A Path-based Recommendations Approach for Online Systems via Hyperbolic Network Embedding

Nikolaos Papadis
School of Electr. & Comp. Engin.
Nat'l Technical University of Athens
Zografou, 15780, Athens, Greece
Email: npapadis@netmode.ntua.gr

Eleni Stai
School of Comp. & Commun. Sciences
École Polytechnique Fédérale de Lausanne
Lausanne, 1015, Switzerland
Email: eleni.stai@epfl.ch

Vasileios Karyotis
School of Electr. & Comp. Engin.
Nat'l Technical University of Athens
Zografou, 15780, Athens, Greece
Email: vassilis@netmode.ntua.gr

***Abstract*—In this paper we introduce and demonstrate new recommendation algorithms for large-scale online systems, such as e-shops and cloud services. The proposed algorithms are based on the combination of network embedding in hyperbolic space with greedy routing, exploiting properties of hyperbolic metric spaces. Contrary to the existing recommender systems that rank products in order to propose the highest ranked ones to the users, our proposed recommender system creates a progressive path of recommendations towards a final (known or inferred) target product using greedy routing over networks embedded in hyperbolic space. Thus, it prepares the user by intermediate recommendations for maximizing the chances that he/she accepts the recommendation of the target product(s). This casts the problem of locating a suitable recommendation as a path problem, where leveraging on the efficiency of greedy routing in graphs embedded in hyperbolic spaces and exploiting special network structure, if any, pays dividends. Two variants of our recommendation approach are provided, namely Hyperbolic Recommendation-Known Destination (*HRKD*), Hyperbolic Recommendation-Unknown Destination (*HRUD*), when the target product is known or unknown respectively. We demonstrate how the proposed approach can be used for producing efficient recommendations in online systems, along with studying the impact of the several parameters involved in its performance via proper emulation of user activity over suitably defined graphs.**

***Index Terms*—Recommendation algorithms; Online social networks; E-commerce; Network embedding; Greedy routing; Hyperbolic spaces; Path problems;**

I. Introduction

Providing recommendations for products and services in modern online systems (e.g., cloud services, e-shops, etc.) can be a very complex problem, especially as the volumes of the involved users, 'products'[1], services and data increase rapidly. The users become associated with products, while they simultaneously develop social interactions (e.g., via online social networks through which they may access the e-shops, e-services, etc.) that may further assist in more efficient recommendations. Given the "long-tail" problem emerging in e-commerce [1], and other recommendable items, novel approaches for efficient recommendations of such items in large systems are needed. As the scales of operation become more complicated, the new recommendation techniques will be required to operate efficiently, successfully, and scale/adapt rapidly to the increase of number/interconnection of users and volume of available data ('products', information, etc.).

In this paper, we focus on addressing these aspects of recommendations for online systems by proposing novel path-based recommendation algorithms for big network data environments. We capitalize on a recent framework for performing efficient path-related computations in large network topologies denoted as Hyperbolic Data Analytics (HDA) [2], which exploits the properties of hyperbolic metric spaces. More specifically, here we introduce novel path-based recommendation algorithms combining the techniques of network embedding and greedy routing. The original graphs are embedded in hyperbolic space via a distance-preserving map, where path-related computations can be performed more efficiently using the hyperbolic coordinates assigned to the network nodes. Utilizing greedy routing techniques over hyperbolic coordinates, it is possible to perform a series of recommendation suggestions more efficiently. The proposed algorithms, *HRKD* and *HRUD*, address the cases where the eventually recommended product is known or unknown, respectively.

HRKD and *HRUD* differ from the existing algorithms in the literature (e.g., [3]) in terms of their objective and methodology. Instead of ranking products in order to propose the highest ranked to the users, our approach creates a progressive path of recommendations towards a final target product (known or inferred). Specifically, the algorithms first recommend some relevant products close to ones already selected by the user before a more diverse target product is suggested. In order to produce such progressive recommendation paths towards target products, we assume products in graph form (i.e., there are relations among them) and map their graph into hyperbolic space. Then, the paths are created dynamically based on greedy routing over hyperbolic coordinates, while considering users' feedback. The paths are initialized by products that the user has already chosen. Furthermore, users' relations, possibly determined by an online social network and represented in a user graph, can assist in introducing recommendation diversity and redirecting greedy paths similarly to collaborative recommendations [1], [4]. The proposed algorithms address different scenarios, i.e., *HRKD* aims at increasing the sales of specific products of an e-shop, whereas *HRUD* addresses the case where acceptance of any of a set of products is targeted,

[1]The term 'product' denotes actual goods sold, e.g., in an e-shop, or other items of recommendation interest, e.g., new services, subscriptions, etc.

similarly to the "unarticulated want" case in the literature [1].

The rest of the paper is organized as follows. Section II explains relevant works, segregating our contribution, while Section III explains the assumed user-product model. Section IV describes our approach and the proposed recommendation algorithms, while Section V provides systematic evaluations. Finally, Section VI concludes the paper.

