In: Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02), pp. 194–202. © SIAM 2002.

NP-Hardness of Broadcast Scheduling and Inapproximability of Single-Source Unsplittable Min-Cost Flow

Thomas Erlebach* Alexander Hall*

Abstract

We consider the version of broadcast scheduling where a server can transmit one message of a given set at each time-step, answering previously made requests for that message. The goal is to minimize the average response time if the amount of requests is known in advance for each time-step and message. We prove that this problem is NP-hard, thus answering an open question stated by Kalyanasundaram, Pruhs and Velauthapillai (Proceedings of ESA 2000, LNCS 1879, 2000, pp. 290–301). Furthermore, we present an approximation algorithm that is allowed to send several messages at once. Using 6 channels for transmissions, the algorithm achieves an average response time that is at least as good as the optimal solution using one channel. The best previous approximation algorithm achieved ratio 1.5 with 6 channels and reached ratio 1 only in the case where there are as many channels as messages.

From the NP-hardness of broadcast scheduling we derive a new inapproximability result of $(2-\varepsilon, 1)$ for the (congestion,cost) bicriteria version of the single source unsplittable min-cost flow problem, for arbitrary $\varepsilon > 0$. The result holds even in the often considered case where the maximum demand is less than or equal to the minimum edge capacity ($d_{\max} \leq u_{\min}$), a case for which an algorithm with ratio $(3, 1)$ was presented by Skutella.

1 Introduction

We will first give a brief introduction to the two problems this paper is concerned with, then go into previous work, and finally sketch the results presented.

1.1 Definitions and notation concerning Broadcast Scheduling In the broadcast scheduling problem a (possibly large) number of clients over time request messages M_i, $i \in \{1 \dots n\}$, from a server. They do this via a slow channel, e.g. a modem connection. The server can answer these requests via *one* high-bandwidth channel to which all clients are connected, e.g. a TV-cable, electrical power supply, or satellite. At each time-step the server can broadcast one of the n messages. The goal is to minimize the average response time (the time a client has to wait on average until her request is satisfied). Because several clients might request the same message it is possible to bundle answers. The broadcast scheduling problem is gaining practical importance due to the increasing availability of infrastructure that supports high-bandwidth broadcast and due to the growth of information-centric applications [1].

Formally, an instance of the offline version of the problem is given by the number of requests for message i at time t: $R_i(t)$, for $i \in \{1 \dots n\}, t \in \{0 \dots T\}$. A (1-feasible) schedule is a set of values $S(t) \in \{1 \dots n\}$, for $t \in \{1 \dots T+n\}$, that denote the message being broadcast at time-step t. Since there are no requests after time T, we can assume that any schedule has satisfied all requests by time $T+n$.

Let $\delta_i(t)$ be the number of time-steps which pass until the server broadcasts message i after time t. This is the time a request for M_i at t takes to be satisfied. Requests can at the earliest be satisfied one time-step after they were made, so $\delta_i(t) \geq 1$. This leads to the following expression for the average response time (ART):

$$\frac{\sum_t \sum_i R_i(t) \cdot \delta_i(t)}{\sum_t \sum_i R_i(t)}.$$

In the following the denominator will be ignored because it is constant for a fixed instance. We will denote this problem by $B|r_i, p_i = 1|\sum F_i$.

A natural generalization of the problem is to allow multiple channels for the responses. A schedule that broadcasts at most s messages simultaneously (alternatively, a schedule that uses s broadcast channels) is called an s-feasible schedule.

Another interesting generalization allows arbitrary length messages and preemption $(B|r_i, pmtn|\sum F_i)$. Furthermore the arrival times of the requests are not required to be integral and the available bandwidth (W) can be distributed arbitrarily among the broadcast messages at every time-step (we call this *bandwidth sharing*). We consider the case where clients cannot buffer the last part of a message, i.e. if a request arrives in the middle of the broadcast of the wanted message

*Computer Engineering and Networks Laboratory (TIK), Gloriastrasse 35, ETH Zentrum, 8092 Zurich, Switzerland, email: {erlebach|hall}@tik.ee.ethz.ch. The second author is supported by the joint Berlin/Zurich graduate program Combinatorics, Geometry, and Computation (CGC), financed by ETH Zurich and the German Science Foundation (DFG).

it has to wait until the next complete broadcast has finished.

An algorithm for the broadcast scheduling problem is called an s-speed ρ-approximation algorithm if it runs in polynomial time and always computes an s-feasible schedule whose average response time (ART) is at most ρ times the ART of an optimal 1-feasible schedule.

1.2 Single source unsplittable min-cost flow problem A directed graph (V, E) is given with edge capacities $u_e > 0$ and costs $c_e \geq 0$, $e \in E$. One vertex $s \in V$ is the designated source vertex. For a set of k commodities we are given destinations $t_i \in V \setminus \{s\}$ and demands $d_i > 0$, $i \in \{1 \dots k\}$. Furthermore the problem instance contains a budget $B \geq 0$. The goal is to route each demand d_i from s to its destination t_i unsplittably without violating the edge capacities and without exceeding the budget B. Formally if each d_i is routed along a $s - t_i$ path P_i with cost $c(P_i) = d_i \cdot \sum_{e \in P_i} c_e$ the following must hold: $\sum_{i: e \in P_i} d_i \leq u_e$ for all $e \in E$ and $\sum_{i=1}^{k} c(P_i) \leq B$.

In this paper we are only concerned with so called bicriteria (a, b)-approximation algorithms with respect to *minimum congestion* and cost, i.e., algorithms computing a flow that violates the edge capacities only by a factor a more than the optimal solution and is at most b times more expensive than the budget.

