Les Cahiers du GERAD ISSN: 0711–2440

Resolving sets and integer programs for recommender systems

A. Hertz

G–2019–12

January 2019

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2019
– Bibliothèque et Archives Canada, 2019

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2019
– Library and Archives Canada, 2019

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Resolving sets and integer programs for recommender systems

Alain Hertz

GERAD, Montréal (Québec), Canada, H3T 2A7

Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7 other example

alain.hertz@gerad.ca

January 2019
Les Cahiers du GERAD
G–2019–12

Abstract: Recommender systems make use of different sources of information for providing users with recommendations of items. Such systems are often based on collaborative filtering methods which make automatic predictions about the interests of a user by collecting taste information from many users. As an alternative approach, we propose to use the concept of resolving set that allows to determine the preferences of the users with a very limited number of ratings. We also show how to make recommendations when user ratings are imprecise or inconsistent, and we indicate how to take into account situations where users possibly don't care about the attribute values of some items. All recommendations are obtained in a few seconds by solving integer programs.

1 Introduction

Recommender systems provide personalized recommendations of products or services to users based on their previous searches, purchases, or ratings. The main challenge in such systems is to predict the rating that a user would give to an item. There is a huge literature dedicated to recommender systems. Since the first paper [5] in the mid 1990s, research in this area got diversified in various directions. Most recommender systems are based on collaborative filtering methods which predict unobserved ratings by exploiting correlations between ratings across a population of users. But other algorithms, such as content based filtering and hybrid filtering, are also popular. For surveys on this topic, we recommend the recent papers by Akhil and Shelbi [1], and by Bobadilla et al. [3].

The aim of this paper is to propose an alternative approach for recommender systems with multiple boolean attributes. It is based on the graph theoretical concept of *resolving set* which is defined as follows. Consider a connected undirected graph G, and let $d(u, v)$ be the distance between two vertices u and v in G. A vertex x *resolves* two vertices u and v if $d(x, u) \neq d(x, v)$. A subset R of vertices is a *resolving set* for G if every two vertices in G are resolved by at least one vertex of R. The problem of determining a resolving set of minimum size is NP-hard [8]. It was introduced independently by Slater [9] and by Harary and Melter [6]. It arises in many areas, including robot navigation [8], telecommunication [2] and chemistry [4]. We show in the next section that when all users precisely indicate the distance between rated items and their expectations, then resolving sets for the n-cube correspond to small sets of recommendations that are sufficient to predict the preferences of all users.

The use of numerical rating scales (e.g., a 5-star rating) often results in imprecise ratings. Also, user ratings may be inconsistent. For example, it is not rare that a user gives two different ratings to two items having the same attribute values. This can be due to a missing attribute in the system. To deal with such situations, we define, in Section 3, a measure that evaluates how every item fits with known user ratings, and we then embed this measure into an integer program to generate a set of recommendations. While the users of a recommender system typically like or don't like the attributes characterizing the items, it may also happen that they don't care about some of them. We show in Section 4 how to deal with such situations.

All proposed models are integer programs with few variables and constraints, which make it possible to generate recommendations in a few seconds.

2 Resolving sets

Let A be an ordered set of n boolean attributes of a recommender system, and let I be a set of items. A vector $\boldsymbol{v}^i = (v^i_1, \ldots, v^i_n) \in \{0, 1\}^n$ can be associated with every item $i \in I$ so that $v^i_j = 1$ if i has the jth attribute in A, and $v^i_j = 0$ otherwise. For example, if I is a set of restaurants in a recommender system with $n = 3$ boolean attributes, the first one being 'low cost', the second one 'offers vegetarian food', and the third one 'with an outdoor terrace', then vector $(1, 0, 1)$ is associated with all low cost restaurants that have an outdoor terrace, but do not offer vegetarian food.

Given a set U of users, we can also associate a vector $\boldsymbol{w}^u = (w^u_1, \ldots, w^u_n) \in \{0, 1\}^n$ with each user $u \in U$, so that $w^u_j = 1$ if u 'likes' the jth attribute in A, and $w^u_j = 0$ otherwise. For the above example, the vector $(0, 1, 1)$ would be associated with all users interested in expensive restaurants with vegetarian food and an outdoor terrace.

The *Hamming distance* $d(\boldsymbol{x}, \boldsymbol{y})$ between two vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$ in $\{0, 1\}^n$ is the number of integers j such that $x_j \neq y_j$. Let Q_n be the n-dimensional hypercube, also called n-cube, with vertex set $\{0, 1\}^n$, and where two vertices $\boldsymbol{x}$ and $\boldsymbol{y}$ are linked with an edge if and only if $d(\boldsymbol{x}, \boldsymbol{y}) = 1$. To simplify the writing, we use notation $x_1 \ldots x_n$ (with no comma and no parenthesis) instead of $(x_1, \ldots, x_n)$ for a vertex $\boldsymbol{x} \in Q_n$. The Hamming distance $d(\boldsymbol{v}^i, \boldsymbol{w}^u)$ between the vertices that represent item i and user u is the number of edges in a shortest path from $\boldsymbol{v}^i$ to $\boldsymbol{w}^u$ in Q_n. This distance is the number of attribute values of i that do not correspond to the preferences of u. For the

above example, the distance $d(101, 011)$ between item i with $\boldsymbol{v}^i = 101$ and user u with $\boldsymbol{w}^u = 011$ is 2, since user u is interested in expensive restaurants with a vegetarian option, while i has opposite values for these two attributes.

