Maximizing the Number of Satisfied Subscribers in Pub/Sub Systems Under Capacity Constraints

Vinay Setty
University of Oslo, Norway
vinay@ifi.uio.no

Gunnar Kreitz
Spotify, Stockholm, Sweden
KTH – Royal Institute of Technology, Sweden
gkreitz@{spotify.com,kth.se}

Guido Urdaneta
Spotify, Stockholm, Sweden
guidou@spotify.com

Roman Vitenberg
University of Oslo, Norway
romanvi@ifi.uio.no

Maarten van Steen
VU University and The Network Institute,
Amsterdam, The Netherlands
steen@cs.vu.nl

***Abstract*—Publish/subscribe (pub/sub) is a popular communication paradigm in the design of large-scale distributed systems. A provider of a pub/sub service (whether centralized, peer-assisted, or based on a federated organization of cooperatively managed servers) commonly faces two fundamental challenges: server provisioning and assignment of workload to a server.**

In this paper, we provide the first formal treatment of this subject to the best of our knowledge. Specifically, we introduce a measure of subscriber satisfaction that lends itself to a large class of pub/sub notification services where (a) publication event message delivery is best effort and (b) every notification is intended to be read by a human user so that having a cumulative delivery rate to a particular subscriber above certain threshold will not bring any benefit. For example, most applications where notifications are generated due to social interaction fall into this class of pub/sub services: following the tweets of selected users in Twitter, monitoring updates to the profiles of user's friends in Facebook, or receiving instant notifications related to favorite artists and albums in *Spotify*. We define two distinct subscriber satisfaction metrics suitable for such applications.

Then, we introduce a principal optimization problem: given a server with a limited capacity, and a workload consisting of (i) a set of topics each with its own publication event rate, and (ii) a set of subscribers with their interests, assign a subset of the workload to the server so as to maximize the cumulative satisfaction of the subscribers. We define two distinct flavors of the problem: a "Budgeted Maximum Multiset Multicover" (B3M) and "Fractional Budgeted Maximum Multiset Multicover" (F-B3M) using the two satisfaction metrics, respectively. We prove that both flavors are NP-Hard. We also show that while B3M does not admit a Polynomial-Time Approximation Scheme (PTAS) unless NP has randomized algorithms that run in sub-exponential time, F-B3M has a polynomial time approximation algorithm with a guaranteed constant ratio of $\frac{1}{2}\left(1-\frac{1}{e}\right)$. Furthermore, we derive an upper bound for the optimal solution of the problem.

We evaluate the proposed heuristics for B3M and F-B3M using a large-scale real data set from the pub/sub system of *Spotify*. We show that the heuristics provide up to 0.9 approximation for F-B3M and 0.82 approximation for B3M for the given dataset, using the derived upper bound on the optimal solution as the baseline. Finally, we propose various optimizations to make the heuristics more efficient and we show that the heuristics can be computed within under 30 seconds in most of the experiments.

I. Introduction

We are witnessing an increasingly widespread use of the publish/subscribe (pub/sub) communication paradigm in the design of large-scale distributed systems. Pub/sub is regarded as a technology enabler for a loosely coupled form of interaction among many publishing data sources and many subscribing data sinks. Many applications report benefits from using this form of interaction, such as online delivery of notifications due to social interaction [21], application integration [20], financial data dissemination [1], RSS feed distribution and filtering [18], [19], and business process management [17]. As a result, many industry standards have adopted pub/sub as part of their interfaces. Examples of such standards include WS Notifications, WS Eventing, the OMG's Real-time Data Dissemination Service, and the Active Message Queuing Protocol.

In this paper, we focus on the topic-based pub/sub model. In a topic-based system, publication events are associated with topics, and subscribers register their interests in receiving all events published to topics of interest.

While traditional pub/sub implementations are either centralized or based on a federated organization of cooperatively managed servers, an increasingly higher number of pub/sub applications are being deployed in P2P environments [23]. In particular, the pub/sub service at *Spotify* [21] is inherently suitable for a peer-assisted implementation, in line with the reported peer-assisted implementation of other *Spotify* services such as music streaming [16]. In a peer-assisted implementation, a limited number of servers are providing a guaranteed high-quality service to a subset of pub/sub subscribers while the rest of subscribers are receiving notifications through the peers, thereby getting a best-effort service that works convincingly well in practice. The part of the workload assigned to a server is dictated by maximizing server utilization as well as the overall quality of service given to the subscribers.

In addition to assigning load to a server, a provider of a pub/sub service (whether centralized or peer-assisted) faces another fundamental challenge: resource provisioning. How many clients can receive an improved service if a new server is purchased? How does the capacity of the new server affect this number of clients? How to strike the balance between maximizing the quality of service and the number of clients to which a higher quality service is provided?

In this paper, we provide the first formal treatment of this subject to the best of our knowledge. Specifically, we introduce a measure of subscriber satisfaction that lends itself to a large class of pub/sub notification services where (a) publication event message delivery is best effort: reliable delivery is desirable but it is not mandatory to deliver all notifications and (b) every notification is intended to be read by a human user so that having a cumulative delivery rate to a particular subscriber above certain threshold will not bring any benefit and may only annoy the user. For example, most applications where notifications are generated due to social interaction fall into this class of pub/sub services: following the tweets of selected users in Twitter, monitoring updates to the profiles of user's friends in Facebook, or receiving instant notifications related to favorite artists and albums in *Spotify*. According to the binary satisfaction metric, we consider a subscriber satisfied in such applications if and only if the user receives all notifications of interest up to a configurable threshold delivery rate. We also provide a fractional satisfaction metric: If a subscriber receives fewer notifications than desired, the satisfaction of the subscriber is defined as a fraction of the actual and desired number of notifications.

Then, we introduce a principal optimization problem: given a server with a limited capacity, and a workload consisting of (i) a set of topics each with its own publication event rate, and (ii) a set of subscribers with their interests, assign a subset of the workload to the server so as to maximize the cumulative satisfaction of the subscribers. We define two distinct flavors of the problem: a **"Budgeted Maximum Multiset Multicover"** (B3M) and **"Fractional Budgeted Maximum Multiset Multicover"** (F-B3M) using the binary and fractional satisfaction metrics, respectively. We prove that both flavors are NP-Hard. We also show that while B3M does not admit a Polynomial-Time Approximation Scheme (PTAS) unless NP has randomized algorithms that run in sub-exponential time, F-B3M has a polynomial-time approximation algorithm with a guaranteed constant ratio of $\frac{1}{2}\left(1-\frac{1}{e}\right)$. Furthermore, we derive an upper bound for the optimal solution of each problem.

