Submitted to *Management Science*
manuscript

Dynamic Optimization of Mobile Push Advertising Campaigns

Xinshang Wang, Van-Anh Truong
Department of Industrial Engineering and Operations Research, Columbia University, New York, NY
{xw2230, vt2196}@columbia.edu

Shenghuo Zhu, Qiong Zhang
Alibaba Group, {shenghuo.zhu, qz.zhang}@alibaba-inc.com

We study a novel resource-allocation problem faced by Alibaba Group. In this problem, mobile "push messages" must be sent over the course of a day to hundreds of millions of users. Each message can be sent to any number of users, and yields a reward when it generates a clickthrough, subject to a budget constraint on the total reward over all users for the message. This budget represents the maximum amount that an advertiser is willing to pay for clickthroughs for the message on a given day. Given users' diverse preferences, the problem aims to deliver the "right messages" to the "right users" to maximize ad revenues without overwhelming each user with too many messages.

Due to the large size of the real application, we analyze algorithms for the above problem in an asymptotic regime. We consider a novel scaling of the problem "size," called *big-data scaling.* In this scaling, as the problem size grows, the number of users, as well as their diversity, grow. The scaling captures the fact that individual user information remains highly granular and distinctive even as the size of the user base increases. We prove that solving the problem as a static assignment problem results in a regret of $O(\sqrt{t})$, where t is the parameter scaling the problem. Furthermore, adding a single recourse opportunity, by sending push messages in two cycles over the course of a day and making use of information observed in the first cycle to adapt decisions in the second cycle, can reduce the regret to $O(t^{1/4}\log t)$. Finally, the difference in regret between the static and dynamic strategy can be $\Omega(\sqrt{t})$. Numerical experiments on three real data sets, each containing several hundred million users, show that the latter strategy improves the regret of the former by at least 10%-50%.

1. Introduction

Recent years have seen tremendous growth in the volume of sales taking place in mobile commerce (m-commerce) markets. In these markets, customers visit online stores and purchase products via mobile platforms, such as apps for iOS and Android systems. In 2015, more than US$200 billion in sales took place via a single mobile app developed by Alibaba Group. Alibaba is an e-commerce company that provides a third-party platform for business-to-customer and customer-to-customer markets, among many other services. In China, the m-commerce market share of its mobile app, which has been installed on several hundred million devices, is rapidly displacing traditional e-commerce markets (Emarketer.com 2016).

Given the size of its user base, the mobile app of Alibaba Group, henceforth referred to simply as *the app*, serves as a new channel for advertising and delivering personalized recommendations. Owners of the app can choose to receive recommendations about products that are tailored to their interests. These recommendations are sent via *mobile push notifications*. When a mobile push notification is sent to a user, a short *message* describing the recommended product appears in the notification zone of the user's mobile phone. The user can either swipe the message away or click on the message to link to a new page containing full details about the recommendation. We call the latter action a *clickthrough*.

Push messages are used to achieve two objectives. First, the messages are used to make personalized recommendations about premium products selected from millions of online stores at Alibaba Group. These recommendations prompt mobile users to visit and browse products in the online markets, eventually generating more sales. Second, the messages are designed to promote products for online stores that have applied to join advertising campaigns organized by Alibaba Group. Such campaigns often aim at achieving a certain impact on the user base, such as a desired number of clickthroughs that the campaign messages need to attract. Retailers who want to advertise their products may offer to pay for clickthroughs to their products.

Push messages are sent to hundreds of millions of mobile users every day (BusinessWire 2016). Given such a large user base, it is challenging to manage the delivery of the “right messages” to the “right users” without overwhelming each user with too many messages.

In this paper, we model the problem faced by Alibaba Group of how to manage push messages over a planning period. Without loss of generality, we take a period to be a day. The problem is a novel resource-allocation problem. In this problem, there is a set of known users, each owning a mobile device on which the app is installed. There is also a set of distinct messages. Each message can be sent to any number of users, and yields a reward when it generates a clickthrough, subject to a budget constraint on the total reward that all the users can generate for the message. The budget represents the maximum amount that an advertiser is willing to pay for clickthroughs to his product or website on a given day. Since sending too many messages to the same user would have an adverse impact on the user’s experience, each user must receive no more than one push message per day. Over the course of a day, push messages can be sent sequentially to different users. Once a message has been sent to a user, we can observe after some time whether the user clicks on the message. The observed user actions can be used to update decisions about what messages we want to send to subsequent users, so that we can make the most use of the remaining budgets. We aim to maximize the total reward earned from all messages sent, by adaptively determining the sequence of users to contact, together with their corresponding messages. We do not consider any multi-day effect.

To achieve our goal, it is necessary to learn and calibrate the clickthrough probability that we might expect from any given message-user pair. A typical method is to construct a mapping $p(x, y)$ between every vector of user features x and every vector of product features y, and then fitting this mapping to historical information, as well as to data gathered in experiments. At Alibaba, the construction and fitting of the mapping $p(\cdot, \cdot)$ is performed as a separate data-mining and calibration problem. Accordingly, in this paper, we take the view that the clickthrough probabilities will be provided as inputs to our model. Thus, we will focus on the remaining optimization problem.

The following simple example illustrates our problem.

Example 1. There are two messages, A and B, to be sent out to two users X and Y (see Figure 1). Each message can be sent to any of the two users, but each user should receive exactly one message. For simplicity, we assume that each message has budget of \$1 and yields a reward of \$1 when clicked. In other words, each message will exhaust all of its budget on the first clickthrough. We assume that the clickthrough probabilities are given by Figure 1.

Suppose that we send message B to both users X and Y. Then with probability $0.6 \times (1 - 0.2)$, only X will click on message B, in which case we earn \$1. With probability 0.6×0.2, both users will click on message B, but since message B has a budget of only \$1, we only earn \$1 even though message B receives two clickthroughs.

The optimal strategy to maximize the expected total reward is to first send message B to user Y. Then we observe whether Y clicks on B. If Y clicks on B (with probability 0.2), thereby exhausting the budget of message B, we next send message A to X. If Y does not click on B, we next send message B to user X. The resulting total expected reward is $0.2 \times 0.4 \times \$1 + 0.8 \times 0.6 \times \$1 = \$0.56$. Note that the decision for user X is a dynamic one. Which message X will receive depends on whether Y clicks on message B.

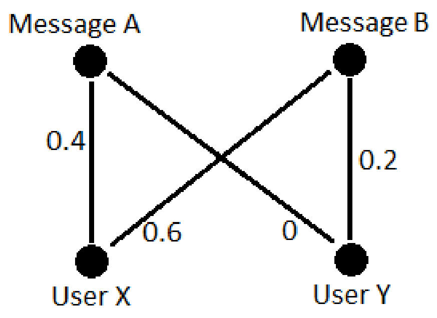


Figure 1

An interesting related question is how to adaptively find the optimal sequence of users to contact, together with their corresponding messages. However, in practice, we cannot implement a solution that makes an adaptive decision for every single user, simply because it would take too long to observe the actions of all users one at a time. Users might not immediately notice new push messages on their mobile devices. In practice, more than half of all user actions (either clickthrough or dismissal) can be observed within 1 hour. Most user actions can be observed within 3 hours. Furthermore, the clickthrough rate of users who respond after 3 hours is negligible. Given such long response time, it is only practically possible to send out messages in a small number of batches on each day. After sending out each batch, we must wait for several hours to observe the feedback of users who were most recently contacted. In this way, we must take advantage of a very small number of recourse opportunities.

1.1. Overview of Algorithms and Contributions

We design an algorithm to determine how to send push messages in a small number of cycles over the course of a day. The decisions for each cycle can be based on the observed actions of all users contacted in previous cycles in the day. In the real application, it is possible to send hundreds of millions of messages in a single cycle.

Due to the large size of the real application, we analyze algorithms in an asymptotic regime. We consider a novel scaling of the problem "size," called *big-data scaling*. In this scaling, as the problem size grows, the number of users, as well as their diversity, as characterized by the number of distinct profiles of user characteristics, both grow. We study the regret of our algorithms, which captures the revenue loss of our algorithms relative to an optimal algorithm, when the scale is large.

We analyze the performance of two algorithms for sending push messages. The first algorithm, which we call the *Static Algorithm*, does not make dynamic decisions based on user feedback, and essentially sends out all messages in one cycle. It represents a simple attempt to tackle this problem without utilizing user feedback. We will show that the Static Algorithm has asymptotic regret $O(\sqrt{t})$. Although this regret is diminishing in the size t of the system, we show that it is possible to further reduce the regret.

To this end, we analyze a second algorithm, which we call the *Reservation Algorithm*. The Reservation Algorithm sends out messages in two cycles. We prove that by making use of only one additional cycle, the Reservation Algorithm is able to reduce the regret to $O(t^{1/4}\log t)$. In other words, the regret of the Reservation Algorithm grows much more slowly than that of the Static Algorithm as we scale up the system size t. Further, we prove that the difference in regret between the Static Algorithm and the Reservation Algorithm is $\Omega(\sqrt{t})$.

Both of algorithms that we analyze make use of a solution to a linear program that matches the expected number of clickthroughs with the budgets of the messages. Linear programming has been used to design algorithms for various resource-allocation problems because optimal solutions to linear programs are often useful guides for making allocation decisions (Wang, Truong and Bank 2015). In our model, an assignment of users to messages given by a static linear program performs the following basic tradeoff. If we send a message to too many users, we might obtain more clickthroughs than are paid for by the budget of that message. Thus, we waste the opportunity to have these users view other messages and bring in rewards from other sources. In other words, we *lose viewing opportunities*. On the other hand, if we send the message to too few viewers, we lose the opportunity of generating more clickthroughs. Thus we *lose potential rewards*.

The Reservation Algorithm performs this tradeoff by using a re-solving heuristic. To be more specific, the algorithm sends out messages in the first cycle based on an optimal solution to a linear program. Then after observing the clickthroughs that each message receives from the first batch of users contacted, the algorithm re-solves a linear program that matches the remaining budgets of the messages to the remaining users.

We test the numerical performance of the above two algorithms by simulating them on production data provided by Alibaba. The data contains three large batches of messages that were sent to several hundred million users in three separate days in March 2016, along with the clickthrough probabilities for all user-message pairs, which were estimated by Alibaba. Our computational results show that by exploiting user feedback, the Reservation Algorithm can reduce the regret of the Static Algorithm by at least 10%-50%. This result suggests that the idea of dynamically optimizing the allocation of push messages is highly promising.

2. Literature Review

Our problem is related to revenue-management problems, as it aims to maximize a total expected reward that can be obtained by dynamically allocating demands (users) to resources (messages). Traditional revenue-management problems focus on situations in which demands arrive randomly and exogenously over time (Talluri and van Ryzin 2004). In contrast, in our model we can control the demands, or the users to be contacted. However, our model limits the number of opportunities at which dynamic decisions can be made.

Reiman and Wang (2008) first prove that for a network revenue-management problem, re-solving a linear program can help to reduce the total regret to $o(\sqrt{t})$. Their algorithm, which only re-solves the linear program once, cannot be applied to our model as they require observing the system state at every point in time. Recall that in our model, we can observe the action of users, or the consequences a decision, only a small number of times due to the long response time needed to acquire user feedback.

Jasin and Kumar (2012) also study a network revenue-management problem for which they propose an algorithm having constant regret that is independent of the system size t. However, their algorithm requires resolving LP, and thus observing the system state, infinitely many times as they scale up the system size t. As a result, their algorithm cannot be easily adapted to our setting.

Agrawal, Wang and Ye (2009) and Jasin (2015) study online resource-allocation problems in the asymptotic regime. However, in their model, the distribution of demand arrivals is unknown and exogenous. Their algorithms either have regret that is at least $O(\sqrt{t})$, or require resolving linear programs too many times and therefore, are impractical for our setting.

Another stream of research that is related to our model is the stream of research on ad allocation. Here, similar to our setting, the goal is to find a high-reward allocation between customers and ads with given budgets. Examples include Jaillet and Lu (2014), Manshadi, Vahideh, Gharan and Saberi (2012), Feldman, Mehta, Mirrokni and Muthukrishnan (2009), and Haeupler, Mirrokni, and Zadimoghaddam (2011). These works focus on finding allocation algorithms with constant bounded performance guarantees relative to an offline algorithm, which knows all future information upfront. In those models, the sequence of customer arrivals is exogenous, and in certain cases the algorithms do not have full distributional information regarding future arrivals. As a result of the differences in the modeling assumptions, they cannot be applied to our model to obtain a regret that is smaller than $O(\sqrt{t})$.

We remark that our problem is different from models of online customer selection (Elmachtoub and Levi 2016, 2015). In these models, although the decision maker has the ability to select which customers to offer resources to, the sequence of customer arrivals is still exogenous.

3. Model Formulation

We now formally state our model. There are m different messages and n users. Each message can be sent to any number of users. We make the following main assumptions:

Clickthrough probabilities. If message $j \in \{1,2,...,m\}$ is sent to user $i \in \{1,2,...,n\}$, the user will click the message with known probability p_{ij}. All the p_{ij}'s are given as inputs. For each user i, we call the vector $p_i = (p_{i1}, p_{i2}, ..., p_{im})$ consisting of clickthrough probabilities for user i for all messages a *user profile*. We also call the set of all user profiles $\{p_i\}_{i=1}^n$ a *user pool*.

User fatigue. To limit user fatigue, each of the n users must not receive more than one message. As mentioned earlier, we only consider a single-day problem, so this assumption is equivalent to requiring that no more than one message must be sent to each user per day.

Reward and budget. Each message $j \in \{1,2,...,m\}$ has budget of $c_j \cdot r_j$. We call c_j the *capacity* of message j. If message j receives k clickthroughs in total, then we earn a reward of $r_j \min(k, c_j)$ from message j. In this way, we view messages as resources, and the capacity of a resource is the number of clickthroughs that will just exhaust the budget.

Special message 1. Among the m messages, message 1 is a personalized recommendation that recommends products selected from all online stores at Alibaba Group. We assume that message 1 has infinite capacity because there is no limit as to how many additional visits the online market can receive. The reward r_1 can be interpreted as the long-term expected marginal reward of a visit.

Other messages $2, 3, ..., m$ correspond to campaigns and commercial ads, and have finite capacity values.

In practice, most users prefer products recommended by message 1 over products in other campaign and advertising messages. The reason is that we personalize this message 1 to every user by selecting his/her favorite products from millions of online stores at Alibaba. It is unlikely that we cannot find a product from such a huge number of stores that a user prefers over products in other messages (except on the infrequent occasions that advertizers offer huge discounts for their products). We formally state these conditions in the following assumption.

Assumption 1.

$$c_1 = \infty, \;\; p_{i1} \geq p_{ij} \;\; \forall i = 1, 2, ..., n, \;\; j = 1, 2, ..., m.$$

According to this assumption, no user will ever receive message j if $r_j < r_1$, because instead of sending message j, it is more beneficial to send message 1 that has a larger clickthrough probability and a larger reward value. Therefore, we assume that $r_j \geq r_1$ for all message $j \in \{2, 3, ..., m\}$. In other words, we assume that the marginal reward earned from campaign and advertising messages is at least r_1.

Number of cycles. The system can sequentially send messages to the n users in several cycles throughout the day. At the beginning of each cycle, the system can observe the clickthroughs in the previous cycle. Then the system sends messages to a subset of users who have not received messages yet. The decision of which messages to be sent to which users in each cycle can be adapted to the clickthroughs observed previously. An algorithm can choose how many cycles to use. However, for the algorithm to be practically useful, the number of cycles used should be small.