Our objective is to match every user $u \in U$ with a vertex $\boldsymbol{x}$ in Q_n so that the preferences of u are known and recommendations can then be made by selecting items i with $\boldsymbol{v}^i = \boldsymbol{x}$, or having a small distance $d(\boldsymbol{v}^i, \boldsymbol{x})$ to $\boldsymbol{x}$. In other words, for every $u \in U$, we want to determine $\boldsymbol{x} \in Q_n$ such that $\boldsymbol{x} = \boldsymbol{w}^u$.

As mentioned in the previous section, a resolving set for Q_n is a subset R of its vertices such that, for every two vertices $\boldsymbol{x}, \boldsymbol{y}$ in Q_n, there is at least one vertex $\boldsymbol{r}$ in R with $d(\boldsymbol{r}, \boldsymbol{x}) \neq d(\boldsymbol{r}, \boldsymbol{y})$. Note that :

- $R = \{000, 100\}$ is not a resolving set for Q_3 since, for example, $d(000, 001) = d(000, 010) = 1$ and $d(100, 001) = d(100, 010) = 2$ (i.e., no vertex in R resolves 001 and 010);
- $R = \{000, 110\}$ is not a resolving set for Q_3 since, for example, $d(000, 100) = d(000, 010) = 1$ and $d(110, 100) = d(110, 010) = 1$ (i.e., no vertex in R resolves 100 and 010);
- $R = \{000, 111\}$ is not a resolving set for Q_3 since, for example, $d(000, 100) = d(000, 010) = 1$ and $d(111, 100) = d(111, 010) = 2$ (i.e., no vertex in R resolves 100 and 010).

By symmetry, we conclude that there is no resolving set for Q_3 with two vertices. There are however resolving sets with three vertices, for example $R = \{\boldsymbol{r}^1 = 000, \boldsymbol{r}^2 = 010, \boldsymbol{r}^3 = 100\}$. Indeed, we observe in Figure 1 that for every $\boldsymbol{x}$ in Q_3, we get a distinct vector $d(\boldsymbol{r}^1, \boldsymbol{x})d(\boldsymbol{r}^2, \boldsymbol{x})d(\boldsymbol{r}^3, \boldsymbol{x})$. Hence, for the above example, if we ask a user u to rate three restaurants i_1, i_2, i_3 with $\boldsymbol{v}^{i_1} = \boldsymbol{r}^1$, $\boldsymbol{v}^{i_2} = \boldsymbol{r}^2$, and $\boldsymbol{v}^{i_3} = \boldsymbol{r}^3$, and if u indicates that he does not agree with 2 attribute values in i_1, 1 in i_2, and 3 in i_3, we can conclude that $\boldsymbol{w}^u = 101$, which means that he is looking for a low cost restaurant with an outdoor terrace, but is not interested in vegetarian food.

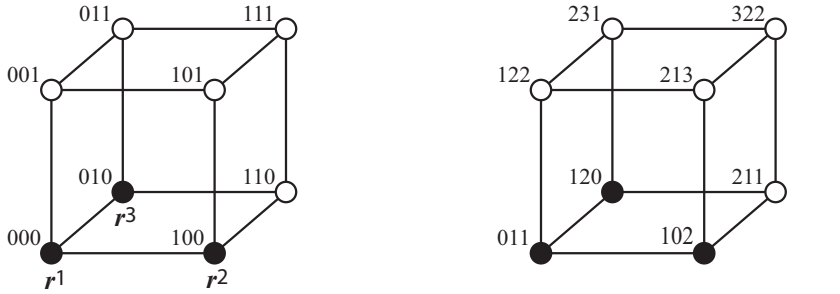

Figure 1: A resolving set $\{\boldsymbol{r}^1, \boldsymbol{r}^2, \boldsymbol{r}^3\}$ in Q_3.

Let R be a subset of vertices in Q_n. In order to check whether R is a resolving set for Q_n, one can consider all pairs $\boldsymbol{x}, \boldsymbol{y}$ of vertices in Q_n and determine whether there exists $\boldsymbol{r} \in R$ such that $d(\boldsymbol{r}, \boldsymbol{x}) \neq d(\boldsymbol{r}, \boldsymbol{y})$. Such a technique requires $O(2^{2n}|R|)$ distance computations, which becomes infeasible for large values of n. Recently, Hertz [7] has observed that one can determine if $R = \{\boldsymbol{r}^1, \ldots, \boldsymbol{r}^{|R|}\}$ is a resolving set for Q_n by solving the following constrained maximization problem:

$$\max \quad d(\boldsymbol{x}, \boldsymbol{y}) \tag{1}$$

$$\text{s.t.} \quad d(\boldsymbol{r^i}, \boldsymbol{x}) = d(\boldsymbol{r^i}, \boldsymbol{y}) \qquad \forall i = 1, \ldots, |R| \tag{2}$$

$$\boldsymbol{x}, \boldsymbol{y} \in \{0, 1\}^n \tag{3}$$

Indeed, notice that if R is not a resolving set for Q_n, then constraints (2) are satisfied by at least two distinct vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, while all feasible solutions have $\boldsymbol{x} = \boldsymbol{y}$ if R is a resolving set for Q_n. Hence, the optimal value of the above maximization problem is 0 if and only if R is a resolving set for Q_n.

