A Graph-Based Recommender for Enhancing the Assortment of Web Shops

Hans Friedrich Witschel, Emidio Galliè, Kaspar Riesen

University of Applied Sciences Northwestern Switzerland (FHNW),
Riggenbachstr. 16, 4600 Olten, Switzerland
{HansFriedrich.Witschel,Kaspar.Riesen}@fhnw.ch,emidio.gallie@students.fhnw.ch

Abstract In this work, we consider a situation where multiple providers (competitors) serve a common market, using a common infrastructure of sales channels. More specifically, we focus on multiple web shops that are run by the same web shop platform provider. Our goal is to recommend new items to complement the assortment of a provider, based on user behaviour in the other shops of the same platform. For this new problem, we propose to capture information on how items sell together in a shared product co-occurrence graph. We then adapt known graph-based recommenders to the problem. Further criteria for ranking recommended items are derived as part of a case study conducted in the context of IT web shops. They are combined with the scores of the graph recommenders in a final ranking function. We evaluate this function with data from our case study context and based on judgments of one shop owner. Our results show that a good ranking can be achieved, reflecting the needs of the shop owner.

1 Introduction

On-line retailers can gain much revenue from cross-selling. For instance, given that a customer put product A in her shopping basket, a web shop might recommend product B, because it is often sold together with A. But what about other products B' that are not in the shop's assortment, but might still sell very well together with A?

In this work, we consider a situation where multiple providers (competitors) serve a common market, using a common infrastructure of sales channels. More specifically, we focus on multiple web shops that are run by the same web shop platform provider. The platform is capable of collecting data about purchases made in each web shop. This situation is becoming more common as shop owners outsource the cost of shop maintenance or make their offerings available to brokers such as price comparison platforms. Examples can be found in many areas, e.g. portals where users can research and book flight offers of many airlines. The example used in our evaluation will be that of multiple web shops offering IT hard- and software which are run by the same provider.

In such a situation, it becomes possible to gather knowledge about user preferences and co-occurrence of products in shopping baskets *across multiple web shops* and use the knowledge to optimise the assortment of each web shop. That is, based on the knowledge that e.g. the two products A and B sell very well together in many web shops, a shop that offers only A could use that knowledge to infer that it should also offer B. Of course, shop owners must agree to trade such knowledge to their competitors – but this is probable since benefits will outweigh the potential losses in most cases. And of course each shop owner will ponder such recommendations carefully in order to keep focused on their particular market segment.

Therefore, the objective of this research was to build a recommender system that can help to optimise the assortment of competing web shops by collecting cross-sales knowledge across multiple shops. We analysed the requirements for such a recommender as part of a case study conducted in the above-mentioned context of IT web shops. We then implemented various variants of a recommender and evaluated them using real data from the case study.

2 Related Work

Our work resembles prior work from several areas:

- Firstly, it shares its goal with various attempts to optimise assortments of shops. Some of these are based on criteria such as profitability of products, but not related to cross-selling potentials [14], [8]. Others do look at cross-selling potentials, but only for one web shop [2], [3]. In [4], the authors look at specific problems in market basket analysis in multiple shops of the same retailer.
- Secondly, since our goal is to build a recommender, prior work in recommender systems is relevant. For the area of e-commerce, an overview of approaches can be found in [11]. As in many other areas, successful recommenders are based on collaborative filtering (e.g. [10]), but also association rule mining has been applied successfully (see [11]).
- Finally, we focus on graph-based recommendation: a graph is a natural way of representing similarities between users or items and has therefore been used in several recommender systems, e.g. [9]. Related to our approaches, researchers have built recommenders based on graph clustering [5] and on random walks [7], [13]. Random walks are usually biased towards a user: the score of a recommended item is derived by computing the probability of reaching the corresponding node via a random walk that starts at (or is biased towards) products already purchased or liked by the user.

Our problem is different from all previous work since we consider a multiple-provider setting where we would like to recommend items not to the end users visiting the shops, but to the web shop providers – but these recommendations should be based on the preferences of the end users. This introduces an additional level of complexity (comparing shopping baskets across multiple stores). Our

contribution is hence a study of how to best adapt known approaches to that modified recommendation problem.

