INCREMENTAL CHANNELIZATION

Fei (Sophie) Che and Errol L. Lloyd [1]
University of Delaware
Newark, DE 19716

July 23, 2009

Abstract

Channelization is the problem of forming multicast groups from inputs of a set of flows, a set of users and a set of user preferences, along with an upper bound on the number of groups. Channelization aims to construct groups such that the network cost is substantially smaller than using a single multicast group which delivers all flows to all users. In this paper we introduce an incremental version of channelization wherein the input may change over time. When such a change occurs, one approach is to totally recompute the channelization solution. In contrast, in the incremental approach taken here, the solution for the original channelization instance is updated to become a solution for the modified channelization instance. The goal is to produce a solution of high quality in substantially less time than it would take to do a full recomputation. In this context we study incremental channelization problems corresponding to adding a flow and adding a user. For these two problems we provide complexity results and incremental algorithms. Specifically we show: 1) that natural incremental approaches are NP-hard; 2) that approximating the optimal solution within a log factor is unlikely; and 3) give a greedy algorithm with a performance within a log factor of the optimal (in light of our result 2, this is the best possible approximation result). In addition to the theoretical results, a case study is given for the problem of adding a flow, providing simulation results comparing the effectiveness of the solutions produced by our incremental algorithms with solutions from algorithms doing full recomputation.

1 Introduction

The problem addressed in this paper considers *channelization* in an *incremental* context:

- In *channelization* [AG01] a specified number of multicast groups are formed from a set of flows, a set of users and a set of user preferences. The goal is to construct groups in a way that the network cost in terms of the total size of the flows delivered to groups, and then to the users, is substantially less in comparison with using a single multicast group wherein all flows are delivered to all users.
- *Incremental* problems arise in situations where the problem instance changes (slowly) over time. That is, the problem instance is not provided "all in one piece" but rather, over time, pieces of the instance may arrive and depart. This situation occurs often in a variety of dynamic problems. Incremental algorithms incorporate changes without any knowledge of the existence and nature of future changes. The goal in solving incremental problems is to develop algorithms that can accept changes in the problem instance (either additions or deletions) and can compute a good solution to the modified problem instance in significantly less time than an algorithm that computes a solution for the modified instance "from scratch".

In this paper we study *incremental channelization*, where the channelization instance can change over time. We focus on the problems of: a) adding a flow (along with the users who want to receive that flow); and b) adding a user (along with the preferences of that user). These are both changes that are likely to occur frequently in real situations. For example, in a military situation, flows may be the outputs of fields of sensors,

[1]Email: {fei, elloyd}@cis.udel.edu. Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

and users may be soldiers located in different geographic areas or with different responsibilities (thus having different sets of sensors in which they are interested). Neither the set of flows nor the set of users is likely to be static. Rather, over time, additional sensor fields (i.e. flows) may become available and/or additional soldiers (i.e. users) may require knowledge of particular flows. This adding of flows and/or users is precisely the situation addressed in incremental channelization.

For the problems of adding a flow and adding a user we provide complexity results and incremental algorithms. Specifically we: 1) show that natural incremental approaches are NP-hard; 2) show that approximating the optimal solution with less than a log factor is unlikely; and 3) describe incremental algorithms that are no more than a log factor of the optimal. In light of result 2, this is the best possible approximation result. In addition to the theoretical results, a case study is given for the problem of adding a flow, providing simulation results comparing the effectiveness of the solutions produced by our incremental algorithms with solutions from algorithms doing full recomputation.

The remainder of this paper is organized as follows. In the next section we provide a formal definition of channelization and summarize existing work. In Section 3 we discuss the complexity and approximation results for the problems of adding a flow and adding a user. Section 4 describes the experimental study. Finally, Section 5 summarizes our results and outlines future work and open problems.

2 Background and Related Work

This section describes background and related work on channelization including a formal definition.

Channelization was first studied in [AG01] and the description given here is based on their formulation, though with modified notation.

