Mining Interesting Itemsets using Submodular Optimization

Jaroslav Fowkes and Charles Sutton
University of Edinburgh, Edinburgh, EH8 9AB, UK
{jfowkes,csutton}@inf.ed.ac.uk

Abstract

We propose a novel technique to retrieve itemsets that best explain a transaction database by leveraging a simple probabilistic model. Our approach is the first to infer such interesting itemsets directly from the transaction database using submodular function optimization and in so doing avoids many of the pitfalls commonly present in frequent itemset mining algorithms. Our proposed approach is theoretically simple, straightforward to implement, trivially parallelizable and exhibits good performance as we demonstrate on both synthetic and real-world examples.

1 Introduction

Over the past ten years, submodular function optimization has been applied to a wide variety of problems in machine learning, including structure learning (Narasimhan & Bilmes, 2004), information gathering (Krause & Guestrin, 2011) and document summarization (Lin & Bilmes, 2011). These successes raise the possibility that a submodular formulation would also be useful for other problems involving the interpretation and understanding of data.

Itemset mining (Han et al., 2007) is a popular technique for exploratory data analysis within data mining, especially for market basket data in which each data point, called a *transaction*, is a set of items drawn from a common universe. The goal of itemset mining is to return a set of small itemsets that represent common patterns in the transaction database. In the vast majority of existing research, this is formalized as the problem of *frequent itemset mining*, which is to return the collection of all itemsets that occur with a frequency greater than some user-specified minimum. Although appealing algorithmically, from a statistical perspective this formulation has a number of issues, such as that many frequent itemsets contain items that are individually frequent but uncorrelated, and that the list of frequent itemset is typically long, highly redundant, and difficult to understand.

We believe that the fundamental issue with frequent itemset mining (FIM) approaches is that they solve the wrong problem: users are not interested in solely itemsets that are frequent but rather in the itemsets that *best explain the transaction database*. In this paper, we propose mining *interesting itemsets*, those that are most informative within a simple but natural probabilistic model of transactions. The collection of interesting itemsets can then be inferred using a structural EM framework (Friedman, 1998), in which the E step employs a submodular formulation of the weighted set cover problem. Our contributions in this paper are as follows:

- Our proposed generative model is the first to capture itemsets that 'best explain' a transaction database. That is to say, our model contains a latent indicator variable z_S that *infers* if itemset S was used to generate a transaction in the database (Section 3).
- Our approach resolves many outstanding issues with FIM, including difficulties mining rare itemsets and the retrieval of large numbers of spurious itemsets (Section 2).
- Our model is theoretically simple and our approach is straightforward to implement and trivially parallelizable (Section 4).

Our prototype implementation finds a hundred potentially interesting itemsets from a million transactions in under an hour on off-the-shelf hardware, but it is possible that more sophisticated algorithms for submodular optimization would allow us to scale this even farther.

2 Related Work

The first FIM algorithm was Apriori (Agrawal et al., 1994) which recursively builds larger frequent itemsets from frequent singleton itemsets. While straightforward to implement and exhaustive, Apriori can spend a long time generating a large number of candidate frequent itemsets and repeatedly scanning the database to evaluate them. The FPGrowth algorithm (Han et al., 2000) was devised to address these issues by mining frequent itemsets without candidate generation. FPGrowth works by compressing the database into an FP-tree and starting from each frequent singleton (a suffix in the tree), constructing a conditional subtree of prefix paths which is then recursively mined. We refer the interested reader to Han et al. (2007) for a survey of FIM algorithms.

A major issue with FIM is that a huge number of frequent itemsets is often retrieved for a given minimum support (frequency) threshold. In an attempt to address this, a large body of research has been devoted to devising condensed representations of frequent itemsets including closed frequent, maximal frequent and non-derivable itemsets (see Han et al., 2007, for a survey). While these approaches are successful at decreasing the number of retrieved itemsets, they cannot solve any of the other issues with FIM, such as difficulties finding rare itemsets and the retrieval of spurious itemsets, as they are merely representations of the underlying frequent itemsets themselves.