II. Background & Contribution

Recommender systems collect and analyze different types of information regarding user preference with respect to accessed products/services, towards providing ranking scores for these items per each user. They aim at improving user experience by discovering and suggesting items of potential user interest. The information required for determining recommendations is either explicitly provided by the users, or implicitly collected, e.g., by monitoring users' behavior [3]. In general, recommender systems are classified into three categories. The first regards the content-based recommendations [3] that rely on comparisons between users' profiles and the characteristics of available items. The second category, namely collaborative filtering [4], infers users' preferences according to information collected by similar users. As such, the k-Nearest Neighbors algorithm [4] applies similarity measures [5] to determine the k-most similar users (k-neighbors) to the target user and generates suggestions based on information collected from the k-neighbors. The third combines the previous two and it is denoted as hybrid filtering [6]. Content-based filtering achieves more accurate results on a per user basis, but since it is based only on the target user's feedback/characteristics, it does not typically provide diverse recommendations. Collaborative filtering alleviates this problem, but suffers from the cold start one (i.e., initial lack of sufficient user ratings), and from the sparsity of the user-item rating matrix [1], especially when the number of items and users grows rapidly. In both cases, recommendation is based on large amounts of data, i.e., user/item features, users' feedback, choices, history, etc.

The proposed recommendation approach is of hybrid type, performing path-based recommendations initiated by products already chosen by a user (content-based recommendation), while path deviations may occur towards products chosen by similar users for introducing diversity and increasing success probability (collaborative filtering). In order to tackle the large scale involved in practical scenarios it leverages on properties of hyperbolic spaces, avoiding the sparse matrix and cold start issues of collaborative recommendations. The proposed approach differs from existing one-shot recommendation schemes, in that it creates a progressive path of recommendations towards a final target product. In our proposed recommender system the graph of products will be mapped to hyperbolic space via the Rigel embedding [7] and path-based recommendations to users will be addressed via greedy routing techniques over hyperbolic coordinates [8], [9], [10].

Network embedding has been suggested lately to cope with emerging challenges in big network data systems. Our recommendation algorithms utilize the holistic HDA framework of [2] that exploits properties of hyperbolic geometry. In HDA, networks or data are embedded in hyperbolic metric spaces, allowing for more efficient computations. In this paper, HDA is exploited for discovering and providing sequences of recommendation products. Exploiting hyperbolic network embedding for recommendations is a new approach, even though hyperbolic network embedding has been lately used for various other purposes. In [11], a social network hyperbolic embedding exploiting social connections and user preferences is proposed, aiming at increasing both the accuracy and diversity of recommendations. Our work shares similar approach and objectives, but develops different algorithms for generating recommendations. Rigel network embedding that is applied in this work maps network nodes in hyperbolic space such that their graph distances are approximated by their hyperbolic distances (i.e., distances defined over hyperbolic coordinates).

According to greedy routing in hyperbolic space, each node forwards traffic to one of its "greedy" neighbors, i.e., neighbors with hyperbolic distance to the destination smaller than itself. Thus, greedy routing uses only local information, i.e., each node is required to know the hyperbolic coordinates of its neighbors and the destination, reducing the complexity of shortest path computations [7], [12], [13] and making it rather suitable for very large-scale systems. A disadvantage of greedy routing emerges when traffic gets blocked, in cases that a node does not have neighbors closer to the destination than itself [8]. This can be significantly alleviated in hyperbolic space [7], [12] and especially via an appropriate choice of hyperbolic network embedding. Especially in scale-free networks, which are considered to have a hidden hyperbolic structure [9], [14], greedy routing based on hyperbolic coordinates achieves a very high success rate in finding a path from the source to the destination. In any case, in this work we do not make any special assumptions on the type of user/product graphs.

III. Users-Products System Model

We assume a given set of products and users. The online system consists of a connected product graph, determined via a similarity function [5] applied for pairs of products, and a graph of users determined via a different similarity function, so that neighboring nodes (users or products) are considered more similar than non-neighboring ones. Typically user links indicate some relationship, e.g., friendship in an online social network. Both graphs can be predefined and provided as inputs. There exist links between pairs of products and users, where such 'vertical' edges denote the selection or preference shown by users for specific products (Fig. 1). Vertical interconnections vary with time, since they represent users' choices over the products.

Also, the graph of products may be dynamic, e.g., when the products' relations depend on users' choices. This is the case we assume in our evaluation, in Section V, where we consider as products videos from a streaming service. The links between pairs of videos in this case are inferred via the "co-views" metric, inspired by previous works on a recommender system for YouTube [15], [16]. In our case, the

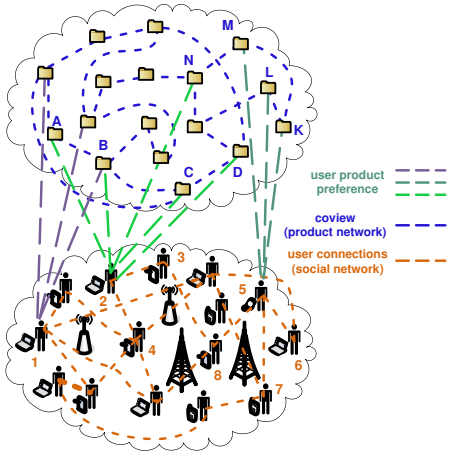


Fig. 1. The user-product graph. User connections denote social relations among them, while product connections denote co-views (two connected products have been chosen by the same user in the past). Vertical links denote that a user has shown preference for the corresponding products.

co-views metric is computed for each pair of products/videos as the number of users who chose both these products divided by the number of users who chose one of the two products. A high value of the co-views metric implies that many users have chosen a pair of products, thus these two products are connected and they are also close to each other (with link weight inversely proportional to the co-view value). Since the choices of users evolve over time as the recommender system proceeds with its operation, the product graph should be updated. In this work, we assume the update and re-embedding of the product graph over regular time-intervals, as it will be explained in the next section.