An instance of the *channelization problem* consists of:

- t, a positive integer indicating an upper bound on the number of multicast groups to be formed
- a set F of flows $F_1, F_2, ..., F_m$ where the *rate* (in bytes) of flow F_i is denoted by s_i
- a set of users $U_1, U_2, ..., U_p$
- for each user U_i, a set $P_i \subseteq F$ – these are the *preferences* of user U_i and specify the minimum set of flows that must be delivered to U_i
- constants w_1 and w_2 each in the range $[0, 1]$ such that $w_1 + w_2 = 1$

A solution to a channelization instance is a surjective mapping M from flows and users to $1, 2, \ldots, t$ such that for each user U_i, $P_i \subseteq \cup_{j \in M(U_i)} G_j$, where $G_j = \{F_k : j \in M(F_k)\}$. Relative to this mapping, if $j \in M(U_i)$ then user U_i is said to *subscribe* or *belong* to group j, and if $j \in M(F_i)$ then flow F_i is said to *belong* to group j. Thus, we can restate the requirement for the mapping M to be: for each user, each of the flows desired by that user must belong to at least one of the groups subscribed to by that user. Note that since the mapping is surjective, each user and each flow may be mapped to more than one group.

The *cost of a solution* is the weighted sum of a) the cost of getting the flows to the groups and b) the cost of multicasting all of the flows assigned to a group to each user subscribing to that group. Those costs are dependent on the rates associated with the flows and, in the case of sending the flows from the group host to the users, on a "topology dependent coefficient" [AG01] that is unique to each flow and user pair. In this paper, as in both the complexity and experimental results in [AG01], we assume that each of these topology dependent coefficients is 1. It follows that the cost of a solution is

$$\sum_{i=1}^{t} \sum_{j:F_j \in G_i} (w_1 * s_j + w_2 * s_j * u_i) \qquad (1)$$

where u_i is the number of users that subscribe to G_i. Note that when $w_1 = 0$ and $w_2 = 1$, this sum is equal to the total size of the flows delivered to users.

In [AG01] two versions of channelization were proposed. In the first, *constrained channelization*, each flow can be assigned to exactly one group, while in the second, *unconstrained channelization*, there is no such restriction. They showed that both versions are NP-complete. [AG01] also proposed five heuristic algorithms for unconstrained channelization and examined their performance using an experimental study. The primary conclusions were that simple heuristics "generally do not provide much improvement over a random assignment scheme", and that a heuristic called *flow-based merge* provides a good approximation to the result obtained by exhaustive search. That heuristic works as follows: Initially there are m multicast groups, each being assigned a single unique flow. Then, while the number

of groups exceeds t, iteratively merge two groups whose merger will least increase the cost of the solution as given by equation (1). Note that in this approach each flow will be assigned to just a single group, while users will generally subscribe to multiple groups.

3 Complexity Analysis

There are a variety of ways in which an instance of (unconstrained) channelization can change in an incremental fashion. In this paper we study the following two incremental channelization problems:

- The addition of a flow. Along with the added flow, a set of (existing) users who add that flow to their preferences is provided.
- The addition of a user. Along with the added user, the flow preferences of that user are provided.

We consider each of these incremental problems to be critical because they require action in terms of the channelization solution. In the case of adding a user for example, the solution *must* be changed so that the user does indeed subscribe to groups in a way that provides the flows of interest to that user. In contrast, if a flow, or user, or even just a user preference, is deleted then the existing channelization solution remains valid (albeit with the deleted items removed).

3.1 Adding a flow

In this section we consider how to handle the addition of a flow. Recall that along with that flow, we are given a list L of existing users who will add that flow to their preferences. To handle this new flow, we consider three approaches.

3.1.1 *Single group approach (SGA)*

In this approach the new flow f, whose rate is denoted by s_f, is added to just a single existing group, and all of the users in L will add a subscription to that group (if they are not already subscribed to that group). The goal in selecting the group is to minimize the addition to the total delivered flow cost. This can be determined in time $O(tp)$ by determining for each group G_i, $\sum_{j:F_j \in G_i} s_j * u'_i + s_f * u''_i$, where u'_i is the number of users in L that do not subscribe to G_i and u''_i is the number of users either in L or subscribed to G_i. Note that the latter term reflects the delivery of the new flow to all subscribers to G_i. The optimal group is then the group with smallest computed value.