Often the case where the maximum demand is less than or equal to the minimum edge capacity ($d_{\max} \leq u_{\min}$) is considered, resulting in better approximation ratios.

1.3 Related work The problem of minimizing the ART for broadcast scheduling with unit-length messages is considered by Kalyanasundaram et al. [7]. An algorithm with ratio $W/(W-2)$, using $W \geq 3$ channels and comparing the solution to the optimum 1-feasible schedule, is presented. This e.g. leads to an approximation ratio of 3 for 3 channels or 1.5 for 6 channels. Determining whether the broadcast scheduling problem is NP-hard is stated as an open question. The paper contains some results about the online version of the problem, one of them being that every online algorithm using only one channel has a competitive ratio of $\Omega(n)$. This additionally motivates the relaxation of allowing an online / approximation algorithm to use several channels and comparing the resulting ART with the ART of the optimal 1-feasible schedule. Relaxations of this kind are called resource augmentation [6, 10].

Edmonds and Pruhs consider in [5] the case where arbitrary length messages and preemption are allowed ($B|r_i, pmtn|\sum F_i$). The authors prove that no 1-speed $\mathcal{O}(1)$-competitive online algorithm exists. They also present an algorithm which is $\mathcal{O}(1)$-speed $\mathcal{O}(1)$-competitive.

In [3] a slightly different version of broadcast scheduling with different message sizes and preemption is treated. For the goal of minimizing the maximum response time a PTAS is given.

[8], [11] and [9] are concerned with stochastic broadcast scheduling problems in which not the exact $R_i(t)$ are known in advance but the probabilities p_i of a user requesting M_i at any time-step. Costs c_i for broadcasting a message M_i are introduced. The goal is to find an infinite (periodic) schedule which minimizes the sum of the expected response time and the broadcast costs of one period. This is motivated e.g. by the TV tele-text system. In [9] the authors look into the case where all messages have unit length and present a PTAS if the number W of channels and the costs c_i are bounded by constants. They state that it is unknown whether the problem is NP-hard, although a somewhat generalized version is proven to be so in [2].

Results of empirical studies and background information about broadcast scheduling can be found in [1].

Recently, Skutella [12] presented upper and lower bounds for the approximation of the single source unsplittable min-cost flow problem. Among other results he gave algorithms that achieve ratio $(3, 1)$ for the $d_{\max} \leq u_{\min}$ case and ratio $(3 + 2\sqrt{2}, 1)$ for arbitrary demands. Furthermore he proved a lower bound of $(1 + \sqrt{5})/2 \approx 1.618$ for the case of arbitrary demands and without costs.[1] Dinitz, Garg and Goemans [4] prove that a ratio of 2 can be achieved if $d_{\max} \leq u_{\min}$ and no costs are present.

1.4 Contribution of this paper The main contribution of this paper is an NP-hardness proof for the broadcast scheduling problem with unit-length messages, which answers the open question of [7]. The proof is based on a reduction from Max Independent Set using different *gadgets* to simulate the nodes and the edge constraints. As a consequence of the NP-hardness, the decision version of broadcast scheduling is NP-complete because the problem is clearly in NP.

We show that this result can be carried over to the problem $B|r_i, pmtn|\sum F_i$ (which is discussed also in [5]), by proving that preemption does not help in the unit-length case.

Using the NP-hardness result, we prove a new inapproximability result of $(2 - \varepsilon, 1)$, for any $\varepsilon > 0$, for the flow problem stated above. In the case of arbitrary demands this improves on the lower bound of 1.618

[1] The conference version of [12] claims a lower bound of $2 - \varepsilon$, but this is clarified in the journal version.

due to Skutella [12], although for the latter no edge costs need to be present. In the $d_{\max} \leq u_{\min}$ case, on the other hand, this is to our knowledge the first inapproximability result of this sort. There is a gap to the so far best known approximation ratio of $(3, 1)$. Skutella in [12] cites Goemans conjecturing that a ratio of $(2, 1)$ can be obtained by generalizing results from [4]. In that case our lower bound would be tight.

Finally, we propose a 6-speed 1-approximation algorithm for broadcast scheduling, i.e., an algorithm achieving ratio 1 using 6 channels compared to the optimal solution using one channel. This improves on the previously known ratio of 1.5 for 6 channels [7].

2 NP-Hardness of Broadcast Scheduling

2.1 Reducing Max Independent Set First we explain the general idea of the reduction from Max Independent Set to single-channel broadcast scheduling. Then we will describe more precisely the gadgets used and prove their correctness. Finally, we put it all together and give concrete values for the parameters in order to obtain a polynomial reduction. We also show how to generalize the result to the multiple channel case, and later the case where preemption and bandwidth sharing is allowed.

2.2 The general idea Let (V, E) be the graph for which a maximum independent set is to be found. Each node $v \in V$ will be represented by $\deg(v) + 1$ pairs of messages, where $\deg(v)$ is the degree of node v. In each pair, one message stands for "v is in the set" (the *on* message) and the other for "v is not in the set" (the *off* message). The goal of the first part of the requests is to "force" a schedule to either have X requests waiting for each *on* message and none for the *off* messages of node v, or vice versa. In the second part of the requests the edge constraints will be modeled. For each edge $\{u, v\}$, so far unused *on/off* messages of u and v are utilized. Finally, in the last part of the requests, nodes being *on* are rewarded using the remaining $|V|$ *on/off* message pairs.

From the description below it will become clear how the schedule S_0 corresponding to an empty independent set in (V, E) would look like. Let ART_0 be the ART of the schedule S_0. We call any schedule whose ART is at most ART_0 a *reasonable schedule*. When we say that a schedule is "forced" to do something we mean that any other way of scheduling would result in an unreasonable schedule (with an ART greater than ART_0).

