Approximation Algorithms for Bandwidth and Storage Allocation Problems under Real Time Constraints

Stefano Leonardi* Alberto Marchetti-Spaccamela * Andrea Vitaletti *

September 11, 2000

Abstract

The problem we consider is motivated by allocating bandwidth slots to communication requests on a satellite channel under real time constraints. Accepted requests must be scheduled on non-intersecting rectangles in the time/bandwidth Cartesian space with the goal of maximizing the benefit obtained from accepted requests. This problem turns out to be equal to the maximization version of the well known Dynamic Storage Allocation problem when storage size is limited and requests must be accommodated within a prescribed time interval.

We present constant approximation algorithms for the problem introduced in this paper using as a basic step the solution of a fractional Linear Programming formulation. We also show the extension to the case in which multiple channels are available for accepting communication requests.

This problem has been independently studied by Bar-Noy et al [BNBYF$^+$00]; our techniques are different and allow to improve the better approximation ratio.

*Dipartimento di Informatica Sistemistica, Università di Roma "La Sapienza", via Salaria 113, 00198-Roma, Italia. E-mail:{leon,marchetti,vitale@dis.uniroma1.it}

1 Introduction

The problem we study in this paper has been encountered in the context of the EU research project Euromednet on scheduling requests for remote medical consulting on a shared satellite channel [Eme]. Every request asks for a number of contiguous bandwidth slots to provide every end user involved in the consulting with a TCP/IP satellite channel. Bandwidth is assigned in slots of 64 kb/sec. The number of slots per end user depends on the type of service desired (typical values are 64 kb/sec for common internet services – 384 Kb/sec for audio/video.) At most 48 slots of 64 Kb/sec are available on the channel in this specific application. Requests also specify a duration of the consulting (typical values are from 1/2 hour to 2 hours) to be allocated within a time interval specified in the request. Requests are typically issued a few days in advance. The service manager will reply soon with a positive or a negative answer on the basis of the pending requests and of the requests already accepted for scheduling. Every accepted request will be allocated starting from a base bandwidth for a contiguous number of slots along a time duration within the indicated time interval. Other details regarding this specific application are available to [Eme].

The problem encountered in this application is a natural interesting combinatorial problem: if we consider a Cartesian space with the bandwidth on the ordinate and the time on the abscissa then every accepted request is scheduled on a rectangle of basis equal to the duration and height equal to the bandwidth requested. Accepted requests must observe the *packing constraint*, that is any two rectangles are non-overlapping. A benefit is also assigned to every request, modelling its relevance or the economic revenue from its acceptance. The objective is to maximize the overall benefit of accepted requests. We denote this problem as the Rectangle Packing (RP) problem in the sequel of the paper.

This problem is related to a number of well studied combinatorial problems. Consider the machine scheduling problem with real time constraints in which every job asks to be scheduled for a given duration between a release time and a deadline. Only one job can be scheduled at any time on every single machine. A benefit is associated with every job with the goal of maximizing the benefit obtained from scheduled jobs. This is an old NP-hard scheduling problem [GJ79] studied in many variants [LLKS93]. Very recently the first constant approximation algorithms has been proposed for this problem [BNGNS99] both on single and parallel machines.

A second related problem is the Dynamic Storage Allocation problem (DSA). DSA concerns the dynamic allocation of contiguous areas in a storage device. In DSA a set of requests for a contiguous area of memory along a specified time duration has to be accommodated while minimizing the storage space required. DSA is a classical problem in computer science [Knu73] whose study backs to the sixties. The rectangle packing problem is a maximization version of DSA where we have to allocate bandwidth rather than storage space. In RP storage space is limited. On the other end we insert real time constraints on the temporal allocation of the process. We believe that this version of DSA is of relevance in many practical settings. DSA has been shown to be tightly related to interval graph coloring. Kierstead and Slusarek [Kie91, Slu89] proposed 3-approximation algorithms for aligned DSA, where the storage space required is a power of 2. More recently Gergov [Ger96] proposed a 5/2 approximation algorithm for aligned DSA that implies a 5 approximation for DSA and Claimed a 3 approximation algorithm [Ger99].