3 Recommendation based on graph analysis

3.1 Analysis of requirements

In order to understand the requirements for an "assortment-optimising recommender", we studied the case of a web shop platform, on which 21 different web shops were run. All web shops can build their assortment from a common pool of products, but most shops offer only a comparatively small selection of items, depending on the market segment that they address.

We interviewed the providers of the platform and elicited the following requirements for a recommender system:

- The recommender should be able to identify new products for a shop that sell well together with parts of the existing assortment of the shop. This is our starting point (see above).
- Besides, the recommender should be able to explain its recommendations. When recommending a product B, it should list all products A already offered by a shop that caused the recommendation, e.g. because they were sold together with B in other shops.
- Finally, the recommender should be able to rank recommended products by their likelihood of enhancing a shop's assortment. For this ranking, several criteria were identified in addition to the likelihood of generating cross-sales – namely a) a product's general likelihood of generating revenue, b) the up-to-dateness of the product and c) the fit with the shop's assortment in terms of manufacturer (if a shop sells only products of manufacturer A, products of other manufacturers might not fit well). This last criterion was only discovered during the experiments (see below).

Based on these requirements, we built a recommender in several steps, as described in the next subsections.

3.2 Notation

First, the situation described in Section 1 is formalised as follows: The platform on which the web shops are run is described as a set $P = (w_1, ...w_n)$ of web shops w_i. A common pool of products (or items) $I = (i_1, ..., i_n)$ is available from which each web shop chooses its assortment. Hence, each web shop $w_i \in P$ is basically defined as a subset of the product pool $w_i \subseteq I$. For simplicity, a purchase b is a (typically small) subset of a web shop: $b \subseteq w_i$ for some $w_i \in P$ – i.e. we do not care about the quantity of each item that was purchased. The set of all purchases within a certain timeframe of analysis will be called B.

3.3 Building a product graph

A first step in building the recommenders consists in representing the knowledge about item co-occurrence across all web shops via a common product co-occurrence graph. This is a graph $G = (V, E)$ where the vertices represent all available items, i.e. $V = I$. The set of edges is established by analysing the co-occurrence of items within all purchases $b \in B$ that were recorded within a given time frame: two items u, v are connected by an undirected edge $(u, v) \in E$ if their joint occurrence in purchases significantly exceeds the number that would be expected by pure chance, given the individual occurrence frequencies of the two items. We used the likelihood ratio significance [6] to assign weights to edges – this significance measure was developed for co-occurrence of words in natural language sentences. Edges $(u, v) \in E$ are weighted by the likelihood ratio significance level $lr(u, v)$. The measure fits well here because the distribution of products in purchases is similarly skewed (power-law distributed) as the distribution of words in text.

3.4 Graph-based recommendation

Based on the co-occurrence graph G, we developed three variants of a graph-based recommender. The input of each variant is a web shop $w_i \subseteq V$ and the output is a set of items R_i that w_i is currently not offering (i.e. $R_i \cap w_i = \emptyset$). Each recommended item $r \in R_i$ has a score $s(r)$ that is assigned by the recommender. In all cases, we recommend items that are in close neighbourhood or "easily" reachable on a path of edges from the vertices w_i in graph G.

The first variant will be called the *Simple Recommender*. It recommends all items that are adjacent to at least one product already offered by the shop, i.e. $R_i = \{r \in V \setminus w_i | \exists j \in w_i : (r, j) \in E\}$. The recommended items r are then scored by the sum of weights of all edges that connect r with any $j \in w_i$, i.e. $s(r) = \sum_{j \in w_i} lr(j, r)$ (where $lr(j, r) > 0$ only if $(j, r) \in E$). The leftmost graphic in Figure 1 illustrates this variant: the items in w_i are highlighted in blue and recommended items $r \in R_i$ are in yellow/red. We can see that each $r \in R_i$ has at least one edge connecting it to a product in w_i. The color indicates the score (yellow = low score, red = high score). Assuming that all edges have equal weight (for simplicity), we will find that the red-colored node is ranked higher since it is connected to three nodes $j \in w_i$ whereas the yellow nodes are adjacent to one node from w_i only. The rationale behind this scoring is that we prefer items that will cross-sell with many existing products of the shop.