We evaluate the proposed heuristics for B3M and F-B3M using a large-scale real data from the pub/sub system of *Spotify*. We show that the heuristics provide up to 0.9 approximation for F-B3M and 0.82 approximation for B3M for the given dataset, using the derived upper bound on the optimal solution as the baseline. Finally, we propose various optimizations to make the heuristics more efficient and we show that the heuristics can be computed within under 30 seconds in most of the experiments.

II. Model and Notations

In this paper we use well known topic-based pub/sub model. However, our model considers a more practical case where different topics have different publication rates and different popularity in terms of number of subscribers. We use the following notations:

T	A collection of l *topics* $\{t_1, t_2, ..., t_l\}$ in the system.
V	A collection of n *subscribers* $\{v_1, v_2, ..., v_n\}$ participating in the pub/sub system. A subscriber can subscribe to one or more topics from T. Subscribers in a typical pub/sub system are generally the end user applications (e.g. *Spotify* client software). In the rest of the paper we use subscribers and users interchangeably.
T_v	The *interest* of subscriber v, that is, the collection of of topics subscribed by v.
Int	The collection of interests $\{T_{v_1}, T_{v_2}, ..., T_{v_n}\}$ for all subscribers in V.
ev_t	*Event rate* of the publications generated for a topic t, that is, mean of events published to topic t during a given period (e.g., per minute or per hour). Without loss of generality we assume that $ev_t > 0$. When we say 'event' in the rest of the paper we mean a publication event message generated by the back-end service for a topic intented for all subscribers of the topic.

III. Motivating Application Scenario: Pub/Sub at Spotify

At *Spotify*, the pub/sub system responsible for managing subscriptions and delivering publication events is implemented as a back-end service service running on *Spotify*'s data centers. Details about *Spotify*'s pub/sub system have been presented in [21]. In this paper, we propose a methodology to select a fraction of the pub/sub workload such that the fraction is within the capacity of a back-end service with limited resources, while user satisfaction is maximized. This approach can help system managers to deal with the trade-off between deploying additional hardware and satisfying more users. It can also be used as a mechanism to drop or offload part of the pub/sub workload to an external lower-cost system with lower quality of service, such as a pool of lower-reliability servers, or a set of computers belonging to end users (peers) forming a peer-to-peer network. In this regard, we propose adding a new component called *Offloading Decision Service* (ODS) to the existing architecture.

The ODS is responsible for deciding what part of the pub/sub load can be handled by the pub/sub back-end service, given a capacity constraint on the resources available to that service, and what part should be offloaded to an external system. The ODS should also take into account that the characteristics of the workload can change over time, so load-assignment decisions need to be revised periodically. Therefore, an additional requirement on the ODS is that is should employ light-weight algorithms that can be executed relatively quickly.

In order to perform its work, the ODS divides the total pub/sub load on a per-topic basis and then decides for each topic whether the topic can be managed by the back-end service without exceeding the capacity. In this context, managing a topic means taking care of delivering the corresponding topic events for all subscribers of that topic. The rationale for this design decision is that we believe that organizing the pub/sub load at this granularity level greatly simplifies system design compared to an approach based on dealing with each (topic, subscriber) subscription pair individually, without seriously affecting the ability of the system to offload the back-end service in a manner that suits the applications for which it will be employed.

IV. Problem Definitions

The two QoS metrics mentioned in Section I prompt problems which are similar in nature but, as we will show later in Sections V and VI, they are very different in hardness. In the first QoS metric, we are interested in maximizing the number of subscribers receiving at least τ (satisfaction threshold) events related to them from the back-end service. A subscriber is considered satisfied if and only if at least τ relevant events are received. This definition of user satisfaction is suitable for applications with events that are relatively infrequent but important for the user. *Spotify* updates about favorite albums and artists fall in this category. In this regard, we define a problem coined *Budgeted Maximum Multiset Multicover* (*B3M*) in this section. In Section V we analyze the hardness of *B3M* and propose a feasible heuristic.

Unlike the first QoS metric, in the second QoS metric every event notification received up to a limit by a subscriber is considered beneficial by itself. We quantify the amount of benefit towards the satisfaction of a subscriber with a fraction of events received by the subscriber relative to the given satisfaction threshold of τ. The goal is to maximize the total benefit achieved by selecting a sub-collection of topics from a given collection of topics. This definition suits better applications where events are frequent but of relatively low importance. An example would be *Spotify*'s updates about the activities of the friends of each given user. In this regard we define the *Fractional Budgeted Maximum Multiset Multicover* (*F-B3M*) problem. In Section VI we analyze the hardness of *F-B3M* and propose a feasible heuristic that also gives a guarantee on the quality of the output.

In both flavors of the problem we want to ensure that the computational cost to serve those events by the back-end service does not exceed a given limit on the computational resources at the back-end service.

Before we define the problem more formally, we introduce the following notations:

τ	A system parameter that represents the *satisfaction threshold* for a subscriber. It is defined as a constant specifying the number of events to be delivered to a subscriber by the back-end service in order for the subscriber to be considered satisfied. The period over which the events are to be delivered is the same as the time unit of ev_t. In the rest of the paper when we say a subscriber is *covered* x times we mean that a subscriber is set to receive exactly x events related to the subscriber from the back-end service in a given period of time.
τ_v	Subscriber-specific satisfaction threshold. In practice, the total event rate of the topics subscribed to by a subscriber is sometimes less than τ. In such cases we need to serve all the events the subscriber is interested in to meet the satisfaction threshold. It is mathematically expressed as follows: $\tau_v = min(\tau, \sum_{t \in T_v} ev_t)$.
V_t	$V_t \subseteq V$ is a non-empty set of subscribers to topic t. Given *Int*, V_t can be derived trivially.
$cost(t)$	Represents the non-zero cost of serving a topic t by the back-end service. We say that the cost of a topic is *normalized* if it costs 1 per event sent by the server related to the topic and hence, *normalized* cost is defined as $cost(t) = ev_t \cdot \lvert V_t \rvert$.
$\mathcal{C}$	Capacity of the back-end service. A constant to quantitatively represent the amount of resources available to the back-end service. $\mathcal{C}$ has same unit as *cost*.
$\mathcal{S}$	Solution ($\mathcal{S} \subseteq T$). It is a set of topics that can be served by the back-end service with a cost that does not exceed a given computational resource constraint expressed by the constant $\mathcal{C}$.
$\sigma(\mathcal{S})$	Represents the sum of the satisfaction for all the subscribers, given a potential solution $\mathcal{S}$. We want to maximize this function.