A third closely related problem is the call control problem on linear networks [GGK97, BNCK$^+$95, AA99]. This problem has been typically considered in its on-line version. At any step a request for establishing a connection between two vertices on the line network with a given bandwidth is presented. The algorithm has to accept or reject the request without knowledge of the requests presented in the future. If the request is accepted, a given benefit is obtained. The objective is to maximize the obtained benefit without violating the bandwidth constraint on any link of the network. In the call control problem every request must be assigned on a fixed path in the line network, while in the RP problem some slackness in the time allocation may be allowed. However, the major difference from RP is that in call control we only require the bandwidth constraint rather than the stronger packing constraint of the RP problem.

Results of the paper. We present a 12 approximation algorithm for the RP problem. As a basic step of the algorithm we solve a fractional LP problem in which we only enforce the bandwidth constraint and requests can be fractionally accepted. We then show that the optimal fractional solution is a convex combination of a set of integral solutions of which we select that with highest benefit. The obtained solution may still contain intersecting rectangles. However we show that it can be partitioned into three feasible solutions of which we select the best one as final solution of the algorithm. The approximation ratio we obtain is 6 if the bandwidth request is a power of 2, 12 in the general case. The proposed solution runs in pseudopolynomial time. It can be transformed into a fully polynomial time algorithm at the expenses of a small increase in the approximation ratio. We also show a combinatorial algorithm with approximation ratio arbitrarily close to 26. This algorithm uses as a basic step the combinatorial algorithm devised in Bar-Noy et al. [BNBYF$^+$00].

Independently from our paper, Bar-Noy et al. [BNBYF$^+$00] proposed a 35 approximation for our problem that they call Benefit DSA. Their approach is to solve a version of the problem where requests are either accepted or rejected in an integral sense, while the packing constraint is relaxed to the milder bandwidth constraint. A solution of this problem is then combined with an algorithm for the DSA problem. In a later version of their paper they improve the result to a $6\gamma - 1$ combinatorial approximation and to a $6\gamma - 3$ LP-based approximation, where γ is the approximation ratio for DSA. If we consider the 5-approximation for DSA of [Ger96] this yields respectively a 29 combinatorial and a 27 LP-based approximation for the problem. If we consider the 3-approximation Claimed in [Ger99], this yields a $17 + \epsilon$ combinatorial and a 15 LP-based approximation for the problem.

We finally show how to extend our algorithm to the multiple channel case for bandwidth allocation or, equivalently, to the multiple storage devices case in the DSA problem.

Structure of the paper. In Section 2 we formally describe the RP problem. In Section 3.1 we describe the LP based approximation algorithm for the RP problem. In Section 4 we present a combinatorial version of the algorithm. Finally, in Section 5 we describe the extension to multiple channels.

2 The RP problem

Given an input set of n requests $\{< r_i, d_i, b_i, l_i, \omega_i >\}_1^n$, where $r_i, d_i, b_i, l_i, \omega_i$ are integers, the generic request asks for a bandwidth interval of size b_i in $[0, B]$ along a continuous time interval of length l_i contained in $[r_i, d_i]$. A request can be either accepted or rejected. A request that is accepted is scheduled on a bandwidth interval $[f(i), f(i) + b_i]$ and a time interval $[t(i), t(i) + l_i]$ and a benefit ω_i is accrued. An accepted request is represented with a rectangle of basis l_i and height b_i on a Cartesian space having the time on the abscissa and the bandwidth on the ordinate. The schedule must observe the constraint that any two rectangles are non-overlapping. In the following we denote by *packing constraint* the constraint that two rectangles are non-overlapping. The packing constraint also implies the *bandwidth constraint* imposing that the overall bandwidth allocated at time t cannot exceed B. In the following we assume $B = 1$. In the aligned version of the RP all the bandwidths are powers of two. The objective of the algorithm is to maximise the overall profit obtained from accepted requests.

3 An LP based approximation algorithm

We present an LP based approximation algorithm for RP. We first round all the bandwidth requests to the nearest higher power of 2. As a basic step of the algorithm we solve a fractional LP problem in which we only enforce the bandwidth constraint. We then show that the optimal solution to the fractional RP problem is a convex combination of a set of integral solutions and that the best among these solution is a schedule that has benefit at least 1/2 the optimum to the LP problem. The selected solution S_{best} may allow rectangle intersection since the packing constraint has not been imposed, but it satisfies the following definition of stability.