The second variant is based on a biased random walk on the product graph and will be called the *PageRank Recommender*. Here, we use PageRank with priors [12] and for any item j, the prior p_j is set to 0 if $j \notin w_i$ and $p_j = \frac{1}{|w_i|}$ for each $j \in w_i$. That is, we bias the random walk towards the initial assortment of shop w_i and hence score a recommended item r by the probability of visiting it on such a biased random walk. By applying n iterations of the biased PageRank algorithm, we obtain scores for all items and we include in R_i all items with a PageRank score greater than 0. This is depicted in the middle of Figure 1.

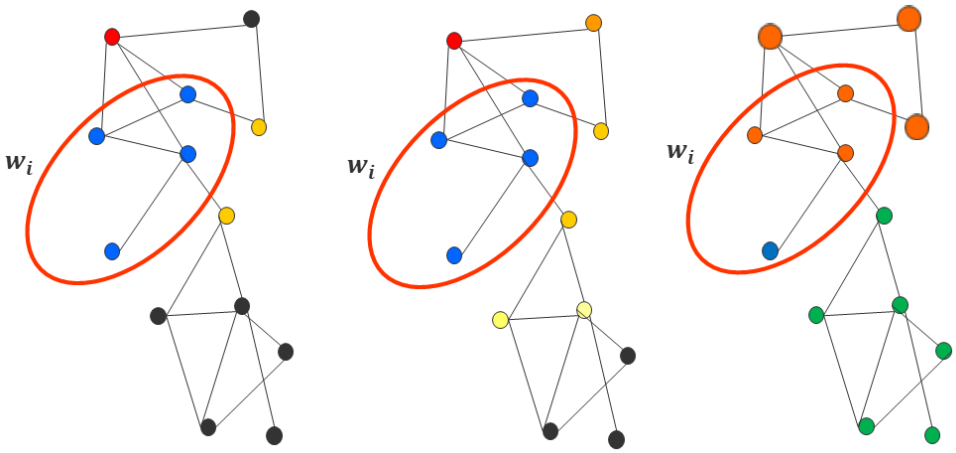

Figure 1. Illustration of the recommender variants Simple (left) PageRank (middle) and Cluster (right) on the same product graph (colour in on-line proceedings)

Again, the items in w_i are in blue and recommended items after two iterations of PageRank are in yellow/orange/red indicating their scores. Here, a recommended item will be scored highly if there are many short paths leading from items in w_i to that item.

The third variant uses graph clustering and is hence called the *Cluster Recommender*: the product co-occurrence graph is clustered via the approach described in [1], resulting in a set of disjoint clusters $C = (c_1, ..., c_m)$, $c_i \cap c_j = \emptyset$. In Figure 1 in the rightmost graphic, the cluster membership of a product is indicated by colors. There are three clusters (orange, blue and green). This recommender includes all items in R_i, which belong to a cluster $c \in C$ that contains at least one of w_i's products, i.e. $R_i = \bigcup_{c \in C, c \cap w_i \neq \emptyset} c \setminus w_i$. In Figure 1, the recommended items from the orange cluster are highlighted (bigger size of node). All recommended items r from the same cluster c_j receive the same score, namely $s(r) = \frac{|c_j \cap w_i|}{|c_j|}$, i.e. the proportion of the cluster that is already offered in the web shop. In Figure 1, the recommended orange items receive a score of 0.5 since 3 out of 6 orange items are already in w_i. The rationale of this scoring method is that we prefer clusters that are already well covered by a shop – i.e. that represent well the market segment addressed by the shop – and that we wish to complete the shop's assortment with the yet missing items of these clusters.

Obviously, both the PageRank Recommender and the Cluster Recommender may recommend items r to a web shop w_i that have never been sold together with any item $j \in w_i$ whereas this is not possible for the Simple Recommender. In terms of explanations (see Section 3.1), the Simple Recommender can explain its recommendations r by listing all those $j \in w_i$ that are connected to r. Similarly, recommendations r of the Cluster Recommender can be explained by those $j \in$

w_i that are in the same cluster as r. For the PageRank Recommender, however, there are no direct explanations available.

