Daily-Deal Selection for Revenue Maximization

Theodoros Lappas
Boston University
tlappas@cs.bu.edu

Evimaria Terzi
Boston University
evimaria@cs.bu.edu

ABSTRACT

Daily-Deal Sites (DDS) like Groupon, LivingSocial, Amazon's Goldbox, and many more, have become particularly popular over the last three years, providing discounted offers to customers for restaurants, ticketed events, services etc. In this paper, we study the following problem: among a set of candidate deals, which are the ones that a DDS should feature as daily-deals in order to maximize its revenue? Our first contribution lies in providing two combinatorial formulations of this problem. Both formulations take into account factors like the diversification of daily deals and the limited consuming capacity of the userbase. We prove that our problems are NP-hard and devise pseudopolynomial – time approximation algorithms for their solution. We also propose a set of heuristics, and demonstrate their efficiency in our experiments. In the context of deal selection and scheduling, we acknowledge the importance of the ability to estimate the *expected revenue* of a candidate deal. We explore the nature of this task in the context of real data, and propose a framework for revenue-estimation. We demonstrate the effectiveness of our entire methodology in an experimental evaluation on a large dataset of daily-deals from Groupon.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce; H.2.8 [**Database Management**]: Database applications—*Data mining*

Keywords

daily deals, revenue maximization, deal selection, e-commerce

1. INTRODUCTION

Daily Deal Sites (DDS), like Groupon[1], LivingSocial[2], and Amazon's Goldbox[3] are among the latest Internet sensations, recommending group deals to users on a daily basis. The deals offered by DDSs are typically geographically localized and include discounted prices to restaurants, cafes, event tickets, spa services etc. Groupon is a characteristic example of a DDS. Since its introduction in 2008, it has been one of the fastest-growing Internet-sales businesses in history. Groupon has since gone public, with its Initial Public Offering (IPO) valuing the company at almost 13 billion dollars.

The business model of DDSs: Every day, a DDS features a small number of *daily deals*. Each deal provides a discounted rate (usually a 40-60% discount) to some item or service. These deals are usually geographically localized, corresponding to services or products that are available to residents of a particular city or a bounded geographic area. For example, deals often correspond to discounted rates to a local restaurant or beauty salon. Every day, the registered DDS users receive an email that notifies them of the available daily deals in their area. Once a user decides to buy the discounted item, he logs in the DDS and expresses his interest in participating in the deal. If a sufficiently large number of users has expressed such interest by the end of a pre-specified period of availability (usually one or two days), the deal gets *materialized*. Then, the users who expressed interest get charged the pre-specified amount and are given a *coupon*, which they can subsequently use in return for the advertised service. In our work, we refer to the number of coupons sold per deal as the *size* of the deal. The *revenue* of the deal is then equal to its size multiplied by the price of the coupon. The DDS typically retains a percentage of the revenue of every deal. For example, for Groupon, a 50% [14] commision is applied on every deal.

Existing work on DDSs: The new e-commerce model adopted by DDSs has motivated a significant body of relevant work. Economists have tried to analyze the factors that affect the generated revenue [1], as well as the effect of daily deals on the merchants that provide the discounted services [11, 12]. More recently, data-mining researchers have been intrigued by the daily-deal phenomenon and conducted large-scale data-analytic studies on datasets from DDSs [5, 4, 21]. These studies focus on discovering (*i*) the effect of different factors on the revenue and size of daily deals [4], (*ii*) the effect of daily deals on customers' reviews [5] and (*iii*) the impact of the users' social network on the propagation of daily deals [21].

Our contribution: Although the above studies provide useful insights on some aspects of the daily-deal paradigm, the algorithmic problems that arise in this domain have not yet

[1] `www.groupon.com`
[2] `www.livingsocial.com`
[3] `http://www.amazon.com/gp/goldbox`

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

been tackled. In fact, despite the immense popularity of DDSs, there has been little research on modeling and solving the relevant computational problems in a principled and efficient manner. Our work is the first to formalize these problems and present appropriate algorithmic techniques for revenue-maximization. Specifically, the contributions of this paper are the following:

- Our work is the first to formalize the following problem: given a set of deals from local businesses, which of them should the DDS select, in order to maximize its expected revenue? In addition to revenue maximization, our formulation takes into account relevant factors with significant impact, such as the *diversity*[18, 15] of the selected set and the limited consuming *capacity* (size) of the market [7].

- From the computational point of view, we study two variants of the above problem, which we refer to as the DEAL SELECTION and the DEAL SCHEDULING problems. The first problem handles the selection of a set of deals for singly day, while the second one models a long-term scheduling strategy for multiple days. In a detailed analysis, we discuss the intricacies of the two problems and prove that they are both NP-hard.

- Considering the large amounts of data that need to be considered by a deal-selection algorithm, we identify the need for efficiency and scalability. Motivated by this, we present a pseudopolynomial optimal algorithm for the DEAL SELECTION problem, and a pseudopolynomial 2-approximation algorithm for the DEAL SCHEDULING problem. We also consider even more efficient heuristics for both problems, and use experiments on real data to demonstrate their competitiveness.

- In order for a deal-selection mechanism to be effective, it requires the ability to estimate the popularity and expected revenue of a deal. We develop a machine-learning methodology that estimates user-interest in a particular deal, by considering a number of contributing factors. These include, among others, the domain, location, and reputation of the participating merchant, the coupon price, and the discount percentage. Our framework performs well on real data and motivates interesting observations on the factors that affect the success of a deal.

Note that our algorithms for the DEAL SELECTION and DEAL SCHEDULING problems can also be applied *retrospectively* on a set of deals that have already been featured. The size and revenue of each deal are then already known. In this setting, our methods for deal-selection can be used to a-posteriori evaluate the strategies of deal allocation. While this is an interesting application, the ability to estimate the interest of users allows us to utilize our framework for *proactive* deal selection and scheduling. We consider this latter setting as the main application of our work.

Roadmap: The rest of the paper is organized as follows: in Section 2 we present the DEAL SELECTION and the DEAL SCHEDULING problems. In Section 4 we discuss our methodology for estimating user-interest in a daily deal. We discuss our experimental results in Section 5 and summarize related work in Section 6. We conclude the paper in Section 7.

2. DEAL SELECTION

In our setting, we focus on the operation of a DDS within a particular geographic area. Within each area, we assume that there are m possible *markets* $\mathcal{M} = \{M_1, \ldots, M_m\}$. Each market corresponds to a particular *type* of business, e.g., restaurants, beauty & spa, coffee shops etc. That is, all the businesses in a particular market serve the same needs for the users. Each market M_i consists of a set of n_i *businesses*. We assume the current mode of operation of DDSs, where each daily deal is about a single business.