Objective. The objective is to maximize the expected total reward, where the expectation is taken over random user responses (i.e., clicking or not). Mathematically, under a policy Π, let I_{ij}^{Π} indicate whether user i clicks message j. The expected total reward V^{Π} of Π is

$$V^{\Pi} = \mathbf{E}[\sum_{j=1}^{m} r_j \min(\sum_{i=1}^{n} I_{ij}^{\Pi}, c_j)].$$

4. Performance Measure

We want to compare the performance of our algorithms against that of an optimal algorithm OPT. As discussed earlier, an algorithm is limited to using a few cycles if it is to be practically useful. However, we relax this requirement for OPT, and allow OPT to take an unlimited number of cycles. Then, OPT will never risk wasting opportunities by sending out too many messages in a single cycle. Therefore, we must have

$$\sum_{i=1}^{n} I_{ij}^{OPT} \le c_j, \ \forall j = 1, 2, ..., m, \ \text{w.p.1}, \tag{1}$$

And the objective value of OPT is

$$V^{OPT} = \sum_{j=1}^{m} r_j \cdot \sum_{i=1}^{n} \mathbf{E}[\min(I_{ij}^{OPT}, c_j)] = \sum_{j=1}^{m} r_j \cdot \sum_{i=1}^{n} \mathbf{E}[I_{ij}^{OPT}]. \tag{2}$$

We define the *regret* of an algorithm Π as the difference between V^{OPT} and V^{Π}.

$$V^{OPT} - V^{\Pi} = \sum_{j=1}^{m} r_j \cdot \sum_{i=1}^{n} \mathbf{E}[I_{ij}^{OPT}] - \mathbf{E}[\sum_{j=1}^{m} r_j \cdot \min(\sum_{i=1}^{n} I_{ij}^{\Pi}, c_j)].$$

5. Linear-Programming Formulation and Upper Bound on OPT

It is difficult to directly analyze the optimal policy OPT due to its complex dynamic properties. We are thus motivated to investigate an upper bound on V^{OPT} that is computationally tractable. In this section, we show that V^{OPT} can be bounded by the optimal objective value of a linear program, which allocates users to messages once in an expected sense. The algorithms we present in subsequent sections will also be based on an optimal solution to the same linear program.

To this end, consider the following pairs of LP's that are dual to each other:

Primal:

$$\begin{aligned} V^{LP} = \max_{s\in\mathbb{R}^{n+m}} & \sum_{i=1}^{n}\sum_{j=1}^{m} r_j s_{ij} p_{ij} \\ \text{s.t.} & \sum_{i=1}^{n} s_{ij}p_{ij} \le c_j,\ \forall j=1,2,...,m \\ & \sum_{j=1}^{m} s_{ij} = 1,\ \forall i = 1,2,...,n \\ & s_{ij}\ge 0,\ \forall i,j. \end{aligned} \tag{3}$$

Dual:

$$\begin{aligned} \min_{\gamma\in\mathbb{R}^m,\ \eta\in\mathbb{R}^n} & \sum_{j=1}^{m} c_j\gamma_j + \sum_{i=1}^{n}\eta_i \\ \text{s.t.}\ & \eta_i \ge (r_j-\gamma_j)p_{ij},\ \forall i=1,2,...,n, j=1,2,...,m \\ & \gamma_j\ge 0,\ \forall j=1,2,...,m. \end{aligned} \tag{4}$$

Both the primal and dual forms of the linear program have physical interpretations. In the primal problem (3), the decision variable s_{ij} represents the fraction of user i to be allocated to message j. For each message j, the linear program matches the expected number of clickthroughs to the capacity c_j. In the dual problem (4), the dual variable γ_j corresponds to the primal constraint $\sum_{i=1}^{n} s_{ij}p_{ij}\le c_j$, and η_i corresponds to $\sum_{j=1}^{m} s_{ij}=1$.

Let s^*, γ^*, η^* be a set of optimal dual variables. According to complementary slackness, if $s^*_{ij} > 0$, i.e., the optimal solution allocates user i to message j with positive probability, we must have

$$j \in \arg\max_{k=1,2,\ldots,m} \{(r_k - \gamma^*_k) p_{ik}\}. \tag{5}$$

If we view γ^*_k as a cost incurred by every clickthrough of message k, then $(r_k - \gamma^*_k)p_{ik}$ is the expected profit of sending message k to user i. In other words, the dual problem implies that every user is allocated a message that maximizes the expected profit in this sense.

The following theorem shows that the expected total reward of OPT can be bounded by the above linear program.

Theorem 1.

$$V^{OPT} \le V^{LP}.$$

The Static Algorithm is a naive implementation of the LP solution. The algorithm sends out all messages in a single cycle based on an optimal solution to the LP (3). The name *static* comes from the fact that it does not adapt to user feedback.

Static Algorithm:

1. Solve the linear program (3). Let s^* be an optimal solution.

2. In a single cycle, send message j to user i with probability s^*_{ij}, for all $i = 1, 2, ..., n$, $j = 1, 2, ..., m$.

We will use analyze the performance of the Static Algorithm in Section 14 and use that as a basis for comparison when analyzing the performance of our Reservation Algorithm, which we will describe in the next section.

6. The Reservation Algorithm

In this section we present the Reservation Algorithm, which refines the Static Algorithm by adapting to user feedback. Specifically, the algorithm sends out messages in two cycles.

In the first cycle, it partially sends out messages based on s^*. This step is similar to the Static Algorithm except that some users are reserved for the second cycle. Then based on user feedback, the algorithm re-solves a linear program for the remainder of capacity and users. In the second cycle, the algorithm sends messages according to an optimal solution to this re-optimized linear program.

Core to the algorithm is the decision of which users should be reserved (i.e., should *not* receive messages in the first cycle). Intuitively, if s^* allocates user i to message j (i.e., $s^*_{ij} = 1$) but we do not send user i any message in the first cycle, we are hoping that in case message j has its budget exhausted in the first cycle, user i can be directed to some other message having positive remaining budget. Thus, for each message j, we want to reserve a certain number of users who are allocated to message j according to s^*, who *also* have relatively high clickthrough probabilities for other messages. These users are good candidates for redirection, in case the budget of message j becomes exhausted after the first cycle.

The actual set of users reserved by the Reservation Algorithm is determined by a parameter Δ, which we call the *reservation level.* Given a problem instance, we can run the algorithm using any value of Δ. The remaining capacity of each message that is unassigned in the first cycle due to reservation will be roughly proportional to Δ. In theory, we will show a way of choosing the parameter Δ to ensure an upper bound on the regret of the algorithm. In practice, we can tune Δ to achieve the best empirical performance.

The Reservation Algorithm works as follows:

1. Solve the linear program (3) to find an optimal solution s^* such that $\sum_{i=1}^{n} \|s^*_i\|_0 \leq n + m^2$, where $\|\cdot\|_0$ stands for the number of non-zero elements in the vector. In other words, the number of users who are simultaneously (fractionally) allocated to multiple

messages should be no more than m^2. This condition is not required to implement this algorithm in practice, but only to simplify the analysis. Our Proposition 1 guarantees that one such s^* always exists and can be easily found by adding some infinitesimally small noise to the coefficients p_{ij}'s of the linear program.

2. For every pair of messages $j,k \in \{1,2,...,m\}$, $j \neq k$, solve the following problem to find a set R^{jk} that contains users who are intended for message j according to s^*, but who will be now reserved for message k in the second cycle.

$$\begin{aligned} \min_{R^{jk}\subseteq\{1,2,\ldots,n\}} & \sum_{i\in R^{jk}} s^*_{ij} p_{ij} \\ \text{s.t. } & p_{ik}p_{lj} \geq p_{lk}p_{ij} \quad \forall i \in R^{jk}, l \neq R^{jk} \\ & \sum_{i\in R^{jk}} s^*_{ij}p_{ij} \geq \min(\Delta, \sum_{i=1}^{n} s^*_{ij}p_{ij}). \end{aligned} \tag{6}$$

The first constraint of this optimization problem states that the users $i \in R^{jk}$ selected should be those having the largest ratios of p_{ik}/p_{ij} among all users. These users are deemed the most promising to be reserved for message k. The second constraint states that either R^{jk} is the set of all users, or the expected number of clickthroughs that message j is to receive from the users in R^{jk}, according to the allocation s^*, is at least Δ. The objective function keeps the expected number of clickthroughs that are reduced due to the reservation as small as possible.

Note that for some user i and message j, it is possible to have $i \in R^{jk}$ for many different k's, meaning that user i is suitable to be redirected to multiple different messages in the second cycle. Furthermore, since the objective function keeps minimal the size of R^{jk}, we must have

$$\sum_{i\in R^{jk}} s^*_{ij}p_{ij} \leq \Delta + 1. \tag{7}$$

3. Send message j to user i in the first cycle with probability

$$x_{ij}^{(1)} \equiv s_{ij}^* \cdot \mathbf{1}(i \notin \bigcup_{k=1,k\neq j}^{m} R^{jk}). \tag{8}$$

Note that the condition $i \in R^{jk}$ only indicates that the algorithm reserves the fraction s_{ij}^* of user i that is allocated to message j; if there is another fraction of user i allocated to some other message, say j', then user i may receive message j' in the first cycle even if $i \in R^{jk}$.

Compared to the Static Algorithm, this step reduces the expected number of clickthroughs that each message j will receive in the first cycle by

$$\sum_{i=1}^{n} \mathbf{1}(i \in \bigcup_{k=1,k\neq j}^{m} R^{jk}) s_{ij}^* p_{ij} \leq \sum_{k=1,k\neq j}^{m} \sum_{i\in R^{jk}} s_{ij}^* p_{ij} \leq (m-1)(\Delta+1), \tag{9}$$

where the last inequality follows from (7).

4. Let D_j be the actual number of clickthroughs that message j receives in the first cycle. Solve the following linear program that allocates the residual demands to capacities.

$$\begin{aligned} \max_{x_{ij}, i=1,2,...,n, j=1,2,...,m} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} r_j x_{ij} p_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} p_{ij} \leq (c_j - D_j)^+, \ \forall j = 1,2,...,m \\ & \sum_{j=1}^{m} x_{ij} = 1 - \sum_{j=1}^{m} x_{ij}^{(1)}, \ \forall i = 1,2,...,n \\ & x_{ij} \geq 0, \ \forall i,j. \end{aligned} \tag{10}$$

5. Let $x^{(2)}$ be an optimal solution to (10). In the second cycle, for any user $i \in \{1,2,...,n\}$ who was not sent message in the first cycle, send message j to the user with probability

$$\frac{x_{ij}^{(2)}}{\sum_{k=1}^{m} x_{ik}^{(2)}}.$$

The following proposition establishes the condition that $\sum_{i=1}^{n} \|s_i^*\|_0 \leq n + m^2$ as required in the Reservation Algorithm.

PROPOSITION 1. *Consider the non-trivial case where $r_1 > 0$ and $\|p_i\|_1 > 0$ for every $i \in \{1,2,...,n\}$. If for any two users $i,l \in \{1,2,...,n\}$, $i \neq l$, and any two messages $j,k \in \{1,2,...,m\}$, $j \neq k$, we have $p_{ik}p_{lj} \neq p_{lk}p_{ij}$, then any optimal solution s^* to the linear program (3) must satisfy $\sum_{i=1}^n \|s_i^*\|_0 \leq n + m^2$.*

The regret of the Reservation Algorithm clearly depends on the value of Δ. In the special case of $\Delta = 0$, the algorithm does not reserve any users and thus reduces to the Static Algorithm. The Reservation Algorithm chooses Δ as

$$\Delta = C \cdot \sum_{j=2}^{m} \sqrt{c_j} \cdot \log \sum_{j=2}^{m} c_j, \tag{11}$$

where $C > 0$ is any small constant that we can further tune to improve the performance of the algorithm. Note that messages 2 to m have finite capacity values (see Section 3), so Δ is finite.

7. Overview of Analysis of the Reservation Algorithm

In this section, we outline a way to bound the regret of the Reservation Algorithm. We prove this result by analyzing the gap between the expected total reward of this algorithm and the upper bound on the optimal total reward given by Theorem 1.

Recall that D_j is the random number of clickthroughs that message j receives in the first cycle, which is a Poisson binomial random variable with mean $\mathbf{E}[D_j] = \sum_{i=1}^n x_{ij}^{(1)} p_{ij}$. Define δ_j to be the noise in D_j

$$\delta_j \equiv D_j - \mathbf{E}[D_j] = D_j - \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij}. \tag{12}$$

Our analysis is based on the idea of viewing δ_j as a small *perturbation to the capacity* c_j. After the first cycle, the remaining capacity of message j is $(c_j - D_j)^+$, which can be written as

$$(c_j - D_j)^+ = [(c_j - \delta_j)^+ - \mathbf{E}[D_j]]^+.$$

Intuitively, we view $(c_j-\delta_j)^+$ as the perturbed capacity of message j. Then the noise δ_j arises from this random capacity $(c_j-\delta_j)^+$.

The following is the *perturbed linear program* in which the capacity of message j is perturbed by δ.

$$\begin{aligned} V^{LP}(\delta) = \max_{s_{ij}, i=1,2,...,n, j=1,2,...,m} & \sum_{i=1}^{n}\sum_{j=1}^{m} r_j s_{ij} p_{ij} \\ \text{s.t.} & \sum_{i=1}^{n} s_{ij}p_{ij} \le (c_j-\delta_j)^+, \ \forall j=1,2,...,m \\ & \sum_{j=1}^{m} s_{ij} = 1, \ \forall i = 1,2,...,n \\ & s_{ij} \ge 0, \ \forall i,j. \end{aligned} \tag{13}$$

Note that $V^{LP}(\delta)$ is a random variable as it is a function of the random noise δ. The following lemma, which will later be used in the analysis, relates our Reservation Algorithm to the perturbed linear program.

Lemma 1. *If there exists an optimal solution $s(\delta)$ to the perturbed linear program (13) such that $s(\delta) \ge x^{(1)}$ component wise, then $x^{(1)} + x^{(2)}$ must be an optimal solution to (13).*

Let V^{RA} be the expected total reward of the Reservation Algorithm. We bound the gap between V^{RA} and the upper bound V^{LP} on the optimal reward by using $V^{LP}(\delta)$ as an intermediate value between the two, and by writing the gap as the sum of two smaller gaps:

$$V^{LP} - V^{RA} = (V^{LP} - \mathbf{E}[V^{LP}(\delta)]) + (\mathbf{E}[V^{LP}(\delta)] - V^{RA}).$$

The first gap, namely $V^{LP} - \mathbf{E}[V^{LP}(\delta)]$, depends on the way in which $V^{LP}(\delta)$ behaves as a function of δ. It is easy to check that $V^{LP}(\delta)$ is concave for all $\delta \le c$, and the first derivatives of $V^{LP}(\delta)$ with respect to δ are equal to the negative of dual variables associated with capacity constraints of (13) (Bertsimas and Tsitsiklis 1997). Therefore, if the size of δ is much smaller than c, we have from Jensen's inequality that

$$\mathbf{E}[V^{LP}(\delta)] \le V^{LP}(\mathbf{E}[\delta]) = V^{LP}(0) = V^{LP}.$$

The gap $V^{LP} - \mathbf{E}[V^{LP}(\delta)]$ is small if the first derivatives of $V^{LP}(\delta)$, or the dual variables of the perturbed linear program, change smoothly in δ. We will prove in Theorem 2 of Section 8 that if $\gamma(c)$ is a vector of optimal dual variables to the linear program (3) when the capacity vector is c and if e_k is the unit basis vector with the k-th element being 1, then provided that there exists a constant number $\bar{\gamma} > 0$ such that

$$|\gamma_j(c) - \gamma_j(c + \alpha e_k)| \le \bar{\gamma} \tag{14}$$

for all $j, k \in \{1, 2, ..., m\}$, all $\alpha \in [0, 1]$ and for all $c \in \mathbb{R}_+^m$ with $c_1 = \infty$, we have

$$V^{LP} - \mathbf{E}[V^{LP}(\delta)] \le m^2(n + \sqrt{n})\bar{\gamma}.$$

The second gap, namely $\mathbf{E}[V^{LP}(\delta)] - V^{RA}$, is largely determined by the difference between the assignments made by the Reservation Algorithm and the perturbed linear program. We will show that when $\|\delta\|_1$ is not too large compared to Δ, $x^{(1)} + x^{(2)}$ will be an optimal solution to the perturbed linear program. That is, the assignment made by the Reservation Algorithm will be very close to the assignment for $V^{LP}(\delta)$, and hence the total expected reward of the Reservation Algorithm will be close to $V^{LP}(\delta)$. We prove a condition by which $x^{(1)} + x^{(2)}$ is optimal for the perturbed linear program, by viewing our model as a *generalized network flow* problem and by utilizing flow properties in the generalized network. We will prove in Theorem 6 of Section 9 that

$$\mathbf{E}[V^{LP}(\delta)] - V^{RA} \le \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta} + 3m^2 + nP(\bar{\mathcal{O}}) + \sqrt{nP(\bar{\mathcal{O}})} \right],$$

where $\mathcal{O}$ denote the event that $\sum_{j=2}^{m}|\delta_j|\frac{r_j}{r_1}\leq\Delta$.