We now show how to rewrite the above maximization problem as an integer program with linear constraints and a linear objective. For every $j = 1, \ldots, n$, we have $|r^i_j - x_j| \in \{-1, 0, 1\}$, $x_j \in \{0, 1\}$ and $r^i_j \in \{0, 1\}$, which implies $|r^i_j - x_j| = (r^i_j - x_j)^2$, $(x_j)^2 = x_j$, and $(r^i_j)^2 = r^i_j$. Hence,

$$d(\boldsymbol{r^i}, \boldsymbol{x}) = \sum_{j=1}^{n} |r^i_j - x_j| = \sum_{j=1}^{n} (r^i_j - x_j)^2 = \sum_{j=1}^{n} x_j(1 - 2r^i_j) + \sum_{j=1}^{n} r^i_j.$$

Similarly,

$$d(\boldsymbol{r^i}, \boldsymbol{y}) = \sum_{j=1}^{n} y_j(1 - 2r^i_j) + \sum_{j=1}^{n} r^i_j.$$

Constraints (2) can therefore be rewritten as :

$$\begin{aligned} & \sum_{j=1}^{n} x_j(1 - 2r^i_j) + \sum_{j=1}^{n} r^i_j = \sum_{j=1}^{n} y_j(1 - 2r^i_j) + \sum_{j=1}^{n} r^i_j && \forall i = 1, \ldots, |R| \\ \Leftrightarrow \quad & \sum_{j=1}^{n} (1 - 2r^i_j)(x_j - y_j) = 0 && \forall i = 1, \ldots, |R| \end{aligned} \tag{4}$$

These are linear equations since x_j and y_j are the variables, while the values r^i_j are given numbers in $\{0, 1\}$. We cannot use the same trick for the computation of the objective $d(\boldsymbol{x}, \boldsymbol{y})$ since $\sum_{j=1}^{n} x_j(1 - 2y_j) + \sum_{j=1}^{n} y_j$ is not a linear function. Consider however the following constrained maximization problem:

$$\max \quad \sum_{j=1}^{n} q_j$$

$$\text{s.t.} \quad \left.\begin{array}{l} q_j \le x_j + y_j \\ q_j \le 2 - x_j - y_j \end{array}\right\} \qquad \forall j = 1, \ldots, n \tag{5}$$

$$x_j, y_j, q_j \in \{0, 1\} \qquad \forall j = 1, \ldots, n \tag{6}$$

Clearly, $d(\boldsymbol{x}, \boldsymbol{y})$ is the optimal value of this integer program. Indeed, constraints (5) and (6) impose $q_j = 0$ when $x_j = y_j$, while q_j can take value 0 or 1 when $x_j \neq y_j$. By maximizing $\sum_{j=1}^{n} q_j$, we therefore count the number of indices j such that $x_j \neq y_j$, which is exactly the Hamming distance $d(\boldsymbol{x}, \boldsymbol{y})$. In summary, one can determine if $R = \{\boldsymbol{r}^1, \ldots, \boldsymbol{r}^{|R|}\}$ is a resolving set for Q_n by solving the following integer program:

$$\max \quad \sum_{j=1}^{n} q_j \tag{7}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} (1 - 2r^i_j)(x_j - y_j) = 0 \qquad \forall i = 1, \ldots, |R| \tag{8}$$

$$q_j \le x_j + y_j \qquad \forall j = 1, \ldots, n \tag{9}$$

$$q_j \le 2 - x_j - y_j \qquad \forall j = 1, \ldots, n \tag{10}$$

$$q_j, x_j, y_j \in \{0, 1\} \qquad \forall j = 1, \ldots, n \tag{11}$$

This integer program has $3n$ variables and $2n + |R|$ constraints, which is considered as a problem of small size for today's computers. The optimal solution can typically be obtained in less than a second.

For $\boldsymbol{x} \in Q_{n-1}$, we denote by $\boldsymbol{x}0$ (resp. $\boldsymbol{x}1$) the vertex of Q_n obtained from $\boldsymbol{x}$ by adding 0 (resp. 1) as nth component. It is not difficult to prove that if $\{\boldsymbol{r}^1, \ldots, \boldsymbol{r}^{|R|}\}$ is a resolving set for Q_{n-1}, then $\{\boldsymbol{r}^1 0, \ldots, \boldsymbol{r}^{|R|} 0, \boldsymbol{r}^1 1\}$ is a resolving set for Q_n. For example, since $\{0\}$ is a resolving set for Q_1, it follows that $\{00, 01\}$ is a resolving set for Q_2, and $\{000, 010, 001\}$ is a resolving set for Q_3. Hence, resolving sets with n vertices for Q_n are easy to determine. A procedure is described in [7] that determines smaller resolving sets. It is a swapping algorithm which can be summarized as follows, where σ is any given positive number.

1 Choose an initial set R of σ vertices in Q_n
2 **while** *R is not a resolving set for Q_n and no stopping criterion is met* **do**
3 | Choose a vertex $\boldsymbol{x} \in R$, and replace it with a vertex $\boldsymbol{y} \notin R$;
4 **end**
5 **if** *R is not a resolving set for Q_n then* **then** Return the message "No resolving set of size σ was found";
6 **else** return R;

Algorithm 1: A swapping algorithm for the generation of a resolving set of size σ.

The above procedure chooses an initial set R of σ vertices in Q_n (instruction 1), and then repeatedly replaces a vertex $\boldsymbol{x} \in R$ with a vertex $\boldsymbol{y} \notin R$ until R is a resolving set for Q_n, or a stopping criterion is met (instructions 2–4). The above integer program is used at instruction 3 to determine whether R is a resolving set for Q_n. More details are given in [7] where it is shown that there are resolving sets of 15 vertices for Q_{27}, which means that 15 ratings are sufficient to determine the preferences of a user among $2^{27} = 134,217,728$ item categories. Examples of resolving sets for Q_3, Q_{10}, Q_{20} and Q_{27} are given in Figure 2.