Every business b, when featured as the vendor of a daily deal, has an *(expected) revenue*, denoted by $R(b)$. Typically, the revenue of a deal is computed by multiplying the price of the coupon with the (estimated) number of sold coupons (recall that the DDS keeps a constant percentage of this amount for every deal). We adopt this definition in our experiments. Alternatively, one could opt for a different instantiation of revenue (e.g. the number of new users that register to the DDS because of this deal). In addition, the DDS may choose to incorporate its marketing strategy to the revenue function. For example, this can be done by adopting a function that favors deals in emergent and promising markets, or deals from high-quality businesses that can help the public image of the DDS. In any case, our methodology is compatible with *any* definition of the revenue function.

A basic assumption in our framework is that it considers the financial, as well as the time-based resources of the registered DDS users. That is, users cannot spend an infinite amount of money, neither do they have the time to accept the services from a very large number of businesses within a limited time interval. Given the population of users (e.g., the registered users of a DDS), we use C to represent the population's *(consuming) capacity*. As with revenue, the capacity of the population can be quantified in different ways [4]. These include the total number of coupons the users are willing to buy, the total number of hours they can spend on recreational activities, and the total amount of dollars they can spend on coupons. In our experiments, we measure C using the total number of coupons the users are willing to buy within a fixed time interval. Capacity can be estimated in a number of different ways by the DDS. These include studying past coupon sales in area and considering the area's population, demographics and income levels. The estimation of capacity is an interesting problem that we plan to study in future work. In this paper, we assume that C is provided as part of the input, and explore its effect on revenue maximization in our experiments. Once a deal from a business b is materialized, it consumes a portion of C. We refer to this portion as the deal's *(expected) size* and denote it by $S(b)$.

The goal of the DDS is to select a set of deals that maximize each revenue within a pre-specified period of time (e.g., a day, a week or a month), given the limited consuming capacity of the user-base. In addition, our framework consider the additional constraint of *diversity* [18, 15]. This constraint enforces the diversification of the set of featured deals, ensuring that it includes different types of markets. In addition to providing users with more options, showing a diverse set also makes the set of deals appealing to a wider range of users. Further, the diversity constraint can be used to limit the number of deals from a specific market, based on estimates on

[4] `http://www.marketingprofs.com/Tutorials/marketsize1.asp`

the demand of the user-base for such deals. We capture this intuition by imposing an upper bound on the total number of deals that can be selected per market. That is, we impose that at most K_i deals with businesses from market M_i can be selected. Taking into consideration the capacity and diversity constraints, we formalize the problem as follows:

PROBLEM 1 (DEAL SELECTION). *Assume a population with capacity C and m markets $\mathcal{M} = \{M_1, \ldots, M_m\}$, where each market M_i includes of a set deals from different businesses. Then, given the expected revenue and size for each deal, select a set of deals $\mathcal{D}$ such that the total revenue*

$$\text{TOTALR}(\mathcal{D}) = \sum_{b \in \mathcal{D}} R(b)$$

is maximized and for every $i = 1, \ldots, m$

$$|\mathcal{D} \cap M_i| \leq K_i \text{ [diversity constraints],}$$

and

$$\sum_{b \in \mathcal{D}} S(b) \leq C \text{ [capacity constraint].}$$

The above problem definition assumes that the K_i's and the value of C are given as part of the input. The effect of these parameters in the obtained results are demonstrated in the experimental section.

From the computational point of view, the DEAL SELECTION problem is NP-hard. This is due to the fact that DEAL SELECTION is a generalization of KNAPSACK, which is NP-hard [10]. More specifically, for $m = 1$ and $K_1 = n_1$, the problem is identical to KNAPSACK. In other words, our problem without the diversity constraints becomes identical to the KNAPSACK problem. The way the diversity constraints affect the algorithmic solution to the DEAL SELECTION problem is explored further in the next section.

2.1 An optimal algorithm for the DEAL SELECTION problem

In this section, we present an optimal algorithm for the DEAL SELECTION problem. Since our Problem 1 is a a generalization of the KNAPSACK problem [10], known techniques for the latter are not applicable. Therefore, we present a new algorithm, based on *nested dynamic-programming*, that is appropriate for our formulation. We refer to this algorithm as `DP-DP`.

In order to present the `DP-DP` algorithm we first need to define the *global revenue* and *market revenue* functions. The *global revenue* function, $\text{GR}(i, c)$, represents the maximum benefit of the optimal solution considering businesses from markets M_1 through M_i, and a population capacity equal to c. Using this definition, the optimal solution to the DEAL SELECTION problem corresponds to the value $\text{GR}(m, C)$.

Within a particular market M_i, we define the *market revenue* $\text{MR}_i(\ell, c, k)$ to be the maximum revenue that the DDS can earn from businesses in market M_i, given that (a) it can feature up to k out of the the first ℓ businesses of M_i, and (b) the total size of the selected businesses is no more than c. For a fixed capacity c, the optimal revenue that the DDS can have from market M_i is then $\text{MR}_i(n_i, c, K_i)$. Note that, although this definition assumes an order of the businesses within M_i, this order can be arbitrary.

Using the notation above, the optimal value of the optimization function TOTALR() (Problem 1) is equal to the value of $\text{GR}(m, C)$. Next, we show that the computation of $\text{GR}(m, C)$ can be done optimally using the following dynamic-programming recursion:

$$\text{GR}(m, C) = \max_{c=0,\ldots,C} \Big\{ \text{GR}(m-1, C-c) + \text{MR}_m(n_m, c, K_m) \Big\}. \quad (1)$$

Observe that the second term in the above recursion, (i.e., $\text{MR}_m(n_m, c, K_m)$ is the optimal revenue from market M_m given that only K_m businesses can be selected from M_m and the capacity allocated to this market can be at most c.

In order to compute the value of $\text{MR}_m(n_m, c, K_m)$, we need to invoke another dynamic-programming computation; in this case the recursion requires the evaluation of a 3-dimensional table. Next, we show the recursion for evaluating this table for market M_i. Observe that the same recursion can be used for every $i \in \{1, \ldots, m\}$. If for every market M_i we assume that the businesses within M_i are arranged in some random order $b_1, \ldots, b_{n_i}$, then we can evaluate $\text{MR}_i(n_i, c, K_i)$ using the following recursion:

$$\text{MR}_i(n_i, c, K_i) = \max \Big\{ \text{MR}_i(n_i - 1, c, K_i), \text{MR}_i(n_i - 1, c - S(b_{n_i}), k_i - 1) + R(b_{n_i}) \Big\}. \quad (2)$$

This recursion can be evaluated using dynamic programming. Observe that the running time for evaluating Recursion (2) is $O(n_i C K_i)$. One such evaluation needs to be done for each one of the m markets. However, since every business belongs to only one market, these computations can be done separately (and in parallel) for every market. Given that the values $\text{MR}_i(n_i, c, K_i)$ are precomputed, evaluating Recursion (1) requires $O(mC^2)$ time. Therefore, the running time of the `DP-DP` algorithm depends on C, which is not necessarily polynomial in the size of the input, making the overall running time of `DP-DP` to be pseudopolynomial.