A. *The problem of* ***Budgeted Maximum Multiset Multicover*** *(***B3M***):*

Given an instance of T, V and their interests *Int*, the goal of the *B3M*$(T, V, ev, cost, Int, \tau, \mathcal{C})$ problem is to find $\mathcal{S} \subseteq T$ so as to maximize the objective function defined below:

$$\text{Maximize} \quad \sigma(\mathcal{S}) = \sum_{v \in V} f(v), \quad \text{Subject to:} \sum_{t \in \mathcal{S}} cost(t) \leq \mathcal{C} \tag{1}$$

$f(v)$ is a function that indicates if subscriber v is receiving a number of events that meets the satisfaction threshold:

$$f(v) = \begin{cases} 1 & \text{if } \sum_{\{t \in \mathcal{S} \cap T_v\}} ev_t \geq \tau_v \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The first condition in the Equation (2) is the case when a subscriber v is receiving publication events at a rate not lower than τ_v. In order for v to contribute to the objective function $f(v)$, the solution $\mathcal{S}$ must include enough topics subscribed by v with a total event rate of at least τ_v.

B. *Fractional*-B3M

We now define a relaxed version of the *B3M* problem in which we quantify the satisfaction relative to the number of events covered for a subscriber v out of τ_v events. Given an instance of T, V and their interests *Int*, the goal of the *F-B3M*$(T, V, ev, cost, Int, \tau, \mathcal{C})$ problem is to find $\mathcal{S} \subseteq T$ so as to maximize the sum of the fractions for all the subscribers.

$$\text{Maximize} \quad \sigma(\mathcal{S}) = \sum_{v \in V} g(v), \qquad \text{Subject to:} \sum_{t \in \mathcal{S}} cost(t) \leq \mathcal{C} \tag{3}$$

$g(v)$ is the fraction of events subscriber v receives, and it is defined as:

$$g(v) = \begin{cases} 1 & \text{if } \sum_{\{t \in \mathcal{S} \cap T_v\}} ev_t \geq \tau_v \\ \frac{\sum_{\{t \in \mathcal{S} \cap T_v\}} ev_t}{\tau_v} & \text{Otherwise} \end{cases} \tag{4}$$

The difference between *B3M* and *F-B3M* lies in the definition of the satisfaction metrics in Equation (2) and Equation 4 respectively. In Equation 2 the satisfaction is defined in a binary fashion i.e. we the satisfaction is 0 when less than τ_v events are received by the subscriber and 1 otherwise. On the other hand in Equation (4) a fraction of events received up to τ_v is considered instead of a binary 1 or 0. This subtle difference makes the two problems fundamentally different in terms of difficulty of solving. We explore this in detail in Section V and Section VI.

V. Hardness of *B3M* and its Solution Approach

In this section we prove that *B3M* is NP-Hard and we also show that *B3M* has no Polynomial-Time Approximation Scheme (PTAS). We further propose an algorithm to give an upper bound on *B3M* instances. We later use this bound to evaluate a greedy heuristic we propose in Section VII-B.

A. *Hardness of* B3M *problem*

To establish the hardness of *B3M* we prove that the well-known hard problem of Densest-k-Subgraph (*DkS*) can be reduced to a special case of *B3M.* We now define the *DkS* problem and an auxiliary unit-cost version of *B3M.*

Definition V.1 (Densest-k-Subgraph)**.** Given an undirected graph $G(U, E)$ the Densest-k-Subgraph ($DkS(U, E, k)$) problem on G is the problem of finding a subset $U' \in U$ of vertices of size $|U'| = k$ with the maximum induced average degree. The average degree of the optimal subgraph is $2|E(U')|/k$. Here $|E(U')|$ denotes the number of edges in the subgraph induced by U'.

The *DkS* problem can be proven to be NP-Hard by reduction from the Max-Clique problem [8]. In [9] it has been shown that *DkS* is also NP-Hard even when restricted to a maximum degree of 3. The best known approximation algorithm achieves a ratio of $\mathrm{O}\left(n^{1/4+\epsilon}\right)$ and runs in $2^{n^{O(1/\epsilon)}}$ time, for any $\epsilon > 0$ [2]. On the other hand, it is known that *DkS* does not admit a PTAS [13].

Definition V.2 (UC-*B3M*)**.** We define an auxiliary problem coined Unit-Cost-*B3M*(*UC-B3M*) which is a restricted version of *B3M.* We define *UC-B3M* to be an instance of *B3M* with unit cost for all the topics $\forall t \in T$: $cost(t) = 1$ and unit event rate $ev_t = 1$, each subscriber subscribes to exactly two topics $\forall v \in V : |T_v| = 2$, no two subscribers subscribe to same set of topics $\forall v_1 \neq v_2 : T_{v_1} \neq T_{v_2}$ and the satisfaction threshold $\tau_v = 2$.

Lemma V.3. UC-B3M *is NP-Hard*

Proof: Given an instance of $DkS(U, E, k)$ we construct an instance of *UC-B3M*$(T, V, ev, cost, Int, \tau, \mathcal{C})$ in the following way: we take T with topics that one-to-one correspond to the vertices in the set U. We take V to one-to-one correspond to the edges in the set E. We build *Int* from the edges incident on the vertices. For example, V_t corresponds to the edges incident on the corresponding vertex in U. We set $\mathcal{C} = k$. We now prove that there is an induced subgraph of $A(U', E')$ with average degree δ and exactly k vertices if and only if there is a solution $\mathcal{S}$ to *UC-B3M* with value at least $|E(U')|$ (i.e., the total number of edges in the induced subgraph).

To see this, we observe that a subscriber in our *UC-B3M* instance only contributes to the objective function if both of her topics are included in $\mathcal{S}$. This precisely corresponds to the condition if and only if that exact edge with the vertices corresponding to those two topics is in the induced subgraph of the *DkS* instance. We can, without loss of generality, assume that $\mathcal{S}$ contains precisely k topics as the cost of each topic is 1 and the objective function is non-decreasing in the number of selected topics.

As we know that *DkS* is NP-Hard [8], it follows that *UC-B3M* is NP-Hard too. ∎

Theorem V.4. B3M *is NP-Hard.*

Proof: *UC-B3M* is a special case of *B3M.* From Lemma V.3 we know that *UC-B3M* is NP-Hard and hence *B3M* is NP-Hard too. ∎

Corollary V.5. *Assuming* NP $\not\subseteq \cap_{\epsilon>0}$BPTIME$(2^{n^{\epsilon}})$, *there is no Polynomial-Time Approximation Scheme (PTAS) for* B3M.