IV. Algorithmic Presentation of the Path-based Recommendation Framework

In this section we describe the proposed algorithms yielding recommendation paths for a specific user. We segregate two cases, namely one that the target product is very specific, e.g., specified explicitly by a marketing campaign (Algorithm Hyperbolic Recommendation-Known Destination, *HRKD*), and a second where the destination product(s) is (are) inferred based on the specific user's profile (Algorithm Hyperbolic Recommendation-Unknown Destination, *HRUD*).

In both algorithms, the recommendation paths begin from a set of sources determined by the recommender system and consist of products that the user has already chosen (content-based filtering) enhanced with products chosen by neighboring users in the user graph (collaborative-based filtering). Each source is associated with a corresponding destination. For known target product, all sources will have the same known destination. Thus from each source, *HRKD* or *HRUD* attempt to construct a recommendation path towards the destination via greedy routing over hyperbolic coordinates (denoted as a "greedy path"). This is obtained via functions *recommend*(.) and $refreshRecommendation$(.) shown in Algorithms 3 and 4, respectively. A recommendation path will reach a destination only if the user accepts the intermediate hops. The existence of multiple greedy paths between pairs of products allows for the search of alternative paths by our recommendation approach.

In terms of evaluation, the recommendation will be considered successful (for both *HRKD*, *HRUD*) if for one source-destination pair, one recommendation (greedy) path from all those beginning at the source towards the corresponding destination is eventually successful. In the following, we analyze in more detail each of the proposed algorithms.

Regarding the operational time-scale of both algorithms, we denote a single execution of *HRKD* for a user and target destination or a single execution of *HRUD* for a user as *recommendation round*. Within *HRKD*, *HRUD*, there are multiple *attempts*, referring to sets of sources and their corresponding set of destinations, as it will be explained below. In an *attempt*, we try to construct greedy (recommendation) paths beginning from a specific set of sources, leading to the corresponding destinations (or the target destination if known). An attempt itself consists of several *instances* over which a set of recommendations is provided to the user and his/her feedback is acquired. At each instance, an intermediate hop of the recommendation path via greedy routing will be achieved in case that the recommendation is accepted by the user.

Algorithm 1: Hyperbolic Recommendation Known Destination (*HRKD*)

```
1  Parameters: maxAttempts, sourcesPerRound, p_w
2  Input: user, dest
3  Output: success (true or false)
4  Global variables: A := products that the user has chosen
     B := products that his/her neighboring users have chosen
5  if dest ∈ A then
6      success := true; Return (Destination reached)
7  else
8      success := false
9      attempt := 1
10     while success = false and attempt < maxAttempts do
11         % Construct current sources set CSS
12         CSS := random selection of min{sourcesPerRound,
             |A ∪ B|} products: with probability p_w from
             set A and the rest from set B \ A (without replacement)
13         % Build destination ordered set
14         CDS := {dest, dest, ...., dest} with cardinality equal
             to |CSS|.
15         % Produce the recommendations
16         success = recommend(user, CSS, CDS)
17         attempt++
```

HRKD (Algorithm 1) performs a recommendation round as follows: Once a user (denoted as *user*) and a product-destination (denoted as *dest*) are specified as input, it produces recommendation paths starting from the source towards the destination, aiming at driving the user to potentially accept the recommendation of *dest*. First, *HRKD* checks if *dest* has

already been chosen by the user. In that case, the algorithm terminates. Otherwise, on each attempt, *HRKD* constructs a set CSS (current sources set) that contains the sources, as follows. CSS has cardinality $|CSS| \leq \min\{sourcesPerRound, |A \cup B|\}$, where *sourcesPerRound* is a parameter of the system. Beginning from the empty set CSS, a product from either the set A or the set B is added to CSS with probabilities p_w, $1-p_w$, respectively, up to reaching $\min\{sourcesPerRound, |A \cup B|\}$ products. The set A consists of products chosen by the target user in the past (aiming at content-based recommendations) and the set B consists of products chosen by neighbors of the target user in the user graph (aiming at collaborative-based recommendations), i.e., B is a union of the corresponding A sets of the neighbors of the specific user. Note that by setting $p_w = 1$ one would apply only content-based filtering at this initial step. If any of the sets A, B cannot provide any more elements, the sources set will still continue to be populated from the other one. Also, note that the sets A and B are considered as global variables, always available in their updated form. Once the sources set, CSS, is finalized, it is provided as input to function *recommend*(.) (Algorithm 3), together with the target *user* and the *dest* to generate the recommendation paths. If for a set of sources, CSS, the recommender system leads to success (as defined above), then *HRKD* (i.e., a recommendation round) terminates considering the target user to have accepted/selected the destination product. If this is not the case, we choose another set of sources up to a maximum number of attempts determined by parameter *maxAttempts*. The latter controls the maximum number of different source sets considered. Moreover, every *roundsToRenewGraph* recommendation rounds (i.e., application of *HRKD* or *HRUD* for multiple users/target products or multiple users correspondingly), the product graph is regenerated (by recalculating the co-views metric) and re-embedded in hyperbolic space, to achieve higher accuracy by incorporating the latest user choices.