An orthogonal research direction to deal with FIM issues has been to devise *measures of interestingness* which are usually applied to prune the associations (correlations) implied by the itemsets. These measures incorporate the underlying frequent itemset supports and their notions of interestingness are often based on significance tests against random itemset distributions. We refer the interested reader to Geng & Hamilton (2006); Han et al. (2007) for surveys of interestingness measures. Note that all such approaches require that a collection of frequent itemsets be first precomputed.

The closest work to our own is by Mampaey et al. (2012) who make use of a maximum entropy model to mine the most *informative itemsets*, i.e., itemsets with frequencies which deviate most from those predicted by their model. The authors enforce the constraint that an itemset must be used to explain a transaction which contains it as opposed to *inferring* when this is the case as in our model. This causes their model to overfit and they are forced to resort to model regularization techniques such as BIC and MDL.

3 Interesting Itemset Mining

In this section we will formulate the problem of identifying a set of interesting itemsets that are useful for explaining a database (i.e., sequence) of transactions. First we will define some preliminary concepts and notation. An *item* i is an element of the universe $U = \{1, 2, \ldots, n\}$ that indexes database attributes. A *transaction* X is a subset of the universe U and an *itemset* S is simply a set of items i. The set of interesting itemsets $\mathcal{I}$ we wish to determine is therefore a subset of the power set (set of all possible subsets) of the universe. Further, we say that an itemset S *supports* a transaction X if $S \subset X$.

Generative Model We propose a simple graphical model for generating a database of transactions from a set of interesting itemsets. The parameters of our model are a set $\mathcal{I}$ of interesting itemsets and for each interesting itemset $S \in \mathcal{I}$ a Bernoulli probability π_S. The generative story for our model is, independently for each transaction X in the database:

1. For each itemset $S \in \mathcal{I}$, decide independently whether to include S in the transaction, i.e., sample
$$z_S \sim \text{Bernoulli}(\pi_S).$$
2. Set the transaction to be the set of items in all the itemsets selected above, i.e.,
$$X = \bigcup \{i \in S \mid S \in \mathcal{I}, z_S = 1\}.$$

Note that the model allows individual items to be generated multiple times from different itemsets.

Inference Given a set of itemsets $\mathcal{I}$, let $\mathbf{z}, \boldsymbol{\pi}$ denote the vectors of z_S, π_S for all itemsets $S \in \mathcal{I}$. Assuming $\mathbf{z}, \boldsymbol{\pi}$ are fully determined, it is evident from the generative model that the probability of generating a transaction X is

$$p(X, \mathbf{z}|\boldsymbol{\pi}) = \prod_{S \in \mathcal{I}} \pi_S^{z_S} (1 - \pi_S)^{1 - z_S} \quad \text{if } X = \bigcup_{z_s = 1} S, \quad \text{otherwise } 0 \tag{3.1}$$

since each $z_S \sim \text{Bernoulli}(\pi_S)$. Now assuming the latent variables $\boldsymbol{\pi}$ are known, we can infer $\mathbf{z}$ for a specific transaction X using maximum likelihood estimation. From (3.1) it is easy to see that the maximum likelihood solution is to maximize the posterior distribution $p(\mathbf{z}|X, \boldsymbol{\pi})$ over $\mathbf{z}$:

$$\begin{aligned} &\max_{\mathbf{z}} \prod_{S \in \mathcal{I}} \pi_S^{z_S} (1 - \pi_S)^{1 - z_S} \\ &\text{s.t. } X = \bigcup \{i \in S \,|\, S \in \mathcal{I}, z_S = 1\}. \end{aligned} \tag{3.2}$$

Taking logs and rewriting (3.2) in a more standard form we obtain

$$\begin{aligned} &\max_{\mathbf{z}} \sum_{S \in \mathcal{I}} z_S \ln \left(\frac{\pi_S}{1 - \pi_S} \right) + \ln(1 - \pi_S) \\ &\text{s.t.} \sum_{S | i \in S} z_S \geq 1 \qquad \forall i \in X \\ &\qquad z_S \in \{0, 1\} \qquad \forall S \in \mathcal{I} \end{aligned} \tag{3.3}$$

which is (up to a constant) the weighted set-cover problem (see e.g. Korte & Vygen, 2012, S16.1) with weights $w_S \in \mathbb{R}$ given by