Proof: The statement follows directly for *UC-B3M* from the reduction given in Lemma V.3 together with a result by Khot [13] saying that unless NP has randomized algorithms that run in sub-exponential time (more formally: NP $\subseteq \cap_{\epsilon>0}$BPTIME$(2^{n^\epsilon})$) there is no PTAS for *DkS*. As *UC-B3M* is a special case of *B3M*, the statement also holds for *B3M*. ■

B. *Greedy heuristic for* **B3M**

The idea behind the greedy algorithm to solve *B3M* is to choose a topic t that maximizes the ratio between its total contribution towards the objective function relative to the already chosen topics $\mathcal{S}'$, and its cost. To define this contribution of a topic, we firstly say that it is the change of the objective function resulting from choosing the topic (i.e., contributing the value 1 for each subscriber "finished" by the topic). This is a very coarse metric, so we secondly also add a contribution for each subscriber who receives new events from t and does not already receive τ_v events from previously selected topics. This addition we have defined as the number of events divided by how many more events the user requires to receive τ_v events. The intuition behind this choice is that it steers the heuristic towards "finishing" subscribers, rather than spending resources to "start" a large number of subscribers by giving them a few, but not τ_v events. In Algorithm 1 we show this in line 3 by computing the number of events a user needs to receive to get to τ_v. We then, in line 5, add the value 1 if ev_t exceeds this value, or the partial contribution otherwise.

The pseudocode of the greedy algorithm to solve *B3M* is sketched in Algorithm 2 and the greedy strategy is to choose a topic that maximizes the objective function. In lines 2 and 3 an array containing the profit-to-cost ratio of the individual topics is initialized using Algorithm 1. In practice, this array can be a max-heap structure optimized for obtaining elements with maximum value. A topic that maximizes the profit-to-cost ratio in each iteration is selected in Line 5. The topic is added to the solution if its addition keeps the cost of the solution within the budget. Otherwise the topic is ignored. If the topic is added to the solution, the profit-to-cost ratio of all the topics not selected so far are updated based on the current solution set $\mathcal{S}'$ (lines 9 and 10). $V_t \cap V_{t'}$ is the set of subscribers common to subscribers of t and subscribers of t'. The algorithm terminates when it has considered all the available topics, or when all subscribers have been covered in which case the profit-to-cost ratio of all the topics would be 0 (line 6).

Theorem V.6. *The run time complexity of Algorithm 2 is* $\mathrm{O}\big(|T|^2(|V| + \log|T|)\big)$.

Proof: Refer to Appendix C. ■

Theorem V.6 gives the worst-case run time complexity, the cost being dominated by updating the cost for all topics in lines 9 and 10 of Algorithm 2 when a topic is added to the solution. We remark that in practice, the code runs significantly faster than this bound would imply.

Algorithm 1: Heuristic value of topic t given partial solution $\mathcal{S}'$

```
1  GetHeuristicB3M($t, ev, cost(t), Int, \mathcal{S}', \tau$)
   Input: $t, ev, cost(t), Int, \mathcal{S}', \tau$
   Data: $h \leftarrow 0$ : Heuristic value
   $rem_v \leftarrow 0$: Events remaining to make user $v$ happy
2  foreach $\{v \in V_t\}$ do
3      $rem_v \leftarrow \tau_v - \sum_{\{t' \in \mathcal{S}' \cap T_v\}} ev_{t'}$
4      if $rem_v > 0$ then
5          $h \leftarrow h + \min\left(1, \frac{ev_t}{rem_v}\right)$
6  return $\frac{h}{cost(t)}$
```

Algorithm 2: Greedy solution for ***B3M***

```
1  GreedyB3M($T, V, ev, cost, Int, \tau, \mathcal{C}$)
   Input: $T, V, ev, cost, Int, \tau, \mathcal{C}$
   Data: $A$ : Array of size $l$
   Result: $\mathcal{S}' \leftarrow \emptyset$ : Output set of topics
2  foreach $t \in T$ do
3      $A[t] \leftarrow$ GetHeuristicB3M($t, ev, cost(t), Int, \mathcal{S}', \tau$)
4  while $T \neq \emptyset$ do
5      $t \leftarrow \mathrm{argmax}_{\{t' \in T\}} A[t']$
6      if $A[t] = 0$ then break
7      if $cost(t) + \sum_{t' \in \mathcal{S}'} cost(t') \leq \mathcal{C}$ then
8          $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{t\}$
9          foreach $\{t' : V_t \cap V_{t'} \neq \emptyset \wedge t' \notin \mathcal{S}'\}$ do
10             $A[t'] \leftarrow$
               GetHeuristicB3M($t', ev, cost(t'), Int, \mathcal{S}', \tau$)
11     $T \leftarrow T \setminus \{t\}$
12 return $\mathcal{S}'$
```

One of the reasons being that the number of updates is bounded by $max_{t \neq t'} |V_t \cap V_{t'}|$, which is usually significantly lower than $|T|$.

We now turn to the subject of computing an upper bound on the optimal solution. For this analysis, we only consider the case when the cost function is normalized, i.e., $cost(t) = ev_t \cdot |V_t|$.

Theorem V.7. *Given an instance* B3M*($T, V, ev, cost, Int, \tau, \mathcal{C}$) where the costs are normal-*

Algorithm 3: Upper bound for ***B3M*** with normalized topic costs

```
1  GetUpperBound($V, T, ev, Int, \mathcal{C}, \tau$)
   Input: $V, T, ev, Int, \mathcal{C}, \tau$
   Data: $C$ : Array of size $n$
   $csubs \leftarrow \emptyset$ : Set of subscribers covered
2  foreach $\{v \in V\}$ do
3      $C[v] \leftarrow \max(\tau_v, \min_{t \in T_v} ev_t)$
4  while $V \neq \emptyset$ do
5      $v \leftarrow \mathrm{argmin}_{\{v' \in V\}} C[v']$
6      if $C[v] + \sum_{v' \in csubs} C[v'] \leq \mathcal{C}$ then
7          $csubs \leftarrow csubs \cup \{v\}$
8      $V \leftarrow V \setminus \{v\}$
9  return $|csubs|$
```

ized, for any solution $\mathcal{S}$ it holds that:

$$\sigma(\mathcal{S}) \leq \max\left\{ |V'| : V' \subseteq V, \sum_{v \in V'} \max\left(\tau_v, \min_{t \in T_v} ev_t\right) \leq \mathcal{C} \right\}$$

Proof (Sketch): With normalized costs, one can see that the amortized cost to cover each subscriber v is at least τ_v. The cost is also bounded by the lowest event rate of any event in which the subscriber is interested. ■

Theorem V.7 presents a way to compute an upper bound on the optimal solution. Since Algorithm 2 gives an unbounded approximation ratio, we make use of Theorem V.7 to evaluate how well our proposed heuristic performs on real-world inputs (see Section VII-B). This theorem can be readily turned into an algorithm as shown in Algorithm 3. In lines 2 and 3 the minimum cost to consider a subscriber satisfied is initialized in an array. Then, in each iteration the subscriber with the least cost is selected until there is no more budget left to cover more subscribers (lines 4 to 8). Finally, the number of selected subscribers is returned as the upper bound for the optimal solution (line 9).