Unfortunately, while this approach is easy to implement, it may result in a very bad solution in the worst case. Consider the following example: There are just two groups, with half the members of L subscribed to one group and the other half subscribed to the second group. Assume that the number of flows in each group is k, that the number of subscribers to each group is s, that each flow rate is 1, and that $w_1 = 0$ and $w_2 = 1$. Then if the flow is added to just one of the two groups, then each of the s subscribers to the other group must also now subscribe to the group where the flow is added. Thus, the addition to the total solution cost is $ks + 2s$. In contrast, if the flow is added to both groups, then no new subscriptions are required and the added solution cost is simply $2s$. Clearly, since there is no apriori bound on k, the single group solution is arbitrarily worse than a solution allowing the flow to be assigned to multiple groups.

3.1.2 *Static subscription approach (SSA)*

In this approach, the new flow is added to a minimum number of groups so that all of the users in L do indeed receive that flow. Here, users do *not* have their subscriptions changed nor are the costs of delivery of the flows from the subscribed groups to the users taken into account in making the decision about to which groups the new flow should be added. Despite the seeming simplicity of this approach, it turns out that computing an optimal solution is quite hard:

Theorem 3.1 *The SSA for adding a flow to an instance of channelization is NP-complete.*

Proof: To show that the decision version[2] of SSA is NP-complete we use a reduction from the known NP-complete problem:

[2]In the decision version of SSA we are given the regular inputs along with a parameter k, and are asked to determine if there is a solution of cost at most k. Formally:
Static Subscription Approach (SSA):

- Instance: Groups $G_1, \ldots, G_t$, a list L of users who will subscribe to a new flow, and a positive integer k.
- Question: Does there exist a set X of at most k groups such that each of the users in L subscribes to at least one of the groups in X.

Hitting Set (HS) [GJ79]:

- Instance: A set M, finite subsets $S_1, S_2, \ldots, S_n$ of M, and a positive integer k.
- Question: Does there exist a subset H of M such that $|H| \leq k$ and such that $|H \cap S_i| \geq 1$ for each i, $1 \leq i \leq n$?

To reduce HS to SSA we use the following construction:

- There is one group in SSA for each element in M from HS.
- For each subset S_i in HS there is a user in SSA. That user then subscribes to each of the groups corresponding to an element in S_i.

Clearly the construction can be completed in linear time. The correctness of the construction is straight-forward and is omitted here due to space constraints. ■

Because this transformation is *approximation preserving* [C97], we immediately have:

Corollary 3.1 *Unless $P = NP$, no approximation algorithm for SSA can have an approximation ratio*[3] *less than $\Omega(logn)$.*

What then is possible in regard to approximation algorithms for SSA? Consider the greedy type algorithm shown in Algorithm 1.

Algorithm 1 – ICNF - INCREMENTAL CHANNELIZATION FOR A NEW FLOW

1: **while** there exist unsatisfied users in L **do**
2: Let G_i be a group having the largest number of unsatisfied subscribers from L;
3: Assign the new flow to G_i;
4: Update L;

[Note: A user in L is **satisfied** *if it subscribes to a group that contains the new flow. Otherwise a user in L is* **unsatisfied***.]*

Using similar results for greedy algorithms for the Hitting Set problem, we can show that:

Theorem 3.2 *The approximation ratio of Algorithm 1 - ICNF is $O(logn)$.*

[3]The approximation ratio of an approximation algorithm is the worst case ratio between the size of the solution produced by the approximation algorithm and the size of an optimal solution.

Thus, from a theoretical perspective, the greedy approach can, on the one hand, have a very bad performance, yielding a solution that is a factor of $\log n$ larger than the minimum. But, on the other hand, from Corollary 3.1, no other algorithm can do better in the worst case. From the experimental perspective the story may be quite different and we examine this issue in the case study of Section **??**.

3.1.3 *Minimize uninterested users (MUS)*

In this approach the new flow is added to multiple groups with the goal of minimizing the sum over all groups where this flow is added, of the number of subscribers to that group. Note that if a user subscribes to two groups, both of which will get this new flow, then that user will be counted twice. Again in this approach user subscriptions do not change.

