How to Grow More Pairs: Suggesting Review Targets For Comparison-Friendly Review Ecosystems

James Cook*
UC Berkeley
Berkeley, CA, USA
jcook@cs.berkeley.edu

Alex Fabrikant
Google Research
Mountain View, CA, USA
fabrikant@google.com

Avinatan Hassidim
Google Research
Tel Aviv, Israel
avinatan@google.com

ABSTRACT

We consider the algorithmic challenges behind a novel interface that simplifies consumer research of online reviews by surfacing relevant *comparable review bundles*: reviews for two or more of the items being researched, all generated in similar enough circumstances to provide for easy comparison. This can be reviews by the same reviewer, or by the same demographic category of reviewer, or reviews focusing on the same aspect of the items. But such an interface will work only if the review ecosystem often has comparable review bundles for common research tasks.

Here, we develop and evaluate practical algorithms for suggesting additional review targets to reviewers to maximize *comparable pair coverage*, the fraction of co-researched pairs of items that have both been reviewed by the same reviewer (or more generally are comparable in one of several ways). We show the exact problem and many subcases to be intractable, and give a greedy online, linear-time 2-approximation for a very general setting, and an offline 1.583-approximation for a narrower setting. We evaluate the algorithms on the Google+ Local reviews dataset, yielding more than 10× gain in pair coverage from six months of simulated replacement of existing reviews by suggested reviews. Even allowing for 90% of reviewers ignoring the suggestions, the pair coverage grows more than 2× in the simulation. To explore other parts of the parameter space, we also evaluate the algorithms on synthetic models.

Categories and Subject Descriptors

H.5.2 [**HCI**]: General; F.2.2 [**Theory of Computation**]: Algorithms

Keywords

reviews, algorithms, graphs, comparing

1. INTRODUCTION

Whether you're choosing a plumber, or a kitchen sink, or a dry hotel room, online crowdsourced review systems now promote extensive consumer research and significantly affect decision-making [8–11]. The volume and coverage of user-generated reviews is one of the key factors driving customers to online businesses over traditional stores [7, 15, 23], and can thus be considered one of the major driving forces of the Internet economy. Yet consumers researching purchase options online are often overwhelmed or confused by the large volume of available information [15]. This is the well-studied phenomenon of *information overload* which in many contexts engenders difficulties with decision-making and "analysis paralysis" [12, 19, 21, 25].

In a typical research scenario, a consumer deciding between several items discovers a variety of unstructured, essay-form reviews for each of the items, each written from the unique perspective of the respective reviewer. So, which item to pick?

Most existing review systems augment unstructured reviews with coarse overall ratings, such as on a scale from 1 to 5 stars, and summarize an item by the arithmetic mean of the reviews [1, 6], occasionally with some filtering or per-user weights [5]. But a numerical average is often a poor tool for decision making. Obscure items may have only a couple of reviews each, addressing different aspects of the items, making the averages themselves statistically meaningless. Also, reviewer grade distributions are neither smooth [16] nor time-independent [18].

The cognitive load of a comparison task is lightest when the options differ in a small number of clearly delineated and contrasted ways [17, 19]. *But can we corral a corpus of unstructured user-generated reviews into an interface for convenient "apples-to-apples" comparisons*? One approach is to use the tools of automatic natural-language analysis to extract sentiment data and other semantics from an existing review corpus. A large and fruitful body of research literature on approach is surveyed in [22]. Here, we propose and study a fundamentally different, orthogonal approach: **pushing reviewers toward specific review targets to enable more comparisons among reviews by the same user**.

The reviewer herself is a major, perhaps dominant, source of variation in the topics, attitudes, and conclusions of a consumer review. When we look at Alice's review, how do we interpret her calling a record store's prices "very affordable"? That phrase carries a different meaning when coming from a hipster than when coming from a hippie. What does Bob mean when he calls a hotel's carpet "terrible"? Is he concerned about the color scheme or about biohazards? If Bob has reviewed both of the hotels Carol is considering, a UI can prominently surface these reviews to Carol: "This hotel

*Work done as part of a research internship at Google

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

and the hotel you just looked at have both been reviewed by Bob," along with both of Bob's reviews side-by-side. Such an interface, especially along with meta-data about Bob often provided by social-oriented review sites[1] will enable Carol to deeply contextualize Bob's reviews relative to his preferences and demographics, and will bring the consumer choice experience closer to the gold standard of getting both reviews from the same well-known friend.

How often can we surface such a *review pair*? In most contexts, e.g. local businesses or consumable goods, if a pair of items is frequently compared by researching users, it is likely that some users have experienced both items and *can* review both. In our target dataset, Google+ Local reviews, we experimentally measured the pre-existing levels of *pair coverage*: how many of the pairs of items frequently searched for in the same session have at least one review pair in common. Given the wide variety of reviewable items, and no existing incentive to create review pairs, we found the pair coverage was far too low to allow for a useful review pair surfacing feature.

But an (honest) reviewer is typically acting altruistically, with utility largely independent of which item she reviews among the items she knows, so many reviewers will be receptive to a suggestion of what to review next. We thus focus on the algorithmic question of optimizing pair coverage by picking review target suggestions:

General question. We are given a set of reviewers, where each reviewer has already reviewed certain items, and the importance of each pair of items, which measures how much we value review pairs for this item pair. How do we assign each reviewer to new items which she will review for us?

1.1 Notation and initial problem statement

Any given review ecosystem and review-comparison UI design will determine several dimensions of our general question that must be specified to arrive at a concrete algorithmic problem. In Section 5 we catalog this variety of settings via a taxonomy for the relevant dimensions, but for the time being we describe the particular form of the problem to which our experimental results apply.

Since our experimental evaluations center on reviews from Google+ Local, we interchangeably refer to the reviewable items as "venues", but our results generalize immediately to other review-suggestion settings as well.

We focus on a set of n venues that we denote by V. We consider a weighted *pair relevance graph* $G = (V, E)$ on the venues where the edge weight c_{uv} gives the fraction of the consuming users who are interested in comparing u and v.

We assume a set of m reviewers R, with each reviewer having already reviewed some initial set of venues.

In every context, there is a cost associated with making more suggestions to any reviewer: there is typically a cap on how much of the reviewer's patience and attention we can use, or on screen space to allocate to suggestions. We can even incur monetary costs for suggestions in scenarios where we incentivize reviewing. In all relevant cases, these budgets are not fungible between reviewers: suggesting 1000 items to one reviewer and no items to the others is not at all interchangeable with suggesting one item to each of 1000 reviewers. Thus, we only allow a single item to be suggested to each $r \in R$, drawn from a set of items A_r that we believe r may be willing to review.

We use a_{uv} to denote the number of review pairs for venues u and v.