Let a, b and c denote lengths of certain non-empty intervals (shown in the figures) satisfying $2 < a, b, c < T$. We will explain later how to set these parameters. In the following we will also need several different request-amounts. The smallest number of clients requesting one message at a time-step will be denoted by X. Let G be a much larger amount of requests, such that $G_1 := G/(a-1)$, $G_2 := G/a$ and $G_3 := G/(a-2)$ are integers. In particular, $G_i > X \cdot (T + n)$ should hold. This ensures that at any time-step it is always better to schedule a message with G_i requests waiting than one with X requests, no matter what happens in future time-steps. The largest possible $R_i(t)$ will be L, $L > r \cdot G \cdot (T + n)$, where r is the total number of requests excluding the requests with $R_i(t) = L$. L requests for a dummy message at time t can be used to force the scheduling of the dummy message at time $t + 1$ in any reasonable schedule, since delaying the dummy message by one time-step is more expensive than all requests for non-dummy messages in total. Thus, we can block big parts of the schedule for the other messages by having L requests for the dummy message in each time-step. In the following pictures of request patterns this dummy message will not be shown explicitly, but it will be requested L times at each time-step during which no other message is requested. As a consequence only in the time-steps directly following ones with requests for other messages, a non-dummy message can be broadcast.

2.3 Duplicating the status of a node In this section we will only regard the first part of the requests (the *duplication part*) and also only take into account its contribution to the ART until its end. By right choice of G we will later show that a suboptimal schedule for the duplication part cannot lead to a reasonable total schedule, no matter what is done in the other parts.

In this part each node v is treated separately in order to achieve the described setting of X waiting requests for each of $\deg(v) + 1$ (*on* or *off*) messages. Let v_i, $i \in \{1 \dots 2 \cdot \deg(v) + 2\}$, be the corresponding messages, even i standing for *off* messages and odd i for *on* messages. We now look at an interval concerning one of these nodes. During the whole interval at each time-step where a non-dummy transmission is possible at least two v_i messages will have $\geq G/a$ requests waiting. All other messages each have at most X requests waiting (in any reasonable schedule). Therefore it will always be better to schedule one of the v_i messages. The contribution of the waiting X requests and the requests for the dummy message to the ART is the same for all reasonable schedules and will be ignored in the following.

Without loss of generality assume that the interval begins at $t = 0$. The first requests in the interval are $R_{v_1}(0) = G_1$ and $R_{v_2}(0) = G_2$. If v_1 is scheduled at time 1 this corresponds to the node being in the set ("on"), otherwise it is not ("off"). For now let us assume v_1 is scheduled. Next in the interval a simple gadget of requests will follow, so that at the end again G_2 requests will be waiting for message v_2, X requests for v_4 and none for v_1 and v_3. This gadget so to speak duplicates the status of v_1, v_2 to v_3, v_4. This is repeated for the pairs of messages v_5, v_6; v_7, v_8 and so on. As a last step in the interval the waiting G_2 requests for message v_2 will be "transformed" into X waiting requests for v_2.

We will now take a closer look at the duplicator gadget. Figure 1 shows it and two possible schedules which minimize its contribution to the ART. In the following we will prove that any other schedule is suboptimal and cannot lead to a reasonable schedule in total. We let A, B, C, D denote time-steps as shown in Figure 1. Since we assume that the interval starts at time 0, we have $A = 0$, $B = a - 1$, $C = 2a - 2$ and $D = 3a - 2$.

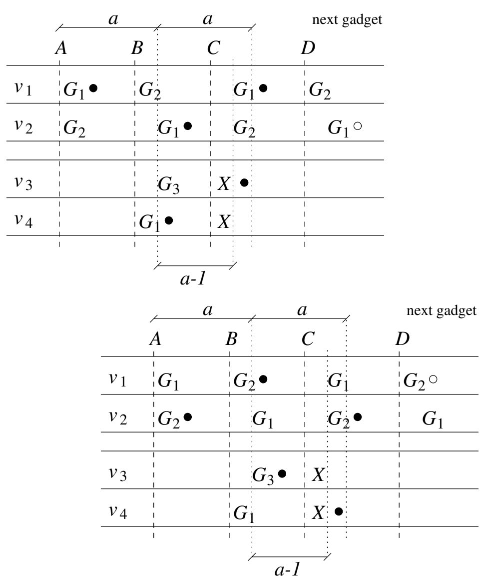

Figure 1: Two schedules which minimize the ART for the duplicator gadget. The dots mark which message is broadcast. The situation atop corresponds to v being in the independent set, the one below to v not being in.

Observation 2.1. *Lower bound: In every schedule, from time-step $A+1$ until B and from $C+2$ until D at least G/a requests are waiting, and from $B+2$ until C at least $2 \cdot G/a$ requests.*

Both schedules in Figure 1 come close: From $A+1$ until B and from $C+2$ until D at most $G/(a-2)+X$ clients are waiting, during the period $[B+2, C]$ at most $2 \cdot G/(a-2)$.

Let a be sufficiently large (for example 100). Consider a schedule that has additional G/a or more requests waiting in any of the periods, compared to the lower bound. Clearly it will perform worse (cost at least G/a more) than either of the schedules in Figure 1.

We use this observation to restrict the number of cases which have to be taken into account at the time-steps $B+1$ and $B+2$. Let $v_i v_j$ denote that $S(B+1) = v_i$ and $S(B+2) = v_j$.