Combining the bounds on the two gaps, we will obtain a bound on the regret of the Reservation algorithm as

$$V^{LP}-V^{RA}\leq m^2(n+\sqrt{n})\bar{\gamma}+\sum_{j=1}^{m}r_j\left[\sqrt{m\Delta}+3m^2+nP(\bar{\mathcal{O}})+\sqrt{nP(\bar{\mathcal{O}})}\right]. \tag{15}$$

Finally, in Section 10, we will show that this bound grows relatively slowly under an appropriate asymptotic scaling of the problem. Part of this analysis involves showing that, under reasonable assumptions, condition (14) holds when we reduce $\bar{\gamma}$ while scaling up the size of the problem.

8. Bound on the First Gap

In this section, we give a bound on the first of the two gaps described in Section 7, namely, $V^{LP}-\mathbf{E}[V^{LP}(\delta)]$.

THEOREM 2. *Let $\gamma(c)$ be a vector of optimal dual variables to the linear program (3) when the capacity vector is c. Let e_k be a unit basis vector with the k-th element being* 1. *If there exists a constant number $\bar{\gamma}>0$ such that*

$$|\gamma_j(c)-\gamma_j(c+\alpha e_k)|\leq\bar{\gamma} \tag{16}$$

for all $j,k\in\{1,2,...,m\}$, all $\alpha\in[0,1]$ and for all $c\in\mathbb{R}_+^m$ with $c_1=\infty$, then we have

$$V^{LP}-\mathbf{E}[V^{LP}(\delta)]\leq m^2(n+\sqrt{n})\bar{\gamma}.$$

The intuition for the above result is that if the dual vector γ changes smoothly as a function of the capacity c, then the difference in value between the two LP's can be bounded.

9. Bound on the Second Gap via Generalized Network Flows

In this section, we cast our static linear-programming formulation as a generalized network flow problem (Wayne 1999). The generalized networks corresponding to the original and the perturbed linear programs are the same, except that some of the edges have different capacity values. Based on network-flow properties, we analyze how an optimal flow, or equivalently, an optimal assignment given by the linear program (3), changes when the capacity values are perturbed by δ. This analysis enables us to bound the second of the two gaps described in Section 7, namely the difference $\mathbf{E}[V^{LP}(\delta)] - V^{RA}$.

As a main result of this section, we show that the resulting changes made to an optimal assignment can be bounded by the size of δ. This result will lead to a condition by which $x^{(1)} + x^{(2)}$, which is the solution used by RA (with objective value V^{RA}), is *optimal* for the perturbed linear program (with objective value $\mathbf{E}[V^{LP}(\delta)]$).

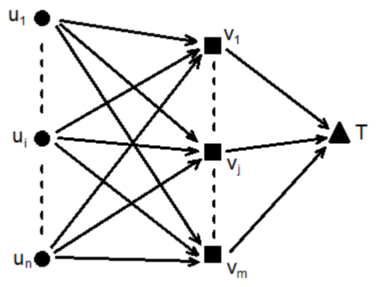


Figure 2 **In a generalized flow network, every user i corresponds to a user node u_i, and every message j corresponds to a message node v_j. T is the sink.**

The network-flow problem we describe next specializes the generalized network-flow problem (Wayne 1999) to a specific graph. There will be a few minor changes that we will point out presently.

9.1. Construction of a generalized flow network

We now construct our generalized network-flow problem. In our flow network, there is a set U of user nodes and a set V of message nodes. Each user $i \in \{1, 2, ..., n\}$ corresponds to a user node $u_i \in U$, and each message $j \in \{1, 2, ..., m\}$ corresponds to a message node $v_j \in V$. There is also a sink T. Every user node u_i has an *initial excess* $e(u_i) = 1$. The initial excesses at all other nodes are 0. The unit initial excess at a user node u_i means that there is a single user i to be assigned to a message.

Figure 2 illustrates the set E of directed edges in the generalized network. From every user node $u_i \in U$ to every message node $v_j \in V$, there is an edge $(u_i, v_j) \in E$ having capacity $c(u_i, v_j) = \infty$. From every message node v_i to the sink, there is an edge $(v_i, T) \in E$ having capacity $c(v_j, T) = c_j$.

A *generalized pseudo-flow* is a function $f : E \rightarrow \mathbb{R}_+$ that satisfies the capacity constraints $0 \leq f(u, v) \leq c(u, v)$, $\forall (u, v) \in E$. Note that according to Assumption 1, starting from any user node $u_i \in U$, there is always a path $u_i \rightarrow v_1 \rightarrow T$ having infinite capacity. Thus, it is always possible to push all initial excesses at all user nodes to the sink.

In a generalized network-flow problem, flows might not be conserved along edges. Every edge $(u, v) \in E$ is associated with a *gain factor* $\mu(u, v)$. When a flow is sent along an edge $(u, v) \in E$ by a generalized pseudo-flow f, the size of the flow leaving from u is $f(u, v)$, while the size of the flow arriving at v is $\mu(u, v) f(u, v)$. In our problem, every edge (u_i, v_j) from a user node u_i to a message node v_j has gain factor $\mu(u_i, v_j) = p_{ij}$. In this way, the size of a flow $f(u_i, v_j)$ leaving u_i represents the allocation s_{ij} in the linear program (3), while the size of flow $\mu(u_i, v_j) f(u_i, v_j) = p_{ij} f(u_i, v_j)$ arriving at v_j represents the expected number of clickthroughs that message j receives from user i. Every edge (v_j, T) from a message node v_j to the sink has gain factor $\mu(v_j, T) = r_j$. In this way, the size of a flow $f(v_j, T)$ leaving v_j represents the total expected number of clickthroughs assigned to message j, while the size of the flow $\mu(v_j, T) f(v_j, T) = r_j f(v_j, T)$ arriving at T represents the expected reward earned from message j.

The *excess* of a node u (as opposed to the initial excess) is the difference between the initial excess at u and the net outflow at u:

$$e(u) - \sum_{(u,v)\in E} f(u,v) + \sum_{(v,u)\in E} \mu(v,u)f(v,u).$$

In this paper, we call a generalized pseudo-flow f a *feasible flow* if f has zero excess at all nodes other than T. Consequently, every feasible flow corresponds to a feasible solution to the linear program (3) in the following way

$$f(u_i,v_j) = s_{ij},\ \forall i = 1,2,...,n, j = 1,2,...,m,$$

$$f(v_j,T) = \sum_{i=1}^{n} s_{ij}p_{ij},\ \forall j = 1,2,...,m.$$

The objective of our generalized network-flow problem is to find a feasible flow that maximizes the excess at the sink, which is by definition

$$\begin{aligned} & e(T) - \textstyle\sum_{(T,v)\in E} f(T,v) + \sum_{(v,T)\in E} \mu(v,T)f(v,T) \\ = & \quad 0 - 0 + \textstyle\sum_{(v,T)\in E} \mu(v,T)f(v,T) \\ = & \quad \textstyle\sum_{(v,T)\in E} \mu(v,T)f(v,T) \\ = & \quad \textstyle\sum_{j=1}^{m} r_j f(v_j,T) \\ = & \quad \textstyle\sum_{j=1}^{m} r_j \sum_{i=1}^{n} s_{ij}p_{ij}. \end{aligned} \tag{17}$$

Thus, every optimal solution s^* to (3) must correspond to an optimal flow f^*.

For any generalized pseudo-flow f, the *residual network* with respect to f is the network $G_f = (U,V,T,E_f,c_f,\mu,e)$, where E_f is the set of *directed* edges in the residual network defined as

$$E_f = \{(u,v)\in E : f(u,v) < c(u,v)\} \cup \{(v,u) : (u,v)\in E, f(u,v) > 0\},$$

and c_f is the residual capacity defined for every edge in E_f in the following way. If $(u,v) \in E$, then $c_f(u,v) = c(u,v) - f(u,v)$; otherwise, $c_f(u,v) = \mu(v,u)f(v,u)$.

For every edge $(u,v) \in E_f$ in the residual network, we further define its *length* as $w(u,v) = -\log \mu(u,v)$ if $(u,v) \in E$, and $w(u,v) = \log \mu(v,u)$ if $(u,v) \notin E$. By this definition, if a flow is pushed in the residual network along a path $d_1 \to d_2 \to \cdots \to d_k$ such that the excesses at all nodes are unchanged except at d_1 and d_k, then the size of the flow arriving at d_k is

$$e^{-[w(d_1,d_2)+w(d_2,d_3)+\cdots+w(d_{k-1},d_k)]} \tag{18}$$

times the size of the flow leaving node d_1 (for a more detailed discussion, see Wayne (1999)). Therefore, the flow arriving at the last node of the path is larger (smaller) than the flow leaving from the first node if the total length of the path $w(d_1,d_2) + w(d_2,d_3) + \cdots + w(d_{k-1},d_k)$ is negative (positive). In particular, if the path is a cycle, i.e., d_1 is the same node as d_k, then the flow returning to d_1 is greater than the flow leaving d_1 if and only if the cycle is a negative cycle, i.e., the total length of the cycle is negative. In other words, if we push a flow along a negative cycle, we can increase the excess at one node in the cycle without changing the excesses at all other nodes.

Previous works have studied various types of *augmentations* for the generalized network flow problem. Given a generalized pseudo-flow, an augmentation modifies flow values on a certain subset of edges in the residual network, such that the resulting flow remains a generalized pseudo-flow. Most often, an augmentation keeps the excesses of most nodes unchanged. In this paper we focus on two types of augmentations: augmentation along paths and along cycles. Here a cycle can be as well seen as a path that starts and ends at the same node. Both types of augmentations aim at increasing the excess at the sink, thereby increasing the objective value of the solution corresponding to the resulting flow as we showed in (17). When an augmentation is performed along a path in the residual network from node v to the sink T, a flow is pushed along the path such that the excess at v is reduced, the excess at T is increased, and the excess at every other node is unchanged. If v is itself the sink, and the augmentation is performed along a cycle that covers $v = T$, then the excess at $v = T$ increases if and only if the cycle is a negative cycle (see (18)).

9.2. Properties of optimal flows in our generalized flow network

Based on special properties of our model, the following theorem strengthens a known result (for example, see Wayne (1999)) that when a generalized pseudo-flow has no negative cycle in the residual network and has zero excess at all nodes other than T, then the maximum flow is attained.

THEOREM 3. *In our model, a feasible flow f is optimal if and only if there is no negative cycle in the residual network G_f.*

Proof. It has been proved that when a generalized pseudo-flow has zero excess at all nodes other than T and has no negative cycle in the residual network, the flow is optimal (for example, see Wayne (1999)). Thus, a feasible flow f with no negative cycles in G_f must be optimal.

To prove the other direction, suppose f is an optimal flow with a negative cycle in the residual network. Then we can push a flow along the cycle to create a positive excess at some user node u while keeping the excess at all other nodes unchanged. We then direct this excess created at node u through the path $u \to v_1 \to T$ that has infinite capacity. In this way, we obtain a feasible flow with a larger size of excess at the sink, which proves that the original flow f is not optimal. $\square$

Now we consider the problem of how to modify an optimal flow when the capacity of some message $k \in \{2, ..., m\}$ is perturbed. Note that perturbing the capacity of message

1 has no impact as it has infinite capacity. We give algorithms that find an optimal flow for the perturbed network by modifying a given optimal solution to the original network. These algorithms are designed mainly for the purpose of providing structural results for our model. The algorithms are not meant to be implemented. Rather, we use them to argue certain facts about the way in which an optimal solution changes when the capacity of some message $k \in \{2, ..., m\}$ changes.

Let $G = (U, V, T, E, c, \mu, e)$ and $\bar{G} = (U, V, T, E, \bar{c}, \mu, e)$ be two generalized networks of our model that differ by the capacity on a single edge (v_k, T) for some given $k \in \{2, ..., m\}$. Given an optimal flow f for the network G, the following algorithms find an optimal flow for $\bar{G}$ by successively performing augmentations along shortest paths or negative cycles in the residual network. We give a different algorithm depending on whether $\bar{c}(v_k, T)$ is greater than or less than $c(v_k, T)$.

Case 1. Algorithm for the case $0 \le \bar{c}(v_k, T) < c(v_k, T)$:

1. Construct a generalized pseudo-flow $\bar{f}$ for $\bar{G}$ as follows. Let

$$\bar{f}(v_k, T) = \min(\bar{c}(v_k, T), f(v_k, T)) \tag{19}$$

for the edge (v_k, T), and let $\bar{f}(u, v) = f(u, v)$ for all other edges $(u, v) \in E$. Since f is optimal for G, there is no negative cycle in the residual network G_f according to Theorem 3. It is easy to see that there is also no negative cycle in the residual network $\bar{G}_{\bar{f}}$ because according to (19), the edge set of G_f must contain the edge set of $\bar{G}_{\bar{f}}$. Thus, if $\bar{f}$ is feasible, then $\bar{f}$ is already an optimal flow for $\bar{G}$.

Now suppose that $\bar{f}$ is not feasible. It must be that the excess at node v_k is positive.

2. Find a shortest path from v_k to T in the residual network $\bar{G}_{\bar{f}}$.

3. Perform an augmentation along the shortest path such that either the excess at node v_k reaches 0 or the residual capacity of one of the edges in the path is reduced to 0. Update the residual network $\bar{G}_{\bar{f}}$.

4. Stop if $\bar{f}$ becomes feasible. Otherwise, go to step 2.

It is easy to check that there will be no negative cycle generated in the residual network throughout the algorithm, because otherwise, the previous shortest path would have passed though the negative cycle to further reduce the total length of the path. Furthermore, the algorithm must terminate in a finite number of iterations because the shortest length from v_k to T in the residual network must be (weakly) increasing after each iteration, and thus at most one augmentation can be performed along every possible path from v_k to T. Therefore, the algorithm outputs an optimal flow $\bar{f}$ for $\bar{G}$.

Case 2. Algorithm for the case $\bar{c}(v_k,T) > c(v_k,T)$.

1. Start with an initial flow $\bar{f} = f$. Suppose $\bar{f}$ is not optimal for $\bar{G}$. Then since $\bar{f}$ is feasible for $\bar{G}$, there must exist some negative cycle in the residual network $\bar{G}_{\bar{f}}$. Moreover, all negative cycles in $\bar{G}_{\bar{f}}$ must include the edge (v_k,T).

2. Find a simple cycle, i.e., a cycle that contains only distinct nodes, having the smallest (negative) total length. This can be solved by finding a shortest path from T to v_k.

3. Perform an augmentation along this shortest negative cycle to increase the excess at T such that the residual capacity of one of the edges in the cycle is reduced to 0. Update the residual network $\bar{G}_{\bar{f}}$. After the augmentation, all negative cycles in $\bar{G}_{\bar{f}}$, if there is any, must still pass through (v_k,T) because the augmentation is performed along the shortest cycle.

4. Stop if there is no more negative cycles in $\bar{G}_{\bar{f}}$. Otherwise, go to step 2.

The algorithm must terminate in a finite number of iterations because the length of the shortest negative cycle must be (weakly) increasing after each iteration, and thus at most one augmentation can be performed along each possible cycle in the residual network. Therefore, this algorithm outputs an optimal flow $\bar{f}$ for the network $\bar{G}$, as $\bar{f}$ is always feasible throughout the algorithm and, by the end of the algorithm, there is no negative cycle in the residual network.

The main structural property of our model that we want to prove is given by the following theorem.

THEOREM 4. *For any two message nodes $v_j, v_k \in V$ and $v_k \neq v_1$, when $c(v_k, T)$ changes to $\bar{c}(v_k, T)$ (with the capacity values of all other edges unchanged), the total size of augmenting flows leaving v_j in the above algorithm is at most $|\bar{c}(v_k, T) - c(v_k, T)| r_k / r_1$.*

This result is driven by the fact that an optimal flow can be strictly improved by adding an augmenting flow through v_1, if the condition above does not hold, thus leading to a contradiction.

9.3. Bounding the second gap

Now we consider how the optimal flow for the perturbed system relates to the assignment made by the Reservation Algorithm.