VI. Hardness of *F-B3M* and its Solution Approach

In this section we analyze the hardness of *F-B3M*. Comparing to the results we obtained for *B3M*, the direct reduction we did from Densest-k-Subgraph no longer works as in that case it is imperative that we are not “paid” for a partially satisfied subscriber. This also means that the approximation-resistance results obtained for *B3M* do not translate. For *F-B3M*, we are instead able to give a greedy approximation algorithm with an approximation ratio of $\frac{1}{2}\left(1 - \frac{1}{e}\right)$. *F-B3M* is still NP-Hard, which we first prove by a reduction from the (unweighted) *Maximum Coverage problem* [11].

Theorem VI.1. F-B3M *problem is NP-Hard.*

Proof: By reduction from Maximum Coverage. Refer to Appendix. A. ■

A. Greedy Heuristic

Theorem VI.2. *The objective function in the* F-B3M *problem from Expression* (3) *is a **submodular** function.*

Proof: Refer to Appendix B, where we also define submodularity. ■

From Theorem VI.2 we infer that the *F-B3M* problem is essentially the budgeted maximization of a submodular function. The generalized greedy heuristic for maximization of submodular functions is known to guarantee a constant approximation factor as shown in [10]. Unfortunately, greedily selecting topics with best profit-to-cost ratio for a budgeted maximization of a submodular function no longer gives a constant approximation guarantee. Greedily choosing the topics to maximize the profit-to-cost ratio similarly to the solution for *B3M* performs arbitrarily poorly.

Algorithm 4: Heuristic value of topic t given partial solution $\mathcal{S}'$

```
1 GetHeuristicFB3M($t, ev, Int, \mathcal{S}', \tau$)
  Input: $t, ev, Int, \mathcal{S}', \tau$
  Data: $h \leftarrow 0$ : Heuristic value
  $rem_v$ : Events remaining to make user $v$ happy
2 foreach $\{v \in V_t\}$ do
3     $rem_v \leftarrow \tau_v - \sum_{\{t' \in \mathcal{S}' \cap T_v\}} ev_{t'}$
4     if $rem_v > 0$ then
5         $h \leftarrow h + \frac{\min(rem_v, ev_t)}{\tau_v}$
6 return $h$
```

Algorithm 5: Appropriate simple greedy algorithm for *F-B3M*, given a type

```
1  GreedyFB3M($T, V, ev, cost, Int, \tau, \mathcal{C}, type$)
   Input: $T, V, ev, cost, Int, \tau, \mathcal{C}, type$
   Data: $A$ : Array of size $l$
   Result: $\mathcal{S}' \leftarrow \emptyset$ : Output set of topics
2  foreach $t \in T$ do
3      $A[t] \leftarrow$ ComputeHeuristic($t, ev, cost(t), Int, \mathcal{S}', \tau, type$)
4  while $T \neq \emptyset$ do
5      $t \leftarrow \text{argmax}_{\{x \in T\}} A[x]$
6      $T \leftarrow T \setminus \{t\}$
7      if $cost(t) + \sum_{t' \in \mathcal{S}'} cost(t') \leq \mathcal{C}$ then
8          $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{t\}$
9          repeat
10             $t' \leftarrow t$
11             $t \leftarrow \text{argmax}_{\{x \in T\}} A[x]$
12             $A[t] \leftarrow$
               ComputeHeuristic($t, ev, cost(t), Int, \mathcal{S}', \tau, type$)
13         until $A[t'] = A[t]$
14 return $\mathcal{S}'$
```

Algorithm 6: Appropriate heuristic, given a *type*

```
1 ComputeHeuristic($t, ev, cost(t), Int, \mathcal{S}', \tau, type$)
  Input: $t, ev, cost(t), Int, \mathcal{S}', \tau, type$
2 if $type = \mathcal{G}$ then
3     return GetHeuristicFB3M($t, ev, Int, \mathcal{S}', \tau$)
4 else if $type = \mathcal{R}$ then
5     return GetHeuristicFB3M($t, ev, Int, \mathcal{S}', \tau$)$/cost(t)$
```

Algorithm 7: Greedy algorithm for ***F-B3M***

```
1 ModifiedGreedyFB3M($T, V, ev, cost, Int, \tau, \mathcal{C}$)
  Input: $T, V, ev, cost, Int, \tau, \mathcal{C}$
2 $\mathcal{S}' \leftarrow$ GreedyFB3M($T, V, ev, cost, Int, \tau, \mathcal{C}, \mathcal{G}$)
3 $\mathcal{S}'' \leftarrow$ GreedyFB3M($T, V, ev, cost, Int, \tau, \mathcal{C}, \mathcal{R}$)
4 if $\sigma(\mathcal{S}') \geq \sigma(\mathcal{S}'')$ then return $\mathcal{S}'$ else return $\mathcal{S}''$
```

To see why the simple greedy approach fails, consider an instance with two topics t_1 and t_2 with $\sigma(t_1) = 1$ and $cost(t_1) = 1$ and $\sigma(t_2) = x$ for some $x > 1$ and $cost(t_2) = x + 1$ and with a $\mathcal{C}$ of $x + 1$. The heuristic of profit-to-cost ratio prefers t_1 against t_2. Having spent a budget of 1 the heuristic can no longer select t_2 and hence terminates with the result $\sigma(t_1) = 1$ while the optimal solution is choosing t_2 with the gain $\sigma(t_2) = x$ giving an approximation ratio of x.

Taking inspiration from [14], we address this problem by running two instances of a greedy algorithm, each using a different heuristic. The first algorithm, which we refer to as being of *type* $\mathcal{G}$, uses σ as shown in Algorithm 4. The second algorithm, of *type* $\mathcal{R}$, uses the profit-to-cost ratio ($\sigma/cost(t)$). The final solution is the best of the two solutions provided by executing the algorithms of type $\mathcal{G}$ and $\mathcal{R}$, respectively. The pseudocode of the simple greedy algorithm is shown in Algorithm 5. Algorithm 7 is the pseudocode for the modified greedy algorithm to solve the *F-B3M* problem that executes the simple greedy algorithms of *type* $\mathcal{G}$ and $\mathcal{R}$ and selects the best solution.