Lemma 2.1. *All cases other than v_4v_2, v_1v_3, v_1v_4 and v_4v_1 lead to unreasonable schedules.*

Proof. The following three observations exclude possible combinations of $v_i v_j$ step by step:

- $S(B+1) = v_1$ or $S(B+1) = v_4$, otherwise at least $3G/a$ instead of at most $2G/(a-2)$ requests are waiting during $[B+2, C]$.
- Analogously:
  If $S(1) = v_1$ then $S(B+2) = v_2$.
  If $S(1) = v_2$ then $S(B+1) = v_1$ or $S(B+2) = v_1$.
- The last case to be excluded, the case v_1v_2, is also out of question because this would lead to $\geq 2G/a$ requests waiting in $[C+2, D]$ instead of $\leq G/(a-2)$.

□

Lemma 2.2. *The cases v_1v_4 and v_4v_1 cost at least G/a more than v_4v_2 and v_1v_3, if only the contribution of $R_{v_1}(B)$, $R_{v_2}(B+1)$, $R_{v_3}(B+1)$ and $R_{v_4}(B)$ until D is considered.*

Proof. In any of the cases $S(C+1)$ can (and has) to be chosen such that neither $R_{v_3}(B+1)$ nor $R_{v_4}(B)$ have to wait after $C+1$. Similarly also $R_{v_1}(B)$ and $R_{v_2}(B+1)$ do not have to wait longer than $C+2$. This leads to the following joint contributions:

- v_4v_2: $(a+1)G_2 + G_1 + (a-1)G_3 + G_1 =$
  $$G \cdot \left(2 + \tfrac{1}{a-2} + \tfrac{2}{a-1} + \tfrac{1}{a}\right)$$
- v_1v_3: $G_2 + aG_1 + G_3 + aG_1 =$
  $$G \cdot \left(2 + \tfrac{1}{a-2} + \tfrac{2}{a-1} + \tfrac{1}{a}\right)$$
- v_1v_4: $G_2 + aG_1 + (a-1)G_3 + 2G_1 =$
  $$G \cdot \left(2 + \tfrac{1}{a-2} + \tfrac{3}{a-1} + \tfrac{1}{a}\right)$$
- v_4v_1: $2G_2 + aG_1 + (a-1)G_3 + G_1 =$
  $$G \cdot \left(2 + \tfrac{1}{a-2} + \tfrac{2}{a-1} + \tfrac{2}{a}\right)$$

□

Lemma 2.3. *In every reasonable schedule the joint contribution of $R_{v_1}(0)$ until time-step $B+1$ and $R_{v_2}(0)$ until time-step $B+2$ is the same.*

Proof. There are two cases to be considered:

- $S(1) = v_1$: The joint contribution is
$$G_1 + (a+1)G_2 = G \cdot \left(\tfrac{1}{a-1} + \tfrac{1}{a} + 1\right)$$
- $S(1) = v_2$: The joint contribution is
$$G_2 + aG_1 = G \cdot \left(\tfrac{1}{a-1} + \tfrac{1}{a} + 1\right)$$

□

LEMMA 2.4. *The contribution of a single duplicator gadget to the ART in the duplication part is minimized by either of the two schedules in Figure 1. Every other schedule costs at least G/a more.*

Proof. Considering only the contribution until time-step $C+2$ and ignoring the requests $R_{v_1}(C+1)$ and $R_{v_2}(C+1)$ (these can be accounted to the next gadget), this follows directly from Lemmas 2.1, 2.2 and 2.3. To the right of $C+2$ in every schedule there will be at least one message with X waiting requests. These messages cannot be scheduled until after the duplication part is over. So it does not matter which of the two messages is waiting. □

Now all that is missing is the special case after the last duplicator gadget, which ends with G_1 requests for v_1 and v_2 (instead of G_2 for the latter): the "transformation" of G_1 requests waiting for v_1 (resp. v_2) into X requests waiting for v_1 (resp. v_2). It is easy to see that the requests depicted in Figure 2 achieve that.

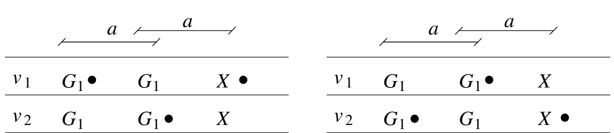

Figure 2: Two possible optimal schedules of the transformation gadget.

2.4 Enforcing the edge constraints In this section we present an *edge* gadget that will model the edge constraints. We only take into account its contribution to the ART in the second part of the requests (the *edges part*). We assume that for each node either all its *on* messages or all its *off* messages have X requests waiting. As in the previous section the gadgets can be treated separately. At each time-step where a message can be scheduled at least one big amount of requests ($\geq G$) concerning the current edge will be waiting and only small amounts for other messages ($\leq X$).

For each edge $\{u, v\}$ previously unused message pairs u_i, u_{i+1} and v_j, v_{j+1} are combined in an *edge gadget*. Figure 3 shows this gadget in three possible situations and resulting optimal schedules. After the requests for the *off* messages three time-steps are given to answer all waiting G requests. The fourth case, with u not being in the set and v being in, is symmetrical to the second picture.