3.5 Ranking function

As mentioned in Section 3.1, the final score of a recommended product should also reflect other criteria besides $s(r)$. In particular, in accordance with the gathered requirements, we defined the following additional variables for each recommended item r:

- The estimated amount of money that web shop w_i lost because it did not offer r, defined by $l(r) = price(r) \cdot P(r) \cdot |B_i|$ where B_i is the set of all purchases that happened in web shop w_i and $P(r)$ is the maximum likelihood estimate of r's probability to be purchased across all web shops (i.e. $P(r) = \frac{|\{b \in B | r \in b\}|}{|B|}$). This reflects the general popularity of the item r.
- The estimated amount of money that the shop w_i lost through missed cross-sales. We estimate this "lost cross-sales" criterion for a recommended item r by looking at each $j \in w_i$ that was ever purchased together with r. For each j, we compute the confidence, i.e. the probability of r being purchased, given that someone purchases j: $conf(j, r) = P(r|j) = \frac{|\{b \in B | j \in b, r \in b\}|}{|\{b \in B | j \in b\}|}$. Hence, the amount $c_j(r)$ lost through missed cross-sales of r related to j can be calculated by multiplying this probability with the number of times j was sold in w_i and r's price: $c_j(r) = conf(j, r) \cdot |B_i^j| \cdot price(r)$ where B_i^j denotes the set of all purchases of product j within shop w_i. Finally, the whole lost cross-sales amount for is computed as $c(r) = \sum_{j \in w_i} c_j(r)$. Note that this only works for recommenders that can explain their recommendations, i.e. for Cluster and Simple recommender; for the PageRank recommender, the corresponding score is always 0.
- The number of weeks $w(r)$ since r was last sold in any other shop.
- The fit $m(r)$ between r and the shop's existing assortment in terms of manufacturer: $m(r) = \frac{|B_i^{man(r)}|}{|B_i|}$ where $B_i^{man(r)}$ is the set of all purchase transactions in w_i that contained an item produced by the same manufacturer who produced r.

We then used the inverse of $w(r)$ and normalised all scores by mapping them into the interval $[0, 1]$. The normalised scores are referred to as $s'(r)$, $c'(r)$, $l'(r)$, $w'(r)$ and $m'(r)$. The final ranking function $f(r)$ is a weighted sum of the normalised scores:

$$f(r) = \alpha_1 s'(r) + \alpha_2 l'(r) + \alpha_3 c'(r) + \alpha_4 w'(r) + \alpha_5 m'(r), \quad \sum_{i=1}^{5} \alpha_i = 1 \qquad (1)$$

4 Experimental Setup

We have performed a preliminary evaluation of our recommender variants by applying it to the data of the above-mentioned web shop platform, comprising

21 web shops and purchase data dating back one year from the date of the experiments. The goal of our evaluation was a) to verify that the recommenders were able to rank promising items highly and b) to compare the recommenders against each other and get a first impression on how to configure the ranking function, i.e. tune the weights in equation 1.

We selected one shop w_i and then built a gold standard as follows:

- We generated recommendations for w_i both with the Simple Recommender and with the Cluster Recommender – using weights (rather arbitrarily) of $\alpha_1 = 0.4$, $\alpha_2 = 0.1$, $\alpha_3 = 0.3$, $\alpha_4 = 0.2$ and $\alpha_5 = 0.0$[1] – and then selected 100 items randomly from the union of the two recommendation sets. The resulting set of items hence contained items with both high and low ranking function scores and both ones with direct connections to items in w_i in the product graph and ones without such connection. We will call this set our gold standard G_i.
- We gave the unsorted list of 100 items to the owner of w_i, providing also the scores $s'(r)$, $l'(r)$, $w'(r)$ and $c'(r)$ for each recommended item r, and its explanation (see example in Table 1). We asked the shop owner to state, for each recommended item r, whether he considered r a useful extension of his product assortment (yes or no). We additionally asked for an explanation of each decision.

Recommendation	$s'(r)$	$l'(r)$	$w'(r)$	$c'(r)$	Explanations
LogiLink Video-/Audio-Adapter – DisplayPort/HDMI	**		*	*	1. Elgato EyeTV Hybrid + AKG K317 headset
Kingston Memory DDR3 8GB 1600MHz	****	*	***	**	1. Microsoft Windows 7 Professional SP1, 64Bit 2. Kingston SSDNow V300 - Solid-State-Disk - 120 GB

Table 1. Example recommendations for our chosen web shop, showing scores and explanations. Scores have been transformed into star ratings (between 0 and 5 stars).