When the capacity of each message k is perturbed by δ_k, we apply Theorem 4 to each message $k = 2, 3, ..., m$ by setting $\bar{c}(v_k, T) = (c_k - \delta_k)^+$ (message 1 has infinite capacity and thus perturbing it has no impact). Then the total size of augmenting flows leaving every message node v_j due to the perturbation δ, summed over all m messages, is at most

$$\sum_{k=2}^{m} |\delta_k| r_k / r_1. \tag{20}$$

The following theorem shows that if (20) is smaller than Δ, then the assignment $x^{(1)} + x^{(2)}$ of the Reservation Algorithm is optimal for the perturbed system. The proof relates the Reservation Algorithm to the network-flow algorithms above, by showing that the shortest augmenting paths or cycles will pass only through user nodes u_i that have the smallest ratios of p_{ij}/p_{ik} for some messages j, k. Such users are just the ones reserved by the Reservation Algorithm. To be more specific, the length of the augmenting path or cycle that goes through nodes $v_j \to u_i \to v_k$ is

$$w(v_j, u_i) + w(u_i, v_k) = \log \frac{p_{ij}}{p_{ik}}.$$

Since the augmenting paths and cycles seek the shortest length, they will only pass through user nodes i that have the smallest values of p_{ij}/p_{ik}. We will show that such users i are reserved in the set R^{jk}. Thus augmenting flows will not affect the assignment $x^{(1)}$, which leads to the conclusion that $x^{(1)}$ is compatible with the optimal assignment for the perturbed system.

THEOREM 5. *If $\sum_{j=2}^{m} |\delta_j| \frac{r_j}{r_1} \le \Delta$, then $x^{(1)} + x^{(2)}$ is an optimal solution to the perturbed linear program* (13)*.*

Next, the gap between the total expected reward V^{RA} of the Reservation Algorithm and the optimal objective value of the perturbed linear program is given by the following theorem.

THEOREM 6. *Let $\mathcal{O}$ denote the event that $\sum_{j=2}^{m} |\delta_j| \frac{r_j}{r_1} \le \Delta$. We have*

$$\mathbf{E}[V^{LP}(\delta)] - V^{RA} \le \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta} + 3m^2 + nP(\bar{\mathcal{O}}) + \sqrt{nP(\bar{\mathcal{O}})} \right],$$

where the expectation is taken with respect to δ, and $\bar{\mathcal{O}}$ is the complement of $\mathcal{O}$.

10. Performance Analysis in an Asymptotic Regime

Due to the huge number of users and large capacity values involved in the real problem, we are interested in studying the regret in the asymptotic regime. That is, we will quantify the rate at which the regret grows as we scale up the size of the system.

11. Big-Data scaling

Define a series of problems with increasing sizes as follows. In the problem of size t, for $t = 1, 2, ..., \infty$, the number of users is $n = t \cdot \bar{n}$, and the capacity of message j is $c_j = t \cdot \bar{c}_j, \forall j = 1, 2, ..., m$, for some fixed $\bar{c}$ and $\bar{n}$. The number m of distinct messages and the reward r_j of each message j are fixed for all t. We require that Assumption 1 always holds.

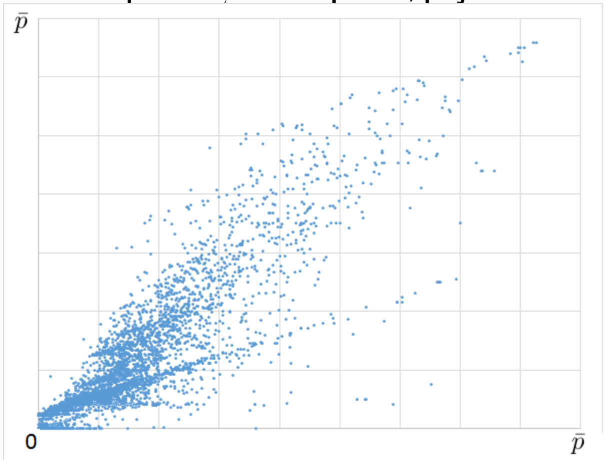
A naive way of scaling n users is to replicate the same set of users t times. However, as we increase the size of the user pool, we also increase its diversity. In reality, it is extremely rare that two active users have identical profiles because the profiles are learned based on users' personal history. A user's history includes, for example, the types of products viewed, purchased and put in the shopping cart over the past 3, 7, 30 days, the operating system of his or her mobile phone, and the type of products sent to the user via push notification over the past several days. Given such high granularity of user histories, it is unlikely that any two active users would have identical profiles. Thus, we can think of the profiles of all $n = t\bar{n}$ users as samples drawn from some continuous distribution. As we scale up t, we are drawing more sample points from this continuous distribution. We called this type of scaling *big-data scaling.*

Hitherto, we have viewed the user pools as fixed. From this point onwards, we will begin to view user pools as random. Formally, let $p_i^t = (p_{i1}^t, p_{i2}^t, ..., p_{im}^t)$ denote the (random) profile of user i in the problem of size t. According to Assumption 1, p_i^t takes values in the region

$$\mathcal{P} = \{y \in [0, \bar{p}]^m : y_1 \geq y_j, \forall j = 2, 3, ..., m\} \tag{21}$$

for some constant $\bar{p}$. By default, we take $\bar{p} = 1$. All of our proofs extend to the case that $0 < \bar{p} < 1$.

Figure 3 **A random sample of $10,000$ user profiles, projected onto two coordinates.**



Assumption 2 **(Big-Data Scaling)**. *Every p_i^t is independently drawn from a continuous distribution with density $f(\cdot)$ having domain $\mathcal{P}$. Furthermore, the density $f(\cdot)$ has upper and lower bounds*

$$0 < \underline{f} \le f(y) \le \bar{f},\ \forall y \in \mathcal{P}.$$

This assumption can be justified by empirical evidence. We verified that in the real dataset, less than 2% of *active users* have profiles that collide with those of other users. In Figure 3, we illustrate the density $f(\cdot)$ by plotting the estimated click-through probabilities for more than $10,000$ users randomly drawn from the real dataset.

Based on big-data scaling, we will analyze the rate at which the regret $V_t^{OPT} - V_t^{\Pi}$ grows as a function of the scaling parameter t. Here, V_t^{Π} and V_t^{OPT} denote the expected total reward of the t-th problem under algorithms Π and OPT, respectively, conditional upon the random user pools for problem t.

Note that as functions of the random user pools for the respective problems, $\{V_t^{\Pi}\}$ and $\{V_t^{OPT}\}$ are sequences of random variables. Thus, our bound on the regret will be a probabilistic bound. More precisely, in Theorem 8 of Section 13, we will show that with probability 1, the regret $V_t^{OPT} - V_t^{RA}$ of the Reservation Algorithm is $O(t^{1/4}\log t)$.

12. Implication on Smoothness of Big-Data Scaling

In this section, we show how big-data scaling leads to a smoothness property for the optimal dual variables to (4). We will use this property to establish condition (14). This condition will help us to prove our asymptotic bound on the regret of the Reservation Algorithm.

Let γ be a vector of dual variables for problem (4). This vector γ must satisfy several conditions for it to be dual-optimal, as stated in the following lemma.

LEMMA 2. *If γ^* is an optimal dual solution to (4), we must have*

$$\gamma^* \in \mathcal{G} \equiv \{\gamma \in \mathbb{R}^m : 0 \le \gamma_j \le r_j - r_1,\ \forall j = 1,2,...,m\}.$$

Proof. First, the assumption $c_1 = \infty$ implies that $\gamma_1 = 0$.

Second, the assumption $p_{ij} \le p_{i1}, \forall j = 2,...,m$, implies that we always have $\gamma_j \le r_j - r_1$. This is because, according to (5), as long as γ_j is greater than $r_j - r_1$, no user will ever be assigned to message j because it is always better to assign users to message 1:

$$\gamma_j \ge r_j - r_1 \Longrightarrow r_j - \gamma_j \le r_1 \Longrightarrow p_{ij}(r_j - \gamma_j) \le p_{i1} r_1,\ \ \forall i = 1,2,...,n.$$

In sum, the set of valid vectors of optimal dual variables γ is included in the region $\mathcal{G}$.

$\square$

Now for each problem of size t, let $c^t(\gamma)$ be a vector of capacity values for which γ is a vector of optimal dual variables to the problem (4) (with c being replaced by $c^t(\gamma)$). It is easy to check that for any $\gamma \in \mathcal{G}$, one such vector $c^t(\gamma)$ can always be found by allocating users to messages $j = 2,3,...,m$ according to the rule (5), i.e.,

$$\begin{aligned} c_j^t(\gamma) &= \sum_{i=1}^{n} p_{ij}^t \mathbf{1}[j = \arg\max_k (r_k - \gamma_k) p_{ik}^t] \\ &= \sum_{i=1}^{n} p_{ij}^t \mathbf{1}[(r_j - \gamma_j) p_{ij}^t - (r_k - \gamma_k) p_{ik}^t \ge 0\ \forall\ k = 1, \dots, m],\ \forall j = 2,3,...,m. \end{aligned}$$

Note that we have ignored the events that $(r_j - \gamma_j)p_{ij}^t - (r_k - \gamma_k)p_{ik}^t = 0$, i.e., a user can be assigned to different messages without affecting dual feasibility, as these events occur with probability 0 for a given γ.

Furthermore, we should set $c_1^t(\gamma) = \infty$ according to Assumption 1. Constructed in this way, $c^t(\gamma)$ is a random vector, as it is a function of $p_1^t, \dots, p_n^t$.

As a main result in this section, we will prove that with probability 1, for all sufficiently large t's, the total expected number of clickthroughs, conditional on the random pool of users, moved from one message to another is at least 1 when γ changes to β or vice versa, whenever $\gamma, \beta \in \mathcal{G}$ with $\|\gamma - \beta\|_\infty \geq t^{-3/4}$.

Theorem 7. *There exists a finite positive random integer t_0 such that, with probability* 1*, for all $t > t_0$ and for any vectors of dual variables $\gamma, \beta \in \mathcal{G}$ with $\|\gamma - \beta\|_\infty \geq t^{-3/4}$,*

$$\sum_{j=2}^{m} |c_j^t(\gamma) - c_j^t(\beta)| > 1.$$

The proof of Theorem 7 relies on a geometric analysis of the space $\mathcal{P}$ of user profiles. Let us define in $\mathcal{P}$ the sub-polytope

$$A_j(\gamma) = \{p \in \mathcal{P} \mid (r_j - \gamma_j)p_j - (r_k - \gamma_k)p_k \geq 0 \ \forall k = 1, \dots, m\}.$$

Then for $j = 2, 3, ..., m$, $c_j^t(\gamma)$ can be expressed equivalently as

$$c_j^t(\gamma) = \sum_{i=1}^{n} p_{ij}^t \mathbf{1}[p_i^t \in A_j(\gamma)].$$

In other words, the capacity $c_j^t(\gamma)$ of message j is the sum of clickthrough probabilities contributed by users whose profiles fall inside the polytope $A_j(\gamma)$.

Consider two different vectors of dual variables $\gamma, \beta \in \mathcal{G}$. If we change the dual variables from γ to β, the users i who were assigned to message j under γ, but who are now moved to other messages under β, must satisfy

$$p_i^t \in A_j(\gamma) \setminus A_j(\beta).$$

For problem t, the total expected number of clickthroughs, conditional on the random pool of users, moved away from message j is

$$\sum_{i=1}^{n} p_{ij}^t \mathbf{1}[p_i^t \in A_j(\gamma) \setminus A_j(\beta)]. \tag{22}$$

It is easy to check that the expectation (over all randomly drawn user profiles) of (22) is

$$\mathbf{E}\left[\sum_{i=1}^{n} p_{ij}^t \mathbf{1}[p_i^t \in A_j(\gamma) \setminus A_j(\beta)]\right] = n\mathbf{E}\left[p_{ij}^t \mathbf{1}[p_i^t \in A_j(\gamma) \setminus A_j(\beta)]\right] = n \int_{A_j(\gamma)\setminus A_j(\beta)} x_j d\vec{x}.$$

The following lemma bounds the expected number of clickthroughs moved from one message to another when γ changes to β.

Lemma 3. *Given any two different vectors of dual variables $\gamma, \beta \in \mathcal{G}$ such that $\beta_l < \gamma_l$ for some index l, we must have*

$$\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 d\vec{x} \geq \frac{1}{m+1}\left(\frac{r_1}{r_l - \gamma_l} - \frac{r_1}{r_l - \beta_l}\right) \prod_{j=2, j\neq l}^{m} \frac{r_1}{r_j - \gamma_j}.$$

For each message $j = 2, 3, ..., m$, Define a number of increments n_j^t as

$$n_j^t \equiv \lfloor \frac{r_j}{0.5t^{-3/4}} \rfloor.$$

Define a discretization of the real line in n_j^t increments as follows. Let $g_{j0}^t = 0$, and let $g_{j1}^t, ..., g_{jn_j}^t$ be such that $g_{jk}^t = g_{j,k-1}^t + 0.5t^{-3/4}$ for $k = 1, 2, ..., n_j^t$.

Let e_k denote the unit vector with the k-th element being one. Define regions $S_{jk}^t \subseteq \mathcal{P}, j = 2, 3, ..., m, k = 1, 2, ..., n_j$, in the user profile space as the difference between successive polytopes $A_1(g_{jk-1}^t e_j)$ and $A_1(g_{jk}^t e_j)$.

$$S_{jk}^t \equiv A_1(g_{jk}^t e_j) \setminus A_1(g_{jk-1}^t e_j).$$

The sets S_{jk}^t, which we call *cells*, $j = 1, \ldots, m$, $k = 1, \ldots, n_j^t$, $t = 1, 2, \ldots$ are a way to divide up the space of user profiles into countably many fixed regions.

The following lemma establishes that there is at least one cell S_{lk}^t in the difference of polytopes $A_1(\gamma) \setminus A_1(\beta)$, for any pair of vectors γ and β in $\mathcal{G}$ such that $\gamma_l \geq \beta_l + t^{-3/4}$.

Lemma 4. *Fix t. For any two dual vectors $\gamma, \beta \in \mathcal{G}$ such that*

$$\gamma_l \geq \beta_l + t^{-3/4}$$

for some $l \in \{2, 3, ..., m\}$, there must exist some $k \in \{1, 2, ..., n_l^t\}$ such that

$$S_{lk}^t \subseteq A_1(\gamma) \setminus A_1(\beta).$$

Lemma 5. *Let $\mathcal{E}_{lk}^t$ denote the event that*

$$\frac{r_1}{\max_{j=1,2,...,m} r_j} \cdot \sum_{i=1}^n p_{i1}^t \mathbf{1}[p_i^t \in S_{lk}^t] > 1.$$

There exist constants C_1 and t^* such that

$$P(\overline{\mathcal{E}_{lk}^t}) \leq e^{\frac{\max_{j=1,2,...,m} r_j}{r_1}} \cdot \left(1 - \frac{C_1}{t^{3/4}}\right)^{t\bar{n}}$$

for all $l = 2, \ldots, m$, $t \geq t^*$, and $k = 1, \ldots, n_l^t$.

13. Asymptotic Regret of the Reservation Algorithm

In this section we will bound the regret $V_t^{LP} - V_t^{RA}$ in a probabilistic sense, in the big-data scaling regime.

Recall that we choose the reservation level for a system of size t as (see (11))

$$\Delta(t) \equiv C \cdot \sum_{j=2}^m \sqrt{c_j} \cdot \log \sum_{j=2}^m c_j = C \cdot \sum_{j=2}^m \sqrt{t\bar{c}_j} \cdot \log \sum_{j=2}^m t\bar{c}_j. \tag{23}$$

Here, $C > 0$ is fixed for all t. Recall that m is constant for all system size t, but $n = t \cdot \bar{n}$, and $c_j = t \cdot \bar{c}_j$, $\forall j = 1, 2, ..., m$. For the instance of size t, let V_t^{LP} denote the optimal objective value of the linear program (3), and let $\gamma^t(c)$ be a vector of its optimal dual variables when the capacity vector is c; let V_t^{RA} denote the expected total reward of the Reservation Algorithm for size t (conditional on the random pool of users); let δ^t be the noise defined as in (12), and let $\mathcal{O}_t$ denote the event that $\sum_{j=2}^m |\delta_j^t| \frac{r_j}{r_1} \leq \Delta(t)$.