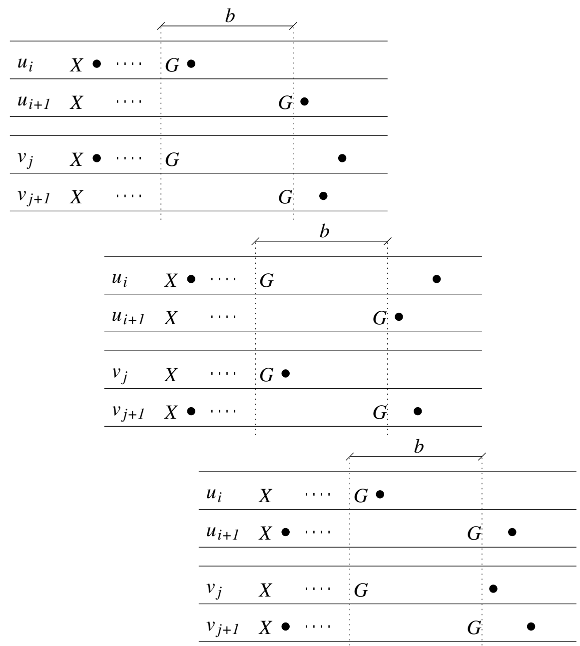

Figure 3: The edge gadget in three possible situations with the corresponding optimal schedule.

LEMMA 2.5. *The contribution of a single edge gadget to the ART in the edges part of the schedule is minimized by one of the three schedules in Figure 3, depending on which messages have waiting requests. The first case (both nodes are in the set) costs $X \cdot b$ more than the other two.*

Proof. The contribution of the G requests can be completely ignored because it is the same in any (optimal) schedule. It is clear that the shown schedules minimize the ART in the corresponding situation. There are other symmetrical schedules which lead to the same ART. A simple calculation gives that the costs of the second and third case are the same and that the first case is $X \cdot b$ more expensive. □

2.5 Maximizing the nodes in the set The last *on/off* message pair of each node is used to reward nodes in the independent set. This is done again in a separate interval for each node. The interval begins with G requests for the *off* message, then it has c empty steps (filled with dummy requests), and it ends with G requests for the *on* message. There is only one reasonable schedule: in the beginning immediately schedule the *off* message and at the end the *on* message. The proof of the following lemma is obvious.

LEMMA 2.6. *If a node is in the set, its* reward *gadget contributes* $X \cdot (c+1)$ *less than if it is not in the set.*

2.6 Combining gadgets and setting the parameter values We now look at the combination of the three previously described parts. Consider some schedule S which is suboptimal in the duplication part. By Lemma 2.4 it contributes at least G/a more to the ART than a schedule which is optimal in this part. The only way S can gain is by possibly earlier scheduled messages with X requests waiting. G is chosen such that G/a is greater than $2 \cdot X \cdot (T+n) \cdot (2 \cdot |E| + |V|)$. So even if all requests of the form $R_i(t) = X$ can be satisfied immediately in S, its ART will still be larger than that of a schedule which is optimal in the duplication part.

This choice of G also suffices to ensure that the cases described in the edge part and the reward part of the requests are the only ones appearing in reasonable solutions.

It remains to choose b such that no edge constraints are violated. The only possibility to gain something by breaking these constraints is via the reward gadgets. If b is greater than $(c+1) \cdot |V|$, violating one edge constraint costs more than what can be gained by scheduling all reward gadgets the optimal way. So it is clear that appropriate values a, b, c, X, G and L can be found.

All reasonable schedules correspond to independent sets of (V, E), and the cost of a reasonable schedule decreases as the cardinality of the corresponding independent set gets larger. Thus, a polynomial algorithm for solving the constructed instance of broadcast scheduling optimally would imply a polynomial algorithm for Max Independent Set, an NP-hard problem.

THEOREM 2.1. *The single-channel broadcast scheduling problem with unit-length messages and given requests is strongly NP-hard.*

Remark: *strongly* NP-hard follows from the fact that all encoded numbers are polynomial in the original input size.

Consider the case with several, say, W, response channels. Simply adding $W-1$ new dummy messages and requesting each of them L times in every time-step ensures that $W-1$ of the W channels are effectively blocked in any reasonable schedule, thus leading to the following theorem.

THEOREM 2.2. *Multiple-channel broadcast scheduling with unit-length messages and given requests is strongly NP-hard for any number W of channels.*

2.7 NP-hardness when preemption is allowed

THEOREM 2.3. *The single-channel broadcast scheduling problem with unit-length messages, preemption, bandwidth sharing and given requests is NP-hard.*

Proof. We transform an optimal, preemptive solution for an instance with unit-length messages, requests arriving at integral time-steps and $W = 1$ into an optimal, non-preemptive one. We consider one time-step after the other starting at $t = 1$. Let S be the modified schedule up to time-step t. Assume that at each time-step to the left of t either exactly one or no message is broadcast in S (this is clear for $t = 1$). Because of this in S no broadcast during time-step t has started earlier than t. Of all messages scheduled in this time-step we choose the one which finishes earliest, say M. From S we construct S' by scheduling M at t and distributing all other broadcasts arbitrarily to the positions $> t$ where M was scheduled in S. By this transformation no broadcast finishes later, only M might finish earlier. Therefore the value of the objective function does not increase. By repeating this step a preemptive solution can be transformed into a non-preemptive one whithout increasing the objective function. With this simple transformation, a polynomial algorithm for the preemptive case would yield one for the non-preemptive case. $\square$

3 Inapproximability of Single-Source Unsplittable Min-Cost Flow

In this section we show that broadcast scheduling with unit-length messages could be solved optimally in polynomial time with a $(2-\varepsilon, 1)$-approximation for single-source unsplittable min-cost flow, i.e., with an algorithm computing a single-source unsplittable flow that does not exceed a given budget B and routes at most $(2-\varepsilon)u_e$ units of flow through each edge e (provided that a single-source unsplittable flow exists that has cost at most B and does not violate any edge capacities).

THEOREM 3.1. *For arbitrary $\varepsilon > 0$ there is no $(2-\varepsilon, 1)$-approximation algorithm for the single source unsplittable min-cost flow problem, unless P=NP.*

Our reduction is inspired by the integral flow model with additional constraints given by Kalyanasundaram, Pruhs and Velauthapillai in [7] to solve single channel broadcast scheduling. By relaxing the flow problem they obtained an approximation algorithm with the ratio mentioned in Section 1. In the following, we briefly recall the flow model of [7].