Evaluating the dynamic-programming recursion expressed in Equation (1) requires another dynamic-programming computation, i.e., the one expressed by Recursion 2. For that reason, we say that `DP-DP` is a *nested dynamic-programming* algorithm. For `DP-DP`, we can state the following:

LEMMA 1. *The `DP-DP` algorithm solves the* DEAL SELECTION *problem optimally.*

If each market M_i is viewed as a super-item with revenue given by the function $\text{MR}_i()$, then Recursion (1) is a standard KNAPSACK-type recursion that computes the optimal global revenue $\text{GR}(m, C)$. In turn, the optimality of the market-revenue values can be argued through Recursion (2). In this recursion, the optimal values for every $\text{MR}_i(n_i, C, K_i)$ are computed. Given optimal market revenues, the global revenue computed by Recursion 1 is also optimal.

2.2 Efficient heuristics for the DEAL SELECTION problem

Although solving the DEAL SELECTION problem is an offline computation for the DDSs, the computational complexity of `DP-DP` could make it inefficient for very large datasets. In this section, we present some alternative heuristic algorithms for the DEAL SELECTION problem. As we show in

our experiments, this algorithms achieve high-revenue solutions, while having a clear advantage over DP-DP in terms of computational cost.

The Fast-DP algorithm: The Fast-DP algorithm is based on the same principles as DP-DP. However, Fast-DP sacrifices optimality for efficiency. Recall that the inefficiency of DP-DP is due to the fact that the dynamic-programming tables GR and MR_i for $i = 1 \dots m$ have a single dimension of size C. When the value of C is very large, then DP-DP becomes rather inefficient. We overcome this computational problem by splitting the value of C into intervals (buckets) of size c. This way, the dimensionality of the dynamic-programming tables GR and MR_i for $i = 1 \dots m$ reduces to $m \times C/c$ and $n_i \times K_i \times C/c$ respectively. Therefore, instead of computing the global and the market revenue functions for all possible number of coupons, we only compute their values for increments of size c. We refer to this version of the DP-DP algorithm as Fast-DP(c); we parameterize it with the value of the increment c. In our experiments, we consider two values of c, namely $c = 10$ and $c = 100$. Other than that, Fast-DP is also a nested dynamic-programming algorithm that evaluates Equations (1) and (2) in a manner identical to that of DP-DP. The only difference is that Fast-DP operates on dynamic-programming tables that keep less information and have smaller size.

Observe that creating buckets of size c does not decrease the worst-case running time of Fast-DP, which remains equal to that of DP-DP. However, our experiments illustrate that, in practice, the Fast-DP algorithm is considerably faster than DP-DP, while achieving near-optimal revenue.

The Sort algorithm: The Sort algorithm is a heuristic approach that ignores the categorization of businesses into markets. The algorithm first sorts all the available businesses in decreasing order of the *sorting criterion*. Then, it traverses the set of businesses in this order and adds into the solution a deal with every business that satisfies the diversity and capacity constraints. Given that the sorting criterion can be computed for every business in constant time, the running time of the Sort algorithm is $O(n \log n)$. In our case we sort the businesses using the benefit-to-size ratio (BSR). Given business b, the benefit-to-size ratio of b is $\text{BSR}(b) = R(b)/S(b)$.

3. DEAL SCHEDULING

In this section, we explore the scenario where the DDS wants to schedule multiple sets of deals to be presented at different time intervals (e.g. days), instead of selecting a single set.

PROBLEM 2 (DEAL SCHEDULING)). *Assume a sequence of time intervals $T = \{t_1, \dots, t_k\}$. and a population that has the same capacity C at every interval. Additionally, assume m markets $\mathcal{M} = \{M_1, \dots, M_m\}$, where each market M_i consists of n_i deals from different businesses. Then, given the expected revenue and size for every deal, select the set of deals $\mathcal{D}_t$ to feature at every interval, such that every deal is featured only once and the following value is maximized (i.e. the total revenue of the DDS):*

$$\text{TOTALR}(\mathcal{D}_1, \dots, \mathcal{D}_t) = \sum_{t \in T} \sum_{b \in \mathcal{D}_t} R(b)$$

under the following constraints:

for every $t \in T$ and $i = 1, \dots, m$

$$|M_i \cap \mathcal{D}_t| \leq K_i \text{ [diversity constraint]}$$

for every $t \in \{1, \dots, T\}$

$$\sum_{b \in \mathcal{D}_t} S(b) \leq C \text{ [capacity constraint].}$$

Observe that, in the definition of Problem 2, we have assumed that the revenue and the size of every deal are fixed across time intervals. A similar assumption was made for the population capacity, which remains the same across intervals. Although we make these assumptions for clarity of presentation, our algorithmic results carry over for the case where these parameters are different for each time interval.

The DEAL SCHEDULING problem is a generalization of the DEAL SELECTION problem and, therefore, it is also NP-hard. In fact, one can observe that the DEAL SCHEDULING problem is a generalized version of the MULTIPLE KNAPSACK problem [8]; for $m = 1$, the DEAL SCHEDULING problem is identical to the MULTIPLE KNAPSACK problem.

3.1 An algorithm for DEAL SCHEDULING

We solve the DEAL SCHEDULING problem using a greedy algorithm that works as follows. Let $\mathcal{M}^{(t)}$ be the set of deals that have not been selected up to time t. Then, the algorithm solves the DEAL SELECTION problem using $\mathcal{M}^{(t)}$ as input. Any of the the DP-DP, Fast-DP or Sort algorithms can be used for solving this problem. Let $\mathcal{D}_t$ be the set of deals selected by this step. Then, we get $\mathcal{M}^{(t+1)} = \mathcal{M}^{(t)} \setminus \mathcal{D}_t$, which is used as input to the DEAL SELECTION problem that needs to be solved for interval $(t + 1)$. We refer to this algorithm as the GreedySchedule algorithm. Depending on the algorithm we use for solving the DEAL SELECTION problem at every interval t, we get a different version of the GreedySchedule. We denote the different versions as GreedySchedule(A) for $A = \{$DP-DP, Fast-DP, Sort $\}$. Clearly, the running time of the GreedySchedule algorithm is T times the running time of the algorithm that is used for solving the DEAL SELECTION problem, where T is the number of time intervals for which a different set of deals needs to be selected.

By applying the results of Cohen *et al.* [9] to our problem, we get the following result.