We then used this gold standard to run different experiments – for each setting, the recommenders produced a ranked list R_i of recommendations, which was then filtered such that it contained only recommendations $r \in R_i \cap G_i$. We then evaluated the filtered rankings using the measures precision, precision at rank 20 (P@20), recall and average precision.

5 Results and Discussion

We first analysed the explanations that the shop owner gave regarding each decision to either accept or reject a recommendation $s(r)$, especially focusing on those cases where the recommendation was rejected – which concerned 40 out of

[1] Note that α_5 was 0 because we only discovered the "manufacturer fit" criterion during the experiments (see Section 5 below).

the 100 items in the gold standard. Since the gold standard was built such as to include also items with low scores, it is no surprise that a large portion of them was rejected – this was actually even intended in order to allow an analysis of rejection reasons. The explanations given by the shop owner were free text, we hence needed to code them. We thus found the following reasons for rejection of recommendations:

1. Product does not fit well because of manufacturer (and/or web shop owner has exclusive contracts with other manufacturers): 11 mentions
2. Web shop owner believes that there is no demand for the product in his addressed market segment: 8 mentions
3. Product is considered too expensive: 7 mentions
4. The manufacturer does not offer good conditions (would need to order large quantities): 6
5. The product does not fit the addressed market segment: 4 mentions
6. Product is outdated: 2 mentions

The analysis shows that most rejection criteria are covered by our ranking function: outdated products (reason 6) and ones that do not fit the addressed market segment (reason 5) should be ranked down by criteria $w(r)$ (number of weeks since recommendation r was last sold) and $s(r)$ (initial score of the recommender), respectively. Similarly, missing demand (reason 2) should be covered by $l(r)$, i.e. the estimated revenue lost by not offering r. For covering reason 1, we introduced additionally the "manufacturer fit" ranking criterion $m(r)$.

We can also see that not all rejection criteria were covered by our ranking function – i.e. it seemed that our requirements analysis (see Section 3.1) was incomplete. However, reasons 3 and 4 (product too expensive or no good conditions available from manufacturer) are hardly reflected in our purchase data (although one may argue that reason 4 is partially covered by ranking criterion $m(r)$). We hence did not further adapt the ranking function.

In a second step, we ran all three recommender variants on our gold standard. For each variant, we tuned the weight vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$ with a brute force approach: we generated all possible vectors by varying each α_i between 0 and 1 with a step width of 0.05, but only considering those $\boldsymbol{\alpha}$ where $\sum \alpha_i = 1$, resulting in 9697 different $\boldsymbol{\alpha}$. The results of the best runs for each recommender variant are shown in Table 2.

Recommender	best $\boldsymbol{\alpha}$	AP	P@20	P	R
Simple	(0.3, 0.2, 0.2, 0.0, 0.3)	0.7946	1.0	0.63	0.87
PageRank	(0.25, 0.2, 0.3, 0.0, 0.25)	0.8573	1.0	0.61	1.0
Cluster	(0.0, 0.2, 0.4, 0.0, 0.4)	0.7041	0.95	0.58	0.82

Table 2. Results of applying recommender variants to the gold standard, measures in average precision (AP), precision at rank 20 (P@20), precision (P) and recall (R).

We first notice that all runs achieve a very good precision at rank 20 (P@20), indicating that it is indeed possible to tune the recommender such that its top-ranked recommendations are perceived as useful by the shop owner.

When comparing recommender variants, we notice that the Cluster Recommender has suboptimal results, inferior to the others in terms of all evaluation measures. The PageRank recommender has a higher average precision and recall than the Simple Recommender, yet scores slightly lower on precision. This is not surprising since the Simple Recommender is not able to retrieve any recommendations that are not directly connected to a $j \in w_i$ – hence necessarily has lower recall. Overall, the PageRank recommender seems to deliver the best performance.