Similarly to the reduction in the prior subsection, we can use a reduction from the *Weighted* Hitting Set problem to show:

Corollary 3.2 *The MUS problem (decision version) for adding a flow to an instance of channelization is NP-complete.*

Likewise, it follows that:

Corollary 3.3 *Unless $P = NP$, no approximation algorithm for MUS can have an approximation ratio less than $\Omega(logn)$.*

To approximate an optimal solution for MUS, we use the heuristic shown in Algorithm 2.

Algorithm 2 – WICNF - WEIGHTED INCREMENTAL CHANNELIZATION FOR A NEW FLOW

1: Let u_i be the number of subscribers to G_i;
2: **while** there exist unsatisfied users in L **do**
3: Let G_i be a group having the smallest *cost ratio* $c = u_i/v$ where v is the number of unsatisfied subscribers to G_i from L;
4: Assign the new flow to G_i;
5: Update L;

Using results for the weighted hitting set problem [GJ79] it follows that:

Corollary 3.4 *The approximation ratio of Algorithm 2 - WICNF is $O(logn)$.*

Again, while the theoretical results are discouraging, they do not address how the algorithm may perform on the average or in typical scenarios. This question is addressed through simulations described in Section **??**.

We conclude this section by noting that although the problem of adding a new flow is *incremental* with respect to the solution to the instance of channelization, the algorithms that we utilize are not themselves incremental with respect to the underlying problem. Thus, our solutions are based on solutions to Hitting Set. That is regular Hitting Set, and not an incremental version of Hitting Set.

3.2 Adding a user

In this section we consider how to change the channelization assignments when a new user is added. Recall that along with that new user, we are given their flow preferences. We consider two approaches, described below, to handling a new user.

3.2.1 *Static assignment approach (SAA)*

In this solution, the new user subscribes to a minimum number of groups so that each of its preferences is met. In this approach, flows do *not* have their assignments changed nor are the costs of delivery of the flows from the subscribed groups to the new user taken into account in making the decision about to which groups the new user should subscribe. As in the case of adding a flow, despite the seeming simplicity of this approach, it turns out that computing an optimal solution is quite hard:

Theorem 3.3 *The SAA (decision version) for adding a user to an instance of channelization is NP-complete.*

Proof: To show that SAA is NP-complete, we use a reduction from the classic NP-complete problem:
Set Cover (SC) [GJ79]:

- Instance: A universal set H, finite subsets $S_1, S_2, \ldots, S_n$ of H, and a positive integer k.
- Question: Do there exist k of the base sets $S_{i1}, S_{i2}, \ldots, S_{ik}$ such that the union of those k base sets is H?

To reduce SC to SAA we use the following construction:

- There is a flow for each element in H.
- The new user has one flow preference for each element in H.
- There is a group G_i for each finite subset S_i.
- A flow is assigned to group G_i if the element in H that corresponds to that flow is in S_i.

Clearly the construction can be completed in linear time. The correctness of the construction is straight-forward and is omitted here due to space constraints. ■

Because this transformation is *approximation preserving* [C97], we immediately have:

Corollary 3.5 *Unless $P = NP$, no approximation algorithm for SAA can have an approximation ratio less than $\Omega(logn)$.*

What then is possible in regard to approximation algorithms for SAA? Consider the algorithm for SAA given in Algorithm 3.

Algorithm 3 – ICNU - INCREMENTAL CHANNELIZATION FOR A NEW USER

1: **while** there exist unsatisfied preferences for the new user **do**
2: Let G_i be a group having the largest number of flows corresponding to currently unsatisfied preferences of the new user;
3: Let the new user subscribe to group G_i;
4: Update the remaining unsatisfied preferences;

Using similar results for greedy algorithms for the Set Cover problem, we can show that:

Theorem 3.4 *The approximation ratio of Algorithm 3 - ICNU is $O(\log n)$ where n is the number of preferences of the new user.*

Thus, from a theoretical perspective, the greedy approach can, on the one hand, have a very bad performance, yielding a solution that is a factor of $\log n$ larger than the minimum. But, on the other hand, from Corollary 3.5, no other algorithm can do better in the worst case.