Definition 3.1 *Given a schedule of requests , the support at time t', denoted by support(t'), is the maximum value such that there exists a set of j non-overlapping requests $i_1, i_2, .., i_j$ scheduled at time t' for which $f(i_1) = 0$, $f(i_k) = f(i_{k-1}) + b_{i_{k-1}}$, $k = 2, .., j$, $f(i_j) = support(t')$.*

Request i^t is (h, t') stable if h is the maximum value of $support(t''), t'' \in [t, t+l_i]$ and $support(t') = h$.

A schedule of requests is stable if for all requests i_j, i_j is (h_i, t_i) stable for some h_i and t_i.

The geometrical interpretation of a (h, t) stable request is rectangle placed on the top of a pile of non-overlapping rectangles of total bandwidth h (see Figure 1).

In the final step fo the algorithm we will show that S_{best} can be decomposed into three feasible solution of which we select that with highest benefit that will form the final solution to the problem. The obtained solution is a 6 approximation for the aligned version and a 2 approximation for the original problem.

Thus the algorithm consists of three main steps:

1. Solve the LP formulation;
2. Find a stable solution S_{best};
3. Obtain a feasible solution.

3.1 The LP formulation

In this section we present the LP formulation we use as a basic step for the solution of the RP problem.

We first round every bandwidth request to the lowest higher power of two, namely $\overline{b_i} = \min_k \left\{ \frac{1}{2^k} : \frac{1}{2^k} \geq b_i \right\}$. Variables x_{it}, $t = r_i, .., d_i - l_i$, are associated with request i. Variable x_{it} is ranging in $[0, 1]$, and denotes the schedule of request i with $t(i) = t$. Every request can be fractionally scheduled along a set of intervals for an overall value not exceeding one, thus we impose $\sum_{t=r_i}^{d_i - l_i} x_{it} \leq 1$. Denote by T the latest deadline of a request. In the LP formulation we also impose the bandwidth constraint at any time $t \in 1, .., T$, namely that the overall bandwidth assigned to the requests fractionally scheduled at time t is at most 1.

$$\max \sum_{i=1}^{n} \sum_{t=r_i}^{d_i - l_i} \omega_i x_{it}$$

$$\sum_{i,t':t\in[t',t'+l_i)} \overline{b}_i x_{it'} \leq 1, \quad \forall \quad t$$
$$\sum_{t=r_i}^{d_i-l_i} x_{it} \leq 1, \quad \forall \quad i$$
$$x_{it} \in [0,1], \quad \forall \quad t,i$$
$$x_{it} = 0, \quad \forall \quad i,t \notin [r_i, d_i - l_i]$$

We will denote with x_{it} both a variable and its value. The optimum of the LP problem is related to the optimum of the RP problem by the following Lemma:

Lemma 3.1 $OPT(LP) \geq OPT(RP)/2$.

Proof: Consider a new formulation LP_1 obtained from LP by replacing the bandwidth constraint with

$$\sum_{i,t':t\in[t',t'+l_i)} b_i x_{it} \leq 1, \forall t \tag{1}$$

and by imposing the integrality constraints $x_{it} \in \{0,1\}$. Observe that any solution to RP is also a solution to LP_1 for which $OPT(LP_1) \geq OPT(RP)$. Since $\overline{b_i} \leq 2b_i$, any solution to LP_1 with values $\frac{x_{it}}{2}$ is also a solution to LP, with a benefit at least $\frac{1}{2}$ of the benefit of LP_1. From a solution to LP_1 we obtain a solution to LP of value at least $\frac{1}{2}$ of the value of the solution to LP_1. Therefore we have $OPT(LP) \geq OPT(LP_1)/2 \geq OPT(RP)/2$. ■

Observe that in if the bandwidth request is a power of $1/2$ the optimum of LP is always bigger than the optimum of RP, for which in the aligned version of RP we save a factor of 2 in the approximation ratio.

3.2 The Algorithm for obtaining a stable solution

In this section we present the LP based algorithm for the RP problem. We denote by i^t request i scheduled at time t.

Given an optimal LP solution we denote by α the largest rational such that x_{it} is an integer multiple of α. Let x_{it} be the value of a variable in the optimal LP solution.