Our simple greedy algorithm (Algorithm 5) includes an optimization that is important in practice, but does not affect the worst-case run time. After selecting a topic, the contribution of other topics needs to be updated. Here we observe that, due to submodularity, the contribution of those topics can only decrease. Thus, we loop over the sorted list of topics in descending order of value and stop updating as soon as the top one does not change. This is done in lines 9 to 12.

B. *Performance of the modified greedy algorithm for* F-B3M

Theorem VI.3. *Algorithm 7 has an approximation ratio of* $\frac{1}{2}\left(1 - \frac{1}{e}\right)$.

Proof: A general result for budgeted maximization of submodular functions was given by Krause and Guestrin [15][Theorem 1]. Our Algorithm 7 is a minor extension of theirs, the difference being that they only select a single element when $type = \mathcal{G}$. ∎

We remark that following [15] one can also create a greedy heuristic with an approximation ratio of $1 - \frac{1}{e}$ at the cost of an additional factor of $|T|^3$ in the run time of the algorithm.

Theorem VI.4. *Given an instance* B3M$(T, V, ev, cost, Int, \tau, \mathcal{C})$ *where costs are normalized, for any solution* $\mathcal{S}$ *it holds that:*

$$\sigma(\mathcal{S}) \leq \max\left\{|V'| : V' \subseteq V, \sum_{v\in V'} \max\left(\tau_v, \min_{t\in T_v} ev_t\right) \leq \mathcal{C}\right\} + 1$$

Note that Theorem VI.4 is an extension of Theorem V.7 with a minor difference in that there may be a fractional contribution to the objective function. This fractional part is upper bounded by 1.

Theorem VI.5. *Algorithm 7 has run time complexity of* $O\left(|T|^2(|V| + \log|T|)\right)$.

Proof: Refer to Appendix C. ∎

VII. Evaluations

A. *Experimental Setup*

We implemented both GreedyB3M and Modified-GreedyFB3M using C++. To evaluate these heuristics we make use of real data from *Spotify*'s deployed pub/sub system. The data consists of about 1.1 million topics, and 4.9 million subscribers. The traces were gathered for 10 days from *Spotify*'s data center at Stockholm. For more information about the data traces refer to [21]. We use the normalized cost function: for each topic $cost(t) = ev_t \cdot |V_t|$. To choose $\mathcal{C}$ we analyzed the full data traces and computed the total capacity needed to handle the full traces in terms of total cost of all the topics $\sum_{t\in T} cost(t)$. For evaluations in this paper we set the capacity constraint $\mathcal{C}$ to be to be 10% of this sum. For τ we use 0.1%(2) to 100%(2763) of the mean event rate of all the topics. All experiments were executed single threaded on a server with 16 cores of Intel Xeon 2.13GHz processors and 32 GB of main memory.

B. *Performance of GreedyB3M*

First we analyze the performance of GreedyB3M (Algorithm 2) comparing it to the upper bound computed by GetUpperBound (Algorithm 3). To visualize the performance we observe that both algorithms iteratively construct solutions. Thus, in Figure 1 we show the progress of the GreedyB3M algorithm after selecting a topic in each iteration, by comparing the service capacity used so far (x-axis) against the number of satisfied subscribers (for a given τ) (y-axis) by the chosen topics. Note that this represents a single run of GreedyB3M until a budget $\mathcal{C}$ of 10% of the workload is reached. However, the intermediate results are equivalent to having stopped GreedyB3M at the corresponding values of $\mathcal{C}$. We can see that the gap between GreedyB3M and the upper bound increases as $\mathcal{C}$ also increases in most cases when the $\mathcal{C}$ is restricted to 10%.

An interesting observation is that, with a capacity constraint $\mathcal{C}$ equivalent to 10% of what is needed for the full workload, the gap between GreedyB3M and the upper bound increases as τ increases from 2 to 276 (the approximation ratio drops from 0.87 to 0.75, as shown in Figure 3.) However this changes when τ is increased to 2763, in which case the approximation ratio of GreedyB3M increases from 0.75 to 0.82. $\tau \approx 27$ is a reasonably realistic value. With this parameter we satisfy around 72% of all subscribers (3.5 million of the total 4.9 million). The upper bound gives that at most 86% (4.2 million) of subscribers can be satisfied, giving an approximation ratio of around 0.83.

C. *Performance of ModifiedGreedyFB3M*

We now analyze the performance of Modified-GreedyFB3M. From the theoretic results, we know

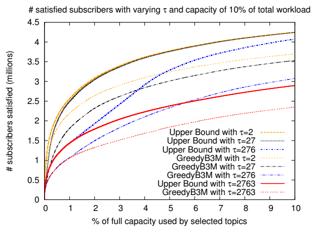

Fig. 1. Comparison of GreedyB3M with the Estimated Upper Bound

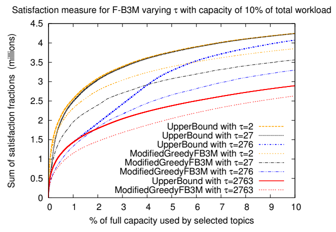

Fig. 2. Comparison of ModifiedGreedyFB3M with the Estimated Upper Bound

that ModifiedGreedyFB3M guarantees an approximation ratio of $\frac{1}{2}\left(1-\frac{1}{e}\right)$. In our real-world data set, we achieve a significantly better ratio (up to 0.9). Analogously to our analysis of GreedyB3M, we use the upper bound given by Theorem VI.4. This theorem can be easily turned into an algorithm identical to Algorithm 3 but with the change that in the last step (line 9) we return $|csubs| + 1$ instead. Since the goal of *F-B3M* is to maximize the total satisfaction fraction among the subscribers of all the topics, the outcome is measured in terms of total fraction instead of number of subscribers. As shown in Figure 2 a similar pattern to GreedyB3M is observed in the approximation ratio when the τ changes from 2 to 2763. However, the gap between the ModifiedGreedyFB3M and the upper bound is much lower compared to the gap between GreedyB3M and its corresponding upper bound. For example for $\tau = 2763$ the approximation ratio between ModifiedGreedyFB3M and the upper bound is 0.9 compared to 0.82 for GreedyB3M, as shown in Figure 3.