$$w_S := \ln \left(\frac{\pi_S}{1 - \pi_S} \right).$$

This is an NP-hard problem in general and so impractical to solve directly in practice. It is important to note that the weighted set cover problem is a special case of maximizing a linear function subject to a submodular constraint, which we formulate as follows (cf. Young, 2008). Given the set of interesting itemsets $\mathcal{T}$ that support the transaction

$$\mathcal{T} := \{S \in \mathcal{I} \,|\, S \subset X\}, \tag{3.4}$$

a real-valued weight w_S for each itemset $S \in \mathcal{T}$ and a non-decreasing submodular function $f : 2^{\mathcal{T}} \to \mathbb{R}$, the aim is to find a covering $C \subset \mathcal{T}$ of maximum total weight, i.e., such that $f(\mathcal{C}) = f(\mathcal{T})$ and $\sum_{S \in \mathcal{C}} w_S$ is maximized. For weighted set cover we simply define $f(\mathcal{C})$ to be the number of items in $\mathcal{C}$, i.e., $f(\mathcal{C}) := |\cup_{S \in \mathcal{C}} S|$, and note that $f(\mathcal{T}) = |X|$ by construction.

We can therefore approximately solve the weighted set cover problem (3.3) using the greedy approximation algorithm for submodular functions. The greedy algorithm builds a covering $\mathcal{C}$ by repeatedly choosing an itemset S that maximizes the weight w_S divided by the number of items not yet covered by the chosen itemset.

Algorithm 1 Greedy Weighted Set Cover

Input: Transaction X, set of itemsets $\mathcal{T}$, weights $\mathbf{w}$
  Initialize $\mathcal{C} \leftarrow \emptyset$
  while $f(\mathcal{C}) \neq |X|$ **do**
    Choose $S \in \mathcal{T}$ maximizing $\frac{w_S}{f(\mathcal{C} \cup \{S\}) - f(\mathcal{C})}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{S\}$
  end while
  return $\mathcal{C}$

It has been shown (Chvátal, 1979) that the greedy algorithm achieves an $\ln|X| + 1$ approximation ratio to the weighted set cover problem and moreover the following inapproximability theorem shows that this is essentially the best possible approximation ratio.

Theorem 1 (Feige, 1998)**.** *There is no* $(1 - o(1)) \ln|X|$*-approximation algorithm to the weighted set cover problem unless* NP $\subseteq$ DTIME$(|X|^{O(\log \log |X|)})$*, i.e., unless* NP *has slightly superpolynomial time algorithms.*

The runtime complexity of the greedy algorithm (Algorithm 1) is $O(|X||\mathcal{T}|)$ however by maintaining a priority queue this can be improved to $O(|X|\log|\mathcal{T}|)$ (see e.g. Cormen et al., 2001). Note that there is also an $O(|X||\mathcal{T}|)$-runtime primal-dual approximation algorithm (Bar-Yehuda & Even, 1981), however this has an approximation order of $f = \max_i |\{S \mid i \in S\}|$, i.e., the frequency of the most frequent element, which would inevitably be worse in our case.

Learning Given a set of itemsets $\mathcal{I}$, consider now the case where both variables $\mathbf{z}, \boldsymbol{\pi}$ in the model are unknown. In this case we can use the hard EM algorithm (Dempster et al., 1977) for parameter estimation with latent variables. The hard EM algorithm in our case is merely a simple layer on top of the inference algorithm (3.3). Suppose there are m transactions $X^{(1)}, \ldots, X^{(m)}$ with supporting sets of itemsets $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(m)}$ defined as in (3.4). Then the hard EM algorithm is the following.

Algorithm 2 Hard Expectation Maximization

Input: Set of itemsets $\mathcal{I}$ and initial estimates $\boldsymbol{\pi}^{(0)}$
$k \leftarrow 0$
do
$k \leftarrow k + 1$
E-step: $\forall\, X^{(j)}$ solve (3.3) using Algorithm 1 to obtain $z_S^{(j)}\ \ \forall\, S \in \mathcal{T}_j$
M-step: $\pi_S^{(k)} \leftarrow \frac{1}{m}\sum_{j=1}^{m} z_S^{(j)} \quad \forall\, S \in \mathcal{I}$
while $\|\boldsymbol{\pi}^{(k-1)} - \boldsymbol{\pi}^{(k)}\| > \varepsilon$
Remove from $\mathcal{I}$ sets S with $\pi_S = 0$
return $\mathcal{I}, \boldsymbol{\pi}^{(k)}$

To initialize $\boldsymbol{\pi}$, a natural choice is simply the support (rel. frequency) of each itemset in the dataset.