The algorithm for obtaining a good stable solution first finds $\frac{2}{\alpha}$ integral solutions $S_1, .., S_{\frac{2}{\alpha}}$ of possibly intersecting rectangles and then chooses the one with highest benefit. Namely the algorithm is composed of the following steps:

1. Order the non-zero x_{it} by non increasing $\overline{b}_i$.
2. Replicate $\frac{x_{it}}{\alpha}$ times request i^t.
3. Assign every replication of i^t to a solution with the following algorithm:
   (a) Select those solutions not containing a copy of i that we indicate by $S_1, ..., S_m$.
   (b) Merge the m solution $S_1, ..., S_m$ of bandwidth 1 into a single solution $\mathcal{S}$ of bandwidth m.(The relative order of the solution is not relevant for the algorithm.)
   (c) Let the replication of i^t be (h, t') stable in $\mathcal{S}$.

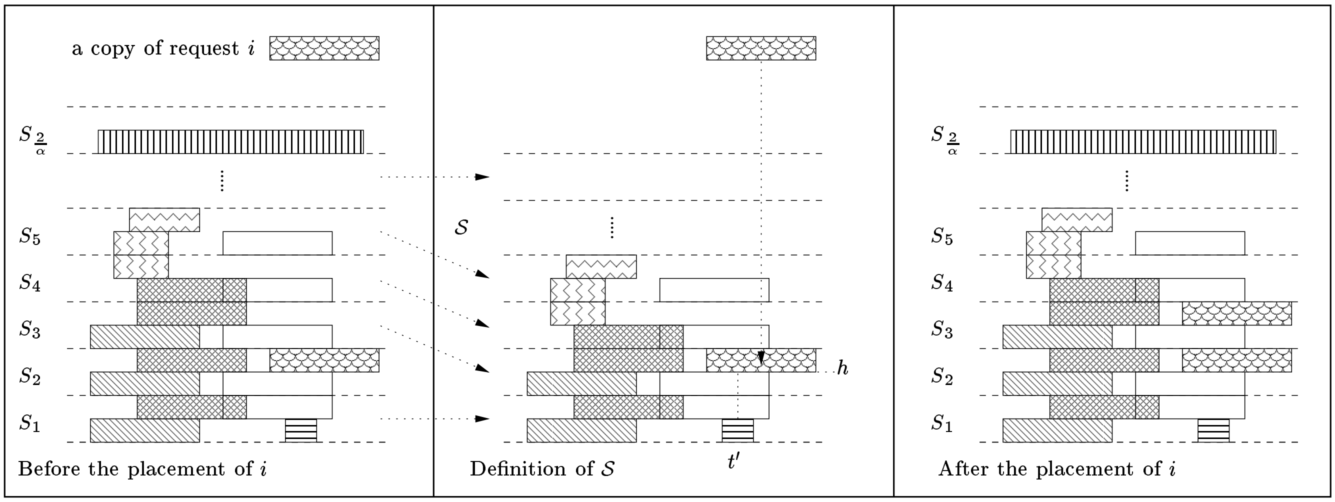

Figure 1: The copy of request i is (h, t') stable in $\mathcal{S}$.

(d) Assign the copy of i^t to solution $S_{\lfloor h \rfloor + 1}$ with $f(i^t) = h \bmod 1$.

4. Select the solution S_{best} with highest benefit out of the $2/\alpha$ solutions.

In the figure the algorithm has to place a copy of request i. The algorithm selects all the solutions with the exception of S_2 that already contains a copy of request i. The copy of request i is (h, t') stable in $\mathcal{S}$ with $\lfloor h \rfloor + 1 = 2$, so it is assigned to S_3.

We observe that the simpler algorithm that places i^t in the first solution where i^t is (h, t) stable with $h \leq 1 - \overline{b}_i$ is not able to place all the copies of the requests in $2/\alpha$ solutions. On the contrary, our algorithm will allow to claim that if a copy cannot be placed in any solution then the bandwidth constraint is violated in the LP problem, thus a contradiction.

The number of solutions we construct may not be bounded by a polynomial. The algorithm constructs a polynomial number of solutions if $1/\alpha$ is rounded at the closest lower value $\frac{1}{n^2 T}$. The resulting degradation in the approximation ratio is negligible.

In the following we show that every copy of a i^t is assigned to a solution S_j, $j \leq m$ and that each integral solution satisfies the bandwidth constraint, namely for every copy of i^t, $f(i^t) + \overline{b}_i \leq 1$.