LEMMA 2. *The GreedySchedule(DP-DP) algorithm is a 2-approximation algorithm for the DEAL SCHEDULING problem.*

Although we omit the complete proof, we simply state here that the results of Cohen *et al.*, translated to our setting, state that the GreedySchedule(A) algorithm is an $(1 + \alpha)$ approximation for the the DEAL SCHEDULING problem. The factor α in this statement corresponds to the approximation factor achieved by algorithm A, which is used to solve the DEAL SELECTION problem. When A is DP-DP, then $\alpha = 1$, since DP-DP is optimal for the DEAL SELECTION problem. Therefore, GreedySchedule(DP-DP) is a 2-approximation algorithm for the DEAL SCHEDULING problem.

4. ESTIMATING DEAL SIZE

Our framework for DEAL SELECTION and DEAL SCHEDULING assumes that we can estimate the expected size and revenue for every deal. The size of the deal is taken to be equal

to the number of coupons that it is estimated to sell. The revenue can then be computed as the size multiplied by the price of the coupon. Since the price of the coupon is set by the merchant and is provided as input, we focus our efforts to estimating the size of the deal.

A naive way of predicting the size of a deal is by observing the sequence of past daily deals and then using the average observed size as an estimator. Obviously, this method disregards the impact of a number of factors that can greatly affect the number of coupons that a deal sells. Identifying these factors and measuring their effect on the size of the deal is a very challenging task. First, we describe the factors that we consider in our work, and discuss how they can be learned from real data. We then incorporate these factors into a machine-learning framework that predicts the size of a given deal. Given a candidate deal, we consider the following factors that can affect its outcome:

Deal location: Each deal is usually available in a particular city or area. It is likely that people in different cities have different tendencies and preferences with respect to different types of offers. Therefore, we consider the location of the merchant (or service) as a factor that influences the size of the deal. Information about a deal's location is usually made available on the DDS's website and is thus easy to retrieve.

Business reputation: DDSs serve as mediators between the users and the merchants. As in every e-commerce model, the reputation of the merchant can greatly affect the number of interested customers. For example, an offer for a restaurant with poor reputation is less likely to be popular.

For some types of merchants (e.g. restaurants), reputation can be quantified using the reviews that have been submitted by users. For example, one can adopt the average star rating or some other review-based quality measure to estimate the reputation of a restaurant. However, for other merchants, reviews may be unavailable or extremely rare. To address this, we quantify the reputation of a business by tapping into an additional resource: the popularity of the merchant's *website*. A number of portals exists that are dedicated to website-evaluation. Arguably, the most well-known among these portals is Alexa (`www.alexa.com`). For each website, Alexa reports two interesting popularity measures: the *global traffic rank* and the website's *reputation*. The former is based on the amount of traffic observed for the site. The latter quantifies popularity by counting the number of incoming hyperlinks to the site. Our intuition is that the reputation of a merchant's website is a reflection of the merchant's overall reputation and popularity. For example, a coupon for a well known, high-profile spa center is more attractive than one from a recently opened competitor. Such phenomena are bound to be captured by considering the popularity of the respective websites. In our experiments, we consider both user reviews and website-evaluation measures as factors that affect the size of a candidate deal.

Business type: The type of the offered product or service (e.g., spa, restaurant, ticketed event etc) is also expected to play a critical role with regards to the deal's selling potential. Although users can identify the type of a deal very easily, the automated extraction of a deal's type by crawling the DDS's website is not a trivial task. The challenge lies in the fact that deals are typically organized on the DDS's website by city, and the type is not always explicitly stated. For the user viewing the deal's page on the website, the type is made clear by pictures and textual information such as the title and the accompanying text. However, in order to automatically deduce the type of a deal, we need a method that can parse the deal's accompanying text and decide the type of the featured business. For this purpose, we use Latent Dirichlet Allocation (LDA) [3]. LDA is a well-known technique used to learn the topic distribution of each document in a given corpus, as well as the term distribution of each distinct topic. In our context, the text that accompanies each deal serves as a single document. We can then group deals based on the assigned topic distribution. One can think that different topics correspond to different markets. More details of this process are given in the experiments section.

Price: Intuitively, a lower-cost deal is expected to sell more than an expensive one. After all, cheap deals are attractive to a wider range of customers. The price of a deal is easy to retrieve, since it is always made available by the DDS.

Discount: The key for the DDSs' success is that they offer products and services at a rate lower than their standard market-price. The size of the discount is thus another factor to be considered, in addition to the price. For example, even though the price of a deal may be high, the deal can still be attractive if it applies a significant discount to the standard market-price. The original value and the offered discount are available in virtually every daily-deal website. **Seasonality:**

Another feature we extract from our data is the time of the year when the offer is made available. For example, it may be the case that people are more likely to opt for solarium treatments before or during the summer months.

Tipping point: The tipping point of a deal is the minimum number of users that need to express interest, in order for the deal to be materialized. This captures the fact that merchants are willing to offer their services at the discounted price only if the DDS guarantees to them a significantly large number of customers. Daily deals featuring established merchants will attract larger number of users and can thus tolerate larger tipping points. Hence, we consider the tipping point as a factor that could correlate with the size of the deal.

Using the above factors as dimensions, we represent every candidate deal as a point in the respective multidimensional space. We can then define the problem of estimating the number of coupons as a regression or classification problem, where the goal is to use the training data to build a model that can then estimate the size of a deal. Our own estimation mechanism is based on regression. Our experiments with different regression mechanisms revealed that the SVM-based regression model suggested by Shevade *et al.* [20] gave the best results. We evaluate our methodology on real datasets in Section 5.4 of the experiments, and present interesting findings that provide valuable insight on the estimation task.

5. EXPERIMENTS

In this section, we describe the experimental evaluation that we performed to verify the efficacy of our methodology. The section is organized as follows: first, we describe the dataset that we use in our experiments. Then, we present the evaluation of our algorithms for solving the Deal Selection and the Deal Scheduling problems. Our evaluation focuses both on the quality of the obtained solutions (as expressed by the objective function, i.e., the total revenue) and on the

running time of the algorithms. Finally, we conclude this section with an evaluation of our machine-learning framework for estimating the size of daily deals.