HRUD (Algorithm 2) provides recommendation paths when the destination is not given. Initially, for a recommendation round a source set S (with cardinality $\min\{totalSources, |A \cup B|\}$) is constructed from elements of sets A and B, with p_w being the percentage of elements selected from A, as in *HRKD*. A respective destination is selected for each source. Specifically, for a source s, the candidate destinations, DS, are nodes with hyperbolic distance from the source s between T_{min} and T_{max}, and they are discovered via the function $findCandidateDestinations$(.). The latter applies Breadth First Search in the product graph, since due to the distance-preserving Rigel embedding the distances in the product graph roughly correspond to hyperbolic distances. Since T_{min} and T_{max} have small values, DS gets populated in a computationally efficient way, up to a maximum number of candidate destinations considered.

Subsequently, the candidate destinations, DS, are ranked according to how many greedy paths beginning at the source s lead to each of the destinations. This enforces the existence of multiple recommendation paths. The number of greedy paths

Algorithm 2: Hyperbolic Recommendation Unknown Destination (*HRUD*)

```
1  Parameters: sourcesPerRound, p_w, T_min, T_max,
     maxAttempts, totalSources
2  Input: user
3  Output: success (true or false)
4  Global variables: A := products that the user has chosen
     B := products that neighboring users have chosen
5  S := random selection of min{totalSources, |A ∪ B|}
     products: with probability p_w from set A and
     the rest from set B \ A (without replacement)
6  success := false
7  attempt := 1
8  while
     success = false and attempt < maxAttempts and S ≠ ∅
   do
9      CSS := ∅ (current sources set)
10     CDS := ∅ (current destinations set)
11     while |CSS| ≤ min{sourcesPerRound, |S|} and S ≠ ∅
        do
12         Extract s from S
13         if s ∉ CDS then
14             DS :=
                 findCandidateDestinations(s, T_min, T_max)
15             if DS ≠ ∅ then
16                 RDS :=
                     rankCandidateDestinations(DS, s)
17                 BD:=first element of RDS not chosen ei-
                     ther by the user or by neighboring users
18                 if such a BD exists then
19                     Append s to CSS
20                     Append BD to CDS
21     if CDS ≠ ∅ then
22         success = recommend(user, CSS, CDS)
23         attempt++
```

is inferred based on a heuristic that computes the number of greedy neighbors of s with respect to each destination and ranks the destinations in decreasing order of this number. RDS is the corresponding ranked set, obtained using the function *rankCandidateDestinations*(.). The best of the ranked nodes BD is picked as the corresponding destination for the source s, $CDS(s) := BD$. A part of the set S denoted as CSS and its corresponding destination set CDS (both with cardinality $\min\{sourcesPerRound, |S|\}$) become input to the function *recommend*(.) for each attempt, which generates the recommendations paths. In both *HRKD*, *HRUD*, the vertical links between products-users are refreshed, as a user chooses some recommended products. Therefore the sets A and B are updated (as mentioned above they are considered as global variables). The above process can be repeated up to *maxAttempts* times using another part of S (new CSS, CDS)

for each attempt, unless the function *recommend*(.) returns $success = true$ or the set S empties in the meanwhile.

Algorithm 3: Function *recommend*(.)

Parameters: R (number of recommendations per instance)
Input: $user$, CSS (current sources set), CDS (current destinations set)
Output: $success$ (true or false)
Global variables: $A :=$ products that the $user$ has chosen
$success := false$
if $CSS \cap CDS \neq \emptyset$ **then**
    $success := true$
    Return
$currentBase := Extract\ first\ R\ elements\ of\ CSS$
$CDB := Extract\ first\ R\ elements\ of\ CDS$
$recommendationsAccepted := 0_{1\times R}$ (an entry not equal to 0 is an accepted recommendation)
$sufficientData := true$
while $sufficientData$ **do**
    if $currentBase \cap CDS \neq \emptyset$ **then**
        $success := true$
        Return
    if $\exists\ i \in \{1...R\} | recommendationsAccepted[i] \neq 0$ **then**
        $recommendationsToChange := \{i \in 1...R :$ $recommendationsAccepted[i] \neq 0\}$
    else
        $recommendationsToChange := \{1...R\}$
    for $i \in recommendationsToChange$ **do**
        $(RCD, currentBase, CDB, CSS, CDS, neighborList) = refreshRecommendation(i, RCD, ...currentBase, CDB, CSS, CDS, neighborList)$
    if $\exists\ i:\ RCD(i) = 0$ *% not enough recommendations* **then**
        $success := false$
        Return
    % Propose to user and get his/her feedback
    $recommendationsAccepted = getUserFeedback(user, RCD)$
    for $i \in \{1...R\} | recommendationsAccepted(i) \neq 0$ **do**
        $currentBase(i) := RCD(i)$
        $Update\ neighborList(i)$ with neighbors of $currentBase(i)$
        $Update\ A$ by adding $RCD(i)$
    if $(neighborList(i) = \emptyset,\ \forall i \in \{1...R\})\ or\ CSS = \emptyset$ **then**
        $sufficientData := false$
        $success := false$