3.1 The flow model The network for the integral flow problem is constructed in the following way. For each message M_i, $i \in \{1 \ldots n\}$, a row of nodes $v_i(t)$, $t \in \{0 \ldots T+n\}$ is added. A flow of 1 passing through node $v_i(t)$ will correspond to M_i being broadcast at t. The extra n nodes to the right are needed to guarantee that every $R_i(t)$ can be answered.

All edges in the network have capacity 1. Between $v_i(t)$ and $v_i(t')$, $0 \leq t < t' \leq T+n$ an edge $e_i(t,t')$ is added. A flow of 1 passing through this edge corresponds to M_i being broadcast at t and t', and not in between. The weight of the edge is the cumulated ART of the requests $R_i(t) \ldots R_i(t'-1)$, if they are answered at t': $w_i(t,t') := \sum_{j=t}^{t'-1}(t'-j)R_i(j)$.

The source node s is connected to the $v_i(0)$ nodes. The nodes $v_i(t)$, $t \in \{T+1 \ldots T+n\}$ are connected to the destination node d. All these edges have cost 0. A demand of n is requested to be routed integrally splittable from s to d. For each row i of the network a flow of 1 can take one specific path, determining by the touched nodes when M_i is broadcast.

So far this would be solvable in polynomial time. The hardness comes in when the additional constraints are posed that at each t only one $v_i(t)$, $i \in \{1 \ldots n\}$ is touched by a flow of 1. This is necessary to ensure a solution with only one channel.

3.2 Enforcing the additional constraints In [7], the flow problem is written as a linear program and additional linear inequalities are added to ensure that at most 1 unit of flow can pass through all the vertices of $v_i(t)$ of each column t. Contrary to this approach, we now transform the flow model into a single source unsplittable flow problem and add nodes, edges and demands to enforce the constraint that only one vertex of each column can carry nonzero flow. All additional edges again have capacity 1. First of all, the (splittable) demand of n units at node d is replaced by n separate demands of 1 unit at node d. We now will explain how to ensure that each column t of the network (nodes $v_i(t)$, $i \in \{1 \ldots n\}$) is only touched by a flow path of one of these demands.

Every node $v_i(t)$ is replaced by two nodes $u_i(t)$ and $w_i(t)$, where $u_i(t)$ receives all in-edges and $w_i(t)$ all out-edges of $v_i(t)$. An edge with cost 0 from $u_i(t)$ to $w_i(t)$ is added. A zero cost edge is added from a new node $x(t)$ to each $u_i(t)$ and similarly from each $w_i(t)$ to a new node $y(t)$.

Let $\varepsilon' > 0$ be a small constant and L be a large number depending on ε' and the problem input (see below). $n-1$ edges with cost L are added from s to $x(t)$. To avoid obtaining a multi-graph these can be split into two edges, adding new nodes (as in Figure 4). At $y(t)$ $n-1$ demands for $1-\varepsilon'$ units are added. Figure 4 shows a simplified example with three rows.

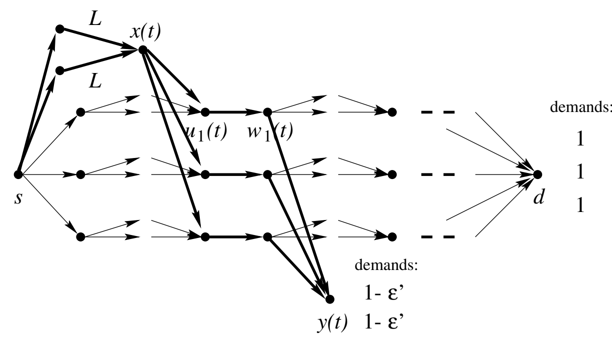

Figure 4: Construction enforcing that at most one flow path touches a column.

If these $n-1$ demands are satisfied via a path that contains $x(t)$, all rows except one will be blocked as wanted. This might be prevented in two cases: First, if a demand at $y(t)$ is routed via $x(t')$, $t' < t$. This will still block a row in column t (and additionally in some previous columns). Second, if one of the demands at d is routed via $x(t)$. Then a demand at $y(t')$ for some $t' \leq t$ must be routed via a row of the network, not touching any $x(t'')$, $t'' \leq t$. Let C be an (easily obtained) upper bound for the cost of a broadcast schedule. If $L > C/\varepsilon'$, this case cannot lead to an optimal cost because the demands at d are greater than the ones for $y(t)$ by ε'.

If there was a $(2-\varepsilon, 1)$-approximation algorithm, with $\varepsilon > 2\varepsilon'$, in all resulting flows no edge would be used by more than one flow path. The minimum cost of a flow that does not violate any edge capacity lies between $X := (T+n) \cdot (n-1) \cdot L \cdot (1-\varepsilon')$ (cost resulting from the demands at the $y(t)$ nodes) and $X + C$. It is clear that the optimal cost will take a value of $X + \delta$, where δ is integer. Therefore an optimal solution for the enhanced integral flow problem could be obtained by a simple binary search over the budget B presented to the approximation algorithm. This would give an optimal solution to the broadcast scheduling problem as well.

4 Approximation of Broadcast Scheduling

In this section, we show that there is a 6-speed 1-approximation algorithm for broadcast scheduling with unit-length messages, i.e., an algorithm that computes a 6-feasible schedule whose ART is at most the ART of the optimal 1-feasible schedule. Previously, no algorithm for computing an $O(1)$-feasible schedule with this property was known.