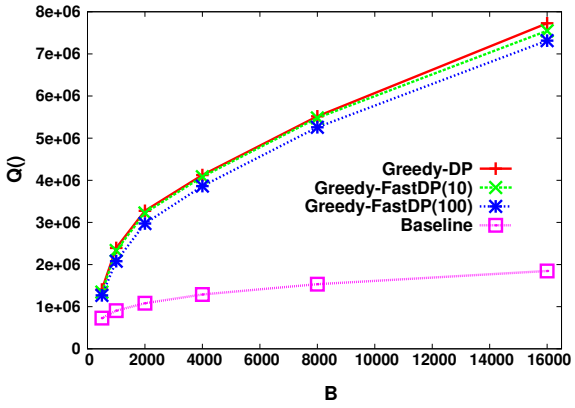
5.1 Data

In order to evaluate our methodology, we collected a set of $n = 100,000$ materialized and closed (i.e., past) deals offered by Groupon. These are deals that started within the last six months, coming from all divisions (i.e., cities) in which Groupon features daily deals. For each deal, we extracted the following set of attributes: the date, the city of the merchant featured in the deal, the name of the merchant, the price of the offered coupon, the discount from the original value, the number of coupons sold, the descriptive text accompanying the deal, and the merchant's website. For every merchant that was featured in a deal, we also quantified the merchant's reputation in the following two ways. First, using the URL of the merchant's website, we probed Alexa (`www.alexa.com` to retrieve the website's *Rank* and *Reputation* scores. The first score quantifies the reputation of the website using as indication the observed traffic. The second score quantifies the website's reputation based on the number of sites that point to the merchant's website. In addition to using Alexa's scores, we quantify the reputation of a merchant based on user reviews. Given the name of the merchant appearing in the deal, we probe the popular review-hosting site Yelp (`www.yelp.com`) and we retrieve the set of reviews for this merchant that are available at Yelp. We do this either by directly querying Yelp for the merchant's name or by following the link to Yelp that is often in Groupon's page for the deal.

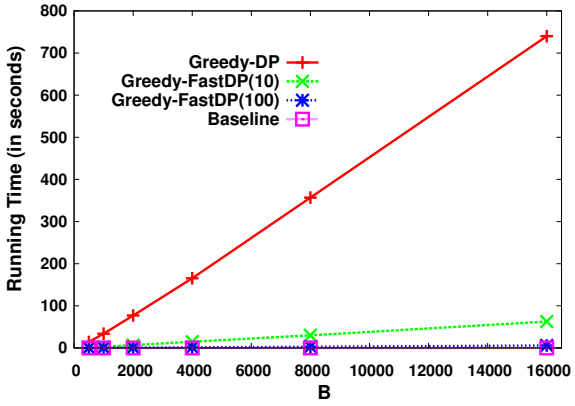
Further, we identify the market that each deal belongs to (i.e. its type). This refers to the type of service that the deal is offering (e.g. restaurants, spas, gym memberships etc.). Unfortunately, the type of each deal is not explicitly mentioned by the DDS. However, since we believe that this is an important feature that will allow us to predict the deal's performance, we learn the type of using the descriptive text of the deal as shown on the DDS website. As we have already described in Section 4, we do this by applying LDA to the dataset of collected deals. We learn a model consisting of 50 topics. The number 50 was chosen after experimentation with different numbers. An inspection of the term distribution of the learned topics showed that this number led to cohesive topics that represent real-life concepts. Therefore, our dataset has $m = 50$ distinct markets and each business belongs to one of these markets (or types). Each deal is assigned to the market corresponding to the topic with the highest probability in the deal's topic distribution.

5.2 Evaluating the algorithms for the Deal Selection problem

In this section, we explore the behavior of the `DP-DP`, `Fast-DP` and `Sort` algorithms for the different values of the parameters of the Deal Selection problem. For each experiment we report the value of the objective function (TotalR) that was achieved by every algorithm, as well as the running time for every approach. For all our experiments, we use two instantiations of the `Fast-DP` algorithm, namely `Fast-DP`(10) and `Fast-DP`(100). In the first instantiation, we consider coupon increments of size 10, while in the second we consider increments of size 100. For these experiments we use the dataset we collected from Groupon and we described in Section 5.1. Recall that this dataset consists of $n = 100,000$ deals and $m = 50$ different markets.



(a) Total revenue (TotalR).



(b) Running time in seconds.

Figure 1: Total Revenue (Figure 1(a)) and Running Time (Figure 1(b)) of `DP-DP`, `Fast-DP`(10), `Fast-DP`(100) and `Sort` algorithms, as a function of the consuming capacity C of the population: $C \in \{500, 1000, 2000, 4000, 8000\}$. For all markets $i \in \{1, \ldots, 50\}$, $K_i = K = 50$.

5.2.1 *Exploring the effect of the capacity constraint*

First, we report the value of the total revenue (TotalR) and the running times of the `DP-DP`, `Fast-DP` and `Sort` algorithms as a function of the population's consuming capacity C. For this, we experiment with values of $C \in \{500, 1000, 2000, 4000, 8000\}$. We also set the value K_i for all 50 markets to be equal to 3; i.e., $K_i = K = 3$. The total revenue of the solutions found by the different algorithms are shown in Figure 1(a). The corresponding running times for each one of the algorithms is shown in Figure 1(b).

As can be seen from Figure 1(a), all the versions of the nested dynamic-programming algorithms (i.e., `DP-DP`, `Fast-DP`(10) and `Fast-DP`(100)) clearly outperform `Sort`. That is, these algorithms report sets of deals that have significantly higher revenue. It is worth observing that `Fast-DP`(10) is extremely competitive, achieving almost-optimal values for all values of C. In addition, while the gap in revenue between `Fast-DP`(100) and the optimal `DP-DP` algorithm is larger than the gap between `Fast-DP`(10) and `DP-DP`, the revenue

achieved by Fast-DP(100) is still competitive, and consistently outmatches Sort.

The results on the revenue of the obtained solutions should be seen together with the actual running times of these algorithms. These are shown in Figure 1(b). The observed running times demonstrate that although DP-DP is feasible to use in real data, it is still significantly slower than the rest of the algorithms. As expected, the Sort algorithm is the fastest, yet its results are the worse in terms of revenue. Noteworthy is the performance of Fast-DP(10) and Fast-DP(100); both algorithms achieve very low running times. This observation, combined with the near-optimal solutions reported by these algorithms, indicates that Fast-DP is an effective algorithmic solution to the Deal Selection problem, providing a balance of efficiency and quality. This observation is consolidated by the experiments that follow.