The main recommendation mechanism is the function *recommend*(.) (Algorithm 3). The function regards a specific *user*, receiving as input a source set (CSS) and a destination set CDS (corresponding source-destination pairs). If the source and destination sets do not intersect (which will always be the case when the function is called by Algorithm 1 or Algorithm 2), recommendation paths from the sources to their corresponding destinations are created, as follows. First, a current base (*currentBase*), consisting of past choices of the user is initialized with the first R elements of the products-sources, CSS. CDB keeps the corresponding destinations of *currentBase* and it is initialized with the first R elements of CDS. At each instance exactly R recommendations are provided to the user, one for each source-destination pair given by corresponding entries of *currentBase*, CDB. These recommendations are included in the vector RCD. A flag R-dimensioned vector, denoted as *recommendationsAccepted* retains the information whether the user chose each of the R recommendations of the previous instance or not (it is initialized to the zero-vector). The user feedback is introduced via the function $getUserFeedback$(.), which can be either real user behavior or its emulation (as it will be the case in our evaluation part). The set *recommendationsToChange* is constructed, containing the indices of the successful recommendations, if such exist, or all the values from 1 to R if the user did not accept any of the previous recommendations. Next, the recommendations whose index appears in the set *recommendationsToChange* are refreshed via the function $refreshRecommendation$(.). If the latter returns zero value for a recommendation then the result of *recommend*(.) will be $success = false$, as we will not be able to provide R recommendations to the user. Based on the user's feedback, the accepted recommendations replace the corresponding elements of the *currentBase*, their neighbors' list is updated (*neighborList*) and also the corresponding products are added to those already chosen by the user in his/her set A (along with the corresponding vertical connections to the user-product graph). If at some step there are no more neighbors for any element of the current base or the set CSS empties, the algorithm terminates unsuccessfully, whereas if the intersection of the *currentBase* and the destinations set CDS is nonempty (namely the user has reached a destination), the algorithm terminates successfully. We should note that the *neighborList*(i) is a set including the neighbors of the corresponding element of the current base, i.e., *currentBase*(i).

Finally, Algorithm 4 shows how a specific (e.g., the i^{th}) entry of the recommendation vector RCD is updated. Firstly, neighbors of the i^{th} element of the current base (*currentBase*) are picked, until one neighbor with hyperbolic distance (function *hyperbolicDistance*(.) [7]) from the destination less than the one of the *currentBase* element is found (i.e., a greedy neighbor is found). This greedy neighbor is set as the i^{th} element of the recommendation vector. If no greedy neighbors are found, the i^{th} source-destination pair is replaced by another pair of CSS, CDS, if existing, and the process of searching for greedy neighbors is repeated, otherwise $refreshRecommendation$(.) returns zero for $RCD(i)$.

The method of refreshing recommendations indicates that a rejection of a recommendation is mostly interpreted as a non-selection instead of a clear rejection. Specifically, if the user accepts $K \geq 1$ recommendations, only the K accepted recom-

Algorithm 4: Function $refreshRecommendation(.)$

```
Input: i (index in RCD), RCD (current recommendations)
  currentBase, CDB, CSS, CDS, neighborList
Global variables: A := products already chosen by user
Output: RCD, currentBase, CDB, CSS, CDS,
  neighborList
while neighborList(i) ≠ ∅ do
    Extract a neighbor n from neighborList(i)
    if n ∉ A and n ∉ RCD then
        if hyperbolicDistance(n, CDB(i)) <
          hyperbolicDistance(currentBase(i), CDB(i))
          then
            RCD(i) := n; Return
    if neighborList(i) = ∅ then
        if CSS ≠ ∅ then
            currentBase(i) :=
              Extract first element from CSS
            CDB(i) := Extract first element from CDS
            Update neighborList(i)
        else
            RCD(i) := 0; Return
```

mendations are renewed and the rest *R-K* remain unchanged. The intuition is that if the user selected a recommended product, then he/she definitely liked it. This does not mean, however, that the user did not like the other products that were recommended. Perhaps he/she did not have time to choose them, so the algorithm recommends them again. On the other hand, if the user does not accept/select any recommendation, then all recommendations are renewed, as most probably the user did not find any recommendation interesting.

For completeness purposes, we need to note that for the parameters, we assume $R \leq sourcesPerRound \leq totalSources$.

V. Evaluation Results

In this section, we provide performance results for the proposed approaches based on appropriately defined metrics and by reasonably emulating user behavior. These results constitute an initial offline evaluation aiming at exemplifying the efficiency of the proposed path-based recommender system and studying the impact of several parameters controlling its performance.

In order to emulate the user behavior in a realistic manner, we considered $X = 5$ user classes and $Y = 10$ product classes. Each user and product class is characterized by a feature vector indicating its characteristics. The number of features is constant and equal to 100, the same for both products and users. Each class has higher weight values at a specific subset of features of the feature vector that mostly characterize it. Both users and products may belong to more than one class having a feature vector that is a linear combination of the feature vectors of the classes they belong to. For a pair of target user and recommended product, we compute the inner product between their feature vectors and consider the product as accepted/chosen by the user with probability proportional to this inner product. Thus, a user is more likely to select products similar to his/her profile, while in order to reduce bias there is always a chance of selecting irrelevant products in place of relevant ones, as it is expected in reality. A similar emulation of user behavior was followed in [17].