Theorem 8. *Suppose that for each problem t, the reservation level used in RA is $\Delta(t)$. Then the regret of RA is $O(t^{1/4} \log t)$ with probability 1.*

14. Regret of the Static Algorithm

In this section, we analyze the regret of the Static Algorithm and compare it to that of the Reservation Algorithm.

Under the Static Algorithm, the random number of clickthroughs that message $j \in \{1,2,...,m\}$ receives has mean

$$b_j \equiv \sum_{i=1}^{n} s^*_{ij} p_{ij} \le c_j \tag{24}$$

and standard deviation

$$\sigma_j \equiv \sqrt{\sum_{i=1}^{n} s^*_{ij} p_{ij}(1 - s^*_{ij} p_{ij})} \le \sqrt{b_j} \le \sqrt{c_j}. \tag{25}$$

Since σ_j is the size of noise in the random number of clickthroughs that message j receives under this algorithm, we can expect that the regret of this algorithm grows as $O(\sigma_j) = O(\sqrt{c_j}) = O(\sqrt{t})$. The following theorem shows that $O(\sqrt{t})$ is an upper bound on the regret of the Static Algorithm. Later, we will further show that $O(\sqrt{t})$ is a tight bound.

THEOREM 9. *The regret of the Static Algorithm is $O(\sqrt{t})$ almost surely.*

Finally, we prove a lower bound on the performance of the Static Algorithm with respect to V_t^{LP}. This result provides a worst-case lower bound on $V_t^{RA} - V_t^{Static}$.

THEOREM 10. *There exist problem instances in which almost surely,*

$$V_t^{RA} - V_t^{Static} = \Omega(\sqrt{t}).$$

15. Numerical Studies

We test the performance of the Reservation Algorithm and the Static Algorithm by simulating their total reward values using real data of clickthrough probabilities.

We use three different data-sets. Each data set contains a set of messages that were sent to several hundred million users on a certain day in March 2016, and the estimated clickthrough probabilities between all the users and messages for that day.

We implement the algorithms on a distributed computing system using the MapReduce framework. The linear program used in the algorithms is solved by smoothing the dual problem (4) and then using a descent method. We refer the reader to Zhong et al. (2015) for a similar distributed algorithm for solving an LP. Problem (6) in the second step of the Reservation Algorithm is solved by using a binary search to find an appropriate ratio ω such that $p_{ik}/p_{ij} \geq \omega$ for every $i \in R^{jk}$.

The expected total reward for both algorithms are simulated based on clickthrough probabilities provided by Alibaba. We vary the reservation level Δ used in the Reservation Algorithm over a range. For each test case, we report the relative regret of the two algorithms as

$$\alpha \equiv \frac{V^{LP} - V^{RA}}{V^{LP} - V^{Static}}.$$

Note that since

$$\frac{V^{LP} - V^{RA}}{V^{LP} - V^{Static}} \geq \frac{V^{OPT} - V^{RA}}{V^{OPT} - V^{Static}},$$

the ratios we report overestimate the actual regret of the Reservation Algorithm compared to that of the Static Algorithm. That is, if compared against V^{OPT} instead of V^{LP}, the actual performance of the Reservation Algorithm will be better than the results reported. Figure 4 summarizes our test results. The actual performance of the Reservation Algorithm highly depends on the types of products sent out on each day. Among the 3 test cases, our Reservation Algorithm improves the total regret of the Static Algorithm by at least 10% and as much as 50%. Thus, implementing the Reservation Algorithm is very promising in improving the overall benefit earned from this mobile-based recommendation system.

References

Agrawal, Shipra, Zizhuo Wang, Yinyu Ye. 2009. A dynamic near-optimal algorithm for online linear programming. *arXiv preprint arXiv:0911.2974* .

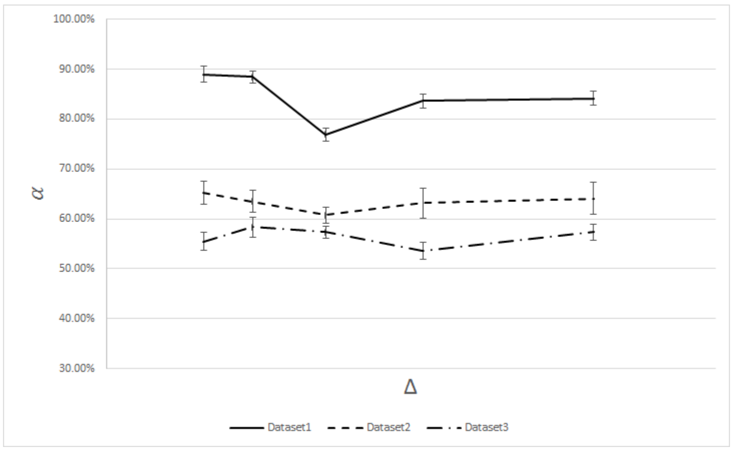


Figure 4 Regret of the Reservation Algorithm relative to the Static Algorithm under different values of Δ.

Bahmani, Bahman, Michael Kapralov. 2010. Improved bounds for online stochastic matching. *Proceedings of the 18th Annual European Conference on Algorithms: Part I*. ESA'10, Springer-Verlag, Berlin, Heidelberg, 170–181. URL `http://dl.acm.org/citation.cfm?id=1888935.1888956`.

Bertsimas, Dimitris, John Tsitsiklis. 1997. *Introduction to Linear Optimization*. 1st ed. Athena Scientific.

BusinessWire. 2016. Alibaba group announces june quarter 2016 results. `http://www.businesswire.com/news/home/20160811005428/en`.

Elmachtoub, Adam N., Retsef Levi. 2015. From cost sharing mechanisms to online selection problems. *Mathematics of Operations Research* **40**(3) 542–557. doi:10.1287/moor.2014.0684. URL `http://dx.doi.org/10.1287/moor.2014.0684`.

Elmachtoub, Adam N., Retsef Levi. 2016. Supply chain management with online customer selection. *Operations Research* **64**(2) 458–473. doi:10.1287/opre.2015.1472. URL `http://dx.doi.org/10.1287/opre.2015.1472`.

Emarketer.com. 2016. E-commerce turns into m-commerce in china. `http://www.emarketer.com/Article/Ecommerce-Turns-Mcommerce-China/1013736`.

Feldman, Jon, Aranyak Mehta, Vahab Mirrokni, S. Muthukrishnan. 2009. Online stochastic matching: Beating 1-1/e. *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '09, IEEE Computer Society, Washington, DC, USA, 117–126. doi:10.1109/FOCS.2009.72. URL `http://dx.doi.org/10.1109/FOCS.2009.72`.

Haeupler, Bernhard, Vahab S. Mirrokni, Morteza Zadimoghaddam. 2011. Online stochastic weighted matching: Improved approximation algorithms. *Proceedings of the 7th International Conference on Internet and Network Economics*. WINE'11, Springer-Verlag, Berlin, Heidelberg, 170–181.

Jaillet, Patrick, Xin Lu. 2014. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research* **39**(3) 624–646. doi:10.1287/moor.2013.0621. URL `http://dx.doi.org/10.1287/moor.2013.0621`.

Jasin, Stefanus. 2015. Performance of an lp-based control for revenue management with unknown demand parameters. *Operations Research* **63**(4) 909–915. doi:10.1287/opre.2015.1390. URL `http://dx.doi.org/10.1287/opre.2015.1390`.

Jasin, Stefanus, Sunil Kumar. 2012. A re-solving heuristic with bounded revenue loss for network revenue management with customer choice. *Mathematics of Operations Research* **37**(2) 313–345. doi:10.1287/moor.1120.0537. URL `http://dx.doi.org/10.1287/moor.1120.0537`.

Manshadi, Vahideh H., Shayan Oveis Gharan, Amin Saberi. 2012. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research* **37**(4) 559–573. doi:10.1287/moor.1120.0551. URL `http://dx.doi.org/10.1287/moor.1120.0551`.

Reiman, Martin I., Qiong Wang. 2008. An asymptotically optimal policy for a quantity-based network revenue management problem. *Mathematics of Operations Research* **33**(2) 257–282. doi:10.1287/moor.1070.0288. URL `http://dx.doi.org/10.1287/moor.1070.0288`.

Talluri, Kalyan T., Garrett J. van Ryzin. 2004. *The Theory and Practice of Revenue Management*. Springer.

Wang, X., V. Truong, D. Bank. 2015. Online advance admission scheduling for services, with customer preferences. Working paper.

Wayne, Kevin Daniel. 1999. Generalized maximum flow algorithms. Ph.D. thesis, Cornell University, Ithaca, NY.

Zhong, Wenliang, Rong Jin, Cheng Yang, Xiaowei Yan, Qi Zhang, Qiang Li. 2015. Stock constrained recommendation in tmall. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15, ACM, New York, NY, USA, 2287–2296. doi:10.1145/2783258.2788565. URL `http://doi.acm.org/10.1145/2783258.2788565`.

16. Appendix

Proof of Theorem 1.

Proof. Define

$$J_{ij}^{\Pi} \equiv \begin{cases} 1, & \text{if } \Pi \text{ sends message } j \text{ to user } i, \\ 0, & \text{otherwise.} \end{cases}$$

Then, the objective value (2) becomes

$$\sum_{j=1}^{m} r_j \cdot \sum_{i=1}^{n} \mathbf{E}[J_{ij}^{OPT}] p_{ij}, \tag{26}$$

which is the same as the objective of the linear program (3) when $\mathbf{E}[J_{ij}^{\mathrm{OPT}}]$ is the decision variable.

Taking expectation on both sides of (1), we get

$$\sum_{i=1}^{n} \mathbf{E}[I_{ij}^{OPT}] \le c_j$$

$$\Longrightarrow \sum_{i \in \mathcal{N}} \mathbf{E}[J_{ij}^{OPT}] \cdot p_{ij} \le c_j, \ \forall j = 1, 2, ..., m,$$

which is the first constraint of the linear program.

Moreover, the model requires $\sum_{j=1}^{m} J_{ij}^{OPT} = 1$, which gives the second constraint of the LP.

In sum, $\mathbf{E}[J_{ij}^{OPT}]$ satisfies all the constraints of the LP, and thus the optimal objective value of the LP is an upper bound on (26). $\square$

Proof of Proposition 1

Proof. Suppose there is a solution s^* that violates the proposition, i.e., $\sum_{i=1}^{n} \|s_i^*\|_0 > n + m^2$. Then there are more than m^2 users who are simultaneously routed to more than one message according to s^*. Since there are $\binom{m}{2} < m^2$ unique ways to choose unique pairs of messages, there must exist two users $i, l \in \{1, 2, ..., n\}$, $i \neq l$, and two messages $j, k \in \{1, 2, ..., m\}$, $j \neq k$, such that $s_{ij}^* > 0$, $s_{ik}^* > 0$, $s_{lj}^* > 0$, $s_{lk}^* > 0$.

Let γ^* be the vector of optimal dual variables to (4) that correspond to the capacity constraints. According to (5), we must have

$$(r_j - \gamma_j^*)p_{ij} = (r_k - \gamma_k^*)p_{ik} \geq (r_1 - \gamma_1^*)p_{i1} = r_1 p_{i1},$$

where the last step follows from the fact that $c_1 = \infty$ and thus $\gamma_1^* = 0$ (see Lemma 2). Similarly, we have

$$(r_j - \gamma_j^*)p_{lj} = (r_k - \gamma_k^*)p_{lk} \geq (r_1 - \gamma_1^*)p_{l1} = r_1 p_{l1}.$$

Combining the above two equations, we have

$$(r_j - \gamma_j^*)p_{ij} \cdot (r_k - \gamma_k^*)p_{lk} = (r_k - \gamma_k^*)p_{ik} \cdot (r_j - \gamma_j^*)p_{lj}$$

$$\Longrightarrow (r_j - \gamma_j^*)(r_k - \gamma_k^*) \cdot p_{ij}p_{lk} = (r_k - \gamma_k^*)(r_j - \gamma_j^*) \cdot p_{ik}p_{lj}.$$

Then if $p_{ik}p_{lj} \neq p_{lk}p_{ij}$, we must have $(r_j - \gamma_j^*)(r_k - \gamma_k^*) = 0$. Thus, either $r_j - \gamma_j^* = 0$ or $r_k - \gamma_k^* = 0$, which implies $r_1 p_{i1} = r_1 p_{l1} = 0$. According to Assumption 1, this implies that $p_{i1} = 0 \Rightarrow \|p_i\|_1 = 0$. Therefore, we must have either $r_1 = 0$ or $\|p_i\|_1 = 0$, which is a contradiction.

□

Proof of Lemma 1

Proof. Suppose there exists an $s(\delta)$ such that $s(\delta) = x^{(1)} + y$ and $y \geq 0$. We can equivalently write the constraints of (13) as

$$\begin{aligned}
&\sum_{i=1}^{n} s_{ij}(\delta)p_{ij} \leq (c_j - \delta_j)^+ \\
\Longleftrightarrow &\sum_{i=1}^{n} y_{ij}p_{ij} \leq (c_j - \delta_j)^+ - \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} \\
\Longleftrightarrow &\sum_{i=1}^{n} y_{ij}p_{ij} \leq (c_j - \delta_j - \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij})^+ \quad \text{(because } y \geq 0\text{)} \\
\Longleftrightarrow &\sum_{i=1}^{n} y_{ij}p_{ij} \leq (c_j - D_j)^+.
\end{aligned}$$

And

$$\sum_{j=1}^{m} s_{ij}(\delta)=1 \Longleftrightarrow \sum_{j=1}^{m} y_{ij}=1-\sum_{j=1}^{m} x_{ij}^{(1)}.$$

Therefore, knowing that $s(\delta)=x^{(1)}+y$ and $y\geq 0$, we can transform the problem of finding y into the linear program (10). Thus, $x^{(2)}$ must be at least as good as y, from which it follows that $x^{(1)}+x^{(2)}$ is optimal for (13). $\square$

Proof of Theorem 2

Proof. Fix δ. We want to find the difference between $V^{LP}=V^{LP}(0)$ and $V^{LP}(\delta)$ in terms of the derivative of the function $V^{LP}(\cdot)$. It is known that the Lagrangian multiplier $\gamma_j(c)$ is the marginal benefit of increasing the capacity c_j of message j (Bertsimas and Tsitsiklis 1997, p.155-156). Thus, for any $u\in\mathbb{R}^m$ such that $u\leq c$, the derivative of $V^{LP}(u)$ with respect to u_j is $-\gamma_j(c-u)$.

Let c be the capacity vector given in the model. Define a vector $\delta^{(j)}\in\mathbb{R}^m$ as

$$\delta_i^{(j)}=\begin{cases}\min(c_i,\delta_i), & \text{for } i\leq j,\\ 0, & \text{for } i>j.\end{cases}$$

For each component $j=1,2,...,m$, we want to integrate the derivative of $V^{LP}(\cdot)$ from c_j to $(c_j-\delta_j)^+$ while keeping other components unchanged. This can be achieved by integrating from vector $c-\delta^{(j-1)}$ to $c-\delta^{(j)}$ for each component j, which can be written as

$$V^{LP}(\delta)=V^{LP}+\sum_{j=1}^{m}\int_0^{|\min(c_j,\delta_j)|} -\gamma_j(c-\delta^{(j-1)}-xe_j)\frac{\delta_j}{|\delta_j|}dx.$$

According to (14), we must have $\forall u\in\mathbb{R}^m$ such that $u\leq c$,

$$|\gamma_j(c-u)-\gamma_j(c)|\leq(\|u\|_1+m)\bar{\gamma},\ \ \forall j=1,2,...,m.$$

This gives us that if $\delta_j<0$ then

$$\int_0^{|\min(c_j,\delta_j)|} -\gamma_j(c-\delta^{(j-1)}-xe_j)\frac{\delta_j}{|\delta_j|}dx=\int_0^{-\delta_j}\gamma_j(c-\delta^{(j-1)}-xe_j)dx$$

$$
\begin{aligned}
&\geq \int_0^{-\delta_j} \left[\gamma_j(c) - (\|\delta^{(j-1)} + xe_j\|_1 + m)\bar{\gamma}\right] dx \\
&\geq \int_0^{-\delta_j} \left[\gamma_j(c) - (\|\delta\|_1 + m)\bar{\gamma}\right] dx \\
&= |\delta_j| \left[\gamma_j(c) - (\|\delta\|_1 + m)\bar{\gamma}\right].
\end{aligned}
$$

And if $\delta_j > 0$ then

$$
\begin{aligned}
\int_0^{|\min(c_j,\delta_j)|} -\gamma_j(c - \delta^{(j-1)} - xe_j)\frac{\delta_j}{|\delta_j|}dx &= \int_0^{\min(c_j,\delta_j)} -\gamma_j(c - \delta^{(j-1)} - xe_j)dx \\
&\geq \int_0^{\min(c_j,\delta_j)} \left[-\gamma_j(c) - (\|\delta^{(j-1)} + xe_j\|_1 + m)\bar{\gamma}\right] dx \\
&\geq \int_0^{\min(c_j,\delta_j)} \left[-\gamma_j(c) - (\|\delta\|_1 + m)\bar{\gamma}\right] dx \\
&= \min(c_j,\delta_j)\left[-\gamma_j(c) - (\|\delta\|_1 + m)\bar{\gamma}\right] \\
&\geq \delta_j\left[-\gamma_j(c) - (\|\delta\|_1 + m)\bar{\gamma}\right],
\end{aligned}
$$

where the last inequality follows from the fact that $\gamma_j(c)$ and $\bar{\gamma}$ are both non-negative.