Inferring new itemsets We can use structural EM (Friedman, 1998) to infer new itemsets, i.e., we add a candidate itemset S' to $\mathcal{I}$ if doing so improves the optimal value $\overline{p}$ of the problem (3.3) averaged across all transactions. We force the candidate S' to be used to explain transactions it supports by setting $\pi_{S'} = 1$ and update $\pi_{S'}$ with the probability corresponding to its actual usage once we have inferred all the coverings. Given a set of itemsets $\mathcal{I}$ and correspoding probabilities $\boldsymbol{\pi}$ along with transactions $X^{(1)}, \ldots, X^{(m)}$, each iteration of the structural EM algorithm is as follows.

Algorithm 3 Structural EM (one iteration)

Input: Set of itemsets $\mathcal{I}$ and probabilities $\boldsymbol{\pi}$
$\forall\, X^{(j)}$, solve (3.3) using Algorithm 1 to obtain the optimum $p^{(j)}$
Set profit $\overline{p} \leftarrow \frac{1}{m}\sum_{j=1}^{m} p^{(j)}$
do
Generate candidate S' using Algorithm 4
$\mathcal{I} \leftarrow \mathcal{I} \cup \{S'\}$, $\pi_{S'} \leftarrow 1$
E-step: $\forall\, X^{(j)}$ solve (3.3) using Algorithm 1 to obtain $z_S^{(j)}\ \ \forall\, S \in \mathcal{T}_j$
M-step: $\pi'_S \leftarrow \frac{1}{m}\sum_{j=1}^{m} z_S^{(j)} \quad \forall\, S \in \mathcal{I}$
$\forall\, X^{(j)}$ get the optimum $p^{(j)}$ of (3.3) using $\pi'_S, z_S^{(j)}\ \ \forall\, S \in \mathcal{T}_j$
Set new profit $\overline{p}' \leftarrow \frac{1}{m}\sum_{j=1}^{m} p^{(j)}$
$\mathcal{I} \leftarrow \mathcal{I} \setminus \{S'\}$
while $\overline{p}' \leq \overline{p}$ {until one good candidate found}
$\mathcal{I} \leftarrow \mathcal{I} \cup \{S'\}$
return $\mathcal{I}, \boldsymbol{\pi}'$

In practice we store the set of candidates that have been rejected by Algorithm 3 and check each potential candidate against this set before running the rest of the algorithm for efficiency.

Candidate generation The structural EM algorithm (Algorithm 3) requires a method to generate new candidate itemsets S' that are to be considered for inclusion in the set of interesting itemsets $\mathcal{I}$. One

possibility would be to use the Apriori algorithm to recursively suggest larger itemsets starting from singletons, however this may not be the most efficient method. For this reason we take a slightly different approach and recursively combine the interesting itemsets in $\mathcal{I}$ with the *highest support first* (Algorithm 4). In this way our candidate generation algorithm is more likely to propose viable candidate itemsets earlier and in practice we find that this heuristic works well.

Algorithm 4 Support-weighted Itemset Combination

Input: Set of itemsets $\mathcal{I}$
  Sort $\mathcal{I}$ by decreasing itemset support
  for all pairs $S_1, S_2 \in \mathcal{I}, S_1 \neq S_2$, highest ranked first **do**
    $S' \leftarrow S_1 \cup S_2$
    Evaluate S' using Algorithm 3
    if S' improves optimum of (3.3) **break**
  end for

In order to determine the supports of the itemsets to be combined, we store the transaction database in a memory-efficient itemset tree (MEI-tree) (Fournier-Viger et al., 2013) and query the tree for the support of a given itemset. A MEI-tree stores itemsets in a tree structure according to their prefixes in a memory efficient manner. To minimize the memory usage of the MEI-tree further, we first sort the items in order of decreasing support (as in the `FPGrowth` algorithm) as this often results in a sparser tree (Han et al., 2000). Note that a MEI-tree is essentially an FP-tree (Han et al., 2000) with node-compression and without node-links for nodes containing the same item. An itemset support query on the MEI-tree efficiently searches the tree for all occurences of the given itemset and adds up their supports (see Figure 4 in Fournier-Viger et al., 2013, for the algorithm). In practice we impose an iteration limit on the number of iterations that Algorithm 4 can spend suggesting candidates.