We initialize the two-layer interconnected graph (Section III) with a user graph of power-law form, as it is the case for most online social networks [18]. The number of nodes is 100, the number of links is 265 and the exponent of the power-law distribution is around 3. The product graph consists of 1000 products which are considered as videos and their connections are determined by the co-views metric (Section III). Computing the latter requires the vertical links of the two-layer interconnected system, which are constructed based on the inner product similarity between the feature vectors of pairs of users and products. We examine several density values of the initial graph of the vertical connections and their impact on the success of the proposed recommender system. Specifically, in each of the figures that follow, Graph 1 has the highest density of vertical connections followed by Graph 2 and Graph 3. Different density values of the vertical interconnections also imply different densities of the product graph (due to applying the co-views metric for building the product graph). Specifically, G_1 has 127274 links, G_2 has 25526, and finally G_3 has 2141 links. Finally the product graph is embedded in hyperbolic space via the Rigel embedding using a total number of 25 landmarks.

We evaluated the proposed *HRKD* and *HRUD* algorithms based on the following performance metrics.

- *Success rate/percentage*: For *HRKD*, it is defined as the number of target user-destination pairs for which *HRKD* is applied with success over all the target user-destination pairs for which *HRKD* is applied. For *HRUD*, it is defined as the number of target users for which *HRUD* is applied with success over all the target users for which *HRUD* is applied.
- *Average Minimum Hyperbolic Distance to the Destination*: The minimum hyperbolic distance to the destination is defined as the minimum hyperbolic distance of the product where *HRKD* or *HRUD* stopped towards the destination for all source-destination pairs considered in a recommendation round in case of non-success. This is averaged over all recommendation rounds.
- *Average Recommendation Precision*: Recommendation precision is defined as the number of recommendations that the target user has chosen over a recommendation round divided by all recommendations made by *HRKD* or *HRUD* for this round. The average is taken over all recommendations rounds.

In the results that follow, we vary the values of the parameters R and $roundsToRenewGraph$, i.e., the number of recommendations provided to the target user at each instance and the

number of rounds after which the product graph is renewed (via the co-views metric) to reflect the updates in the vertical links (i.e., new choices by the users). Their default values (if they are not varying) are $R = 5$ and $roundsToRenewGraph = 5$. The rest of the parameters are kept constant and will be specified separately for each one of the *HRKD*, *HRUD*. Each result (point in the figures) was obtained by averaging over 5 independent evaluation runs.

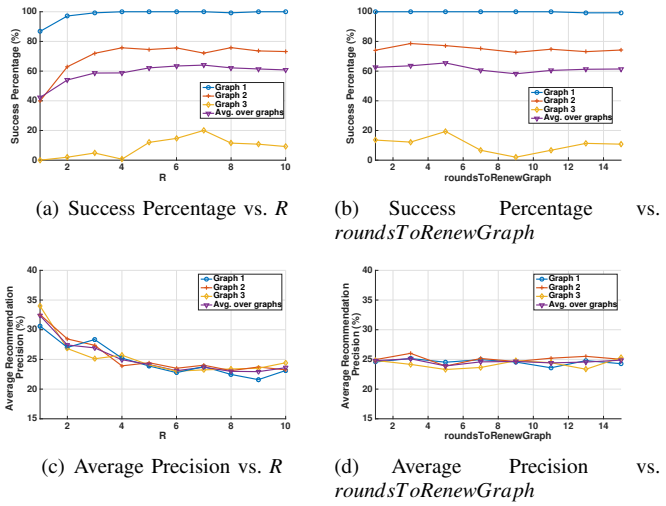
A. Evaluation of HRKD

For the evaluation of *HRKD*, we assign $maxAttempts = 10$, $sourcesPerRound = 20$, $p_w = 0.8$. The evaluation results for *HRKD* are shown in Fig. 2. Firstly, we observe that higher density of the vertical links (e.g., Graph 1) leads to higher success rate of *HRKD*. Also, in cases that *HRKD* is not successful, the minimum distance of the lastly recommended product to the user from the destination is small (lower average minimum hyperbolic distance to the destination metric). This is expected, as higher density of the vertical links means that more information is initially available for the users' choices leading to a more dense and informative product graph (constructed via the co-views metric).

In Fig. 2(a), we observe a concave behavior of the success rate of *HRKD* with respect to R. A maximum is obtained for R = 4-6 in the average case (purple curve). This is a nontrivial result, which favors the efficiency of the proposed recommender system by reducing its complexity per instance, i.e., less greedy neighbors of already chosen products by the user need to be searched for. From Fig. 2(b), we observe that the parameter $roundsToRenewGraph$ does not affect much the success rate of *HRKD*, favoring the application of higher values to reduce computational complexity (i.e., from reconstructing via the co-views metric and re-embedding in hyperbolic space the product graph). However, a slightly better behavior (i.e., higher success rate) is obtained for lower values of the parameter $roundsToRenewGraph$, especially for lower density product graphs, as expected.