These review suggestions aim to optimize the experience of a researching user. We capture this with an objective function, which is to maximize the weighted sum over all item pairs of a concave function of the number of review pairs, i.e. $O_F = \sum_{u,v} c_{uv} F(a_{uv})$ for some concave F. Concavity corresponds to the natural constraint of decreasing marginal utility of additional review pairs. In many cases we are simply interested in having at least one reviewer, so we use the objective $F_1(0) = 0$ and $F_1(a) = 1$ for any $a \geq 1$. We call the corresponding objective O_{F_1} or simply O_1.

1.2 Our contributions

We begin in Section 2 with two approximation algorithms for the problem described above:

- An online greedy 2-approximation algorithm in Section 2.1
- An offline $(e-1)/e$-approximation based on rounding a convex program similar to the Max Cover problem in Section 2.2. This can also be simplified to an LP, in the case of the O_1 objective.

We generalize the convex program to assign each reviewer a constant number k new venues instead of just one.

In Section 3, we evaluate these algorithms in two settings. First, we take all public venue reviews posted on Google+ Local, and conduct hypothetical experiments that remove all reviews created after a time threshold, and consider what would have happened to our objectives if all or some of the same reviewers were instead redirected to review targets suggested by our algorithms.

The $(e-1)/e$-approximation algorithm improves over the greedy algorithm when the review coverage is dense. We demonstrate this effect in Section 4 on two synthetically-generated datasets.

We then delve into the exact problems. We observe that optimizing even the simple objective O_1 exactly is NP-complete, and we find that the problem remains intractable after being simplified in various ways. In particular, we show the following lower bounds in Section 6:

THEOREM 5. *Optimizing O_1 is NP-complete, even when the graph is a clique (every pair has equal relevance c_{uv}).*

THEOREM 6. *Optimizing O_1 is NP-complete, even with only one reviewer with one prior review, if the reviewer is willing to review k new venues instead of one.*

THEOREM 7. *Optimizing O_1 is NP-complete, even when the graph is a clique and every reviewer is either willing to review nothing or anything. (Every A_r is empty or V.)*

COROLLARY 9. *Optimizing O_1 is NP-complete, even when every reviewer is willing to review any restaurant.*

(In Section 5 we propose a taxonomy of problems in order to get a handle on the numerous variations.)

Having established that the exact problem is NP-Complete even after various simplifications, we also give tractable exact algorithms for a handful of particularly simple settings:

[1]such as Bob's rating distributions [2], third-party "helpfulness" ratings [1], or social profile data [4]

- A polynomial algorithm for the O_1 objective, in the case that each reviewer only has one prior venue, but is willing to review k arbitrary new venues. This may be useful for optimizing suggestions for the many reviewers with one review.
- A linear-time online greedy algorithm for the O_1 objective in the very simple case where the graph G is a clique and each reviewer has a single previous review and is willing to review one more arbitrary item.

Lastly, in Section 7, we address generalizations to "comparable reviews" notions broader than just reviews written by the same person, and discuss several open problems.

2. ALGORITHMS

We now consider our two approximation algorithms to optimize the objective O_F. The exact question's NP-Completeness is a corollary of Theorem 7.

2.1 Greedy

The first algorithm assigns the reviewers one at a time in an on-line fashion. At each step, the algorithm chooses the venue of the next reviewer in order to cause the greatest increase in the objective function.

Put another way: after the algorithm has made t assignments, let $z_{rw}(t)$ indicate whether r has reviewed w and $a_{uv}(t)$ indicate the number of reviewers who have reviewed both u and v. Then at step $t+1$, we assign reviewer r_{t+1} to the venue w which maximizes the total increase in O_F created by that assignment:

$$\text{new}_{t+1} = \sum_{u \in V} c_{uw} z_{r_{t+1} u} \big[F(a_{uw}(t)+1) - F(a_{uw}(t))\big]. \quad (1)$$

THEOREM 1. *The value of the greedy solution is at least half the optimal value.*

PROOF. The intuition is that every time the greedy algorithm causes a pair of venues to gain another common reviewer (increasing a_{uv}), then the only way it can be wasteful is if a different reviewer should have co-reviewed u and v: but at least one of the two reviewers is being used efficiently.

More carefully, after the algorithm has made t assignments, let $\text{opt}(t)$ be the value for O_F if the remaining assignments are made optimally, and let $\text{val}(t)$ be the value of objective so far (that is, replacing a_{uv} with $a_{uv}(t)$ in O_F). Let $\text{opt} = \text{opt}(0)$ be the unrestricted optimal value. Then it's enough to show that for every t, $\text{opt} \le \text{val}(t) + \text{opt}(t)$. This is trivially true at $t = 0$. Now, if we take a complete solution that achieves $\text{opt}(t)$, and change only the $(t+1)$-st venue w_{t+1}, the objective decreases by at most[2] new_{t+1}: so $\text{opt}(t+1) \ge \text{opt}(t) - \text{new}_{t+1}$. To complete the proof, note that $\text{val}(t+1) = \text{val}(t) + \text{new}_{t+1}$. □

2.2 Linear and Convex Programming

The second algorithm optimizes the objective O_F by rounding a convex programming solution. In our implementation, we were only concerned with the O_1 objective function, in which case the program becomes a linear program. We start by describing this case.

[2]This argument depends on the fact that each reviewer is assigned to at most one new venue, and that the function F is concave.

The linear program as described here can only assign each reviewer to one new venue. For the results in Section 3, we used an extension of the algorithm which can assign multiple venues per reviewer, but our proofs do not apply in this case.

For every two venues u, v for which $c_{uv} > 0$, we add a variable x_{uv}. In any optimal integer solution of the linear program, x_{uv} will take the value 1 if some reviewer reviewed both u and v ($a_{uv} \ge 1$) and 0 otherwise.

Let $r \in R$ be a reviewer, and $P_r \subseteq V$ the set of venues which r can review. For every venue $w \in P_r$, we add a variable y_{rw} which corresponds to us asking r to review w. We denote by $Q(r,w)$ the set of venues which will be reviewed by r if he reviews w in addition to all the prior reviews she made. We assume that r will agree to review w with some probability p_{rw}. (When solving the problem outlined in Section 1.1, take $p_{rw} = 1$ when $w \in A_r$ and $p_{rw} = 0$ otherwise.) The LP now looks like

$$\begin{array}{lll} \text{Maximize} & \sum_{u,v} x_{uv} & \\ \text{Subject to} & & \\ \forall u, v & x_{uv} & \le \sum_{r,w:\ u,v \in Q(r,w)} y_{rw} p_{rw} \\ & x_{uv} & \le 1 \\ \forall r \in R & \sum_w y_{rw} & \le 1 \end{array}$$

To do the rounding, we sample from the distributions $y_{r,w}$ for every r. This means that the requests are always "legal", and each reviewer gets one request.