000	0110010110	01000100101001000000	110110001010110100110000100
100	0001010101	10000100110110010100	000101001001010000010011111
010	1010111111	11101000100011110000	010111101011101111111011111
	1110000101	00001101101000111100	011110110000111011110001001
	1100011000	00101111000011000100	000110110011111000001101011
	1000000110	11001011101110001000	111011100001110100011010011
	1011010100	10100001111001001100	101110101011001001010000010
		10111101101010000010	111100101110100010000011011
		01010001010000100000	001100101001111110110110001
		11011100011011011000	110000111011111101000010100
		00000000000010001010	010000101000001100101000011
		00000000000010001011	101010001010011110011111111
			010000010111010110111011001
			101101000011101100101011000
			011001100010000001110110101

Figure 2: Resolving sets for Q_3, Q_{10}, Q_{20}, and Q_{27}.

The resolving sets generated by the above algorithm are valid for all users. However, one might prefer to determine resolving sets that take into account the items that a user has eventually already rated. Assume, for example, that a user has rated a set I' of items and let R' be the corresponding subset of vertices in Q_n (i.e., $R' = \{\boldsymbol{v}^i | i \in I'\}$). Rather than generating a resolving set from scratch, it is preferable to determine a set R'' of vertices so that $R = R' \cup R''$ is a resolving set for Q_n. To determine the preferences of a user, it is then sufficient to ask him to rate items associated with vertices in R''. The generation of such a resolving set R can be achieved with minor modifications to the above swapping algorithm : the initial set R of σ vertices built at instruction 1 should contain R' as subset, and when a vertex $\boldsymbol{x}$ is removed from R at instruction 3, it is chosen among those that do not belong to R'.

Assume, for example, that $R' = \{001, 110\}$. Note that $R = R' \cup \{111\}$ is not a resolving set for Q_3 since 011 and 101 are not resolved by any vertex in R. Indeed, $d(001, 011) = d(001, 101) = 1$, $d(110, 011) = d(110, 101) = 2$, and $d(111, 011) = d(111, 101) = 1$. By symmetry, $R' \cup \{\boldsymbol{x}\}$ is not a resolving for every $\boldsymbol{x} \in Q_3$. However, it is not difficult to check that $R = R' \cup \{111, 011\}$ is a resolving set for Q_3 with four vertices, but with only two item types that the user has not rated. This is better than the resolving set $\{000, 010, 100\}$ that contains only three vertices, but three item types that the user has not rated.

3 Imprecise and inconsistent ratings

The message contained in the last section is that very few ratings are sufficient to determine the preferences of a user. We have implicitly assumed that all ratings can be translated into Hamming distances. In other words, we have supposed that all ratings precisely indicate how many attribute values of every rated item correspond to the user expectations. However, this rarely happens in practice. One possible reason is that the use of numerical rating scales limits the number of possible values to some integer s which can be much smaller than the number n of attributes. Also, the item characteristics that a user is looking for may not appear as attributes. For example, a user is possibly interested in restaurants near his home, while no attribute is related to this preference. User ratings are also possibly inconsistent. For example, after having looked for a low cost restaurant and having rated it very well because it gave him complete satisfaction, it is possible that a user decides to turn to a high-end restaurant. He will then not be satisfied with recommendations for low cost restaurants. As a consequence, the user may give different ratings to items having the same attribute values.

The objective of this section is to show how integer programs can be used to make recommendations, knowing that the available ratings are possibly imprecise and inconsistent. For this purpose, consider a function τ that translates ratings into distances. The worst ratings should be translated into distance n, and the best ones into distance 0. For example, consider an s-star rating, where users give the highest score of s stars to items that perfectly match their preferences, and the lowest score of 1 star when they did not like any of the attribute values of the rated item. We can then define $\tau : \{1, \ldots, s\} \to [0, n]$ as follows:

$$\tau(r) = n - \frac{n(r-1)}{s-1}. \tag{12}$$

For example, for a 5-star rating of items with $n = 3$ boolean attributes, we have $\tau(1) = 3$, $\tau(2) = \frac{9}{4}$, $\tau(3) = \frac{3}{2}$, $\tau(4) = \frac{3}{4}$, and $\tau(5) = 0$.

Consider a set I' of items that a user $u \in U$ has already rated, let ρ_i be the score given by u to $i \in I'$, and let $\delta_i = \tau(\rho_i)$ be the corresponding distance. If $\boldsymbol{x}$ is the vertex that represents u in Q_n and if all ratings precisely indicate how many attribute values correspond to the expectations of u, then we should have $d(\boldsymbol{v}^i, \boldsymbol{x}) = \delta_i$ for all $i \in I'$. The following function $f_{I'} : Q_n \to [0, n|I'|]$ therefore estimates the error made by assuming $\boldsymbol{x} = \boldsymbol{w}^u$:

$$f_{I'}(\boldsymbol{x}) = \sum_{i \in I'} |d(\boldsymbol{v}^i, \boldsymbol{x}) - \delta_i|.$$

Let R' be the subset of vertices in Q_n associated with the subset I' of items rated by user u (i.e., $R' = \{\boldsymbol{v}^i | i \in I'\}$). In the previous section, we have described a procedure that determines a subset R'' of vertices so that $R' \cup R''$ is a resolving set for Q_n. We can use function $f_{I'}$ to discriminate among different resolving sets, so that the vertices in R'' (which correspond to item categories that u will have to rate) best fit the preferences of u. This can be done by replacing the original objective function $d(\boldsymbol{x}, \boldsymbol{y})$ in objective (1) of the previous section with the following new objective:

$$(n|I'||R''| + 1)d(\boldsymbol{x}, \boldsymbol{y}) - \sum_{\boldsymbol{r} \in R''} f_{I'}(\boldsymbol{r}) \tag{13}$$

We thus maximize this new objective (13) under constraints (2) and (3). We then have two possible cases:

- If $R' \cup R''$ is not a resolving set for Q_n, then constraints (2) are satisfied by at least two distinct vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Since $f_{I'}(\boldsymbol{r}) \leq n|I'|$ for all $\boldsymbol{r} \in R''$, the value of objective (13) is then at least $n|I'||R''| + 1 - n|I'||R''| = 1$.
- If $R' \cup R''$ is a resolving set for Q_n, then $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ (i.e., $\boldsymbol{x} = \boldsymbol{y}$) for all $\boldsymbol{x}, \boldsymbol{y}$ satisfying constraints (2), and the value of objective (13) is then at most 0.