Figs. 2(c), 2(d), show the results for the metric of precision, which is around 20-30%. Precision reduces with the increase of R, which is logical, as the more recommendations are provided to the user, the less the percentage of them selected by the user. Also, precision seems invariant with respect to the parameter $roundsToRenewGraph$ and with respect to the density of the graphs. In general, the values of precision leave space for further improvements in our proposed recommender system, since we need to propose around a triple number of products than those finally accepted by the user to build his/her path towards the target product. In other words, this result signifies that the greedy paths often encounter 'obstacles' (non-accepted products by the user) and need to be redirected. However, it is significant that the existence of multiple greedy paths between pairs of products allows redirection options, leading eventually to high success percentages.

At this point, we should also note that the aim of our algorithm is to achieve the acceptance of the recommendation of the target product by the user, i.e., the important metric is the success percentage. The recommendation precision shows only an indication of the re-directions of the greedy paths towards the target product.



(a) Success Percentage vs. R (b) Success Percentage vs. $roundsToRenewGraph$

(c) Average Precision vs. R (d) Average Precision vs. $roundsToRenewGraph$

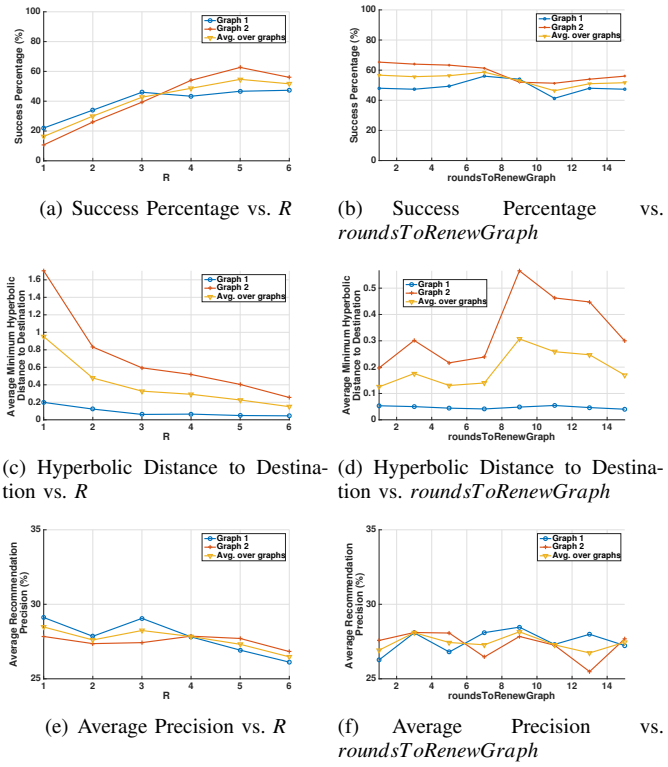
Fig. 2. Performance of the Algorithm *HRKD*

B. Evaluation of HRUD

For the evaluation of *HRUD*, we assign $maxAttempts = 2$, $totalSources = 25$, $sourcesPerRound = 10$, $p_w = 0.6$, $T_{\min} = 2.2$, $T_{\max} = 4.2$. The results for *HRUD* are shown in Fig. 3. The observations are in general similar to those for *HRKD*.

In Fig. 3(a), we observe that there is a concave behavior of the success rate of *HRUD* with respect to parameter R, presenting a maximum at around R = 4-5 that can be regarded as the optimal values of R for both graphs examined. From Fig. 3(b), we observe that the parameter $roundsToRenewGraph$ does not affect much the success rate of *HRUD*, favoring the application of higher values to reduce computational complexity. However, as expected, for both product graph densities, the success percentage is higher for lower values of the parameter $roundsToRenewGraph$, i.e., when the product graph is updated more often with the user choices. In addition, from Figs. 3(a), 3(b), we can observe that the improvements in the success percentage with the increase of the values for the parameter R and the decrease of those for the parameter $roundsToRenewGraph$ are more important for the graph with the lower density (red line). Compared to the algorithm *HRKD*, we obtain lower success rate values due to the different parameters applied in this case, and specifically, lower number of $sourcesPerRound$ and of $maxAttempts$.

In Figs. 3(c), 3(d), we show the values for the metric of the average minimum hyperbolic distance to the destination in case of the algorithm *HRUD*. We observe that Graph 1, which is characterized by a higher density in the product graph, has much lower values (i.e., better values) for the metric of the average minimum hyperbolic distance to the destination, which in this case is not affected much by the parameters R and $roundsToRenewGraph$. However, for Graph 2, the average minimum hyperbolic distance to the destination



(a) Success Percentage vs. R

(b) Success Percentage vs. *roundsToRenewGraph*

(c) Hyperbolic Distance to Destination vs. R

(d) Hyperbolic Distance to Destination vs. *roundsToRenewGraph*

(e) Average Precision vs. R

(f) Average Precision vs. *roundsToRenewGraph*

Fig. 3. Performance of the Algorithm *HRUD*

significantly reduces for higher R values, i.e., when we provide more recommendations to the user at each instance, and for lower values of the *roundsToRenewGraph* parameter, i.e., when we renew the product graph more often based on the user choices. We need to note here that the destinations are in hyperbolic distance between $T_{\min} = 2.2$, $T_{\max} = 4.2$ from the sources, i.e., from products that the user has already chosen at the beginning of each attempt. Thus, the hyperbolic distance to the target product achieved by *HRUD* is much smaller, i.e., although it failed it has approached very close to the target product, considering also that Rigel embedding preserves hop distances. One should recall that for this metric the averages are computed only over users (i.e., recommendation rounds) for which *HRUD* failed.