GreedyB3M and ModifiedGreedyFB3M algorithms are intended to run on a regular basis, it is important that they are fast. In Figure 4 the running times of the greedy approaches proposed in this paper are shown in seconds (mean of 3 runs). To evaluate the gain of our optimization of lazily updating costs in ModifiedGreedyFB3M as explained in Section VI-A we also introduce a naive version, coined ModifiedGreedyFB3MSlow. ModifiedGreedyFB3MSlow is identical to ModifiedGreedyFB3M except lines 9 to 12 of Algorithm 5. Instead of lazily updating topic costs, all the topics that have a common subscriber with the chosen topic in the current iteration are updated (same as lines 6 and 7 of Algorithm 2). From Figure 4 it is clear that ModifiedGreedyFB3M outperforms ModifiedGreedyFB3MSlow and runs in less than 20 seconds for all values of τ. Finally, ModifiedGreedyFB3M takes a maximum of 33 seconds to run for $\tau = 276$. It is clear that these algorithms are in general fast to run in large-scale setting and can be run on a regular basis.

VIII. Related Work

There are many flavors of pub/sub systems proposed in the literature [7]. Proposals from the last 15 years come

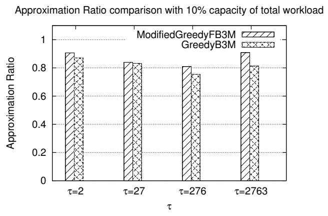

Fig. 3. Computed approximation ratios for *B3M* and *F-B3M* with varying τ

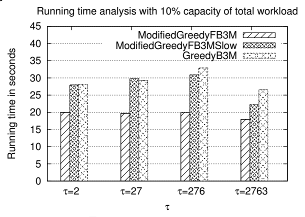

Fig. 4. Running time comparison for Greedy Heuristics with varying τ

from both industry [1], [20], [21] and academia [3], [4], [6], [12], [22]. Subscriber placement and relocation to minimize metrics like publication-notification delay and system load in content-based pub/sub systems have been proposed before [5]. However, to the best of our knowledge, we are the first to formalize the subscriber satisfaction metrics and formulate the problem of maximizing the number of subscribers satisfied.

The formal definition we arrive at bears a strong resemblance to (set) coverage problems; the problem of Budgeted Maximum Coverage (BMC) [14] being the closest match. However, a significant difference is that in our setting a subscriber may need to be "covered" more than once. The family of coverage problems are generally proven NP-Hard using reductions from Max-Cover problem [11]. In this paper we instead reduce *DkS* to our *B3M* problem, which allows us to rule out the existence of a PTAS.

Seminal work on analysis of the maximizing submodular functions was originally done in [10]. In this paper we exploit the submodularity property of the objective function of *F-B3M* to derive a constant approximation ratio for its greedy heuristic and to speed up the corresponding algorithm.

IX. Conclusions

In this paper, motivated by practical scenarios in a real deployed pub/sub system at *Spotify*, we proposed a new approach to maximizing subscriber satisfaction. In the process, we introduced a new set of problems (*B3M* and *F-B3M*) to address the maximization of the number of satisfied subscribers in a pub/sub system and proposed greedy heuristics to solve both problems. We proved that *B3M* is NP-Hard by reduction from the *DkS* problem and, as a corollary, also proved that *B3M* has no PTAS under a standard assumption. *F-B3M* is a relaxed version of *B3M* that is relatively easy to solve. We proved that the objective function of *F-B3M* is submodular, derived a constant approximation bound for its greedy heuristic, and proposed a way to exploit the submodularity of the objective function to improve the running time of the heuristic for typical scenarios. We evaluated our heuristics for both problems using a large-scale real data set from *Spotify*'s pub/sub system and compared their performance with upper bounds we derived for the optimal solutions of both problems. We illustrated that, with a realistic pub/sub workload as input, our heuristics achieve an approximation ratio of up to 0.9 and they are fast enough to be run on a regular basis in a real-world scenario. We conclude that we have demonstrated that there is theoretical and practical evidence that pub/sub systems (like *Spotify*'s pub/sub) can benefit from the algorithms presented in this paper.

Acknowledgments

The authors would like to thank Per Austrin for suggesting the reduction from *DkS* to *UC-B3M*.

References

[1] Tibco rendezvous. http://www.tibco.com.

[2] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *ACM symposium on Theory of computing (SOTC)*, pages 201–210, 2010.

[3] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems (TOCS)*, 19(3):332–383, 2001.

[4] M. Castro, P. Druschel, A-M. Kermarrec, and A.I.T. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489–1499, 2002.

[5] A.K.Y. Cheung and H.-A. Jacobsen. Publisher placement algorithms in content-based publish/subscribe. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 653–664. IEEE, 2010.

[6] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2007.

[7] P.T. Eugster, P.A. Felber, R. Guerraoui, and A-M Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

[8] U. Feige, G. Kortsarz, and D. Peleg. The dense k-subgraph problem1. *Algorithmica*, 29:410–421, 2001.

[9] U. Feige, M. Seltser, et al. On the densest k-subgraph problem. *The Weizmann Institute, Rehovot, Tech. Rep*, 1997.

[10] M.L. Fisher, G.L. Nemhauser, and L.A. Wolsey. An analysis of approximations for maximizing submodular set functions-I. In M.L. Balinski and A.J. Hoffman, editors, *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pages 73–87. Springer Berlin Heidelberg, 1978.

[11] Dorit S. Hochbaum. Approximation algorithms for NP-hard problems. chapter Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems, pages 94–143. PWS Publishing Co., Boston, MA, USA, 1997.

[12] H.-A. Jacobsen, A.K.Y. Cheung, G. Li, B. Maniymaran, V. Muthusamy, and R.S. Kazemzadeh. The padres publish/subscribe system. *Principles and Applications of Distributed Event-Based Systems*, pages 164–205, 2010.

[13] S. Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

[14] S. Khullera, A. Mossb, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70:39–45, 1999.

[15] A. Krause and C. Guestrin. A note on the budgeted maximization of submodular functions. Technical Report CMU-CALD-05-103, Carnegie Mellon University, 2005.

[16] G. Kreitz and F. Niemela. Spotify – large scale, low latency, P2P music-on-demand streaming. In *P2P*, 2010.

[17] G. Li, V. Muthusamy, and H.A. Jacobsen. A distributed service-oriented architecture for business process execution. *ACM Transactions on the web*, 2010.

[18] H. Liu, V. Ramasubramanian, and E. Sirer. Client behavior and feed characteristics of RSS, a publish-subscribe system for web micronews. In *IMC*, 2005.