3.2.2 *Minimize delivered flows (MDF)*

In this approach the new user is added to multiple groups with the goal of minimizing the sum over all groups where this flow is added, of the total rate of the flows assigned to that group. That is, we want to minimize the

total flow rate for the new user - these flows of course, include both the ones given in its preferences as well as the other flows delivered by the groups to which the user subscribes (the flows the user did not desire). Note that if a user receives the same flow from two different groups, then that flow rate is counted twice since the flow is actually delivered twice. Note that in this solution flow assignments to groups do not change.

Analogous to the prior section, we can show the following. The proofs are omitted due to space constraints:

Corollary 3.6 *The MDF problem for adding a flow to an instance of channelization is NP-hard.*

Corollary 3.7 *Unless $P = NP$, no approximation algorithm for MDF can have an approximation ratio less than $\Omega(logn)$.*

Corollary 3.8 *There is a polynomial time approximation algorithm for MDF having an $O(logn)$ approximation ratio.*

3.2.3 Adding a single user preference

In this section we consider how to handle the situation where a user u adds a single preference to its set of preferences. It turns out that this is relatively easy to accommodate as follows. Let F_i be the newly desired flow. There are two possibilities. Either F_i is located in a group to which user u already subscribes, or it is not. In the former case there is nothing to do. In the latter case, there are two possible actions: Either user u adds a subscription to a group that includes F_i, or F_i is added to a group to which user u already subscribes. Accordingly, we calculate two values:

- Determine the least total flow rate of the flows in a group to which F_i is assigned. Call this least total flow rate r_1.
- Determine the least number of users of any group to to which user u already subscribes. Let G_j be that group and let N be the number of users in that group. Then let $r_2 = N \times s_i$, which is the total additional delivered flow rate for existing subscribers to group G_j.

The action that is taken (u adds a subscription or F_i is added to some group) depends on which of the values r_1 and r_2 is smaller. Clearly this determination can be made in polynomial time.

4 A Case Study - Adding a Flow

This section provides an experimental case study of the performance of heuristics for the problem of incrementally adding a flow. Recall (section 3.1) that we outlined three approaches for handling the addition of a flow, namely SGA, SSA, and MUS, and gave incremental algorithms for handling each approach. In this section we compare the performance of those incremental algorithms with the results obtained by running the Flow Based Merge (FBM) algorithm (section 2) of [AG01] on the entire instance.

Our experiments are designed as follows: First, a specific size instance of channelization is generated and solved using FBM. Then, additional flows are added to the instance one at a time, with the algorithms for SGA, SSA and MUS each being applied [4]. At the end of the process, FBM is run on the entire final instance and the final outputs of SGA, SSA and MUS are compared against the output of that final run of FBM. In the next several paragraphs we provide some details of this process.

As noted, we begin by randomly generating problem instances (sets of flows and users, along with flow rates and preferences). This is done using parameters:

- high rate *or* low rate – Each flow is assigned either "high" flow rate r_H with probability $Prob_r$ or "low" flow rate r_L with probability $1 - Prob_r$.
- user preference – Each flow is either *popular* with probability $Prob_p$ or *unpopular* with probability $1 - Prob_p$. Note that the popularity of a flow is uncorrelated to its rate. A user will add a popular flow to its preferences with probability $Prob_{pop}$ and an unpopular flow with probability $Prob_{unp}$. Usually, $Prob_{pop} >> Prob_{unp}$.

As mentioned in Section 2, for the cost function we assume each topology dependent coefficient is 1. We also let $w_1 = w_2 = 1$. Thus, the cost of a solution to a channelization instance is $\sum_{i=1}^{t} \sum_{j:F_j \in G_i} (s_j + s_j * u_i)$ where s_j is the rate of flow F_j and u_i is the number of users that subscribe to G_i.

The parameters utilized in our experiments were set as follows: Initially a channelization instance was generated having 25 flows, 10 groups and 100 users, with

[4] Those are: the algorithm in section 3.1.1 for SGA; the INCNF algorithm for SSA; the WICNF algorithm for MUS. In this section we refer to each algorithm by the name of the problem it handles.