For the metric of recommendation precision shown in Figs. 3(e), 3(f), we can make exactly the same observations as for algorithm *HRKD*.

VI. Conclusion

In this paper we presented an alternative recommendation approach for large-scale online systems, which combines hyperbolic network embedding with greedy routing. We developed two recommendation algorithms, *HRKD* and *HRUD*, for the cases when the target product is known or it is inferred, and evaluated their performance with respect to key involved parameters. The path-based recommendation algorithms require local only information for their operation, thus their functionality can scale efficiently and provide diverse recommendations for users. Our future work will be involved in the implementation and evaluation of the approaches in an operational system with real users, signifying their practical merit. In addition, we will investigate another direction of our path-based recommendation framework [2] that regards the recommendations of common target products to pairs of users possibly bearing different and diverse interests, by locating rendezvous points of greedy paths starting from different products already chosen by each one of them.

References

[1] J. Leskovec, A. Rajaraman, J. Ullman, "Mining of Massive Datasets", *Cambridge University Press*, 2nd Ed., Dec. 2014.

[2] E. Stai, V. Karyotis, S. Papavassiliou, "A Hyperbolic Space Analytics Framework for Big Network Data and their Applications", *IEEE Network Mag.*, Vol. 30, No. 1, pp. 11-17, Jan.-Feb. 2016.

[3] J. Bobadilla, F. Ortega, A. Hernando, A. Gutierrez, "Recommender Systems Survey", *Elsevier Knowledge-Based Systems*, Vol. 46, pp. 109-132, Apr. 2013.

[4] B. Schafer, D. Frankowski, J. Herlocker, S. Sen, "Collaborative Filtering Recommender Systems", *The Adaptive Web*, Springer Lecture Notes in Computer Science, Vol. 4321, pp. 291-324, 2007.

[5] S.H. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions", *Int'l J. Math. Models & Meth. App. Science*, Vol. 1, No. 4, pp. 300-307, 2007.

[6] C. Porcel, A. Tejeda-Lorente, M.A. Martinez, E. Herrera-Viedma, "A Hybrid Recommender System for the Selective Dissemination of Research Resources in a Technology Transfer Office", *Information Sciences, Elsevier*, Vol. 184, No. 1, pp. 13-32, 2012.

[7] X. Zhao, A. Sala, H. Zheng, B.Y. Zhao, "Efficient Shortest Paths on Massive Social Graphs", *IEEE Coll. Communic.*, pp. 77-86, 2011.

[8] A. Cvetkovski, M. Crovella, "Hyperbolic Embedding and Routing for Dynamic Graphs", *IEEE INFOCOM*, pp. 1647-1655, April 2009.

[9] F. Papadopoulos, D. Krioukov, M. Boguñá, A. Vahdat, "Greedy Forwarding in Dynamic Scale-Free Networks Embedded in Hyperbolic Metric Spaces", *in Proc. of IEEE INFOCOM*, pp. 14-19, March 2010.

[10] F. Papadopoulos, C. Psomas, D. Krioukov, "Network Mapping by Replaying Hyperbolic Growth", *IEEE/ACM Trans. Netw.*, Vol. 23, No. 1, pp. 198-211, February 2015.

[11] V. Pouli, J.S. Baras, A. Arvanitis, "Increasing Recommendation Accuracy and Diversity via Social Networks Hyperbolic Embedding", *Proc. of 11th IEEE Consumer Communications and Networking Conference (CCNC)*, Jan. 2014.

[12] X. Ban, J. Gao, A. van de Rijt, "Navigation in Real-World Complex Networks through Embedding in Latent Spaces", *Proc. of ALENEX*, pp. 138-148, January 2010.

[13] J. Zhang, "Greedy Forwarding for Mobile Social Networks Embedded in Hyperbolic Spaces", *Proc. of the ACM SIGCOMM*, New York, NY, USA, pp. 555-556, 2013.

[14] F. Papadopoulos, M. Kitsak, M.A. Serrano, M. Boguñá, D. Krioukov, "Popularity vs. Similarity in Growing Networks", *Nature*, Vol. 489, pp. 537-540, Sept. 2012.

[15] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, "The YouTube Video Recommendation System", *Proc. 4th ACM Conference on Recommender Systems (RecSys)*, Sept. 2010.

[16] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, M. Aly, "Video Suggestion and Discovery for YouTube: Taking Random Walks Through the View Graph", *Proc. 17th Int'l Conference on World Wide Web (WWW)*, pp. 895-904, Apr. 2008.

[17] E. Stai, S. Kafetzoglou, E.E. Tsiropoulou, S. Papavassiliou, "A Holistic Approach for Personalization, Relevance Feedback & Recommendation in Enriched Multimedia Content", *Multimedia Tools and Applications, Springer*, pp. 1-44, 2016.

[18] V. Karyotis, E. Stai, S. Papavassiliou, "Evolutionary Dynamics of Complex Communications Networks", *CRC Press - Taylor & Francis Group*, 2013.