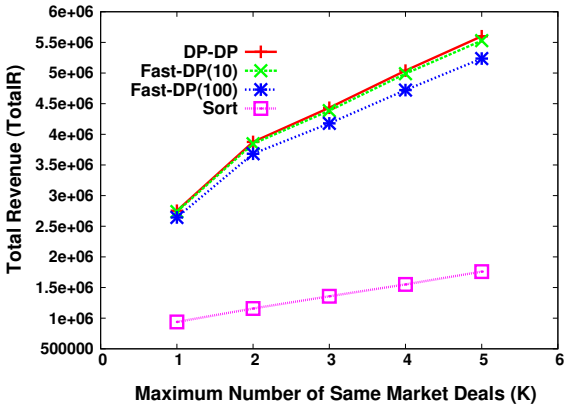
5.2.2 *Exploring the effect of the diversity constraints*

In this experiment, we explore the effect of the diversity constraints on the performance of the different algorithms. For this we consider that the same upper bound on the number of same-market deals is imposed for all 50 markets. That is for every market i we assume that $K_i = K$. Given this, we report the total revenue achieved by the different algorithms (Figure 2(a)) and the corresponding running times (Figure 2(b)) as a function of $K \in \{1, 2, 3, 4, 5\}$. For this experiment, the value of C is set to 5000. The findings of this experiment are similar to the ones we reported in the previous section. As anticipated, larger values of K allow for more flexibility, and thus lead to higher revenues. As can be seen from Figure 2(a), the algorithms that are based on the nested dynamic-programing principle utilize this increased flexibility; all three of them clearly outperform Sort for all values of K. Again, Fast-DP(10) achieves near-optimal revenue values, overcoming Sort and even Fast-DP(100) – which is still extremely competitive.

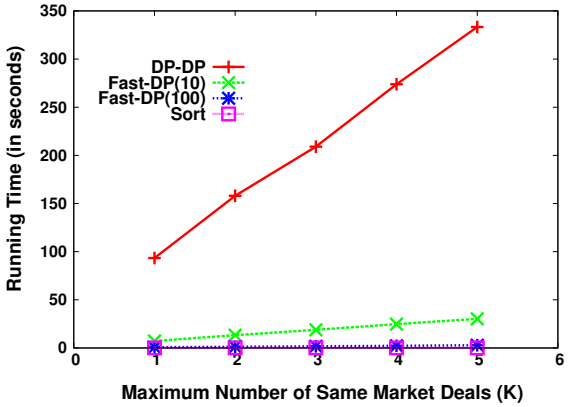
The running times reported in Figure 2(b) verify our previous observation that Fast-DP(10) can combine near-optimal results with low computational times. This makes Fast-DP ideal to use in real settings, where the number of deals is large.

5.2.3 *Scalability experiment*

In this section, we evaluate the scalability of the algorithms for Deal Selection, as a function of the number of candidate deals n. For this experiment, we fix $C = 5000$ and for each market i we use the same value of $K_i = K = 3$. Given the original dataset consisting of $N = 100,000$ deals, we create four more datasets by randomly sampling $N/16$, $N/8$, $N/4$ and $N/2$ deals from the whole collection. Then, we run DP-DP, Fast-DP(10), Fast-DP(100) and Sort on each one of the five datasets. In Figures 3(a) and 3(b) we show the total revenue and the running times of these algorithms as a function of the number of deals in the input. The results shown in Figures 3(a) and 3(b) consolidate the findings of the previous experiments, and are a testament to the scalability and effectiveness of the Fast-DP algorithm. Both variants of this approach follow the optimal DP-DP very closely in terms of the achieved revenue. This fact becomes more impressive as one considers the small running time of these two algorithms, demonstrated in Figure 3(b). In fact, Fast-DP algorithms are able to solve the Deal Selection problem for even the largest of our datasets within few seconds. In



(a) Total revenue (TotalR).



(b) Running time in seconds.

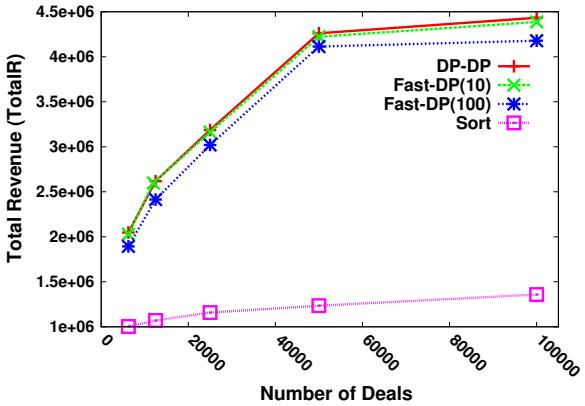
Figure 2: Total Revenue (Figure 2(a)) and Running Time (Figure 2(b)) of DP-DP, Fast-DP(10), Fast-DP(100) and Sort algorithms, as a function of the number of the upper bound K on the number of same-type deals: $K \in \{1, 2, 3, 4, 5\}$; $C = 5000$.

our experiments, the running time of these two algorithms is comparable to the running time of Sort. However, the latter fails to find solutions of satisfactory revenue. Finally, as anticipated, the optimal DP-DP algorithm fails in terms of scalability, as its computational time rises fast with the size of the input.

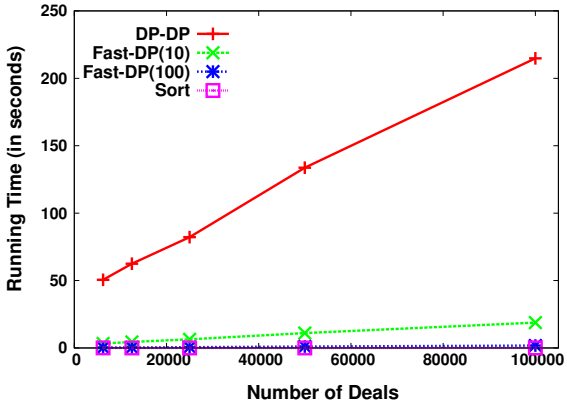
Combining the above scalability results with the findings we reported in the previous experiments, it becomes clear that Fast-DP is by far the most attractive algorithm for the Deal Selection problem, since it consistently offers a combination of near-optimal solutions at a low computational cost.

5.3 Evaluating the algorithms for the Deal Scheduling problem

Here, we study the performance of the GreedySchedule algorithm for the Deal Scheduling problem. As we have already discussed, GreedySchedule is parameterized on the algorithm A, that it invokes for solving the Deal Selection problem for every interval. For solving the Deal Selection problem we use the four algorithms we experimented with in the previous section. That is, $A \in \{$ DP-DP, Fast-DP(10), Fast-DP(100), Sort $\}$. We experiment with the

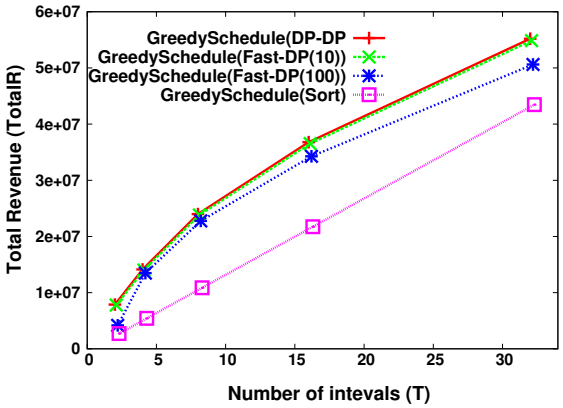


(a) Total revenue (TotalR).