Lemma 3.2 *Every copy of i^t is placed in a solution S_j, $j \leq m$.*

Proof: At most $1/\alpha$ distinct copies of i^t need to be placed for every request i. Since $2/\alpha$ solutions are available, at least $m \geq 1/\alpha + 1$ solutions $S_1, .., S_m$ not containing a copy of i are available for a single i^t. We prove that i^t is (h, t') stable for a value $h < m$. Assume by contradiction $h \geq m$. It follows that at time t' the whole bandwidth has been assigned on all the m solutions, namely for any S_j, $\sum_{i^t \in S_j : t' \in [t, t+l_i]} \overline{b_i} = 1$.

From the packing constraint in LP, at time t' we have : $\alpha \sum_{i,t: t' \in [t, t+l_i]} \frac{\overline{b_i} x_{it}}{\alpha} \leq 1$.

It follows that at time t':

$$1 \geq \alpha \sum_{i,t:t' \in [t,t+l_i]} \frac{x_{it}\overline{b_i}}{\alpha} = \alpha \sum_{S_j} \sum_{i^t \in S_j : t' \in [t,t+l_i]} \overline{b_i} \geq m\alpha \geq 1 + \alpha,$$

thus a contradiction. ■

Lemma 3.3 *For each solution S_j and each time t', support(t') and $1 - $support$(t')$ are either zero or a multiple of $\overline{b}_i$, where $\overline{b}_i$ is the size of the last request scheduled at time t'.*

Proof: We first prove the Lemma for *support*(t'). Since the requests are presented in non-increasing order and we consider the aligned version of the problem, each request scheduled at time t' is a multiple of those request scheduled at time t' afterwards. this implies that *support*(t') is a multiple of the size of the last request scheduled at time t'

Suppose that the size of the last request scheduled at time t' is $1/2^k$ for some positive integer k, this implies that *support*$(t') = a/2^k$ for some positive integer a less than 2^k. We have

$$1 - support(t') = 1 - a/2^k = (2^k - a)/2^k.$$

this completes the proof. ■.

Lemma 3.4 *For any replication of any request i^t, $f(i^t) + \overline{b}_i \leq 1$.*

Proof: By definition of the algorithm each replication is scheduled at $f(i^t)$ if it is $(f(i^t), t')$ stable with $f(i^t) < 1$. By the previous Lemma we have that $1 - f(i^t)$ is a multiple of the last request assigned at time t'; since we consider the aligned version of RP and the replications are placed by decreasing height it follows that $1 - f(i^t)$ is a multiple of $\overline{b}_i$ and the thesis follows. ■

3.3 Obtaining a feasible solution

We now show how to obtain a feasible solution from the S_{best} solution that has been obtained in the previous step.

We first construct the intersection graph of solution S_{best} by assigning a vertex to every rectangle and connecting with an edge every pair of vertices representing intersecting rectangles. We then show that the obtained intersection graph is 3-colourable and that this can be done in polynomial time. By choosing the set of rectangles of the same color having maximum benefit we obtain a collection of non-overlapping intervals in which every request is scheduled at most once.

The algorithm is as follows:

1. Construct the intersection graph of solution S_{best};
2. Colour the intersection graph with three colours;
3. Accept those requests coloured with same colour having total highest benefit;

Note that in the final solution to any rectangle's height $\overline{b}_i$ is given the original b_i.

Lemma 3.5 *The intersection graph of a solution S_j is 3-colourable.*

Proof: We first give two preliminary Claims. The first Claim states that if i and j are overlapping requests the two associated time intervals are not nested.

Claim 3.1 *For any two overlapping requests i, j, it never holds $[t(i), t(i)+l_i] \subseteq [t(j), t(j)+l_j]$.*

Proof: Let i be (h_i, t_i) stable and j be (h_j, t_j) stable. Assume by contradiction $[t(i), t(i)+l_i] \subseteq [t(j), t(j)+l_j]$. If i is placed before j then $h_j \geq h_i + \overline{b}_i$, contraddicting the fact that i and j overlap. If j is placed before i then assume that j is (h_j, t') stable. Since $[t(i), t(i)+l_i] \subseteq [t(j), t(j)+l_j]$ we have that if i is (h_i, t'') stable then either $h_i \geq h_j + \overline{b}_j$ or $h_i < h_j$. In the first case we contradict the fact that i and j overlap. In the second case observe that since $\overline{b}_i \leq \overline{b}_j$, by Lemma 3.3 h_i and h_j are both multiples of $\overline{b}_i$; since $h_i < h_j$ it follows that $h_j - h_i \geq \overline{b}_i$, contraddicting the fact that i and j overlap. ■

Claim 3.2 *The maximum clique size of the intersection graph is 2.*

Proof: Assume by contradiction that requests i, j and k form a clique of size 3 and that k is assigned to a solution after i and j. Assume i is (h_i, t_i) stable and j is (h_j, t_j) stable. Request k must be completely contained in the interval (t_i, t_j), otherwise k is either $(h_i + \overline{b_i}, t_i)$ stable or $(h_j + \overline{b_j}, t_j)$ stable, thus it does not overlap with i or j.