THEOREM 2. *The value of the rounded LP solution is at least $(e-1)/e$ times the optimal value in expectation.*

PROOF. Consider any pair of venues $u, v \in V$. Each reviewer that has reviewed either u or v has some (possibly zero) probability of reviewing the other venue. These probabilities add to x_{uv}, so the probability that (u,v) is not co-reviewed is at most $e^{-x_{uv}}$. □

We also present a simple integrality gap for the LP, which shows that our analysis of the LP is tight. consider a setting with just two venues denoted V, W, and n reviewers who have all reviewed venue V. For each reviewer r the probability that r agrees to review W is $p_{rw} = 1/n$. The value of the LP would be 1. The optimal assignment[3] is to assign all the reviewers to W. However the value of this assignment is $1 - 1/e$.

An extension to the linear program allows us to deal with the objective O_F for a general concave function $F : \mathcal{N} \to \mathcal{R}$, where the user benefits more when there are multiple co-reviews on the items u and v she wants to compare. The user's utility is $F(a_{uv})$ if a_{uv} is the number of co-reviews on u and v. In this case, the following convex program[4] gives an upper bound on the value of the optimal solution:

$$\begin{array}{lll} \text{Maximize} & \sum_{u,v} F(a_{uv}) & \\ \text{Subject to} & & \\ \forall u, v & a_{uv} & \le \sum_{r,w:\ u,v \in Q(r,w)} y_{rw} p_{rw} \\ \forall r \in R & \sum_w y_{rw} & \le 1 \end{array}$$

[3]We don't have much of a choice here.

[4]We are maximizing a concave function, which is equivalent to minimizing a convex function.

Where again y_{rw} is a variable for every reviewer r and venue w, $Q(r,w)$ is the set of venues which will be reviewed by r if he reviews w in addition to all the prior reviews she made and p_{rw} is the probability r agrees to review w. Asking reviewer r to review w with probability y_{rw} gives a solution to the problem, and:

THEOREM 3. *The value of the rounded convex program solution is at least $(e-1)/e$ times the optimal value in expectation.*

The proof is similar to that of Theorem 2, and is omitted here. Since Theorem 3 is a strict generalization of Theorem 2, and given the the integrality gap presented for the LP, Theorem 3 is tight. However, for objectives other than O_1 (where $F(a)$ is constant for $a \geq 1$) the bound is strictly better than $(e-1)/e$. For example, if $F(a) = a$, the rounding is optimal[5]. If $F(a) = \min\{2, a\}$, then the approximation ratio is $1 - 2/e^2$. In general:

THEOREM 4. *For a general utility function $F : \mathcal{N} \to \mathcal{R}$ with decreasing marginal returns, the approximation ratio of the convex program solution is exactly*

$$\min_{k \in \mathcal{N}} \frac{\mathbf{E}_{a \sim \text{Poisson}(k)}[F(a)]}{F(k)}.$$

PROOF. Consider a single pair of venues, and suppose the j-th reviewer reviews the pair with probability y_j, and $\sum y_j = k$. Let the random variable a be the number of reviewers who review the pair. The approximation ratio for this venue pair is $\mathbf{E}[F(a)]/F(k)$. If there are kn total reviewers, then the worst-case value for this expression is achieved when every $y_j = 1/n$. The ratio gets smaller as n increases, so for fixed k, the approximation ratio is:

$$\min_{k \in \mathcal{N}} \lim_{n \to \infty} \frac{\mathbf{E}_{a \sim \text{Binom}(kn, 1/n)}[F(a)]}{F(k)}.$$

As n increases, the Binomial distribution approaches the Poisson distribution. □

In many recommendation engines it is common to suggest several options, trying to maximize the expected number of suggestions a user accepts ([1,3,4]). We consider the case in which the number of suggestions shown to the user together is constant. In this case, let p_{rS} denote the probability the user agrees to review subset S of the venues, if a subset $T \supset S$ is presented to her. Similarly, let $Q(r,S)$ denote the set of restaurants reviewed by r, if she agrees to review S in addition to what she has already reviewed. We now add a variable y_{rS}, for every reviewer r and set of venues S. Since we only need to worry about S with constant size, the number of variables is polynomial. Finally the LP is

$$\begin{array}{rlll} \text{Maximize} & \sum_{u,v} x_{uv} & & \\ \text{Subject to} & & & \\ \forall u,v & x_{uv} & \leq & \sum_{r,S:\ u,v \in Q(r,S)} y_{rS} p_{rS} \\ & x_{uv} & \leq & 1 \\ \forall r \in R & \sum_w y_{rw} & \leq & 1 \end{array}$$

[5]We note that if $F(i) = i$ then greedy would have also been optimal.

Rounding this LP is similar to rounding the one with just a single additional review. We also get the same approximation ratio. We note that if we make no assumptions on p_{rS}, we show hardness of approximation even in degenerate cases.

3. EXPERIMENTAL RESULTS

We evaluated the two algorithms of Section 2 on a task based on real reviews data. We started with a corpus of reviews from January 2007 to the start of July 2012. For our graph G of venues that are worth comparing, we used a weighted graph of pairs of venues, based on two sources. First, venues that appear as prominent results in the same search session were considered interesting to compare. Second, we held out half of the authors from our reviews corpus, and consider pairs of venues co-reviewed by the same author in this held-out set to be interesting. To evaluate the performance of each algorithm, we performed two experiments, described in Sections 3.2 and 3.3 below.

3.1 Adapting the algorithms for real data

To implement the greedy and LP algorithms on real data, we made a small change to each algorithm.

When running the greedy algorithm, sometimes there is a reviewer who will not immediately increase the objective function, no matter which venue they review. This can happen because all pairs of venues the reviewer could co-review have already been reviewed by someone else, or (more commonly) because the reviewer has no previous reviews. In both cases, we want to ask the reviewer to review a venue which will help us if he comes back to the site. To do this, we added a heuristic for assigning reviewers who cannot immediately increase the objective. Our heuristic satisfies these two properties:

1. All other things being equal, we prefer to send a reviewer to a venue which has a heavy outgoing edge. This way, if he comes once more, we can at least get this edge

2. All other things being equal, we prefer to send a reviewer to a venue which has not been reviewed many times. This way, if she comes once more, the edges of the venue will hopefully not be taken.

The LP algorithm as described in Section 2.2 is only able to assign each reviewer to one new venue. The instances that arise in our experiments may ask the algorithm to assign a reviewer to any number of new venues, so it was necessary to adapt the algorithm. We did so by adding a new variable b_{ruv} for each triple (r,u,v), where r is a reviewer and u and v are venues that r might end up reviewing. The variable x_{uv} is then bounded by the sum of the variables b_{ruv}, and b_{ruv} is in turn bounded by both y_{ru} and y_{rv}.