In sum, we can write

$$
\int_0^{|\min(c_j,\delta_j)|} -\gamma_j(c - \delta^{(j-1)} - xe_j)\frac{\delta_j}{|\delta_j|}dx \geq -\delta_j\gamma_j(c) - |\delta_j|(\|\delta\|_1 + m)\bar{\gamma}
$$

$$
\Longrightarrow V^{LP}(\delta) \geq V^{LP} + \sum_{j=1}^m [-\delta_j\gamma_j(c) - |\delta_j|(\|\delta\|_1 + m)\bar{\gamma}] = V^{LP} - \sum_{j=1}^m \delta_j\gamma_j(c) - (\|\delta\|_1^2 + m\|\delta\|_1)\bar{\gamma}.
$$

Since $\mathbf{E}[\delta] = 0$, we then have

$$
\begin{aligned}
V^{LP} - \mathbf{E}[V^{LP}(\delta)] &\leq \sum_{j=1}^m \mathbf{E}[\delta_j]\gamma_j(c) + (\mathbf{E}[\|\delta\|_1^2] + m\mathbf{E}[\|\delta\|_1])\bar{\gamma} \\
&= (\mathbf{E}[\|\delta\|_1^2] + m\mathbf{E}[\|\delta\|_1])\bar{\gamma}.
\end{aligned}
$$

Using Jensen's inequality, we can obtain $\mathbf{E}[|\delta_j|] \leq \sqrt{Var(\delta_j)}$. Thus,

$$
\mathbf{E}[\|\delta\|_1] = \sum_{j=1}^m \mathbf{E}[|\delta_j|]
$$

$$
\begin{aligned}
&\leq \sum_{j=1}^{m} \sqrt{Var(\delta_j)} \\
&= \sum_{j=1}^{m} \sqrt{\sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} (1 - x_{ij}^{(1)} p_{ij})} \\
&\leq m\sqrt{n}.
\end{aligned}
$$

Furthermore,

$$
\begin{aligned}
\mathbf{E}[\|\delta\|_1^2] &= \sum_{j=1}^{m}\sum_{k=1}^{m} \mathbf{E}[|\delta_j \delta_k|] \\
&\leq \sum_{j=1}^{m}\sum_{k=1}^{m} \frac{1}{2}\mathbf{E}[\delta_j^2 + \delta_k^2] \\
&= \sum_{j=1}^{m}\sum_{k=1}^{m} \frac{1}{2}(Var(\delta_j) + Var(\delta_k)) \\
&= \sum_{j=1}^{m}\sum_{k=1}^{m} \frac{1}{2}\left(\sum_{i=1}^{n} x_{ij}^{(1)} p_{ij}(1 - x_{ij}^{(1)} p_{ij}) + x_{ik}^{(1)} p_{ik}(1 - x_{ik}^{(1)} p_{ik})\right) \\
&\leq m^2 n.
\end{aligned}
$$

Combining the above inequalities, we get

$$
V^{LP} - \mathbf{E}[V^{LP}(\delta)] \leq m^2(n + \sqrt{n})\bar{\gamma}.
$$

□

Proof of Theorem 4.

Since the bound stated in the theorem is additive and linear in $|c(v_k, T) - \bar{c}(v_k, T)|$, it suffices to prove the case that the gap $|c(v_k, T) - \bar{c}(v_k, T)|$ is very small. It is easy to see that when $|c(v_k, T) - \bar{c}(v_k, T)|$ is small enough, the network flow algorithms will perform no more than one augmentation. Thus, we only focus on the case where the algorithms perform only one augmentation.

For Case 1, recall that $0 \leq \bar{c}(v_k, T) < c(v_k, T)$. Suppose initially $\bar{f}$ is not optimal for $\bar{G}$, i.e., $\bar{f}(v_k, T) < f(v_k, T)$. In other words, some users are originally assigned to message k, but now have to move to other messages due to the decrease in the capacity.

Suppose that the algorithm performs exactly one augmentation along a shortest path from v_k to T. Let θ_j be the size of the augmenting flow that leaves node v_j, $\forall v_j \in V$. We need to show that $\forall v_j \in V$,

$$\theta_j \leq \theta_k \frac{r_k}{r_1}.$$

This is clearly true when $j = k$, as our model requires $r_k \geq r_1$ (see Section 3. Note that the theorem holds even if we allow $r_k < r_1$, in which case changing the capacity of message k will have no affect at all).

Suppose there is some $l \neq k$ such that $\theta_l > \theta_k \frac{r_k}{r_1}$. Let u_i be the user node preceding v_l in the augmenting path. Then the size of the augmenting flow passing through u_i is θ_l / p_{il}. Let P be the part of the augmenting path from v_k to u_i. According to (18), we can obtain the total length of P from the relative size of the augmenting flow at nodes v_k and u_i. The total length of P is

$$\log \frac{\theta_k}{\theta_l / p_{il}}.$$

Consider the cycle $P \to v_1 \to T \to v_k$ that returns to v_k. This cycle must exist in the residual network because the edges (u_i, v_1) and (v_1, T) always have infinite capacity, and the edge (T, v_k) exists in G_f since $f(v_k, T) > 0$. The total length of the cycle is

$$\begin{aligned}
&\log \frac{\theta_k}{\theta_l / p_{il}} + w(u_i, v_1) + w(v_1, T) + w(T, v_k) \\
=&\log \frac{\theta_k}{\theta_l / p_{il}} - \log p_{i1} - \log r_1 + \log r_k \\
=&\log \left(\frac{\theta_k}{\theta_l} \cdot \frac{r_k}{r_1} \cdot \frac{p_{il}}{p_{i1}} \right) \\
\leq&\log \left(\frac{\theta_k}{\theta_l} \cdot \frac{r_k}{r_1} \right) \quad \text{(by Assumption 1)} \\
<&0.
\end{aligned}$$

In other words, the cycle is a negative cycle. According to Theorem 3, this contradicts the fact that f is optimal for G. Thus, we must have $\theta_j \leq \theta_k \frac{r_k}{r_1}$ for every $v_j \in V$.

For Case 2, recall that $\bar{c}(v_k,T) > c(v_k,T)$. Suppose initially $\bar{f} = f$ is not optimal for $\bar{G}$. Recall that any negative cycle in the residual network $\bar{G}_{\bar{f}}$ must pass through (v_k,T). Again suppose there is exactly one augmentation performed by the algorithm and let θ_j be the size of the augmenting flow that leaves node $v_j \in V$. Similarly, we need to prove that $\theta_j \le \theta_k \frac{r_k}{r_1}$, $\forall j = 1,2,...,m$. The case of $j = k$ is again trivially true as we only consider the case $r_k \ge r_1$.

Suppose there is some $l \ne k$ such that $\theta_l > \theta_k \frac{r_k}{r_1}$. Let u_i be the user node following v_l in the augmenting cycle. Then the size of the augmenting flow that passes through u_i is θ_l/p_{il}. Furthermore, the size of the augmenting flow that leaves T is $\theta_k r_k$. Let P be the part of the augmenting cycle from T to u_i. According to (18), we can obtain the total length of P from the relative size of the augmenting flow at T and u_i. The total length of P is

$$\log \frac{\theta_k r_k}{\theta_l / p_{il}}.$$

Consider the new cycle $P \to v_1 \to T$ which returns to T. This new cycle must exist in the residual network because the edges (u_i,v_1) and (v_1,T) have infinity capacity. The total length of the new cycle is

$$\begin{aligned}
&\log \frac{\theta_k r_k}{\theta_l / p_{il}} + w(u_i,v_1) + w(v_1,T) \\
=&\log \frac{\theta_k r_k}{\theta_l / p_{il}} - \log p_{i1} - \log r_1 \\
=&\log \left(\frac{\theta_k}{\theta_l} \cdot \frac{r_k}{r_1} \cdot \frac{p_{il}}{p_{i1}} \right).
\end{aligned}$$

Similar to the previous argument, this total length is negative, because $\theta_l > \theta_k \frac{r_k}{r_1}$ and according to Assumption 1, $p_{il} \le p_{i1}$. In other words, this new cycle is also a negative cycle. However, this new cycle does not pass through edge (v_k,T), which forms a contradiction. Thus, for Case 2 we also have $\theta_j \le \theta_k \frac{r_k}{r_1}$, $\forall j = 1,2,...,m$. $\square$

Proof of Theorem 5.

Starting with an optimal flow f for the unperturbed network, as defined by the optimal solution s^* to the LP (3), we apply the network-flow algorithms above to obtain an optimal flow $\bar{f}$ for the perturbed network, as defined by the solution of (13). According to Lemma 1, it suffices to show that if $\sum_{j=2}^{m} |\delta_j| \frac{r_j}{r_1} \le \Delta$, we must have $\bar{f}(u_i, v_j) \ge x_{ij}^{(1)}$ for all $u_i \in U$ and $v_j \in V$.

By the definition (8) of $x^{(1)}$, we must have $s^* \ge x^{(1)}$ and thus initially $f(u_i, v_j) \ge x_{ij}^{(1)}$ for all $u_i \in U$ and $v_j \in V$. Note that for any edge (u_i, v_j), the only way to reduce its flow value is to send an augmenting flow that passes (v_j, u_i) in the residual graph. We claim that if $\sum_{j=2}^{m} |\delta_j| \frac{r_j}{r_1} \le \Delta$, no augmenting flow will ever pass through any edge (v_j, u_i) such that $x_{ij}^{(1)} > 0$. This claim implies that for any $x_{ij}^{(1)} > 0$, the flow value on (u_i, v_j) can only increase during the network flow algorithm, which proves that eventually $\bar{f}(u_i, v_j) \ge f(u_i, v_j) \ge x_{ij}^{(1)}$ for any $x_{ij}^{(1)} > 0$.

To see the claim, suppose that at some step during augmentation, the augmenting flow passes through (v_j, u_i) for some $v_j \in V$ and $u_i \in U$. The node following u_i in the augmenting flow must be another message node $v_k \in V$. The total length of $v_j \to u_i \to v_k$ is

$$w(v_j, u_i) + w(u_i, v_k) = \log p_{ij} - \log p_{ik} = \log \frac{p_{ij}}{p_{ik}}.$$

Since the augmenting path or cycle has the *shortest length*, user i must have the smallest value of p_{lj}/p_{lk} among all users l such that (v_j, u_l) exists in the residual network right before the augmentation is performed. Recall that in the Reservation algorithm, we defined a set R^{jk} of users l who have the smallest values of the ratio p_{lj}/p_{lk}. Therefore, if any user in R^{jk} has a flow going to v_j right before the augmentation is performed, we must have $i \in R^{jk}$ and thus $x_{ij}^{(1)} = 0$. When there is a tie in determining a user having the smallest ratio, we assume without loss of generality that the network-flow algorithms first chooses for augmentation a user from the set R^{jk}.

Initially, according to Step 2 of the Reservation algorithm, the total size of flow that v_j receives from the users in R^{jk} according to the solution s^* is at least Δ (or else all users are in R^{jk}). Then if $\sum_{j=2}^m |\delta_j|\frac{r_j}{r_1} \le \Delta$, we know by Theorem 4 that the total amount of augmenting flows leaving v_j is no more than Δ. Thus, the flows that v_j receives from users in R^{jk} can be reduced by at most Δ throughout the network flow algorithms, which implies that at any augmentation step of the network flow algorithms, at least one user in R^{jk} has a flow going to v_j. Combined with the argument above, this proves the claim. □

Proof of Theorem 6.

Let $D_j^{(2)}$ be the number of clickthroughs that message j receives in the second cycle. The total expected reward of the Reservation Algorithm is

$$\begin{aligned}
V^{RA} &= \sum_{j=1}^m r_j \mathbf{E}[\min(c_j, D_j + D_j^{(2)})] \\
&= \sum_{j=1}^m r_j \mathbf{E}[\delta_j + \min(c_j - \delta_j, D_j - \delta_j + D_j^{(2)})] \\
&= \sum_{j=1}^m r_j \mathbf{E}[\min(c_j - \delta_j, D_j - \delta_j + D_j^{(2)})] \\
&= \sum_{j=1}^m r_j \mathbf{E}[\min(c_j - \delta_j, \mathbf{E}[D_j] + D_j^{(2)})] \\
&= \sum_{j=1}^m r_j \mathbf{E}[\min(c_j - \delta_j, \sum_{i=1}^n x_{ij}^{(1)} p_{ij} + D_j^{(2)})]. \qquad (27)
\end{aligned}$$

Here we interpret $c_j - \delta_j$ as the perturbed capacity of message j, and $\sum_{i=1}^n x_{ij}^{(1)} p_{ij} + D_j^{(2)}$ as the 'actual' number of clickthroughs that message j receives in both two cycles. In other words, we view δ as an exogenous noise that arises in the capacity values.

According to Theorem 5, $x^{(1)}+x^{(2)}$ is an optimal solution to the perturbed linear program conditional on $\mathcal{O}$. Then the expected objective value of the perturbed linear program can be written as

$$\begin{aligned}\mathbf{E}[V^{LP}(\delta)] &= \mathbf{E}[V^{LP}(\delta)|\mathcal{O}]P(\mathcal{O}) + \mathbf{E}[V^{LP}(\delta)|\bar{\mathcal{O}}]P(\bar{\mathcal{O}}) \\ &= \mathbf{E}[\sum_{j=1}^{m} r_j \sum_{i=1}^{n}(x_{ij}^{(1)}+x_{ij}^{(2)})p_{ij}|\mathcal{O}]P(\mathcal{O}) + \mathbf{E}[V^{LP}(\delta)|\bar{\mathcal{O}}]P(\bar{\mathcal{O}}) \\ &\le \mathbf{E}[\sum_{j=1}^{m} r_j \sum_{i=1}^{n}(x_{ij}^{(1)}+x_{ij}^{(2)})p_{ij}|\mathcal{O}]P(\mathcal{O}) + n\cdot \max_{j=1,2,\ldots,m} r_j \cdot P(\bar{\mathcal{O}}). \end{aligned} \tag{28}$$

Here the last inequality follows from the fact that $n\cdot\max_{j=1,2,\ldots,m} r_j$ is an upper bound on the total reward value of any allocation.