Without loss of generality, assume $h_i \leq h_j$ and k is (h_k, t_k) stable. We now observe that $h_h < h_i$. In fact if $h_k \geq h_i$, since k is completely contained in (t_i, t_j), it follows that $h_k \geq h_i + b_i$ contraddicting the fact that i and k overlap. To complete the proof we observe that by Lemma 3.3 $(1 - h_k)$ is a multiple of $\overline{b_k}$ and h_i is a multiple of $\overline{b_i}$ and h_k is a multiple of $\overline{b_k}$. Since $\overline{b_i} \geq \overline{b_k}$, then $\overline{b_i}$ is a multiple of $\overline{b_k}$ and we have that $h_i - h_k \geq b_k$. This contraddicts the facts that k and i overlap. ■

We finish the proof of Lemma 3.5 by showing that all the requests assigned to a solution can be coloured with three colours. We colour requests in order of decreasing bandwidth. Repeatedly, until all the requests are coloured, consider the requests of highest bandwidth $1/2^k$ not yet coloured. We separately consider requests with different $f(i) = \overline{f}$. Lemma 3.3 implies that request with the same $\overline{b_i}$ and different $f(i)$ do not overlap. Consider the requests of bandwidth $1/2^k$ assigned with $f(i) = \overline{f}$.

Order the requests by starting point from left to right. By Claim 3.1 this is a well defined order. A sequence is a set of overlapping requests of the same bandwidth $1/2^k$ and same $f(i) = \overline{f}$. Consider the requests in the order from left to right. By Claim 3.1 and by Claim 3.2, overlaps are possible only between consecutive requests in a sequence, or between the leftmost request in a sequence and the left hand side coloured request of higher bandwidth and between the rightmost request in a sequence and the right hand side coloured request of higher bandwidth. It follows that a sequence can be consistently coloured using the three available colours. Then we move to colour the next sequence of requests of bandwidth $1/2^k$ scheduled with $f(i) = \overline{f}$ until this specific set of requests is terminated. ■

We finally show the approximation ratio we obtain.

Theorem 3.6 *The algorithm for the RP problem is 12-approximated.*

Proof: We first prove that the algorithm selects a solution S_{best} whose benefit is at least $OPT(LP)/2$: $OPT(LP) = \sum_{S_l} \sum_{i^t \in S_l} \alpha\omega_i \leq \frac{2}{\alpha} \sum_{i^t \in S_{Best}} \alpha\omega_i \leq 2 \sum_{i^t \in S_{Best}} \omega_i$.

By Lemma 3.1 $OPT(RP) \geq OPT(LP)/2$, for which the benefit of S_{best} is at least $1/4$ of $OPT(RP)$. Moreover we colour the requests of S_{best} with 3 colours and select the set of intervals with same colour of highest benefit, for which the final solution has a benefit of at least $1/3$ of the benefit of S_{best}. Altogether we obtain an approximation ratio of 12. ■

Observe that for the aligned version of RP in which all the b_i's are powers of 2 we obtain an approximation ratio of 6.

We finally observe that the number of constraints in the LP formulation is $O(nT)$, thus leading to a pseudopolynomial algorithm. Bar-Noy et al.[BNGNS99] showed how to reduce the number of time slots to a polynomial in n in a LP formulation for the maximum throughput scheduling problem under real-time constraints. The application of their technique to our case allows to express the LP solution as a convex combination of $3/\alpha$ rather than $2/\alpha$ integral solutions, therefore leading to an integral solution with a benefit at least $1/3$ of the optimum LP. This results in an approximation ratio of 18 for the general version and of 9 for the aligned version.

4 A combinatorial algorithm

In this section we show how to replace the basic step of the approximation algorithm based on the solution of a fractional LP formulation with a combinatorial algorithm that delivers a constant approximation solution to the LP problem.