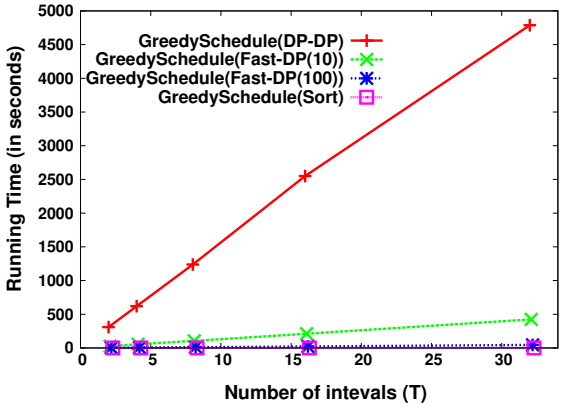


(b) Running time in seconds.

Figure 3: Total Revenue (Figure 3(a)) and Running Time (Figure 3(b)) of DP-DP, Fast-DP(10), Fast-DP(100) and Sort algorithms, as a function of the number of deals n in the input: $n \in \{N/16, N/8, N/4, N/2, N\}$ with $N = 100,000$. For all experiments, $C = 5000$ and $K_i = K = 3$ for all $i \in \{1, \ldots, 50\}$.



(a) Total revenue (TotalR).



(b) Running time in seconds.

Figure 4: Total Revenue (Figure 4(a)) and Running Time (Figure 4(b)) of the GreedySchedule(A) algorithm, where $A \in \{$ DP-DP, Fast-DP(10), Fast-DP(100), Sort$\}$, as a function of the number of intervals T. For every interval T, $C = 5000$ and $K_i = K = 3$ for all $i \in \{1, \ldots, 50\}$.

dataset we described in Section 5.1, which consists of a set of 100,000 candidate deals and 50 markets. We report results for $T \in \{2, 4, 8, 16, 32\}$. Following the setup and findings of the previous experiments, we set the maximum number of same-market deals to be the same across markets (i.e., ($K_i = K = 3$ for all $i \in \{1, \ldots, 50\}$). We also assume that, for each interval, the population's capacity is the same and equal to $C = 5000$. The results for revenue and computational time achieved by GreedySchedule(A) are shown in Figures 4(a) and 4(b), respectively.

Clearly, the performance of GreedySchedule(A) is correlated with the performance of the algorithm that it uses for the Deal Selection problem (algorithm A). Specifically, when either version of Fast-DP is used, GreedySchedule(A) achieves both low computational cost and high-revenue solutions. The revenue achieved by GreedySchedule(Fast-DP(10)) is almost identical to that achieved by GreedySchedule (DP-DP). In fact, even GreedySchedule(Fast-DP(100)) achieves near optimal solutions. On the other hand, while the revenue of GreedySchedule(Sort) increases with the number of intervals, its results are far from optimal.

5.4 Deal-size estimation

In this section, we evaluate our methodology for estimating the size of deals, i.e., the number of coupons they are expected to sell. For this experiment, we use the dataset described in Section 5.1. As discussed in Section 4, we consider a rich set of features for each deal. Given these features, we estimate deal-size by using the SVM-based regression method proposed by Shevade *at al.* [20]. We measure the accuracy of our framework using two standard measures: the *correlation coefficient* (CC) and the *mean absolute error* (MAE). For a given set of deals, CC is the value of the Pearson correlation coefficient between the actual and the predicted sizes for these deals. Similarly, the MAE is the average absolute difference between the actual and the predicted sizes. We use the absolute (rather than the relative) error for the evaluation of the accuracy because our algorithms for the Deal Selection and the Deal Scheduling problems take the actual size of the deals as input. Therefore, our goal is to measure the error that our predictor introduces in the input given to the corresponding algorithms for the two problems.

Market-based grouping of deals: A careful inspection of our data reveals that, although the 50 markets discovered by

LDA correspond to real-life concepts, many of these markets are really sparse, i.e., the have a small number of businesses in them. Further inspection of the term distribution of the original 50 markets reveals that they can be further clustered into larger *marketgroups* that correspond to more general markets. For example, the following 8 of the 50 markets can be classified under a more general food-related marketgroup: *Indian cuisine, pizza, Japanese cuisine, Mediterranean cuisine, Spanish cuisine, restaurants, chocolate & sweets, organic & healthy food*. Another identified marketgroup is "personal care", consisting of the topics *tanning, hair & beauty, skin & face treatment, massage*. Intuitively, we expect deals that belong in the same marketgroup to follow similar trends with respect to their appeal to the population. Given these observations, we use the Kullback-Liebler distance to cluster the initial 50 markets into 12 marketgroups (based on their term distributions). In this way, each original deal is associated with one marketgroup, namely the marketgroup that contains the original market of the deal. Alternatively, one could use a clustering method such as hierarchical LDA [2].

Given the 12 marketgroups, our regression mechanism builds a different regression model for each marketgroup. Note that the grouping of markets into marketgroups serves two purposes: first, by merging the original markets it increases the density of our data. Second, it provides a way of predicting the size of businesses that belong to smaller or nascent markets that are strongly correlated with denser markets.

The CC and MAE obtained for the deals within each of the discovered marketgroups are shown in the 2nd and 3rd columns of Table 1, respectively. One can observe that the CC values are more than 0.5 for most markets. The observed MAE is also a relatively small number when compared to the actual size of the deals, which is in the order of thousands of coupons.

Location-based grouping of deals: While the results obtained using this regression mechanism are promising, they remain far from perfect, especially for specific markets. Motivated by this, we enhance our methodology by applying a location-based partitioning of the deals.

Our experiments indicate that the city where the deal takes place (i.e., the deal's location) is an extremely influential feature in the context of size estimation. This is indeed a reasonable and easily interpretable finding: customers from a particular city share common preferences and requirements (e.g motivated by geography or local trends), which may be different from the preferences of people from other areas.

Taking this finding into consideration, we use the city of every deal to further partition the set of deals into groups that share the same marketgroup *and* the same city. In this case, for every (marketgroup,city) combination we have a different regression model. The results for this are shown are shown in the 2nd and 3rd columns of Table 1, respectively. These are the average values of the CC and the MAE of the models used for every city for all the deals that belong in the same marketgroup. As can be seen from the table, the results are clearly better than those reported without the location-based grouping step. This is true both for the CC (which now exhibits higher values) and the MAE (which exhibits lower values). Observe that, in many of the cases, CC values are near-perfect (very close to 1). In addition, the observed MAE values are consistently low, indicating that the predicted values were close to the actual size of the deals. Again, one should consider the MAE values as indicative of how much error the regression model introduces to the input of the Deal Selection and Deal Scheduling problems. Observe, that the largest error is no more than 185 coupons for deals that actually sell thousands.