Hence, a strictly positive optimal value means that $R' \cup R''$ is not a resolving set, and we therefore swap a vertex in R'' with a vertex not in $R' \cup R''$. When the optimal value is at most 0, we know that $R' \cup R''$ is a resolving set, but instead of stopping the swapping algorithm, we can continue with the objective of maximizing (13) over all resolving sets, which is equivalent to minimizing $\sum_{\boldsymbol{r} \in R''} f_{I'}(\boldsymbol{r})$ (since $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ when $R' \cup R''$ is a resolving set).

For example, suppose that user u has rated two items i_1 and i_2, giving them $\rho_{i_1} = 4$ stars and $\rho_{i_2} = 2$ stars in a 5-star scale. We thus have $\delta_{i_1} = \tau(4) = \frac{3}{4}$ and $\delta_{i_2} = \tau(2) = \frac{9}{4}$. Assume $\boldsymbol{v}^{i_1} = 000$ and $\boldsymbol{v}^{i_2} = 100$. If we try to add vector 111 to $R' = \{000, 100\}$, we do not get a resolving set since 101 and 110 are both at distance 2 from 000, and at distance 1 from 100 and 111. However, if we add 001, 010, 110, or 101 to R', we get four different resolving sets. We have :

- $f_{I'}(001) = |d(000, 001) - \frac{3}{4}| + |d(100, 001) - \frac{9}{4}| = \frac{1}{2}$;
- $f_{I'}(010) = |d(000, 010) - \frac{3}{4}| + |d(100, 010) - \frac{9}{4}| = \frac{1}{2}$;
- $f_{I'}(101) = |d(000, 101) - \frac{3}{4}| + |d(100, 101) - \frac{9}{4}| = \frac{5}{2}$;
- $f_{I'}(110) = |d(000, 110) - \frac{3}{4}| + |d(100, 110) - \frac{9}{4}| = \frac{5}{2}$.

Hence resolving sets $\{000, 100, 001\}$ and $\{000, 100, 010\}$ are preferred to $\{000, 100, 101\}$ and $\{000, 100, 110\}$. Instead of minimizing $\sum_{\boldsymbol{r} \in R''} f_{I'}(\boldsymbol{r})$ over all resolving sets $R' \cup R''$, a variant would consist in assigning a weight $\omega(\boldsymbol{r})$ to every vertex $\boldsymbol{r} \in Q_n$ and to maximize $\sum_{\boldsymbol{r} \in R''} \omega(\boldsymbol{r})$. Such a weight $\omega(\boldsymbol{r})$ could be, for example, the average score that a population of users has given to items $i \in I$ with $\boldsymbol{v}^i = \boldsymbol{r}$. Hence, when a new user u enters the system (i.e., there is no available rating for u), the system would generate a resolving set containing popular items.

Assume a user $u \in U$ has rated a subset I' of items. Because of the above-mentioned imprecisions and inconsistencies, there is typically no vertex in Q_n that perfectly matches these ratings. Hence, instead of generating a resolving set, one can prefer to determine a set $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m\}$ of m vertices in Q_n that best fit with the user ratings, where m is any fixed positive integer. In other words, we aim to generate vertices in Q_n that are close to $\boldsymbol{w}^u$. This can be achieved by solving the following constrained minimization problem:

$$\min \quad \sum_{k=1}^{m} f_{I'}(\boldsymbol{x}^k) \tag{14}$$

$$\text{s.t.} \quad \boldsymbol{x}^k \neq \boldsymbol{x}^{k'} \qquad \forall 1 \leq k < k' \leq m \tag{15}$$

$$f_{I'}(\boldsymbol{x}^k) \leq f_{I'}(\boldsymbol{x}^{k+1}) \qquad \forall k = 1 \ldots, m-1 \tag{16}$$

$$\boldsymbol{x}^k \in \{0,1\}^n \qquad \forall k = 1, \ldots, m \tag{17}$$

Constraints (15) and (16) impose that vertices $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m$ must be all different and sorted by increasing value $f_{I'}(\boldsymbol{x}^k)$. This minimization problem can be rewritten as follows, where $\alpha_{i,k} = |d(\boldsymbol{x}^k, \boldsymbol{v}^i) - \delta_i|$ for all $i \in I'$ and $1 \leq k \leq m$.

$$\min \quad \sum_{k=1}^{m} \sum_{i \in I'} \alpha_{i,k} \tag{18}$$

$$\text{s.t.} \quad \boldsymbol{x}^k \neq \boldsymbol{x}^{k'} \qquad \forall 1 \leq k < k' \leq m \tag{19}$$

$$\left.\begin{array}{l} \alpha_{i,k} \geq d(\boldsymbol{v}^i, \boldsymbol{x}^k) - \delta_i \\ \alpha_{i,k} \geq \delta_i - d(\boldsymbol{v}^i, \boldsymbol{x}^k) \end{array}\right\} \qquad \forall k = 1 \ldots, m \quad \forall i \in I' \tag{20}$$

$$\sum_{i \in I'} \alpha_{i,k} \leq \sum_{i \in I'} \alpha_{i,k+1} \qquad \forall k = 1 \ldots, m-1 \tag{21}$$

