point of the iteration is taken as the
           solution generated in line 9. For each possible metric
           make an attempt to insert the solution in $\mathcal{X}$;
11         Set currentpath = next shortest path ;
12 return $\mathcal{X}$ ;
```



(a) Average similarity score of the recommended products per recommendation id



(b) Average number of products recommended per recommendation id



(c) Running times per problem instance for outputting top-6 recommendations

Figure 1: Comparison of the proposed heuristic and the dynamic program in terms of the average similarity scores of the recommended products, average number of products recommended and their running times for infrequent shoppers.

7 `kShoRe`: A Heuristic for SPBR problem

We now present a heuristic, `kShoRe`, for the SPBR problem. The heuristic makes top-*k Sho*pping *Re*commendations and is essentially adapts a two step approach. In the first step, time constraint is handled and in the second step, the budget constraint and the selection of the queried items are handled. The pseudo-code for the heuristic is given in Algorithm 2. The heuristic satisfies the time and cost constraint but relaxes the item covering constraint. The time-complexity of the heuristic is polynomial since the (i) Eppstien's Algorithm (Eppstein 1999); (ii) greedy approach for Multiple Choice Knapsack Problem (Pisinger 1995) and (iii) while loop on Line 5 in Algorithm 2 are all polynomial.

8 Experimental Setup and Evaluations

The experiments were performed using a mix of real-world data and randomly generated data. The road network was generated using the real-world data, i.e., stores, their locations, inventory and their costs, landmarks and commute time between each of these nodes were generated from real-world data whereas user queries (products to purchase, source and destination, cost and time constraints) were generated randomly. Specifically, the data was generated as follows. The road network (i.e. landmarks, stores, and commute times) was generated using Google Directions, Places and Geocode APIs for a locality called Whitefield in Bengaluru city in India. In that locality, 39 stores and 39 landmarks were extracted within 7 km from its center. The universe of products U was populated by crawling the web of those retail stores that have their own websites. Then the inventory for each of the retail stores in the road network were filled with the relevant products from the universe U. 20 user queries were randomly generated in which 4 queries were for purchasing 2 products and 16 queries were for purchasing 3 products. The cost and budget constraints were set to INR 2500 (Indian Rupee) and 35 mins respectively. The source and destination of the shopper for each query were chosen such that there is at least one path that satisfies the time constraint. From now on, each query is referred to as as problem instance. Out of the 20 generated problem instances, 10 were for luxuriant and 10 were for infrequent shoppers. Accordingly, for each instance, similarity scores were computed as discussed in Section 4; similarity score for a given product is a real number in the range 0 to 1.

We ran both the dynamic program and the heuristic for each of these 20 problem instances to obtain top-6 recommendations. For each recommendation number/id, the average similarity scores, the average number of products recommended by both the approaches were observed. Further, the running times of both the approaches for each problem instance to give the top-6 recommendations were also observed. These observations are plotted for infrequent shoppers in Figure 1. As can be seen from Figure 1b and Figure 1a, the performance of the heuristic is very close to that of the optimal dynamic program in terms of the average number of products recommended and their average similarity scores for each of the recommendations. The cumulative average similarity score of heuristic is approximately only 8% lower than that of the dynamic program with the maximum difference being less than 15%. This difference being less than 8% for the number of products recommended metric. As can be seen in Figure 1c, the heuristic outperforms the dynamic program w.r.t. the running time of the algorithms as it ran at least twice faster compared to the dynamic program. Similar performance behavior was observed in experiments for luxuriant shoppers as well.

9 Conclusions

This work studied the problem of persona based shopping recommendation for physical retail stores and proposed a dynamic program and a heuristic for making top-k recommendations considering shopper persona and shopper's constraints. Dynamic program is an optimal solution with a non-polynomial time-complexity whereas heuristic is a non-optimal solution with polynomial time-complexity. In our experimental evaluations using a combination of real-world and simulated data, the performance of the heuristic closely matched that of the dynamic program as the recommendations provided by the heuristic differed with that of the dynamic program in terms of shopping persona and the number of products recommended by only 8% and ran at least twice faster. In future, we intend to design an approximation algorithm for this problem, obtain relaxed local solutions using branch and bound techniques, perform more rigorous experimental evaluations to show the performance of the solutions and verify the goodness of the recommendations using humans-in-the-loop.

References

Andersen, R.; Borgs, C.; Chayes, J.; Feige, U.; Flaxman, A.; Kalai, A.; Mirrokni, V.; and Tennenholtz, M. 2008. Trust-based recommendation systems: an axiomatic approach. In *Proceedings of the 17th international conference on World Wide Web*, 199–208. ACM.

Bao, J.; Zheng, Y.; and Mokbel, M. F. 2012. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, 199–208. ACM.

Decker, C.; Kubach, U.; and Beigl, M. 2003. Revealing the retail black box by interaction sensing. In *null*, 328. IEEE.

Elahi, M.; Braunhofer, M.; Ricci, F.; and Tkalcic, M. 2013. Personality-based active learning for collaborative filtering recommender systems. In *AI* IA 2013: Advances in Artificial Intelligence*. Springer. 360–371.

Eppstein, D. 1999. Finding the k shortest paths. *SIAM J. Comput.* 28(2):652–673.

Goldberg, L. R. 1990. An alternative" description of personality": the big-five factor structure. *Journal of personality and social psychology* 59(6):1216.

Hu, R., and Pu, P. 2011. Enhancing collaborative filtering systems with personality information. In *Proceedings of the fifth ACM conference on Recommender systems*, 197–204. ACM.

Kim, J. K.; Cho, Y. H.; Kim, W. J.; Kim, J. R.; and Suh, J. H. 2003. A personalized recommendation procedure for internet shopping support. *Electronic commerce research and applications* 1(3):301–313.

Linden, G.; Smith, B.; and York, J. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE* 7(1):76–80.

Muralidharan, K.; Gottipati, S.; Ramasubbu, N.; Jiang, J.; and Balan, R. K. 2014. mydeal: a mobile shopping assistant matching user preferences to promotions. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 238–247. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Pisinger, D. 1995. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research* 83(2):394–410.

Van der Heijden, H.; Kotsis, G.; and Kronsteiner, R. 2005. Mobile recommendation systems for decision making'on the go'. In *Mobile Business, 2005. ICMB 2005. International Conference on*, 137–143. IEEE.

Yin, H.; Sun, Y.; Cui, B.; Hu, Z.; and Chen, L. 2013. Lcars: a location-content-aware recommender system. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 221–229. ACM.