$r_H = 10, r_L = 1, Prob_r = 0.2, Prob_p = 0.4, Prob_{pop} = 0.6, Prob_{unp} = 0.1$. The FBM algorithm was run on this initial instance to produce an initial solution (i.e. assignment of flows and users to groups). Then, 25 new flows were introduced along with user preferences for those flows. The flow rate and user perferences were generated using the same parameters as for the original 25 flows. These 25 new flows were added to the original 25 flow instances one by one and each of SGA, SSA and MUS was run for each added flow. In addition, FBM was run on the entire instance containing the 50 flows. For each of SGA, SSA, MUS and FBM, the cost of the solution that was produced when all 50 flows are included was calculated. Figure 1 shows the results. There, results are given for 30 different channelization instances as outlined above. The x-axis shows the experiment number and the y-axis shows the solution cost.

From the figure, we see that SGA provides the best results of the three heuristics, and that the results provided by SGA are almost as good as FBM (sometimes even better[5]!). Note that this is precisely the result that one hopes for in designing incremental algorithms! Notice also that both SGA and FBM implicitly include a constraint that no flow will be assigned to more than one multicast group. In contrast, SSA and MUS allow each flow to be assigned to multiple multicast groups. It seems that a side affect of this assignment is that higher numbers of unnecessary flows are delivered, hence the higher costs of SSA and MUS. This also explains why MUS performs worse than SSA because MUS tends to assign the new flow to more groups than SSA since smaller sized groups may achieve better cost ratios. Again, this increases the possibility of unnecessary flows being delivered to users.

Along with the specific simulation reported above, we have run additional simulations varying problem parameters. Those results, not reported due to space constraints, all show the same general results (i.e. SGA almost as good as FBM, etc) as those described here.

5 Conclusions

In this paper we studied two incremental problems associated with channelization. We showed that most approaches to these problems are based on NP-hard problems and cannot be approximated within $\log n$ factors. We also gave algorithms that achieve those $\log n$ approximation factors. In addition we provided an experimental case study of the problem of adding a flow, comparing the performance of our incremental algorithms with solutions based on full recomputation of a solution.

[5]Recall that FBM is also a heuristic that does not guarantee an optimal solution.

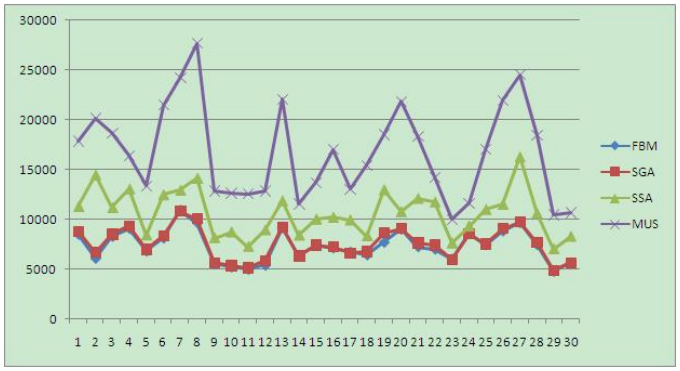

Figure 1: Cost incremental versus full

One open problem is how to handle changes in the number of groups. For instance, if the group count is decremented, what should be done? One approach is to delete a group and reallocate its subscribers. There are two issues: First, what is the best method of selecting a group to delete, and second, once that group is selected, how should the flows and users associated with that group be reassigned? This seems to be a more difficult version of the basic channelization problem, since groups already have some assigned users and flows.

Acknowledgement: We thank Professor S.S. Ravi of the University at Albany for his insights.

Disclaimer: The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government.

References

[AG01] M. Adler, Z. Ge, J. F. Kurose, D. Towsley and S. Zabele. "Channelization Problem In Large Scale Data Dissemination", ICNP, 2001, pp. 100–109.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman and Co., San Francisco, CA, 1979.

[C97] P. Crescenzi. "A Short Guide To Approximation Preserving Reductions", Proc. 12th IEEE Conf. on Computational Complexity, 1997, pp. 262–273.