Conditional on $\mathcal{O}$, the constraint of the perturbed linear program gives

$$\sum_{i=1}^{n}(x_{ij}^{(1)}+x_{ij}^{(2)})p_{ij} \le (c_j-\delta_j)^+, \forall j=1,2,\ldots,m.$$

This implies that, if $\sum_{i=1}^{n} x_{ij}^{(1)}p_{ij} > 0$, we must have $c_j > \delta_j$. On the other hand, if $\sum_{i=1}^{n} x_{ij}^{(1)}p_{ij} = 0$, we must have $D_j = 0 \Longrightarrow \delta_j = 0$. In sum, conditional on the event $\mathcal{O}$, we always have $c_j \ge \delta_j$, and thus

$$\sum_{i=1}^{n}(x_{ij}^{(1)}+x_{ij}^{(2)})p_{ij} \le c_j-\delta_j, \forall j=1,2,\ldots,m. \tag{29}$$

Define

$$\delta_j^{(2)} \equiv D_j^{(2)} - \sum_{i=1}^{n} x_{ij}^{(2)}p_{ij}.$$

Combining (27) and (28), we get

$$\begin{aligned}&\mathbf{E}[V^{LP}(\delta)] - V^{RA} \\ \le&\mathbf{E}[\sum_{j=1}^{m} r_j \sum_{i=1}^{n}(x_{ij}^{(1)}+x_{ij}^{(2)})p_{ij}|\mathcal{O}]P(\mathcal{O}) + n\cdot \max_{j=1,2,\ldots,m} r_j \cdot P(\bar{\mathcal{O}}) \\ &-\sum_{j=1}^{m} r_j \mathbf{E}[\min(c_j-\delta_j, \sum_{i=1}^{n} x_{ij}^{(1)}p_{ij} + D_j^{(2)})]\end{aligned}$$

$$\leq \mathbf{E}[\sum_{j=1}^{m} r_j \sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij}|\mathcal{O}]P(\mathcal{O}) + n \cdot \max_{j=1,2,\ldots,m} r_j \cdot P(\bar{\mathcal{O}})$$
$$- \sum_{j=1}^{m} r_j \mathbf{E}[\min(c_j - \delta_j, \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} + D_j^{(2)})|\mathcal{O}]P(\mathcal{O})$$
$$= \sum_{j=1}^{m} r_j \mathbf{E}[\sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij} - \min(c_j - \delta_j, \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} + D_j^{(2)})|\mathcal{O}]P(\mathcal{O}) + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}})$$
$$= \sum_{j=1}^{m} r_j \mathbf{E}[\sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij} - \min(c_j - \delta_j, \sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)}) p_{ij} + \delta_j^{(2)})|\mathcal{O}]P(\mathcal{O}) + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}}),$$

where $\delta_j^{(2)} = D_j^{(2)} - \sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}$. We apply the identity $a - \min(b, a+c) \leq \max(0, -c)$ for any $a \leq b$ to the above equation and continue to deduce that

$$\sum_{j=1}^{m} r_j \mathbf{E}[\sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij} - \min(c_j - \delta_j, \sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)}) p_{ij} + \delta_j^{(2)})|\mathcal{O}]P(\mathcal{O}) + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}})$$
$$\leq \sum_{j=1}^{m} r_j \mathbf{E}[\max(0, -\delta_j^{(2)})|\mathcal{O}]P(\mathcal{O}) + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}}) \quad \text{because of (29)}$$
$$\leq \sum_{j=1}^{m} r_j \mathbf{E}[\max(0, -\delta_j^{(2)})] + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}})$$
$$\leq \sum_{j=1}^{m} r_j \mathbf{E}[|\delta_j^{(2)}|] + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}}). \tag{30}$$

For any user i who satisfies $\|s_i^*\|_0 = 1$, where s^* is an optimal solution to the LP (3), we must have either $\|x_i^{(1)}\|_1 = 1$ or $\|x_i^{(2)}\|_1 = 1$. That is, we know for sure in which cycle this user will receive a message. Then according to the condition $\sum_{i=1}^{n} \|s_i^*\|_0 \leq n + m^2$ required in Step 1 of the Reservation Algorithm, there are at most m^2 users i that satisfy $\|s_i^*\|_0 > 1$. This implies that at most m^2 users i have $0 < \|x_i^{(2)}\|_1 < 1$ (i.e., each of these users has positive probabilities to both receive and not receive a message in the first cycle).

Let $U^{(2)}$ be the set of users who are sent messages in the second cycle. Then there are at most m^2 users who have positive probabilities to be either in or not in $U^{(2)}$. Thus,

$$\mathbf{E}[|\delta_j^{(2)}|] = \mathbf{E}[|D_j^{(2)} - \sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}|]$$

$$
\begin{aligned}
&\le \mathbf{E}[|D_j^{(2)} - \sum_{i\in U^{(2)}} \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij}|] + \mathbf{E}[|\sum_{i\in U^{(2)}} \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij} - \sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}|] \text{ by the triangle inequality}\\
&\le \mathbf{E}[|D_j^{(2)} - \sum_{i\in U^{(2)}} \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij}|] + m^2 \quad \text{since at most } m^2 \text{ users } i \text{ have } 0 < \|x_i^{(2)}\|_1 < 1\\
&\le \sqrt{\mathbf{E}\left[\left(D_j^{(2)} - \sum_{i\in U^{(2)}} \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij}\right)^2\right]} + m^2 \quad \text{by Jensen's inequality}\\
&= \sqrt{\mathbf{E}\left[\sum_{i\in U^{(2)}} \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij}\left(1 - \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij}\right)\right]} + m^2 \text{ by the variance of binary variables}\\
&\le \sqrt{\mathbf{E}\left[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}\left(1 - x_{ij}^{(2)} p_{ij}\right)\right]} + 2m^2 \quad \text{since at most } m^2 \text{ users } i \text{ have } 0 < \|x_i^{(2)}\|_1 < 1\\
&\le \sqrt{\mathbf{E}\left[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}\right]} + 2m^2. \qquad (31)
\end{aligned}
$$

By Theorem 5, conditional on $\mathcal{O}$, $\sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij}$ and $\sum_{i=1}^{n} s_{ij}^* p_{ij}$ stand for the amount of the capacity of message j allocated to users in the perturbed and original linear programs, respectively. In the generalized residual network, an increment in the expected number of clickthroughs that message j receives corresponds to an augmenting flow that passes through the edge (v_j, T). According to Theorem 4, the total increase in the expected number of clickthroughs that message j receives due to the perturbation δ must be no more than $\sum_{k=2}^{m} |\delta_k| \frac{r_k}{r_1}$. In other words, conditional on $\mathcal{O}$,

$$
\mathbf{E}[\sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij} - \sum_{i=1}^{n} s_{ij}^* p_{ij} | \mathcal{O}] \le \sum_{k=2}^{m} |\delta_k| \frac{r_k}{r_1} \le \Delta,\ \forall j = 1, 2, ..., m.
$$

$$
\begin{aligned}
\Longrightarrow \mathbf{E}[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij} | \mathcal{O}] &\le \Delta + \mathbf{E}[\sum_{i=1}^{n} s_{ij}^* p_{ij} - \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} | \mathcal{O}]\\
&= \Delta + \mathbf{E}[\sum_{i=1}^{n} s_{ij}^* p_{ij} \mathbf{1}(i \in \bigcup_{k=1, k\ne j}^{m} R^{jk}) | \mathcal{O}]\\
&\le \Delta + (m-1)(\Delta + 1)\\
&= m\Delta + m - 1,
\end{aligned}
$$

where the last inequality follows from (9). From this, we can deduce that

$$\begin{aligned}\mathbf{E}[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}] &\leq \mathbf{E}[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij} | \mathcal{O}] P(\mathcal{O}) + nP(\bar{\mathcal{O}}) \\ &\leq m\Delta + m - 1 + nP(\bar{\mathcal{O}}).\end{aligned}$$

Combining this result with (30) and (31), we can deduce that

$$\begin{aligned}\mathbf{E}[V^{LP}(\delta)] - V^{RA} &\leq \sum_{j=1}^{m} r_j \left[\sqrt{\mathbf{E}\left[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}\right]} + 2m^2 \right] + n \max_{j=1,2,...,m} r_j P(\bar{\mathcal{O}}) \\ &\leq \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta + m - 1 + nP(\bar{\mathcal{O}})} + 2m^2 \right] + n \max_{j=1,2,...,m} r_j P(\bar{\mathcal{O}}) \\ &\leq \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta} + 3m^2 + nP(\bar{\mathcal{O}}) + \sqrt{nP(\bar{\mathcal{O}})} \right].\end{aligned}$$

□

Proof of Lemma 3.

Proof. We must have $l \neq 1$ because $\beta_1 = \gamma_1 = 0$.

Define a vector $\theta \in \mathcal{G}$ to be $\theta_j = \gamma_j$, $\forall j \neq l$, and $\theta_l = \beta_l$. Next, we will prove the lemma by showing

$$\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 d\vec{x} \geq \int_{A_1(\gamma)\setminus A_1(\theta)} x_1 d\vec{x} \geq \frac{1}{m+1}\left(\frac{r_1}{r_l - \gamma_l} - \frac{r_1}{r_l - \beta_l}\right) \prod_{j=2, j\neq l}^{m} \frac{r_1}{r_j - \gamma_j}.$$

Any $x \in A_1(\theta)$ must satisfy

$$x_1 r_1 \geq x_j (r_j - \theta_j), \ \forall j = 2, 3, ..., m$$

$$\Longrightarrow x_1 r_1 \geq x_j (r_j - \gamma_j), \ \forall j = 2, 3, ..., m,$$

which implies that

$$A_1(\theta) \subseteq A_1(\gamma). \tag{32}$$

Furthermore, since γ and θ only differ by the l-th component, for any $x \in A_1(\gamma) \setminus A_1(\theta)$ we must have

$$x_1 r_1 < x_l (r_l - \theta_l) \quad \Longrightarrow \quad x_1 r_1 < x_l (r_l - \beta_l),$$

which implies that

$$A_1(\gamma) \setminus A_1(\theta) \subseteq A_1(\gamma) \setminus A_1(\beta). \tag{33}$$

Then (32) and (33) give us

$$\begin{aligned}\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 \vec{dx} &\geq \int_{A_1(\gamma)\setminus A_1(\theta)} x_1 \vec{dx} \\ &= \int_{A_1(\gamma)} x_1 \vec{dx} - \int_{A_1(\theta)\cap A_1(\gamma)} x_1 \vec{dx} \\ &= \int_{A_1(\gamma)} x_1 \vec{dx} - \int_{A_1(\theta)} x_1 \vec{dx}.\end{aligned}$$

Recall that

$$A_j(\gamma) = \{p \in \mathcal{P} \mid (r_j - \gamma_j)p_j - (r_k - \gamma_k)p_k \geq 0 \ \forall k = 1, \ldots, m\}$$

and $\gamma_1 = 0$. Therefore, the above integrals can be expressed in closed form as

$$\begin{aligned}\int_{A_1(\gamma)} x_1 \vec{dx} &= \int_0^1 x_1 \left(\int_0^{\frac{x_1 r_1}{r_2 - \gamma_2}} dx_2 \right) \left(\int_0^{\frac{x_1 r_1}{r_3 - \gamma_3}} dx_3 \right) \cdots \left(\int_0^{\frac{x_1 r_1}{r_m - \gamma_m}} dx_m \right) dx_1 \\ &= \int_0^1 x_1 \prod_{j=2}^m \frac{x_1 r_1}{r_j - \gamma_j} dx_1 \\ &= \int_0^1 x_1^m dx_1 \cdot \prod_{j=2}^m \frac{r_1}{r_j - \gamma_j} \\ &= \frac{1}{m+1} \prod_{j=2}^m \frac{r_1}{r_j - \gamma_j}.\end{aligned}$$

Thus,

$$\begin{aligned}\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 \vec{dx} &\geq \frac{1}{m+1} \prod_{j=2}^m \frac{r_1}{r_j - \gamma_j} - \frac{1}{m+1} \prod_{j=2}^m \frac{r_1}{r_j - \theta_j} \\ &= \frac{1}{m+1} \left(\frac{r_1}{r_l - \gamma_l} - \frac{r_1}{r_l - \beta_l} \right) \prod_{j=2, j\neq l}^m \frac{r_1}{r_j - \gamma_j}.\end{aligned}$$

$\square$

Proof of Lemma 4.

Proof. Since $\gamma_l \geq \beta_l + t^{-3/4}$ and $n_l^t \leq \frac{r_l}{0.5t^{-3/4}}$, there must exist a k such that $\beta_l \leq g_{l,k-1}^t < g_{lk}^t \leq \gamma_l$.

For any $x \in A_1(g_{lk}^t e_l)$, by definition we have $x_1 r_1 \geq x_l(r_l - g_{lk}^t)$ and $x_1 r_1 \geq x_j r_j$ for all $j \neq l, j = 2, 3, ..., m$, which gives

$$x_1 r_1 \geq x_l(r_l - g_{lk}^t) \geq x_l(r_l - \gamma_l)$$

and

$$x_1 r_1 \geq x_j r_j \geq x_j(r_j - \gamma_j) \ \forall j \neq l, j = 2, 3, ..., m.$$

This implies that $x \in A_1(\gamma)$. It follows that $A_1(g_{lk}^t e_l) \subseteq A_1(\gamma)$.

Next, for any $x \in A_1(g_{lk}^t e_l) \cap A_1(\beta)$, we must have $x_1 r_1 \geq x_l(r_l - \beta_l) \geq x_l(r_l - g_{l,k-1}^t)$ and $x_1 r_1 \geq x_j r_j$ for all $j \neq l$, $j = 2, 3, ..., m$. This implies $x \in A_1(g_{l,k-1}^t e_l)$. It follows that $A_1(g_{lk}^t e_l) \cap A_1(\beta) \subseteq A_1(g_{l,k-1}^t e_l)$.

In sum, we conclude that $A_1(g_{lk}^t e_l) \setminus A_1(g_{l,k-1}^t e_l) \subseteq A_1(\gamma) \setminus A_1(\beta)$.

□

Proof of Lemma 5.

Proof. Recall that

$$S_{lk}^t \equiv A_1(g_{lk}^t e_l) \setminus A_1(g_{lk-1}^t e_l).$$

Let $\gamma = g_{lk}^t e_l$ and $\beta = g_{lk-1}^t e_l$. Note that

$$\begin{aligned}
\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 d\vec{x} &\geq \frac{1}{m+1}\left(\frac{r_1}{r_l - \gamma_l} - \frac{r_1}{r_l - \beta_l}\right) \prod_{j=2, j\neq l}^{m} \frac{r_1}{r_j - \gamma_j} \\
&\geq \frac{1}{m+1}\left(\frac{r_1}{r_l - \gamma_l} - \frac{r_1}{r_l - (\gamma_l - 0.5t^{-3/4})}\right) \prod_{j=2, j\neq l}^{m} \frac{r_1}{r_j - \gamma_j} \\
&= \frac{1}{m+1}\left(\frac{r_1}{r_l - \gamma_l} - \frac{r_1}{r_l - \gamma_l + 0.5t^{-3/4}}\right) \prod_{j=2, j\neq l}^{m} \frac{r_1}{r_j - \gamma_j} \\
&= \frac{1}{m+1}\left(\frac{0.5 r_1 t^{-3/4}}{(r_l - \gamma_l)(r_l - \gamma_l + 0.5t^{-3/4})}\right) \prod_{j=2, j\neq l}^{m} \frac{r_1}{r_j - \gamma_j}
\end{aligned}$$

$$\begin{aligned}
&\geq \frac{1}{m+1}\left(\frac{0.5t^{-3/4}}{r_1+0.5t^{-3/4}}\right)\prod_{j=2,j\neq l}^{m}\frac{r_1}{r_j-\gamma_j}\\
&\geq \frac{1}{m+1}\left(\frac{0.5t^{-3/4}}{r_1+0.5t^{-3/4}}\right)\\
&= \frac{1}{m+1}\left(\frac{1}{2t^{3/4}r_1+1}\right)\\
&\geq Ct^{-3/4}
\end{aligned}$$

for all $t \geq t^*$, for some constant C and t^*. Above, the first inequality follows from Lemma 3; the second inequality from $\beta_l = \gamma_l - 0.5t^{-3/4}$; the third inequality from $\gamma_l \geq r_l - r_1$; and the fourth inequality from $\gamma_j \geq r_j - r_1$ for all $j > 1$.