3.2 Experiment: re-assigning the last six months

In the first experiment, we removed all of the reviews in our corpus from the year 2012, and let the greedy algorithm choose how to re-assign some of those reviews. Each reviewer would with probability 0.9 ignore the recommendation and keep their original reviewed venue, and only with probability 0.1 would review the suggested venue instead. (We did not include the linear programming algorithm in this experiment, since it cannot assign reviewers on-line and react to

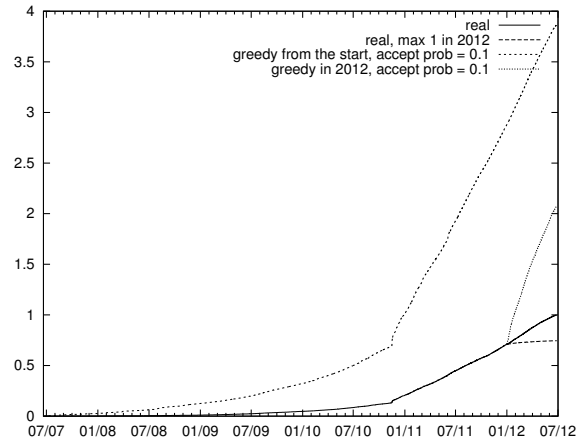

Figure 1: The growth of the objective function as more reviews are added, and a simulation of what an on-line greedy algorithm could accomplish if its suggestions in 2012 were accepted with probability 10%, or if its suggestions from the start were accepted with probability 10%. Also shown is the growth of the objective function if authors are only allowed to review one venue in 2012.

a reviewer refusing a suggestion.) Figure 1 shows how our objective function grows over time: for each date on the time axis, we consider all pairs of venues that were both reviewed by a common author before that date, and add the weights of all those edges in V. For comparison, we show the growth of the objective function in reality (not allowing the algorithm to intervene at all), and the performance when the algorithm can re-assign reviewers with probability 10%. We also show the progression of the objective if each author only reviews one venue in 2012.

3.3 Experiment: re-assigning shorter periods

In our second experiment, we consider 20 different dates in the year 2012. For each date, we allow our greedy and LP algorithms to re-assign reviews only after that date, with each suggestion being accepted with probability 100%. Figure 2 shows the improvement for each such date.

In Figure 3, we again consider 20 different dates in the year 2012. This time, we begin by restricting each author to review at most one venue in 2012, and throwing out the other reviews. Then, for each date, we allow our greedy and linear-programming algorithms to re-assign all reviews after that date, with each suggestion being accepted with probability 100%.

3.4 Conclusions

Based on these results, it seems likely that suggesting venues according to either algorithm even for a fairly short period of time would have a noticeable impact on the proportion of relevant pairs of venues with a common reviewer. The greedy algorithm performed slightly better than rounding the solution of the LP. The situation between the greedy and LP algorithms is reversed in Section 4 where we evaluate both algorithms on synthetic instances (which are more dense).

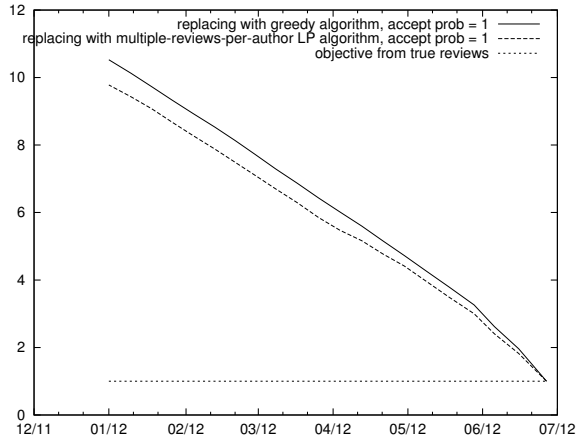

Figure 2: Replacing all reviews after a certain time (time on the x-axis) with what a greedy algorithm gets using a random order, and what rounding the multiple-new-reviews-per-author LP solution gets.

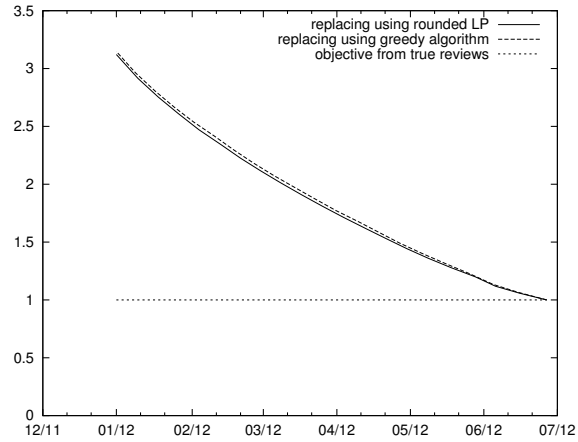

Figure 3: Replacing all reviews after a certain time (time on the x-axis) with what a greedy algorithm or rounded LP gets. Before the experiment, we make sure no author writes more than one review in 2012.

In an earlier experiment with an integer program solver, we found that the optimal integral solution was within 0.02% of the optimal unrounded LP solution, and that both the linear programming and greedy algorithms performed within 5% of optimal. These early experiments used a different graph of venues-worth-comparing c_{uv}, so they cannot be taken to exactly apply to the results in this section.

4. SYNTHETIC MODELS

In addition to the simulations on the reviews corpus mentioned above, we also generated synthetic datasets from two models. Each model describes how to generate a graph of venues that users might want to compare, which venues each author has already reviewed, and which venues the author is willing to review. The first models local reviews, where users are interested in comparing venues that are close geographically. The second models planning a vacation: each vacation has a set of attributes like price or location, and users are interested in comparing similar vacations.

4.1 The Geographic Model

In this setting, we want to model reviewers who are willing to review the local shops, regardless of what they have to offer. However, the user of the site is usually not interested (say) in comparing the local restaurant to the local bookshop. She could either be interested in having a Chinese meal, in which case she wants to read comparisons between Chinese restaurants in the city, or she wants to buy a book, and would like to see comparisons between bookstores.

There are two aspects to this setting which model the Google+ Local reviews: geographic vicinity is important, and there are venues which no user is interested in comparing, but which a reviewer might be equally willing to review.