Mining Interesting Itemsets Our complete interesting itemset mining (IIM) algorithm is given in Algorithm 5. Note that the parameter optimization step (Algorithm 2) need not be performed at

Algorithm 5 Interesting Itemset Miner (IIM)

Input: Database of transactions $X^{(1)}, \ldots, X^{(m)}$
  Initialize $\mathcal{I}$ with singletons and $\boldsymbol{\pi}$ with their supports
  Build EIM-tree from transaction database
  while not converged **do**
    Add itemsets to $\mathcal{I}, \boldsymbol{\pi}$ using Algorithm 3
    Optimize parameters for $\mathcal{I}, \boldsymbol{\pi}$ using Algorithm 2
  end while
  return $\mathcal{I}, \boldsymbol{\pi}$

every iteration, in fact it is more efficient to suggest several candidate itemsets before optimizing the parameters. We can then rank the retrieved itemsets according to their *interestingness*, that is the ratio of transactions they explain to transactions they support. We define this formally as follows.

Definition 1. The *interestingness* of an itemset $S \in \mathcal{I}$ retrieved by IIM (Algorithm 5) is defined as

$$int(S) = \frac{\sum_{j=1}^{m} z_S^{(j)}}{supp(S)}$$

and ranges from 0 (least interesting) to 1 (most interesting).

Any ties in the ranking can be broken using the itemset probability π_S. Note that all operations on transactions in our algorithm are independent and so trivially parallelizable. We therefore perform the E and M-steps in both the hard and structural EM algorithms in parallel for each transaction.

4 Numerical Experiments

We evaluate IIM on both synthetic and real-life transaction datasets. In order to minimize CPU time spent solving the weighted set cover problem (3.3), we cache the itemsets and coverings for each transaction as needed.

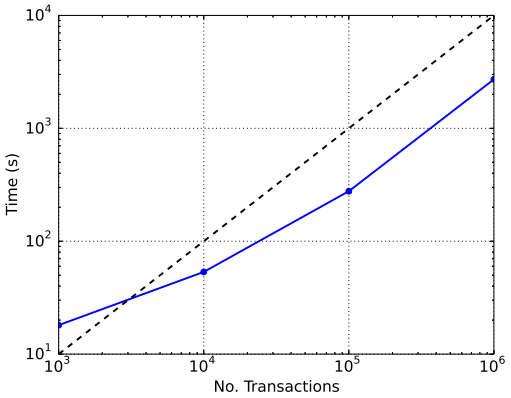

Figure 1: IIM scaling as the number of transactions in the database increases.

Interesting Itemsets	Frequent Itemsets
associ rule	algorithm mine
state art	algorithm base
global local	algorithm problem
support vector machin svm	algorithm set
neural network	algorithm result
social network	algorithm approach
posit negat	algorithm method
parameter parameters	algorithm propos
sequenc sequential	approach base
anomali detect	base method

Table 1: The top ten itemsets of size ≥ 2 as found by IIM and FPGrowth.

Scalability We created a synthetic database in order to investigate the scaling of IIM as the number of transactions in the database increases. To achieve this, we first ran IIM for $1,000$ iterations on the plants database (USDA, 2008) to obtain interesting itemsets $\mathcal{I}$ and associated probabilities $\boldsymbol{\pi}$. This enabled us to use the probabilistic model from Section 3 on $\mathcal{I}, \boldsymbol{\pi}$ to generate synthetic transaction databases of various sizes containing the mined itemsets. We then ran IIM for 100 iterations on these databases (Figure 1) and one can see the transaction scaling is linear as expected. Our prototype implementation can process one million transactions in 30 seconds on 64 cores each iteration, so there is reason to hope that a more highly tuned implementation could scale to even larger data sets.