Next, using Chernoff's bound, we have for all $t \geq t^*$,

$$\begin{aligned}
&P\Big(\frac{r_1}{\max_{j=1,2,\ldots,m} r_j}\cdot\sum_{i=1}^{n} p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma)\setminus A_1(\beta)] \leq 1\Big)\\
\leq & e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot\left(\mathbf{E}[e^{-p_{i1}^t\mathbf{1}[p_i^t\in A_1(\gamma)\setminus A_1(\beta)]}]\right)^n.
\end{aligned}$$

Further, it is easy to check that for any $x \in [0,1]$, we have $e^{-x} \leq 1-(1-e^{-1})x$. Thus,

$$\begin{aligned}
\mathbf{E}[e^{-p_{i1}^t\mathbf{1}[p_i^t\in A_1(\gamma)\setminus A_1(\beta)]}] \leq & \mathbf{E}[1-(1-e^{-1})p_{i1}^t\mathbf{1}[p_i^t\in A_1(\gamma)\setminus A_1(\beta)]]\\
= & 1-(1-e^{-1})\mathbf{E}[p_{i1}^t\mathbf{1}[p_i^t\in A_1(\gamma)\setminus A_1(\beta)]]\\
= & 1-(1-e^{-1})\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 f(\vec{x})d\vec{x}\\
\leq & 1-(1-e^{-1})\underline{f}\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 d\vec{x}\\
\leq & 1-(1-e^{-1})\underline{f}Ct^{-3/4}.
\end{aligned}$$

It follows that

$$\begin{aligned}
&P\Big(\frac{r_1}{\max_{j=1,2,\ldots,m} r_j}\cdot\sum_{i=1}^{n} p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma)\setminus A_1(\beta)] \leq 1\Big)\\
\leq & e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot\left(1-(1-e^{-1})\underline{f}Ct^{-3/4}\right)^n
\end{aligned}$$

$$=e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot \left(1-(1-e^{-1})\underline{f}Ct^{-3/4}\right)^{t\bar{n}}$$
$$=e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot \left(1-\frac{C_1}{t^{3/4}}\right)^{t\bar{n}}$$

for $C_1=(1-e^{-1})\underline{f}C$ and for $t \geq t^*$.

□

Proof of Theorem 7.

Proof. We have shown in Theorem 4 that when γ changes to β, the expected number of clickthroughs (22) (conditional on the random pool of users) leaving message j can be bounded using the change in the capacity values as follows:

$$\sum_{i=1}^{n} p_{ij}^t \mathbf{1}[p_i^t \in A_j(\gamma)\setminus A_j(\beta)] \leq \sum_{k=2}^{m} |c_k^t(\gamma)-c_k^t(\beta)| r_k/r_1.$$

Note that in the above bound we do not consider the change made to the capacity of message 1, because message 1 has infinite capacity and thus any finite change made to this capacity value has no impact on the assignment of users. The above bound leads to

$$\sum_{j=2}^{m} |c_j^t(\gamma)-c_j^t(\beta)| \geq \frac{r_1}{\max_{j=1,2,\ldots,m} r_j} \cdot \sum_{i=1}^{n} p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma)\setminus A_1(\beta)]. \tag{34}$$

Therefore, it suffices to show that with probability one,

$$\frac{r_1}{\max_{j=1,2,\ldots,m} r_j} \cdot \sum_{i=1}^{n} p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma)\setminus A_1(\beta)] > 1$$

for all sufficiently large t and all $\gamma,\beta \in \mathcal{G}$ with $\gamma_l \geq \beta_l + t^{-3/4}$ for some index l.

Based on Lemma 4, to prove the theorem, we just need to prove that there exists a finite random number t_0 such that $\mathcal{E}_{jk}^t$ holds for all $t \geq t_0$, $j=2,3,\ldots,m$, $k=1,2,\ldots,n_j^t$. By the Borel-Cantelli Lemma, it suffices to show that

$$\sum_{t=1}^{\infty} P\left(\overline{\cap_{j=2}^{m} \cap_{k=1}^{n_j^t} \mathcal{E}_{jk}^t}\right) < \infty.$$

Indeed,

$$
\begin{aligned}
P\left(\overline{\cap_{j=2}^{m}\cap_{k=1}^{n_j^t}\mathcal{E}_{jk}^t}\right) &= 1-P\left(\cap_{j=2}^{m}\cap_{k=1}^{n_j^t}\mathcal{E}_{jk}^t\right)\\
&\le 1-\prod_{j=2}^{m}\prod_{k=1}^{n_j^t}P\left(\mathcal{E}_{jk}^t\right) \text{ (because the events } \mathcal{E}_{jk}^t\text{'s are positively correlated)}\\
&= 1-\prod_{j=2}^{m}\prod_{k=1}^{n_j^t}(1-P(\overline{\mathcal{E}_{jk}^t}))\\
&\le 1-\prod_{j=2}^{m}\prod_{k=1}^{n_j^t}\left[1-e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot\left(1-C_1t^{-3/4}\right)^{t\bar{n}}\right] \text{ (by Lemma 5)}\\
&\le 1-\left[1-e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot\left(1-C_1t^{-3/4}\right)^{t\bar{n}}\right]^{C_2t^{3/4}}
\end{aligned}
$$

for an appropriately defined constant C_2. That the events $\mathcal{E}_{jk}^t$'s are positively correlated can be explained by the fact that $\mathcal{E}_{jk}^t$'s are independent if the corresponding sets S_{jk}^t do not overlap. On the other hand, they contain a common set of profiles if they do overlap.

We can check that when $t\to\infty$,

$$
\left(1-C_1t^{-3/4}\right)^{t\bar{n}} = e^{-C_1\bar{n}t^{1/4}+o(1)}
$$

$$
\begin{aligned}
\Longrightarrow &\left[1-e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot\left(1-C_1t^{-3/4}\right)^{t\bar{n}}\right]^{C_2t^{3/4}}\\
&=\left[1-\frac{e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}}{e^{C_1\bar{n}t^{1/4}+o(1)}}\right]^{C_2t^{3/4}}\\
&=\exp\left(-C_2t^{3/4}\cdot e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot e^{-C_1\bar{n}t^{1/4}}+o(e^{-C_1\bar{n}t^{1/4}})\right)\\
&=1-C_2t^{3/4}\cdot e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot e^{-C_1\bar{n}t^{1/4}}+o(e^{-C_1\bar{n}t^{1/4}})
\end{aligned}
$$

$$
\begin{aligned}
\Longrightarrow &1-\left[1-e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot\left(1-C_1t^{-3/4}\right)^{t\bar{n}}\right]^{C_2t^{3/4}}\\
&=C_2t^{3/4}\cdot e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot e^{-C_1\bar{n}t^{1/4}}+o(e^{-C_1\bar{n}t^{1/4}}).
\end{aligned}
$$

This proves that

$$\sum_{t=1}^{\infty} P\left(\overline{\cap_{j=2}^{m} \cap_{k=1}^{n_j^t} \mathcal{E}_{jk}^t}\right) < \infty.$$

□

Proof of Theorem 8.

First, we claim that with probability 1, there exists a finite random integer t_0 such that for all $t > t_0$, condition (14) holds when $\bar{\gamma}$ is set to be $\bar{\gamma}_t \equiv t^{-3/4}$ in the problem of size t.

Let t_0 be the random variable in Theorem 7. Suppose (14) does not hold for some $t > t_0$, i.e., with positive probability, for some $t > t_0$, we can find c, α, j, and k, such that

$$|\gamma_j^t(c) - \gamma_j^t(c + \alpha e_k)| > t^{-3/4}.$$

Then Theorem 7 implies that $\|c - (c + \alpha e_k)\|_1 > 1 \Longrightarrow \|\alpha e_k\|_1 > 1$, which is a contradiction.

Therefore, (14) holds for $t > t_0$ when $\bar{\gamma}_t = t^{-3/4}$. It follows that Theorem 2 holds with probability 1 for $t > t_0$. Let $U_t \equiv \sigma(p_1^t, p_2^t, ..., p_n^t)$ denote the random user pool for problem t. Then the regret (15) of the algorithm becomes, for $t > t_0$,

$$V_t^{LP} - V_t^{RA} \leq \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta(t)} + 3m^2 + nP(\bar{\mathcal{O}}_t|U_t) + \sqrt{nP(\bar{\mathcal{O}}_t|U_t)}\right] + m^2(n + \sqrt{n})t^{-3/4}.$$

Since

$$\Delta(t) = C \cdot \sum_{j=2}^{m} \sqrt{t\bar{c}_j} \cdot \log \sum_{j=2}^{m} t\bar{c}_j = O(\sqrt{t}\log t),$$

we have

$$\sum_{j=1}^{m} r_j \sqrt{m\Delta(t)} = O(t^{1/4}\log t).$$

Furthermore,

$$m^2(n + \sqrt{n})t^{-3/4} = O(t^{1/4}).$$

Define

$$Var(\delta_j^t|U_t) \equiv \mathbf{E}[(\delta_j^t)^2|U_t] - (\mathbf{E}[\delta_j^t|U_t])^2$$

as the variance of δ_j^t given U_t as input. According to the central limit theorem, when the standard deviation of δ_j^t becomes large, $\delta_j^t/\sqrt{Var(\delta_j^t|U_t)}$ approaches a standard normal distribution. Furthermore, since

$$Var(\delta_j^t|U_t) = \sum_{i=1}^n p_{ij}x_{ij}^{(1)}(1-p_{ij}x_{ij}^{(1)}) \le \sum_{i=1}^n p_{ij}x_{ij}^{(1)} \le c_j = \bar{c}_j t,$$

we must have, according to the central limit theorem,

$$P\left(\frac{|\delta_j^t|}{\sqrt{\bar{c}_j t}} > \bar{C}\log t \middle| U_t\right) = O(P(|Z_t| > \bar{C}\log t))$$

for any constant $\bar{C} > 0$ and for a sequence $Z_1, Z_2, Z_3, ...$ of i.i.d. standard normal random variables. It is easy to check that

$$\begin{aligned} P(|Z_t| > \bar{C}\log t) &= 2\cdot\frac{1}{\sqrt{2\pi}}\int_{\bar{C}\log t}^{\infty} e^{-x^2/2}dx \\ &< \frac{2}{\sqrt{2\pi}}\int_{\bar{C}\log t}^{\infty}\frac{x}{\bar{C}\log t}e^{-x^2/2}dx \\ &= \frac{2}{\sqrt{2\pi}}\frac{1}{\bar{C}\log t}e^{-(\bar{C}\log t)^2/2} \\ &= o(1/t), \end{aligned}$$

from which we can deduce that

$$\begin{aligned} &P\left(\frac{|\delta_j^t|}{\sqrt{\bar{c}_j t}} > \bar{C}\log t \middle| U_t\right) = o(1/t) \\ \Longrightarrow\ &P\left(\frac{|\delta_j^t|}{\sqrt{\bar{c}_j t}}\cdot\frac{r_j}{r_1} > \frac{r_j}{r_1}\bar{C}\log t \middle| U_t\right) = o(1/t) \\ \Longrightarrow\ &P\left(\sum_{j=2}^m \frac{|\delta_j^t|}{\sqrt{\bar{c}_j t}}\frac{r_j}{r_1} > \sum_{j=2}^m \frac{r_j}{r_1}\bar{C}\log t \middle| U_t\right) \le \sum_{j=2}^m P\left(\frac{|\delta_j^t|}{\sqrt{\bar{c}_j t}}\cdot\frac{r_j}{r_1} > \frac{r_j}{r_1}\bar{C}\log t \middle| U_t\right) = o(1/t) \\ \Longrightarrow\ &P\left(\sum_{j=2}^m |\delta_j^t|\frac{r_j}{r_1} > \left(\max_{j=2,\ldots,m}\sqrt{\bar{c}_j}\right)\sum_{j=2}^m \frac{r_j}{r_1}\bar{C}\sqrt{t}\log t \middle| U_t\right) = o(1/t). \end{aligned}$$

Then by choosing an appropriate value of $\bar{C}$, we can easily obtain

$$\left(\max_{j=2,\ldots,m}\sqrt{\bar{c}_j}\right)\sum_{j=2}^m \frac{r_j}{r_1}\bar{C}\sqrt{t}\log t = \Delta(t),$$

and hence,

$$P(\bar{\mathcal{O}}_t|U_t) = P(\sum_{j=2}^{m} |\delta_j^t| \frac{r_j}{r_1} > \Delta(t)|U_t) = o(1/t).$$

Thus, $nP(\bar{\mathcal{O}}_t|U_t) = t\bar{n}P(\bar{\mathcal{O}}_t|U_t) = o(1)$.

In sum, $V_t^{LP} - V_t^{RA} = O(t^{1/4}\log t)$ with probability 1. This proves the theorem because V_t^{LP} is an upper bound on V_t^{OPT}. □

Proof of Theorem 9.

We first focus on the analysis of a single problem of size t and suppress t in notation. Fix the set of user profiles. Let V^{Static} be the expected total reward of the static algorithm. We have

$$\begin{aligned}
V^{OPT} - V^{Static} &\leq V^{LP} - V^{Static} \\
&= \sum_{j=1}^{m} r_j [b_j - \mathbf{E}[\min(c_j, \sum_{i=1}^{n} I_{ij}^{Static})]] \\
&= \sum_{j=2}^{m} r_j [b_j - \mathbf{E}[\min(c_j, \sum_{i=1}^{n} I_{ij}^{Static})]] \\
&= \sum_{j=2}^{m} r_j \mathbf{E}[\max(b_j - c_j, b_j - \sum_{i=1}^{n} I_{ij}^{Static})]] \\
&\leq \sum_{j=2}^{m} r_j \mathbf{E}[\max(0, b_j - \sum_{i=1}^{n} I_{ij}^{Static})]] \\
&= \sum_{j=2}^{m} r_j \mathbf{E}[|b_j - \sum_{i=1}^{n} I_{ij}^{Static}|] \\
&\leq \sum_{j=2}^{m} r_j \sigma_j \\
&\leq \sum_{j=2}^{m} r_j \sqrt{c_j}.
\end{aligned}$$

In the asymptotic regime, since $c_j = t \cdot \bar{c}_j$, we have almost surely,

$$V_t^{OPT} - V_t^{Static} \leq \sum_{j=2}^{m} r_j \sqrt{c_j} = O(\sqrt{t}).$$

□

Proof of Theorem 10.

Proof. Again let U_t denote the random user pool for problem t. Fix some message $j \in \{2,3,...,m\}$. Conditional on U_t, let K_t be the number of clickthroughs that message j receives under the Static Algorithm.

Assume that $\bar{p} = 0.5$ in (21), i.e., all clickthrough probabilities are less than 0.5. Also assume that the capacity of each message $j \geq 2$ is fully allocated to users by the linear program (3) almost surely for all t. This condition can be ensured by having small capacities for all messages $j \geq 2$. Then we have for all large t, $\mathbf{E}[K_t|U_t] = \bar{c}_j t$. It is easy to see that we must have, conditional upon the random user pools,

$$V_t^{LP} - V_t^{Static} \geq r_j \mathbf{E}[\max(0, \bar{c}_j t - K_t)|U_t] = r_j \mathbf{E}[\max(0, \mathbf{E}[K_t] - K_t)|U_t]$$

with probability 1 for each t. Furthermore, let $Var(K_t|U_t) = \mathbf{E}[K_t^2|U_t] - (\mathbf{E}[K_t|U_t])^2$ denote the variance of K_t given input U_t. We must have

$$\sqrt{Var(K_t|U_t)} = \sqrt{\sum_{i=1}^{n} s_{ij}^* p_{ij}(1 - s_{ij}^* p_{ij})} \geq \sqrt{\sum_{i=1}^{n} s_{ij}^* p_{ij} 0.5} = \sqrt{\mathbf{E}[K_t|U_t]0.5} = \sqrt{\bar{c}_j t 0.5} = \Omega(\sqrt{t}).$$

When t is large, $K_t/\sqrt{Var(K_t|U_t)}$ approaches a normal random variable with standard deviation 1 almost surely. Thus, we must have almost surely

$$\mathbf{E}[\max(0, \mathbf{E}[K_t] - K_t)|U_t] = \Omega(\sqrt{Var(K_t|U_t)}) = \Omega(\sqrt{t})$$

$$\Longrightarrow V_t^{LP} - V_t^{Static} = \Omega(\sqrt{t}).$$

In the proof of Theorem 8, we have proved $V_t^{LP} - V_t^{RA} = o(\sqrt{t})$. This leads to $V_t^{RA} - V_t^{Static} = \Omega(\sqrt{t})$.

$\square$