$$\boldsymbol{x}^k \in \{0,1\}^n \qquad \forall k = 1, \ldots, m \tag{22}$$

Constraints (20) impose $\alpha_{i,k} \geq |d(\boldsymbol{v}^i, \boldsymbol{x}^k) - \delta_i|$, and equality follows from the fact that we minimize the sum over all variables $\alpha_{i,k}$. Hence,

$$\sum_{i \in I'} \alpha_{i,k} = \sum_{i \in I'} |d(\boldsymbol{v}^i, \boldsymbol{x}^k) - \delta_i| = f_{I'}(\boldsymbol{x}^k)$$

which means that constraints (21) and objective (18) are equivalent to constraints (16) and objective (14). We can now transform this constrained minimization problem into a integer program with linear constraints and a linear objective as follows. Using the same trick as in the previous section, constraints (20) can be rewritten as

$$\alpha_{i,k} \geq \sum_{j=1}^{n} x_j^k(1-2v_j^i) + \sum_{j=1}^{n} v_j^i - \delta_i \qquad \forall k = 1\ldots,m \quad \forall i \in I' \tag{20.1}$$

$$\alpha_{i,k} \geq \delta_i - \sum_{j=1}^{n} x_j^k(1-2v_j^i) - \sum_{j=1}^{n} v_j^i \qquad \forall k = 1\ldots,m \quad \forall i \in I' \tag{20.2}$$

Also, $\boldsymbol{x}^k \neq \boldsymbol{x}^{k'}$ is equivalent to $d(\boldsymbol{x}^k, \boldsymbol{x}^{k'}) \geq 1$. Hence, as was done with equations (5) and (6) in the previous section, we can replace constraints (19) and (22) with the following ones.

$$\left.\begin{array}{l} \beta_j^{k,k'} \leq x_j^k + x_j^{k'} \\ \beta_j^{k,k'} \leq 2 - x_j^k - x_j^{k'} \end{array}\right\} \qquad \forall 1 \leq k < k' \leq m \quad \forall j = 1\ldots,n \tag{19.1}$$

$$\sum_{j=1}^{n} \beta_j^{k,k'} \geq 1 \qquad \forall 1 \leq k < k' \leq m \tag{19.2}$$

$$\beta_j^{k,k'} \in \{0,1\} \qquad \forall 1 \leq k < k' \leq m \quad \forall j = 1\ldots,n \tag{22.1}$$

$$x_j^k \in \{0,1\} \qquad \forall k = 1,\ldots,m \quad \forall j = 1\ldots,n \tag{22.2}$$

We can thus consider objective (18) under constraints (19.1), (19.2), (20.1), (20.2), (21), (22.1), and (22.2), which is a integer program. It has $O(nm^2 + m|I'|)$ variables and constraints. For example, if we are looking for only one recommendation (i.e., $m = 1$), then we have $n + |I'|$ variables (n variables x_j^k and $|I'|$ variables $\alpha_{i,k}$) and $2|I'|$ constraints (those of type (20.1) and (20.2)).

4 Don't care values

It may happen that some users don't care about some of the item attributes. Their ratings will then not depend on the values of these attributes. For our example with restaurants, a user that does not care about eating outside or inside will give the same rating to items i_1 and i_2 if $v_1^{i_1} = v_1^{i_2}$, $v_2^{i_1} = v_2^{i_2}$, $v_3^{i_1} = 0$, and $v_3^{i_2} = 1$

To handle such situations, we propose to slightly modify the models of the previous section by associating a vector $\boldsymbol{w}^u \in \{-1,0,1\}^n$ to every user $u \in U$, so that $w_j^u = -1$ if u does not like the jth attribute, $w_j^u = 0$ if u does not care about it, and $w_j^u = 1$ if u likes it. For our example, if a user u looks for an expensive restaurant with vegetarian food, but does not care about the existence of an outdoor terrace, we then have $\boldsymbol{w}^u = -110$.

If $w_j^u = 0$, the ratings of u do not depend on the value of the jth attribute. To take this into account, we have to replace the Hamming distance by a new one which, given a vertex $\boldsymbol{v} \in \{0,1\}^n$ and a vertex $\boldsymbol{x} \in \{-1,0,1\}^n$, counts the number of components j with $v_j = 1$ and $x_j = -1$, or $v_j = 0$ and $x_j = 1$. For this purpose, we consider function $d' : \{0,1\}^n \times \{-1,0,1\}^n \to \{0,\ldots,n\}$ defined as follows:

$$d'(\boldsymbol{v},\boldsymbol{x}) = \sum_{j=1}^{n} q_j \tag{23}$$

where

$$\left.\begin{array}{l} q_j \geq x_j(1-2v_j) \\ q_j \leq x_j(\frac{1}{2} - v_j) + \frac{1}{2} \\ q_j \in \{0,1\} \end{array}\right\} \qquad \forall j = 1,\ldots,n \tag{24}$$

Hence,

- if $v_j = 0$, then $x_j \leq q_j \leq \frac{1}{2}(x_j + 1)$, which means that $q_j = 0$ if $x_j = -1$ or 0, and $q_j = 1$ if $x_j = 1$;
- if $v_j = 1$, then $-x_j \leq q_j \leq \frac{1}{2}(1 - x_j)$, which means that $q_j = 0$ if $x_j = 0$ or 1, and $q_j = 1$ if $x_j = -1$.

In other words, this new distance between a vector $\boldsymbol{v} \in \{0,1\}^n$ and a vector $\boldsymbol{x} \in \{-1,0,1\}^n$ is equal to the number of indices j such that $v_j = 0$ and $x_j = 1$, or $v_j = 1$ and $x_j = -1$, which is exactly what we wanted.