Kalyanasundaram et al. presented an algorithm which yields a W-speed $W/(W-2)$-approximation for broadcast scheduling, for any $W \geq 3$. We will obtain our result by using their algorithm with $W = 4$ and extending it by two "random channels". From their fractional relaxation of the broadcast scheduling problem (see Section 3) they obtain values $p_i(t)$, for $1 \le i \le n$ and $1 \le t \le T+n$, that can be interpreted as the fraction of message i that is broadcast at time t. (The value $p_i(t)$ is simply the flow through $v_i(t)$ in the fractional relaxation.) These values $p_i(t)$ satisfy the conditions

- $\sum_{i=1}^{n} p_i(t) \le 1$ (the total amount of messages which are broadcast at time t does not exceed 1) and
- if $R_i(t) > 0$, then there exists $t' > t$ such that M_i has been broadcast until t': $\sum_{j=t+1}^{t'} p_i(t) \ge 1$.

The ART of the fractional schedule given by the values $p_i(t)$ is denoted by L_{OPT}. L_{OPT} is a lower bound on the ART of the optimal integral 1-feasible schedule.

The algorithm by Kalyanasundaram et al. is parameterized by a positive value $\varepsilon < \frac{1}{2}$ such that $W = \frac{1}{\varepsilon}$ is an integer. Let $b_i(j)$ be the j-th breakpoint of message i, i.e., the smallest t such that $\sum_{k=1}^{t} p_i(k) \ge j\varepsilon$. Their method of converting the fractional schedule into an integral $\frac{1}{\varepsilon}$-feasible schedule has the following property:

> In any interval $[b_i(j), b_i(j+1)]$ between consecutive breakpoints of the same message, message i is broadcast at least once in the resulting schedule.

Now we are ready to describe our 6-speed 1-approximation algorithm:

Algorithm A

(1) compute the values $p_i(t)$ of the optimal fractional schedule;

(2) on four channels, use the schedule computed by the algorithm from [7] for $\varepsilon = \frac{1}{4}$;

(3) on two channels, select the message to be scheduled at time t randomly with probability $p_i(t)$ for message i.

Let A denote the expected ART of the schedule produced by Algorithm A. We want to show that $A \le L_{\mathrm{OPT}}$.

The ART of the optimal fractional schedule is the sum of the contributions of all requests $R_i(t)$. Fix some $R_i(t) > 0$. Let $t_1, t_2, \ldots, t_\ell$ be the earliest time-steps after time t in which $p_i(t_j) > 0$, $1 \le j \le \ell$, such that $\sum_{j=1}^{\ell-1} p_i(t_j) \le \frac{1}{2}$ and $\sum_{j=1}^{\ell} p_i(t_j) > \frac{1}{2}$. The contribution of $R_i(t)$ to L_{OPT} is at least

$$x := R_i(t) \cdot \left(\sum_{j=1}^{\ell-1} p_i(t_j)(t_j - t) + \left(1 - \sum_{j=1}^{\ell-1} p_i(t_j)\right) \cdot (t_\ell - t) \right). \tag{4.1}$$

Now we consider the expected contribution of $R_i(t)$ to the schedule produced by Algorithm A. On four channels the algorithm uses the schedule produced by the algorithm from [7] for $\varepsilon = \frac{1}{4}$. As remarked above, this ensures that any message i is broadcast in every interval between two consecutive breakpoints of that message. Since $\sum_{j=1}^{\ell} p_i(t_j) > \frac{1}{2}$, the interval $[t+1, t_\ell]$ must contain two consecutive breakpoints. Therefore, it is ensured that message i is broadcast at least once in this interval, so that the requests $R_i(t)$ are satisfied no later than at time t_ℓ.

Furthermore, with $q_j := (1-p_i(t_j))^2$ the probability that $R_i(t)$ is satisfied by one of the two random channels at time t_1 is $1-q_1$. The probability that $R_i(t)$ is not satisfied by the random channels at times $t_1, t_2, \ldots, t_{j-1}$, but is satisfied at time t_j for some $j < \ell$ is

$$\left(\prod_{m=1}^{j-1} q_m \right) \cdot (1 - q_j)$$

The probability that $R_i(t)$ is not satisfied before time t_ℓ on one of the random channels is $\prod_{m=1}^{\ell-1} q_m$. Therefore, the expected contribution of $R_i(t)$ to A is at most:

$$y := \Bigg((1-q_1) \cdot (t_1 - t) + \cdots + \left(\prod_{m=1}^{\ell-2} q_m \right) \cdot (1 - q_{\ell-1}) \cdot (t_{\ell-1} - t) + \left(\prod_{m=1}^{\ell-1} q_m \right) \cdot (t_\ell - t) \Bigg) \cdot R_i(t) \tag{4.2}$$

Now we claim that $y \le x$ always holds.

Lemma 4.1. *Let x and y be as defined by (4.1) and (4.2), where $t < t_1 < t_2 < \cdots < t_\ell$ is an increasing sequence of arbitrary real numbers, $R_i(t)$ is any positive number, and the $p_i(t_j)$ are positive real numbers satisfying $\sum_{j=1}^{\ell-1} p_i(t_j) \le \frac{1}{2}$. Then $y \le x$.*

Proof. The lemma can be proved by induction on ℓ. For $\ell = 1$, we have $x = R_i(t) \cdot (t_1 - t) = y$.