In general, we can consider k classes of venues, with n/k venues in each class. To simplify the model, we always take $k = 2$: think of book stores and restaurants. Each venue $v \in V$ is assigned a point x_v on a two-dimensional sphere. Users are not interested in comparing venues that are in different classes: the weight of an edge (u, v) is zero if u and v are in different classes. If u and v belong to the same class, then the weight of the edge (u, v) corresponds to the fraction of users who are interested in comparing u and v and is determined by the formula[6] $w_{uv} = c(x_u, x_v) = e^{-|x_u - x_v|^2/2\sigma^2}$. We also assign each reviewer $r \in R$ a point y_r in space, inducing a probability distribution over nearby venues which chooses venue v with probability proportional to $c(y_r, x_v)$. From this distribution, we choose n_{prev} venues without replacement as previously reviewed venues, and then choose n_{willing} more among the remaining venues as ones the author would be willing to review. We choose σ based on n so that the expected weighted degree of each venue[7] is 10. While the points sit on a two dimensional sphere, we use distances from the ambient three-dimensional space.

[6]One way to arrive at this formula is to have users uniformly distributed in space, and make each user more likely to compare venues that are nearby. For efficiency, we delete edges with weight less than 0.05.

[7]If D is the expected number of (other) venues in every one-by-one square, and the other venues are uniformly distributed on the plane, then the expected degree of a venue is $2\pi\sigma^2 D$. Since we use a sphere rather than a plane, this is only an approximation.

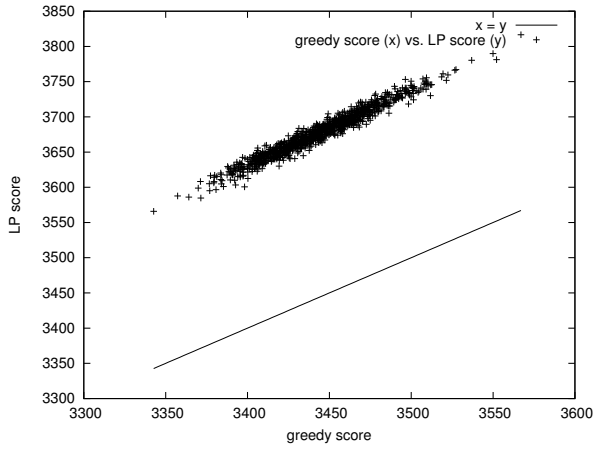

Figure 4: Performance of the greedy and linear-programming algorithms on 1000 synthetic datasets generated using the geographic model with 1000 venues and 10,000 authors.

4.2 Results for the Geographic Model

Figure 4 compares the performance of the greedy algorithm to the rounded linear programming solution on 1000 random instances of the geographic model generated with the same parameters. The x-coordinate of each point is the sum of edge weights w_{uv} for each new co-reviewed pair of venues in the solution found by the greedy algorithm. Similarly, the y-coordinate shows the performance of the algorithm based on linear programming. Figure 5 shows the ratio of greedy to LP performance on synthetic datasets of different sizes.

In both graphs we see that the linear programming algorithm consistently does better than the greedy algorithm. It is interesting that the greedy heuristic, which is essentially an online algorithm, only does slightly worse than the offline. Even when the market is large, the difference is around 7%. This is good news, as in most natural architectures we expect an online component to decide which review to ask the reviewer to fill, without knowing which other reviewers would log into the site.

4.3 The Vacations Model

For this model, it is helpful to think of vacations $v \in V$ rather than venues. Each vacation has several properties, such as price, location, style (like "romantic" or "adventure"), and others. A user could want to go to Western Europe for a honeymoon on a generous budget, and then would like to see comparisons between luxurious vacations to France and Italy. Alternatively, she could want to go on a tight budget to South America, and will want to see comparisons between cheap vacations in Peru and Bolivia. It is less likely that a user would want to compare a backpacking trip in Peru, to a romantic vacation in Paris[8].

The reviewer, on the other hand, has been in $O(1)$ vacations in her life. These vacations may have been to very dif-

[8]In the full version of the paper we also model the user who wants to compare all vacations which share a property (say all vacations to Italy). The results are similar to what we present here.

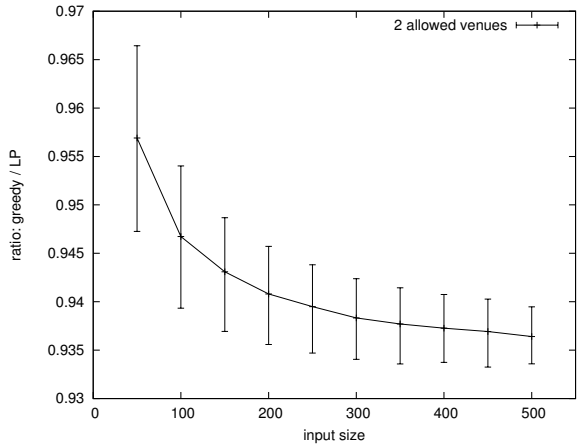

Figure 5: ratio of greedy to lp performance on synthetic datasets generated using the geographic model. for each input size, 1000 random datasets were generated and both algorithms run on each one. 90% of instances fell within the error bars, and a line joins the medians. There are $2n$ venues (n in each class) and $20n$ authors. Each author is willing to review one of two venues. $n = 500$ corresponds to Figure 4.

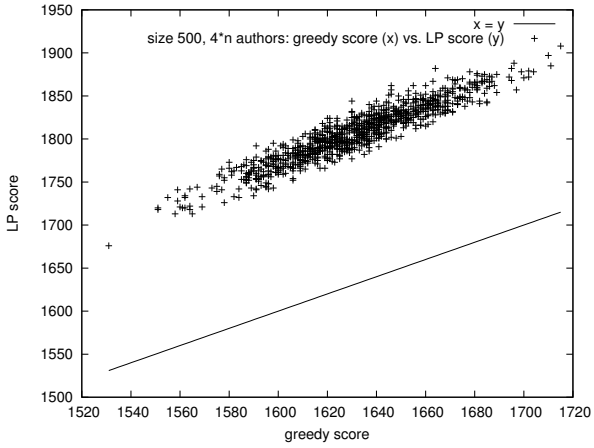

Figure 6: Performance of the greedy and linear-programming algorithms on 1000 synthetic datasets generated using the vacations model with 500 venues, 2000 authors, and $k = 5$-dimensional attributes for the venues.

ferent places and in very different circumstances (academic conference in China and a honeymoon in Ireland), and is equally willing to review any one of them.

Based on this intuition, we assign each venue $v \in V$ a vector of attributes $a_v \in [0,1]^k$, with an edge of weight 1 between each pair of venues with ℓ_∞ distance below some threshold $|a_u - a_v| < \tau$. We choose τ as a function of the number n of venues, so that the expected degree of each venue is 10. We choose n_{prev} venues P_r uniformly at random to be the venues each reviewer r has previously reviewed, and then n_{willing} reviews uniformly at random without replacement from the neighborhood of P_r to be the venues r is willing to review.

4.4 Results for the Vacations Model

Figure 6 compares the performance of the greedy algorithm to the rounded linear programming solution and is organized in the same way as Figure 4. Figure 5 shows the ratio of greedy to LP performance on synthetic datasets of different sizes.