We can now determine a set of m vectors that best fit with the known ratings of a user with the following constrained minimization problem, which is similar to minimizing (18) under constraints (19), (20), (21), (22), except that the Hamming distance d in constraints (20) is replaced by the new distance d', and variables x_j^k can now take value -1:

$$\min \quad \sum_{k=1}^{m} \sum_{i \in I'} \alpha_{i,k} \tag{25}$$

$$\text{s.t.} \quad \boldsymbol{x}^k \neq \boldsymbol{x}^{k'} \qquad \forall 1 \leq k < k' \leq m \tag{26}$$

$$\left.\begin{array}{l} \alpha_{i,k} \geq d'(\boldsymbol{v}^i, \boldsymbol{x}^k) - \delta_i \\ \alpha_{i,k} \geq \delta_i - d'(\boldsymbol{v}^i, \boldsymbol{x}^k) \end{array}\right\} \qquad \forall k = 1 \ldots, m \quad \forall i \in I' \tag{27}$$

$$\sum_{i \in I'} \alpha_{i,k} \leq \sum_{i \in I'} \alpha_{i,k+1} \qquad \forall k = 1 \ldots, m-1 \tag{28}$$

$$\boldsymbol{x}^k \in \{-1,0,1\}^n \qquad \forall k = 1, \ldots, m \tag{29}$$

Constraints (26) are equivalent to $\sum_{i=1}^n |x_i^j - x_i^k| \geq 1$, which can be rewritten with the following linear model:

$$\left.\begin{array}{rcl} \frac{1}{2}(x_j^k - x_j^{k'}) & \leq & a_j^{kk'} \\ a_j^{kk'} & \leq & \frac{1}{2}(x_j^k - x_j^{k'}) + 1 \\ b_j^{kk'} & \leq & 2a_j^{kk'} \\ c_j^{kk'} & \leq & 2(1 - a_j^{kk'}) \\ b_j^{kk'} - c_j^{kk'} & = & x_j^k - x_j^{k'} \\ a_j^{kk'} & \in & \{0,1\} \\ b_j^{kk'}, c_j^{kk'} & \in & \{0,1,2\} \end{array}\right\} \qquad \forall 1 \leq k < k' \leq m \quad \forall j = 1, \ldots, n \tag{26.1}$$

$$\sum_{j=1}^{n} (b_j^{kk'} + c_j^{kk'}) \geq 1 \qquad \forall 1 \leq k < k' \leq m \tag{26.2}$$

Indeed, let us show that constraints (26.1) imply $b_j^{kk'} + c_j^{kk'} = |x_j^k - x_j^{k'}|$. The difference $x_j^k - x_j^{k'}$ is an integer in $\{-2,-1,0,1,2\}$, and it is easy to check that the above constraints impose the following:

- if $x_j^k > x_j^{k'}$ then $a_j^{kk'} = 1$, $c_j^{kk'} = 0$, and $b_j^{kk'} = x_j^k - x_j^{k'}$;
- if $x_j^k < x_j^{k'}$ then $a_j^{kk'} = 0$, $b_j^{kk'} = 0$, and $c_j^{kk'} = x_j^{k'} - x_j^k$;
- if $x_j^k = x_j^{k'}$ then $b_j^{kk'} = c_j^{kk'} = 0$.

It follows that equations (26.1) and (26.2) impose $\sum_{i=1}^n |x_i^j - x_i^k| \geq 1$, which is equivalent to $\boldsymbol{x}^k \neq \boldsymbol{x}^{k'}$ for $1 \leq k < k' \leq m$. As explained above, constraints (27) can be rewritten as follows:

$$\left.\begin{array}{rcl} \alpha_{i,k} \geq & \sum_{j=1}^{n} q_j^{ik} - \delta_i \\ \alpha_{i,k} \geq & \delta_i - \sum_{j=1}^{n} q_j^{ik} \end{array}\right\} \qquad \forall k = 1, \ldots, m \quad \forall i \in I' \tag{27.1}$$

$$\left.\begin{array}{rcl} q_j^{ik} \geq & x_j^k(1 - 2v_j^i) \\ q_j^{ik} \leq & x_j^k(\frac{1}{2} - v_j^i) + \frac{1}{2} \\ q_j^{ik} \in & \{0,1\} \end{array}\right\} \qquad \forall k = 1, \ldots, m \quad \forall i \in I' \quad \forall j = 1, \ldots, n \tag{27.2}$$

We can thus determine a set of m vectors that best fit with the known ratings of a user by considering objective (25) under constraints (26.1), (26.2), (27.1), (27.2), (28), and (29), which is an integer program with $O(nm^2 + nm|I'|)$ variables and constraints. For example, if we are looking for only one recommendation (i.e., $m = 1$), then we have $n + n|I'| + |I'|$ variables (n variables x_j^k, $n|I'|$ variables q_j^{ik}, and $|I'|$ variables $\alpha_{i,k}$), and $5n + 2|I'| + 2n|I'|$ constraints ($5n$ of type (26.2), $2|I'|$ of type (27.1), and $2n|I'|$ of type (27.2)).