[19] M. Petrovic, H. Liu, and H. Jacobsen. G-ToPSS: fast filtering of graph-based metadata. In *WWW*, 2005.

[20] J. Reumann. GooPS: Pub/Sub at Google. Lecture & Personal Communications at EuroSys & CANOE Summer School, 2009.

[21] V. Setty, G. Kreitz, R. Vitenberg, M. van Steen, G. Urdaneta, and S. Gimåker. The hidden pub/sub of spotify. In *ACM International Conference on Distributed Event-Based Systems (DEBS)*. ACM, 2013.

[22] V. Setty, M. van Steen, R. Vitenberg, and S. Voulgaris. Poldercast: Fast, robust, and scalable architecture for P2P topic-based pub/sub. In *International Middleware Conference*, pages 271–291. Springer-Verlag New York, Inc., 2012.

[23] P. Triantafillou and I. Aekaterinidis. Peer-to-peer publish-subscribe systems. *Encyclopedia of Database Systems 2009: 2069-2075*, 2009.

Appendix

A. Proof of Theorem VI.1

Definition A.1 (Maximum Coverage)**.** In the (unweighted) *Maximum Coverage* problem, input consists of a collection of sets $S = \{s_1, s_2, \dots, s_n\}$ and a parameter k. The goal is to find a subset $S' \subseteq S$ maximizing $|\bigcup_{s \in S'} s|$ subject to $|S'| \leq k$.

Proof: Given an instance of Maximum Coverage(S, k) we construct an instance of *F-B3M*$(T, V, ev, cost, Int, \tau, \mathcal{C})$ in the following way: we take T with topics that one-to-one correspond to the sets in the collection S and let $cost(t) = 1$. We take V that one-to-one correspond to the elements of $\bigcup_{s \in S}$ and construct *Int* from set membership relationship of sets in S. We further let $ev_t = 1$, set $\tau = 1$, and let $\mathcal{C} = k$.

From this construction it is easy to see that there is a solution of size d of the Maximum Coverage instance iff there is a solution of value d of the *F-B3M* instance. As Maximum Coverage is NP-Hard, this concludes the proof. ∎

B. Proof for Theorem VI.2

Definition A.2 (Submodularity)**.** A function σ is said to be submodular for any set $A \subseteq B$ if the following holds:

$$\sigma(A \cup x) - \sigma(A) \geq \sigma(B \cup x) - \sigma(B)$$

for any element $x \notin B$.

Proof: Intuitively, the objective function for *F-B3M* is submodular because the incremental gain from adding a new topic is fractional i.e, reaching a threshold of τ to have incremental gain is not a requirement. However, a larger set is more likely to have covered more subscribers and higher number of times hence the gain is incremental. In addition to that adding a topic with subscribers already covered τ to a larger set of topics gives no incremental gain in the objective function. On the other hand adding it to a smaller set of topics would give larger incremental gain. Let us now capture the intuition mathematically. Assume that we have two solution sets $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\mathcal{S}_2 \subseteq \mathcal{S}_1$. Adding a topic $t \notin \mathcal{S}_1$ to these sets always has non-negative incremental gain in their respective objective functions. However, the amount of incremental gain depends on the following scenarios:

1) The subscribers V_t of topic t are already covered τ times in both $\mathcal{S}_1$ and $\mathcal{S}_2$. Hence, adding t results no incremental gain for both sets. Note that this case can be extended to both sets already covering equal number of times, and the incremental gain will be same for both.
2) V_t are covered in $\mathcal{S}_1$ x times and they are covered y times in $\mathcal{S}_2$ such that $x \geq y$ (again, note that other way round is not possible since $\mathcal{S}_2 \subseteq \mathcal{S}_1$). The following sub-cases are possible:
   a) If $x + ev_t \geq \tau$ and $y + ev_t \geq \tau$ then, since we know that $x \geq y$, $\mathcal{S}_1$ will have lower gain because $\sum_{v \in V_t} \frac{\tau_v - x}{\tau_v} \leq \sum_{v \in V_t} \frac{\tau_v - y}{\tau_v}$
   b) If $x + ev_t \geq \tau$ and $y + ev_t < \tau$ then, the incremental gain for $\mathcal{S}_2$ is higher because the incremental gain for $\mathcal{S}_1$ is $\sum_{v \in V_t} \frac{\tau_v - x}{\tau_v} \leq \sum_{v \in V_t} \frac{ev_t}{\tau_v}$ since we know that $x + ev_t \geq \tau$.
   c) Finally, if $x + ev_t < \tau$ and $y + ev_t < \tau$ then, both $\mathcal{S}_1$ and $\mathcal{S}_2$ have same incremental gain.
3) V_t are covered τ times in $\mathcal{S}_1$ but not in $\mathcal{S}_2$ (note that other way round is not possible since $\mathcal{S}_2 \subseteq \mathcal{S}_1$). Hence, adding t to $\mathcal{S}_1$ results in no incremental gain while the objective function for $\mathcal{S}_2$ is incremented with exactly $\sum_{v \in V_t} \frac{\min(rem_v, ev_t)}{\tau_v}$, where, $rem_v = \tau_v - \sum_{\{t' \in \mathcal{S}' \cap T_v\}} ev_{t'}$.

All possible scenarios are covered using the above cases. It is easy to see that in all of the above scenarios the following always holds for any $t \notin \mathcal{S}_1$.

$$\sigma(\mathcal{S}_1 \cup t) - \sigma(\mathcal{S}_1) \leq \sigma(\mathcal{S}_2 \cup t) - \sigma(\mathcal{S}_2)$$

■

C. Proof for Theorem VI.5 and Theorem V.6

Proof: The data structure A (both in case of GreedyB3M and GreedyFB3M) can be any max-heap structure supporting insertion, update, and extracting the maximum element in time $\mathrm{O}(\log n)$, e.g., a binary heap. The initialization of the array to store the heuristic values per-topic done in Line 3 of the Algorithm 2 and Algorithm 7 has complexity of $\mathrm{O}(|T||V| \log |T|)$. Once a topic is selected a while loop (lines 9 to 12) is exectued to update the topics in the top of the heap until there is no more change. This loop runs $|T|$ times in the worst case. Within the loop, re-evaluating the heuristics has complexity of $|V|$ and updating A has takes time $\mathrm{O}(\log |T|)$. Hence the run time complexity of the Algorithm 7 and Algorithm 2 is

$$\mathrm{O}\big(|V||T| \log |T| + |T|^2 \left(|V| + \log |T|\right)\big) \approx \mathrm{O}\big(|T|^2 (|V| + \log |T|)\big)$$

■