Finally, we look at the optimal weight vector $\boldsymbol{\alpha}$. When we focus on the PageRank Recommender, we find that the only criterion that does not seem to play a role is $w(r)$, i.e. the number of weeks since a recommended item was last sold (actually, we find that $\alpha_4 = 0$ across all recommender variants). This is quite consistent with the qualitative analysis above that showed only two cases where a recommendation was rejected because it was perceived as outdated. The importance of the other criteria is roughly balanced, also consistent with the qualitative analysis. Of course, this analysis is only valid for one shop and its strategy – we expect to derive different optimal weight vectors for other shops. But the example shows that it seems possible to adapt the recommender to the needs of a shop owner.

6 Conclusions

In this work, we have considered the situation of a platform through which several (competing) providers offer products (or services) to end users. Our goal was to recommend portfolio optimisations to those providers, based on user preferences. More specifically, we propose to analyse co-occurrence of products in shopping baskets across multiple shops (providers) and use the derived information to recommend additional products with cross-selling potential to complete the existing assortment of the shop. We have introduced a common product co-occurrence graph to capture cross-selling knowledge across providers and, based on that graph, derived adaptations of several graph-based recommendation algorithms to our new problem. These were combined with other ranking criteria – that we derived from interviews with the providers of a web shop platform – into a final ranking function to be used for ranking recommended items.

We have experimentally verified the approach in a preliminary evaluation, building a gold standard in cooperation with a shop owner. Our results show that the recommender variants are capable of ranking good recommendations highly. The recommender based on random walks on the product graph achieved the best performance. The results also indicate that the additional criteria that our ranking function comprises are useful and that it is possible to tune the corresponding weights to meet the needs of the shop owner.

In the future, we would like to work with a larger sample of shops to verify our findings. And we would like to measure the actual (cross) sales achieved by adding recommended items to a shop instead of using the subjective judgment of the shop owner for evaluation.

Acknowledgments

This work is supported by the Swiss Commission for Technology and Innovation, CTI, (Grant No. 15330.1 PFES-ES).

References

1. Biemann, C.: Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems. In: Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing. pp. 73–80 (2006)
2. Brijs, T., Swinnen, G., Vanhoof, K., Wets, G.: Using association rules for product assortment decisions. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '99. pp. 254–260 (1999)
3. Brijs, T., Swinnen, G., Vanhoof, K., Wets, G.: Building an Association Rules Framework to Improve Product Assortment Decisions. Data Mining and Knowledge Discovery 8(1), 7–23 (2004)
4. Chen, Y.L., Tang, K., Shen, R.J., Hu, Y.H.: Market basket analysis in a multiple store environment. Decision Support Systems 40(2), 339–354 (2005)
5. Demir, G.N., Uyar, A.S., Ögüdücü, S.G.: Graph-based Sequence Clustering Through Multiobjective Evolutionary Algorithms for Web Recommender Systems. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. pp. 1943–1950 (2007)
6. Dunning, T.: Accurate Methods for the Statistics of Surprise and Coincidence. Computational Linguistics 19(1), 61–74 (1994)
7. Fouss, F., Pirotte, A., Renders, J.M., Saerens, M.: Random-Walk Computation of Similarities Between Nodes of a Graph with Application to Collaborative Recommendation. IEEE Transactions on Knowledge and Data Engineering 19(3), 355–369 (2007)
8. Hansen, P., Heinsbroek, H.: Product selection and space allocation in supermarkets. European Journal of Operational Research 3(6), 474–484 (1979)
9. Huang, Z., Chung, W., Ong, T.H., Chen, H.: A Graph-based Recommender System for Digital Library. In: Proceedings of the 2Nd ACM/IEEE-CS Joint Conference on Digital Libraries. pp. 65–73 (2002)
10. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing 7(1), 76–80 (2003)
11. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of Recommendation Algorithms for e-Commerce. In: Proceedings of the 2nd ACM Conference on Electronic Commerce. pp. 158–167. EC '00 (2000)
12. White, S., Smyth, P.: Algorithms for Estimating Relative Importance in Networks. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 266–275 (2003)
13. Zhang, L., Zhang, K., Li, C.: A Topical PageRank Based Algorithm for Recommender Systems. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 713–714. SIGIR '08 (2008)
14. Zufryden, F.S.: A Dynamic Programming Approach for Product Selection and Supermarket Shelf-Space Allocation. Journal of the Operational Research Society 37(4), 413–422 (1986)