6. RELATED WORK

Most of the published work on DDSs focuses on data-analytic studies and economic models that try to explain several phenomena in the domain of daily-deals. However, to the best of our knowledge, our work is the first to exploit data-analytic findings in order to solve a combinatorial problem that arises in this context.

Recent work by Byers *et al.* [4] has extensively analyzed the characteristics of daily-deals and the predictability of the success of a particular deal. Another relevant line or work [5, 19, 6, 17] has focused on the propagation of daily deals in social networks, as well as the effect that deals have on the customer reviews written for different businesses after their corresponding deal materializes. The data-analysis performed by Byers *et al.* provides a comprehensive view of the daily-deals domain and the social and business effects associated with it. In another relevant paper, Ye *et al.* [21] analyze data from daily-deal sites to study the propagation of daily deals within a social network. We consider our own work complementary to these efforts. Our focus is to analyze daily-deal datasets, and use our findings to create selection and scheduling algorithms for revenue maximization.

Models for evaluating the benefits and the drawbacks of Groupon from the point of view of the merchant have been considered by Edelman *et al.* [12]. Along the same lines, Dholokia [11] performed an empirical study on the experience of businesses that used Groupon. Finally, models that allow for merchants to analyze the business model of Groupon and identify the optimal ways to participate in DDSs has been studied by Arabshasi [1]. Although related, these papers are complementary to ours since they focus on analyzing and modeling the experience of the merchant, while our focus is to maximize the benefit of the DDS.

Kauffman and Wang [16] studied group-buying paradigms that can be considered as alternatives to DDSs. The focus of that work was to empirically analyze behavioral patterns customer data from `MobShop.com`. Finally, Grabchak et al. [13] explore a variant of the stochastic knapsack problem for online Ad campaigns. Their relevance to our work is limited to the fact that they consider a model similar to Groupon's mass-deal setting, in which a deal is activated only if a minimum number of users opt for it. Nonetheless, their application setting, problem formulation and algorithmic contribution are radically different from ours.

7. CONCLUSIONS

In this paper, we studied the problem of selecting daily deals in order to maximize the expected revenue of the DDS. We provided two combinatorial formulations for this problem and proposed algorithms for their solution. Both formulations take into account natural constraints, such as the limited consuming ability of the population and the diversification of deals across different markets. Our algorithms combine greedy and dynamic-programming techniques and have provable approximation bounds. We also discussed efficient variants of these algorithms that perform extremely well in practice. In order to utilize our deal-selection frame-

Table 1: Regression results per marketgroup: Correlation Coefficient (CC) and Mean Absolute Error (MAE)

	(Without location-based grouping)		(With location-based grouping)	
Marketgroup	**CC**	**MAE**	**CC**	**MAE**
Food	0.26	385.08	0.67	185.07
Personal Care	0.44	248.14	0.76	65.14
Art & Photography	0.54	105.84	0.84	25.91
Fashion & Apparel	0.6073	159.9062	0.72	50.38
Children/Parenthood	0.504	234.96	0.81	42.76
Fitness	0.57	118.015	0.81	43.65
Cars	0.52	207.50	0.91	37.25
Cinema/Theater	0.4717	173.95	0.7	19.51
Medical	0.5952	51.19	0.95	3.56
Hunting & Fishing	0.6653	175.61	0.96	31.72
Bed and Breakfast	0.52	170.92	0.96	46.91
Bowling	0.65	92.34	0.97	11.13

work, the DDS needs to have an estimate of how popular a deal about a particular business (or service) is going to be. We showed that predicting the popularity of a daily deal is feasible, given a set of different characteristics of the deal. Our experimental results demonstrated the efficacy of our algorithms for deal selection and scheduling, and the accuracy of our predictor for the success of a deal. Our work opens up a lot of directions for further research. For example, in the future, we plan to explore how correlations between businesses or markets, as well as external parameters (e.g., news stories, political and economic events), can help DDSs towards even more effective choices.

8. REFERENCES

[1] A. Arabshahi. Undressing groupon: An analysis of the groupon business model. 2010. Available at: `http://www.ahmadalia.com`.

[2] D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *NIPS*, 2003.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[4] J. W. Byers, M. Mitzenmacher, M. Potamias, and G. Zervas. A month in the life of groupon. *CoRR*, 2011.

[5] J. W. Byers, M. Mitzenmacher, and G. Zervas. Daily deals: Prediction, social diffusion, and reputational ramifications. *CoRR*, 2011.

[6] J. W. Byers, M. Mitzenmacher, and G. Zervas. The groupon effect on yelp ratings: A root cause analysis. *CoRR*, abs/1202.2369, 2012.

[7] J. R. Campbell and H. A. Hopenhayn. Market size matters. Working Paper 9113, National Bureau of Economic Research, August 2002.

[8] C. Chekuri and S. Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J. Comput.*, 35(3):713–728, 2005.

[9] R. Cohen, L. Katzir, and D. Raz. An efficient approximation for the generalized assignment problem. *Inf. Process. Lett.*, 100(4):162–166, 2006.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

[11] U. M. Dholakia. How effective are groupon promotions for businesses. 2010. Available at: `http://www.ruf.rice.edu/~dholakia`.

[12] B. Edelman, S. Jae, and S. Kominers. To groupon or not to groupon: The profitability of deep discounts. 2010. Available at: `http://www.scottkom.com/research.html`.

[13] M. Grabchak, N. Bhamidipati, R. Bhatt, and D. Garg. Adaptive policies for selecting groupon style chunked reward ads in a stochastic knapsack framework. WWW '11.

[14] Groupon, inc. s-1 filing. filed with the u.s. securities and exchange commission, 2011.

[15] A. I. Strategies for diversification. *Management Sci*, 35(5):113–124, 1957.

[16] R. J. Kauffman and B. Wang. New buyers' arrival under dynamic pricing market microstructure: The case of group-buying discounts on the internet. *J. Mange. Inf. Syst*, 18(2):157–188, 2001.

[17] V. Kumar and B. Rajan. Social coupons as a marketing strategy: a multifaceted perspective. *Journal of the Academy of Marketing Science*, 40:120–136, 2012. 10.1007/s11747-011-0283-0.

[18] C. A. Montgomery. Product-market diversification and market power. *Academy of Management Journal*, 28(4):789–798, 1985.

[19] M. Potamias. The warm-start bias of yelp ratings. *CoRR*, abs/1202.5713, 2012.

[20] S. K. Shevade, S. S. Keerthi, C. Bhattacharyya, and K. R. K. Murthy. Improvements to the smo algorithm for svm regression. *IEEE Transactions on Neural Networks*, 11(5):1188–1193, 2000.

[21] M. Ye, C. Wang, C. Aperjis, B. A. Huberman, and T. Sandholm. Collective attention and the dynamics of group deals. *CoRR*, 2011.