Dataset	Items	Itemsets	IIMs	Runtime	Iterations
Plants	70	34,781	271	63 min	1,000
ICDM	3,933	859	5,685	121 min	1,000

Table 2: Summary of the real datasets used and IIM results.

Real Datasets To evaluate the quality of the itemsets produced by our method, we ran IIM for $1,000$ iterations on the ICDM abstracts dataset (De Bie, 2011) along with FPGrowth. We show the top ten nontrivial interesting itemsets (ranked according to interestingness) and frequent itemsets (ranked according to support) in Table 1. Once can clearly see that the interesting itemsets are considerably more informative. Moreover, the interesting itemsets suggest that the stemmer used to process the ICDM dataset could be improved as we retrieve the itemsets {parameter, parameters} and {sequenc, sequential}.

Redundancy We also evaluate whether the collection of itemsets returned by IIM are less redundant than those produced by standard FIM methods. We ran FPGrowth on the plants dataset using a minimum support chosen s.t. we obtain roughly the same number of frequent itemsets as interesting itemsets using IIM. This enabled us to measure, for each dataset, the average size of the minimum symmetric distance between the itemsets and see how close the itemsets are to each other. We found that, excluding singletons, this difference is 4 items for IIM and 1 item for FIM, showing that the IIM itemsets are less redundant.

5 Conclusions

We presented a novel approach to inferring itemsets that best explain a transaction database and demonstrated the efficacy of our approach on both synthetic and real-world databases. In future we would like to extend our approach to inferring associations implied by the itemsets directly and parallelize our approach to large clusters so that we can efficiently scale to larger databases. Exploiting more modern solvers from the discrete optimization literature could further improve scalability.

References

Agrawal, R., Srikant, R., et al. Fast algorithms for mining association rules. In *VLDB*, volume 1215, pp. 487–499, 1994.

Bar-Yehuda, R. and Even, S. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.

Chvátal, V. A greedy heuristic for the set-covering problem. *Math. O.R.*, 4(3):233–235, 1979.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. *Introduction to Algorithms*. MIT Press, 2001.

De Bie, T. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Mining and Knowledge Discovery*, 23(3):407–446, 2011.

Dempster, A.P., Laird, N.M., and Rubin, D.B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, pp. 1–38, 1977.

Feige, U. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

Fournier-Viger, P., Mwamikazi, E., Gueniche, T., and Faghihi, U. MEIT: Memory Efficient Itemset Tree for targeted association rule mining. In *Advanced Data Mining and Applications*, volume 8347 of *Lecture Notes in Computer Science*, pp. 95–106. Springer, 2013.

Friedman, N. The Bayesian structural EM algorithm. In *UAI*, pp. 129–138, 1998.

Geng, L. and Hamilton, H.J. Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38 (3):9, 2006.

Han, J., Pei, J., and Yin, Y. Mining frequent patterns without candidate generation. In *SIGMOD Record*, volume 29, pp. 1–12. ACM, 2000.

Han, J., Cheng, H., Xin, D., and Yan, X. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.

Korte, B. and Vygen, J. *Combinatorial Optimization: Theory and Algorithms*. Algorithms and Combinatorics. Springer, 2012.

Krause, A. and Guestrin, C. Submodularity and its applications in optimized information gathering. *TIST*, 2 (4):32, 2011.

Lin, H. and Bilmes, J. A class of submodular functions for document summarization. In *ACL*, pp. 510–520, 2011.

Mampaey, M., Vreeken, J., and Tatti, N. Summarizing data succinctly with the most informative itemsets. *TKDD*, 6(4):16, 2012.

Narasimhan, M. and Bilmes, J. PAC-learning bounded tree-width graphical models. In *UAI*, pp. 410–417, 2004.

USDA, NRCS. The PLANTS Database, 2008. URL http://plants.usda.gov/.

Young, N.E. Greedy set-cover algorithms (1974-1979, Chvátal, Johnson, Lovász, Stein). In Kao, M.Y. (ed.), *Encyclopedia of Algorithms*, pp. 379–381. Springer, 2008.