As before, the linear programming algorithm consistently does better than the greedy algorithm. However, the difference here is larger than in the geographic model. We attribute the larger difference to the fact that the problem has a higher dimension, and therefore the local decisions that greedy makes fare less favorably. Still, the loss in efficiency is not large and is bounded by 10%.

5. PROBLEM TAXONOMY

Many variations are possible on the problem of suggesting review targets. For example, in a review system with a very limited domain, like restaurants in Rexford, we could consider any pair of items to be equally worth comparing, restricting our pair relevance graph G to be a clique with

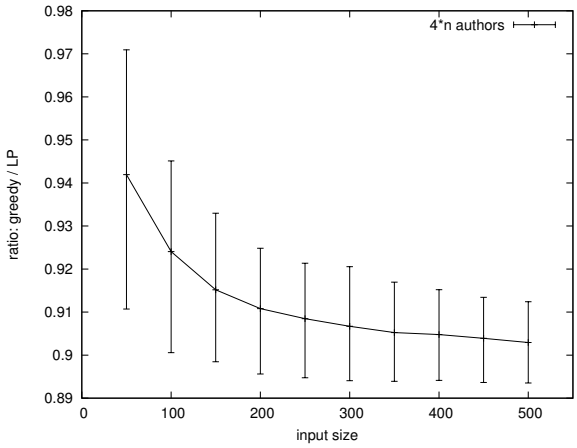

Figure 7: Ratio of greedy to LP performance on synthetic datasets generated using the vacations model. The x-axis is the number of venues, and there are always four times as many authors as venues.

equal edge weights. Other variations might be motivated by a user interface — for example, one number of new targets to suggest to a reviewer at one time might be limited by the method chosen to convey these suggestions.

Here we describe four independent dimensions along which our original problem statement of Section 1.1 can be changed. Besides illuminating a small part of the problem space, this taxonomy is quite useful in Section 6, where we will give algorithms and hardness results for a number of variations.

We focus on a set of n venues that we denote by V. We consider a weighted *pair relevance graph* $G = (V, E)$ on the venues where the edge weight c_{uv} gives the fraction of the consuming users who are interested in comparing u versus v.

To begin with, our notion of a general graph G of pair relevances might be more general than we really need. Our first dimension is to restrict G to belong to one of the following **relevance graph Categories**:

- C1: Homogeneous body of comparable items with the same weight, e.g. all laundromats in Juneau, considered in isolation from all other reviewable items, since comparisons among them are roughly equiprobable and much more likely than comparisons between these and other venues.
- C2: Disjoint cliques of things comparable within each cluster, e.g. all new car dealerships, with a clique connecting each brand's dealerships, since researchers will rarely compare dealerships of different brands.
- C3: The fully general case of arbitrary weighted graphs with each venue pair (u, v) having arbitrary weight c_{uv}

To begin with, each reviewer has already reviewed some set of venues. Our second problem dimension is to consider a **Starting state** where each reviewer has either:

- S1: One prior review (an unrealistically simple scenario more useful for thought experiments and lower bounds than for practical use)
- S2: An arbitrary set of prior reviews, the general case.

There is a cost to soliciting new reviews. In Section 1.1, we modelled this by choosing L4 from the following list of possible **per-reviewer Limits** on what the optimization outputs. We may be looking for:

- L1: One more item to suggest to each reviewer $r \in R$. This is one natural setting for an online algorithm, where we may compute the next suggestion later if the reviewer comes back again ready for another suggestion.
- L2: Only some subset $O \subseteq R$ of reviewers can receive a review suggestion, so we can pick an item to suggest to each reviewer $r \in O$. This is somewhat more practical than L1, such as when we can use previous user activity patterns to decide whether the user will be interested in more review suggestions, or will even visit the site again.
- L3: k more items to suggest to each $r \in R$, to match a reviewer's attention budget.
- L4: One more item to suggest to each $r \in R$, from a set of items A_r that we believe r may be willing to review. These sets can be generated from explicit actions such as social network "check-ins" or from machine-learned reviewer models which might infer, e.g., that the reviewer frequents various taquerias in San Diego.
- L5: A parallel of L4 for k more items with per-reviewer allowed sets A_r (generalizing L2 and L3).

In Section 1.1, we allow our objective O_F to depend on an arbitrary concave function F, or sometimes the particular function $F_1(a) = \min\{a, 0\}$ which only cares whether an item has at least one reviewer. These correspond, respectively, to choices O3 and O1 along our final problem dimension of **Objective functions**:

- O1: Maximizing the total weight of item pairs (v_1, v_2) with at least one review pair ($O_1 = \sum_{u,v|a_{uv}\geq 1} c_{uv}$). This measures the relevance of a review-pair-based UI as a whole: how frequently will we have any such pairs to surface to a researching user?
- O2: Maximizing the total weight of item pairs with at least h review pairs ($O_2 = \sum_{u,v|a_{uv}\geq h} c_{uv}$), a natural "next step up" from O1, representing, e.g., a limit on the screen real estate available to display review pairs without scrolling.
- O3: Maximizing the weighted sum over all item pairs of a concave function of the number of review pairs, i.e. $O_3 = \sum_{u,v} c_{uv} F(a_{uv})$ for some concave F. Concavity corresponds to the natural constraint of decreasing marginal utility of additional review pairs. Notably, this generalizes O1 but not O2, since O2 does not credit an item pair until we've filled up a "screenful" quota of review pairs, creating a convex region in F.

Our original problem statement in Section 1.1 is described using this taxonomy as C3-S2-L4-O3 (or -O1).

6. EXACT PROBLEMS: LOWER BOUNDS AND AN ALGORITHM

To ground our decision to explore approximation algorithms, we now show that several problems in our taxonomy that are significantly simpler than the C3-S2-L4-O3 case approximated above are already NP-hard in their exact form (each is trivially in NP).

THEOREM 5. *It is NP-hard to exactly optimize C1-S2-L4-O1.*

PROOF. We reduce from POS-NAE-3SAT (Positive Not-All-Equals 3SAT) [24], the satisfiability problem on disjunctions of 3-literal "not-all-equals" clauses, with only positive literals (i.e. if use x as a literal, we can't use not-x as a literal elsewhere).