For the induction step, let $\ell \ge 2$ and assume that the claim holds for $\ell - 1$. We rewrite y as a sum of $\ell - 1$ terms instead of ℓ terms by combining the last two terms:

$$R_i(t) \cdot \Bigg((1-q_1) \cdot (t_1 - t) + q_1(1-q_2) \cdot (t_2 - t) + \cdots + \prod_{m=1}^{\ell-2} q_m \cdot \underbrace{((1-q_{\ell-1}) \cdot (t_{\ell-1} - t) + q_{\ell-1} \cdot (t_\ell - t))}_{t'_{\ell-1} - t} \Bigg)$$

By the induction hypothesis, we get

$$y \le R_i(t) \cdot \left(\sum_{j=1}^{\ell-2} p_i(t_j)(t_j - t) + \left(1 - \sum_{j=1}^{\ell-2} p_i(t_j)\right) \cdot (t'_{\ell-1} - t) \right)$$

Then we get

$$\begin{aligned} \frac{x-y}{R_i(t)} &\ge p_i(t_{\ell-1})(t_{\ell-1} - t) + \left(1 - \sum_{j=1}^{\ell-1} p_i(t_j)\right) \cdot (t_\ell - t) \\ &\quad - \left(1 - \sum_{j=1}^{\ell-2} p_i(t_j)\right) \cdot (t'_{\ell-1} - t) \\ &= (t_\ell - t_{\ell-1}) p_i(t_{\ell-1}) \cdot \\ &\quad \left(\left(2 - p_i(t_{\ell-1})\right) \left(1 - \sum_{j=1}^{\ell-2} p_i(t_j)\right) - 1 \right) \end{aligned}$$

This value is positive, because $t_\ell - t_{\ell-1} > 0$, $p_i(t_{\ell-1}) > 0$ and $(2 - p_i(t_{\ell-1}))(1 - \sum_{j=1}^{\ell-2} p_i(t_j)) > 1$. To see the latter, note that $\sum_{j=1}^{\ell-1} p_i(t_j) \le \frac{1}{2}$ implies $\sum_{j=1}^{\ell-2} p_i(t_j) \le \frac{1}{2} - p_i(t_{\ell-1})$ and, thus, $(2 - p_i(t_{\ell-1}))(1 - \sum_{j=1}^{\ell-2} p_i(t_j)) \ge (2 - p_i(t_{\ell-1}))(\frac{1}{2} + p_i(t_{\ell-1})) = 1 + p_i(t_{\ell-1})(\frac{3}{2} - p_i(t_{\ell-1})) > 1$. □

By Lemma 4.1, the expected contribution of $R_i(t)$ to the ART achieved by Algorithm A is at most the contribution of $R_i(t)$ to the ART of the optimal fractional schedule. By linearity of expectation, we obtain that $A \le L_{\mathrm{OPT}}$.

Furthermore, Algorithm A can easily be derandomized by the method of conditional probabilities. In each time-step, there are at most n^2 possible choices for the messages broadcast on the two random channels. For each of these choices, the expected ART of the schedule can be computed in polynomial time, and it suffices to select the choice that minimizes the expected ART. Therefore, we obtain a deterministic algorithm with the same performance guarantee.

THEOREM 4.1. *There is a* 6*-speed* 1*-approximation algorithm for the broadcast scheduling problem with unit-length messages.*

5 Conclusion

We have resolved the complexity of the broadcast scheduling problem by proving its NP-hardness. From this we have derived a $(2-\varepsilon, 1)$ inapproximability result for the single-source unsplittable min-cost flow problem. For broadcast scheduling with unit-length messages, we also showed that an offline algorithm with 6 channels can achieve an average response time that is at least as good as the optimal solution with 1 channel. Regarding future work, it would be interesting to determine the approximability of broadcast scheduling without resource augmentation. The reduction used in our NP-hardness proof does not provide any indication that a good 1-speed approximation algorithm cannot exist.

References

[1] S. Acharya, S. Muthukrishnan. *Scheduling on-demand broadcasts: new metrics and algorithms*, in Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom), 1998, pp. 43–54.

[2] A. Bar-Noy, R. Bhatia, J. Naor, B. Schieber. *Minimizing service and operation costs of periodic scheduling*, in Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1998, pp. 11–20.

[3] Y. Bartal, S. Muthukrishnan. *Minimizing Maximum Response Time in Scheduling Broadcasts*, in Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2000, pp. 558–559.

[4] Y. Dinitz, N. Garg, M. Goemans. *On the single source unsplittable flow problem*, Combinatorica, 19 (1999), pp. 1–25.

[5] J. Edmonds, K. Pruhs. *Broadcast Scheduling: When Fairness is Fine*, in these proceedings.

[6] B. Kalyanasundaram, K. Pruhs. *Speed is as powerful as clairvoyance*, Journal of the ACM, 47 (2000), pp. 617–643.

[7] B. Kalyanasundaram, K. Pruhs, M. Velauthapillai. *Scheduling Broadcasts in Wireless Networks*, in Proceedings of the 8th Annual European Symposium on Algorithms (ESA), 2000, pp. 290–301, Springer LNCS 1879.

[8] C. Kenyon, N. Schabanel. *The data broadcast problem with non-uniform transmission times*, in Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1999, pp. 547–556.

[9] C. Kenyon, N. Schabanel, N. Young. *Polynomial-Time Approximation Scheme for Data Broadcast*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC), 2000, pp. 659–666.

[10] C. Phillips, C. Stein, E. Torng, J. Wein. *Optimal time-critical scheduling via resource augmentation*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC), 1997, pp. 140–149.

[11] N. Schabanel. *The data broadcast problem with preemption*, in Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science (STACS), 2000, pp. 181–192, Springer LNCS 1770.

[12] M. Skutella. *Approximating the single source unsplittable min-cost flow problem*, in Mathematical Programming B (to appear). Extended abstract in the proceedings of FOCS, 2000, pp. 136–145.