To illustrate the difference between the recommendations generated by the above integer program and those produced by the integer program of Section 3, we consider a recommender system with $n = 5$ boolean attributes. We suppose that a user $u \in U$ has rated a set I' of 24 items with a 5-star scale. The ratings ρ_i as well as the corresponding distances $\delta_i = \tau(\rho_i)$ are shown on the left-hand side of Figure 3. Note that some items belong the same category (they have the same attribute values) while the user has given them different scores. This is the case, for example, for items 1 and 16 which are both represented by vector 01100 while one score is $\rho_1 = 5$, and the other is $\rho_{16} = 3$. This illustrates possible inconsistencies in user ratings.

i	$\boldsymbol{v}^i$	ρ_i	δ_i
1	01100	5	0
2	00110	3	2.5
3	01110	5	0
4	01011	5	0
5	00110	5	0
6	00001	5	0
7	00100	3	2.5
8	01011	4	1.25
9	01100	5	0
10	11000	4	1.25
11	10001	4	1.25
12	01001	4	1.25
13	10000	5	0
14	01000	4	1.25
15	00110	4	1.25
16	01100	3	2.5
17	10000	3	2.5
18	11001	2	3.75
19	00100	3	2.5
20	01001	3	2.5
21	10000	3	2.5
22	00110	4	1.25
23	00011	3	2.5
24	00000	1	5

k	$\boldsymbol{x}^k$	$f_{I'}(\boldsymbol{x}^k)$
1	01010	28
2	01000	32.5
3	01110	32.5
4	00010	32.5
5	01100	32.5

The 5 best recommendations with like (1) and don't like (0) options

k	$\boldsymbol{x}^k$	$g_{I'}(\boldsymbol{x}^k)$
1	-1-10	23.5
2	-1-1-	24
3	-11--	24
4	-111-	24.5
5	01-10	25

The 5 best recommendations with like (1), don't like (0), and don't care (-) options

Figure 3: Comparisons of the models with and without don'care values.

The model of Section 3 has produced the five best recommendations, assuming that the user likes or does not like every item attribute. We observe that all recommendations have a 0 at the first and last components, which means that the user seems to prefer items without the first and the fifth attribute. The best recommendation has value $f_{I'}(\boldsymbol{x}^1) = 28$, which means that the average error over all rated items is $\frac{28}{24} = 1.17$. For example, for the first item, we would expect $\delta_1 = 0$ while $d(01100, 01010) = 2$.

For comparison, we give the solution generated by the integer program of this section, with $m = 5$. Every recommendation $\boldsymbol{x}^k$ is represented by a vector with components in $\{0, -, 1\}$ (instead of $\{-1, 0, 1\}$),

where "0" stands for "don't like", "-" for "don't care", and "1" for "like". We indicate the error $g_{I'}(\boldsymbol{x}^k)$ of each recommendation $\boldsymbol{x}^k$, where

$$g_{I'}(\boldsymbol{x}) = \sum_{i \in I'} |d'(\boldsymbol{x}, \boldsymbol{v}^i) - \delta_i|$$

is similar to $f_{I'}(\boldsymbol{x})$, except that distance d' is used instead of the Hamming distance d. We observe that all recommended items have a 1 as second component, which means that the user seems to prefer items with the second attribute. For comparison, only the first three recommendations obtained with the previous model had a 1 as second component. Also, looking at the first four recommendations, the user does not seem to care about the first attribute, while the previous model was suggesting that the user does not like it. The best recommendation has value $g_{I'}(\boldsymbol{x}^1) = 23.5$, which means that the average error over all rated items is $\frac{23.5}{24} = 0.98$. For example, for the first item, we would expect $\delta_1 = 0$ while $d'(\boldsymbol{v}^1, \boldsymbol{x}^1) = 1$.

5 Conclusion

We have described integer programs that produce recommendations when all item attributes have boolean values. We have first supposed that when a user rates an item, he always precisely indicates how many attribute values correspond to his expectations. With such an assumption, using the concept of resolving sets, we have shown that very few ratings are needed to determine the preferences of the users. For example, 15 ratings are sufficient to determine the preferences of a user among more than 134 million item categories.

To take into account the fact that user ratings are typically imprecise and inconsistent, we have defined a measure that indicates how much an item category fits with a set I' of known ratings, and we have then embedded this measure into an integer program that generates the m best recommendations, where m is any fixed number. We have finally modified this model to take into account the fact that users possibly don't care about some of the item attributes. All integer programs have $O(n)$ variables and constraints (if m and I' are considered as fixed numbers) which means that recommendations can be generated in a few seconds.

The proposed approach appears as interesting because it makes recommendations not only to new users for which no rating is available, but also to those who possibly do not care about some of the attributes, and for which we have inconsistent and imprecise ratings. As final validation, the proposed integer programs need to be implemented and tested by recommender system owners, and compared to other popular techniques such as collaborative filtering method that make recommendations by collecting information from many users.

References

[1] P. Akhil and J. Shelbi. A survey of recommender system types and its classification. International Journal of Advanced Research in Computer Science, 8(9):486–491, 2017.

[2] Z. Beerliova, F. Eberhard, T. Erlebach, A. Hall, M. Hoffmann, M. Mihalak, and L. Ram. Network discovery and verification. IEEE Journal on Selected Areas in Communications, 24(12):2168–2181, 2006.

[3] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. Knowledge-Based Systems, 46:109–132, 2013.

[4] G. Chartrand, L. Eroha, M. Johnson, and O. Oellermann. Resolvability in graphs and the metric dimension of a graph. Discrete Applied Mathematics, 105:99–113, 2000.

[5] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. ACM Transactions on Information Systems, 22(1):143–177, Jan. 2004.

[6] F. Harary and R. Melter. On the metric dimension of a graph. Ars Combinatoria, 2:191–195, 1976.

[7] A. Hertz. An ip-based swapping algorithm for the metric dimension and minimal doubly resolving set problems in hypercubes. Optimization Letters, 08 2017. URL: https://doi.org/10.1007/s11590-017-1184-z.

[8] S. Khuller, B. Raghavachari, and A. Rosenfeld. Landmarks in graphs. Discrete Applied Mathematics, 70(3):217–229, 1996.

[9] P. Slater. Leaves of trees. Congressus Numerantium, 14:549–559, 1975.