With such a formula, make each variable into a reviewer r_i, and each clause into a venue v_j, which has already been reviewed by the three reviewers r_{j1}, r_{j2}, r_{j3} (per S2). Set $A_r = \{v_T, v_F\}$ (per L4), which corresponds to the variable picking a Boolean value. Set all $c_{uv} = 1$. The POS-NAE-3SAT formula is satisfiable if and only if there is a correct assignment of reviews such that $2|V|$ new pairs are created – i.e. review pairs are created for all item pairs (v, v_T) and (v, v_F), which is equivalent to $O_1 = 2|V|$. □

THEOREM 6. *It is NP hard to exactly optimize C3-S1-L3-O1, even with a single reviewer, and even if the per-reviewer budget in L3 is given in unary for some obscure reason.*

PROOF. By reduction from Max-Clique in graph $G = (V, E)$. Set the relevance graph to $(V \cup \{v_0\}, E)$, and create a single reviewer with budget k who starts out with just the disconnected v_0. She can increase O_1 to $k(k+1)/2$ if and only if G has a clique of size k. □

THEOREM 7. *It is NP hard to exactly optimize C1-S2-L2-O1.*

PROOF. The reduction is from set cover. Starting with a set cover instance of m_s sets and n_s elements, we will show that one can cover all the elements using k_s sets, if and only if one can get $n_s + m_s - k_s$ new review pairs in a some graph of venues G with a set of reviewers R. We begin by describing the vertices of G. The graph G is a clique, so all edges exist.

Given a collection of elements $X = \{x_1, \ldots, x_n\}$ and a collection of sets $S_1, \ldots, S_m \subseteq X$, construct an instance of the review problem as follows. There are three kinds of venues:

- Each element x_i is represented by an *element venue* e_{x_i}.
- There are two special venues: a target for the elements ET and a target for the sets ST
- For each set S_j, there is a *set venue* denoted v_{S_j}.

The review graph G is a clique, so every pair of venues is equally interesting. There are m_s reviewers who have one more venue to review, and $m_s + 3$ reviewers who finished their reviews and cannot review any more.

- For every set S_j, there is a reviewer $r_j \in O \subset R$ who already reviewed v_{S_j} and e_{x_i} for every $x_i \in S_j$. This reviewer has one more venue they can review.
- For every set S_j, there is another reviewer $r'_j \in R \setminus O$ who already reviewed v_{S_j} and e_{x_i} for every $x_i \notin S_j$. This reviewer has finished their reviews.
- There is a reviewer who finished their reviews who reviewed ET and ST.
- There is a second reviewer who finished their reviews who reviewed ST and all the element venues.
- There is a third reviewer who finished their reviews who reviewed ET and all the set venues.

Given these reviewers, the pairs of venues which have not been reviewed are:

- All the pairs of the form e_{x_i}, ET where e_{x_i} is an element venue.
- All the pairs of the form v_{S_j}, ST where v_{S_j} is a set venue.

Therefore, in an optimal solution every reviewer r_j would choose either the element target ET, which would add all the edges e_{x_i}, ET where $x_i \in S_j$, or the set target ET, which would add the venue pair v_{S_j}, ST. The following claim is easy:

CLAIM 8. *The original set cover problem can be covered with k_s edges if and only if one can add $n_s + m_s - k_s$ new edges in the venue graph.*

Given a set cover solution, assign reviewer r_j to ET if set S_j was chosen, and to ST otherwise.

Conversely, given an assignment of reviewers to a set cover solution, include all sets S_j such that reviewer j chose ET and exclude the others. It is possible that this solution will not cover every element. If any element x_i is missing, it must be because every reviewer j for whom $x_i \in S_j$ chose to review ST If the assignment of reviewers is optimal, then this can only happen if every such S_j has been completely covered except for the element x_i. For each missing element x_i, pick an arbitrary set S_j containing it and add it to the solution. □

This last C1-S2-L2-O1 result also generalizes to C3-S2-L1-O1, immediately yielding the corollary:

COROLLARY 9. *It is NP hard to determine the optimal assignment for the reviewers for C3-S2-L1-O1.*

PROOF. Reduce from C1-S2-L2-O1 by taking any pair (u, v) that was co-reviewed by a reviewers who will not receive a suggestion (i.e. by some $r \in R \setminus O$), and zeroing out the weights c_{uv}, since O_1 has already been satisfied with regard to that review pair. This makes the relevance graph no longer a clique, moving the problem from C1 to C3, but lets us replace the reviewer set R with $R \setminus O$, changing the L2 setting to L1. □

C1-S2-L2-O1 and C3-S2-L1-O1 are fairly minimal models for most real applications, which makes for a strong argument for approximation algorithms. In a couple of even simpler models, though, we can get tractable exact algorithms — by restricting reviewers to only have a single prior review. It is no surprise that we have observed that the distribution of number of items reviewed per reviewer is long-tailed with a mode at 1, so these algorithms could be used for a partial solution for the large subpopulation of reviewers who have reviewed only one item so far:

THEOREM 10. *C3-S1-L4-O1 can be solved exactly in time:*

$$O(\min\{m^2|E|^2 \log(m|E|) + m|E|\alpha, \alpha m^2|E|^2\})$$

where $\alpha = \sum_r |A_r|$.

PROOF. We reduce the problem to maximum-weight bipartite matching on the following bipartite graph B. On the "left" side, for each venue v_i that has been reviewed by some set of reviewers R_i, make a node $v_{i,r}$ for every $r \in R_i$. On the "right" side, put a node $e_{v,w}$ for each relevant item pair: i.e. each edge $(v, w) \in E$. In B, $e_{v,w}$ is connected to each $v_{i,r}$ where r, having already reviewed v, is willing to also review w ($w \in A_r$); and, conversely, to each $w_{i,r}$ where r is willing to review v, having already reviewed w. Weigh all edges of B incident on $e_{v,w}$ with c_{vw}. Since we're in S1, there are exactly m nodes on the "left" side, one for each reviewer. Thus, a bipartite matching is exactly the assignment of each reviewer r to one additional venue in A_r, and maximizing its weight maximizes precisely O_1.

The running time is bounded by the known bounds on maximum-weight bipartite matching [13] on a graph with $m|E|$ nodes and α edges. □

This running time, while polynomial, is still fairly steep for global-scale applications, and it turns out that we should not expect much better:

THEOREM 11. *Even C1-S1-L4-O1 has a linear-time reduction from unweighted maximum bipartite matching.*

PROOF. The general maximum bipartite matching problem on unweighted graph (U, V, E) is equivalent to the following C1-S1-L4-O1 instance. Create a reviewer for each $u \in U$, a venue for each $v \in V$, plus a special venue v_0,

with each pair of venues weighted equally. Every reviewer starts out with just v_0 (S1), and may get exactly one suggestion from her set $A_u = \{v|(u,v) \in E\}$. Given that we can only assign one new review per node, only venue pairs containing v_0 can get a review pair, so the O_1 objective function is exactly the objective function of maximum bipartite matching. □

This makes the problem *complete for Maximum Bipartite Matching*, another practical, well-studied problem where the field does not expect significant running-time improvements over the current best known $O(\min\{V^\omega, \sqrt{V}E\})$ bound [14, 20]. Thus, we should not expect that even C1-S1-L4-O1, with n items and m reviewers, will be solvable faster than $\Theta(\min\{(nm)^\omega, \alpha\sqrt{nm}\})$. Since we can throw out any reviewer with empty A_r, $\alpha = \Omega(m)$, making the algorithm $\Omega(m^{1.5})$ at the very least, which ensures that the problem will not have exact online solutions.

The only setting in which we have a truly scalable exact algorithm, which runs in $\tilde{O}(n+m)$, is the most minimal setting in our taxonomy, which we expect to be of mostly theoretical interest:

THEOREM 12. *C1-S1-L1-O1 can be solved in a time* $\tilde{O}(n+m)$.

PROOF SKETCH. Label the venues $\{1, \ldots, n\}$ so that venue i starts with r_i prior reviewers, with $r_1 \geq r_2 \geq \ldots$. The greedy algorithm operates on the venues in this order. For algorithm steps $t = 1, \ldots, n$, let $s_i(t)$ denote the number of the people who reviewed restaurant i before this step. At each step, we distribute the r_t reviewers initially assigned to venue t in lexicographical order over the following 3-tuple objective function, listed highest-precedence first:

1. First, (A) venues $i < t$ that haven't sent a reviewer to r_t, then (B) all venues $i > t$
2. Within (A) and (B), vertices i with lower $s_i(t)$ come earlier
3. Break ties of $s_i(t)$ by giving i a reviewer before j whenever $i > j$ (i.e. $r_i < r_j$).

The surprisingly convoluted case analysis for the correctness proof is deferred to the full version of this paper. □

7. GENERALIZATIONS

Our approximation algorithms and hardness results describe the feasibility of most parts of the problem taxonomy in Section 5. We expect that our positive results will be applicable in the many settings where comparison research can benefit from the highlighting of review pairs written by the same reviewer.

Of course, there are other approaches to making a review ecosystem more comparison-friendly by actively suggesting item-reviewer pairs. Our algorithms can be useful in several broader settings, but with limitations:

Reviewer categories. When reviews or reviewers are scarce, we may need a weaker notion of comparability such as pairing reviews by people in the same category: e.g. "both golf clubs have been reviewed by an author to Golf magazine" or "both kilts have been reviewed by 30–35-year-old men." Our algorithms can run in this setting if each reviewer is assigned to one relevant category in advance, and all reviewers in each category are merged into a "meta-reviewer" for the purposes of the algorithm. However, we then need to determine relevant categories in advance, and cannot exploit the structure of a membership of a reviewer in multiple categories.

Item aspects. When items have several known aspects which may not get covered by any one review, we can consider suggesting, to prospective reviewer, either an item-aspect pair ("Could you review Bob's diner and focus on their breakfast menu variety?"), or, for a reviewer already set on reviewing a particular item, suggesting aspects of interest ("While you're reviewing Bob's hotel, could you rate the pool and the A/C system?"). Our algorithms can approximate this by expanding the relevant pairs graph G to representing each item-aspect pair as a separate "item" node, putting weight on identical aspects of relevant item pairs: "breakfast quality at Alice's diner" and "breakfast quality at Bob's diner" are then a relevant pair if Alice's and Bob's are frequently compared, but "Alice's appetizers" and "Bob's bathrooms" aren't. However, this will not capture this combinatorial structure of the item-aspect space, so we will not be able to consider that it's easier to review 3 aspects of a single item than 3 unrelated item-aspect pairs.

We leave open the many questions of how to best optimize suggestions for these and other broader settings while utilizing more of the additional combinatorial structure. We also leave open the design questions of how to combine these suggestion-planting algorithms with the existing NLP-based passive review mining, as well as the design and HCI considerations in building a review-comparison UI to best exploit these algorithms.

Acknowledgments

We are indebted to Ed Chi, John Hawkins, Walter Korman, Gueorgi Kossinets, Yishay Mansour, and Fil Zembowicz for a number of insightful conversations that helped shape this work.

8. REFERENCES

[1] Amazon. http://www.amazon.com/.
[2] Easychair. http://www.easychair.org/.
[3] Facebook. http://www.facebook.com/.
[4] Google+. http://plus.google.com.
[5] Internet movie database. http://www.imdb.com/.
[6] Newegg. http://www.newegg.com/.
[7] C. Avery, P. Resnick, and R. Zeckhauser. The market for evaluations. *Amer. Econom. Rev.*, 89(3):564–584, 1999.
[8] J. Chevalier and D. Mayzlin. The effect of word of mouth on sales: Online book reviews. *J. Marketing Res.*, 43(3):345–354, 2006.
[9] comScore/the Kelsey group. Online consumer-generated reviews have significant impact on offline purchase behavior. Press release, November 2007. http://www.comscore.com/press/release.asp?press=1928.
[10] Chrysanthos Dellarocas. The digitization of word of mouth: Promise and challenges of online feedback mechanisms. *Management Science*, 49(10):1401–1424, 2003.
[11] DoubleClick. DoubleClick's Touchpoints II: The changing purchase process, mar 2004.

[12] Angela Edmunds and Anne Morris. The problem of information overload in business organisations: a review of the literature. *International Journal of Information Management*, 20:17–28, 2000.

[13] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, July 1987.

[14] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[15] John A. Horrigan. Online shopping. Pew Internet & American Life Project Report, 2008.

[16] N. Hu, P.A. Pavlou, and J. Zhang. Can online reviews reveal a product's true quality? empirical findings and analytical modeling of online word-of-mouth communication. In *Proceedings of the 7th ACM conference on Electronic commerce (EC'06)*, pages 324–330, 2006.

[17] Cynthia Huffman and Barbara E. Kahn. Variety for sale: Mass customization or mass confusion? *Journal of Retailing*, 74(4):491–513, 1998.

[18] Xinxin Li and Lorin M. Hitt. Self-selection and information role of online product reviews. *Information Systems Research*, 19(4):456–474, 2008.

[19] Naresh K. Malhotra. Information load and consumer decision making. *Journal of Consumer Research*, 8(4):419–430, March 1982.

[20] M. Mucha and P. Sankowski. Maximum matchings via Gaussian Elimination. In *Proc. 45st IEEE FOCS*, pages 248–255, 2004.

[21] C. Oppenheim. Managers' use and handling of information. *International Journal of Information Management*, 17(4), 1997.

[22] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135, 2008. Ch. 6.

[23] C. Piller. Everyone is a critic in cyberspace. *Los Angeles Times*, December 3, 1999.

[24] A.A. Schäffer. Simple local search problems that are hard to solve. *SIAM journal on Computing*, 20(1):56–87, 1991.

[25] A.J. Stanley and P.S. Clipsham. Information overload — myth or reality? In *IEE Colloquium on IT Strategies for Information Overload Digest*, number 1997/340, pages 1–4, 1997.