We partition the requests into *wide* requests, that ask at least $1/2$ of the available bandwidth, and *narrow* requests whose bandwidth requirement is less than $1/2$.

We solve the RP problem separately for wide requests and narrow requests and we choose the best solution. If all requests are wide then RP is equivalent to interval scheduling for which a 2 approximation algorithm is known [Spi99]. For narrow requests we will obtain a $24+1/k$ approximation algorithm. It follows that the best between the two solutions on wide requests and narrow requests is $26 + 1/k$ approximated.

In the following we show the algorithm on narrow requests. Given an instance of the RP problem on narrow requests we first divide each request in k identical requests each one with a $1/k$ of the bandwidth and of the profit of the original request and with the same time requirements. Similarly to the algorithm of [BNBYF$^+$00], we relax the packing constraint into the milder bandwidth constraint, but still keep the integrality constraint. Note that each integral feasible solution to this modified problem is also a feasible solution of the LP formulation presented before (where each variable has a fractional value that is a multiple of $1/k$).

We now apply the unified algorithm of [BNBYF$^+$00] for finding an approximate solution to the above defined problem; Lemma 3.2 of [BNBYF$^+$00] implies the following Lemma.

Lemma 4.1 *For each integer k there exists a combinatorial algorithm that finds a $2+1/k$ approximate solution to the LP formulation.*

The analysis of [BNBYF$^+$00] implies that the running time of the algorithm is polynomial in n and k.

The above Lemma and the analysis of Section 3.2 implies that, if all requests have a bandwidth requirement less than $1/2$, then there exists a $24 + 1/k$ approximation algorithm to the RP problem. Taking into account also wide requests we obtain:

Theorem 4.2 *For each k there exists a $26 + 1/k$ combinatorial approximation algorithm for the RP problem.*

5 The multiple channel case

In this section we assume that m channels, each one with a bandwidth $B_j \leq 1$, are available. For the sake of simplicity we assume B_j's to be powers of 1/2. We briefly sketch the extension of known techniques [BNGNS99, BEY98, AAF$^+$96], to obtain a $c+1$ throughput maximization approximation algorithm for m parallel unrelated machines provided a c algorithm for a single machine. We consider a Linear Programming formulation with variables x_{ijt} indicating the allocation of request i at time t on machine j. We set $x_{ijt} = 0$ for those machines j where $\overline{b}_i > B_j$. We then solve the LP problem and apply our rounding algorithm in order from channel 1 to channel m while we disregard on channel j requests already accepted on a previous channel. We then conclude with the following theorem:

Theorem 5.1 *Provided a c approximation algorithm for the RP problem on a single channel, there exists a $c+1$ approximation algorithm for the RP problem on multiple channels.*

References

[AA99] R. Adler and Y. Azar. Beating the logarithmic lower bound: randomized preemptive disjoint paths and call control algorithms. In *Proc. of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages 1–10, 1999.

[AAF$^+$96] B. Awerbuch, Y. Azar, A. Fiat, S. Leonardi, and A. Rosén. On-line competitive algorithms for call admission in optical networks. In *Proceedings of the 4th Annual European Symposium on Algorithms, Lecture Notes in Compter Science 1136*, pages 431–444. Springer-Verlag, 1996.

[BEY98] A. Borodin and R. El-Yaniv. *On-line Computation and Competitive Analysis.* Cambridge University Press, 1998.

[BNBYF$^+$] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling.

[BNCK$^+$95] Amotz Bar-Noy, Ran Canetti, Shay Kutten, Yishay Mansour, and Baruch Schieber. Bandwidth allocation with preemption. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 616–625, Las Vegas, Nevada, 29 May–1 June 1995.

[BNGNS99] Amotz Bar-Noy, Sudipto Guha, Joseph (Seffi) Naor, and Baruch Schieber. Approximating the throughput of multiple machines under real-time scheduling. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 622–631, Atlanta, GA USA, 1–4 May 1999.

[Eme] http://www.estec.esa.nl/artes3/projects/12telbios/telbios.htm.

[Ger96] Jordan Gergov. Approximation algorithms for dynamic storage allocation, 1996.

[Ger99] Jordan Gergov. Algorithms for compile-time memory optimization, 1999.

[GGK97] Juan A. Garay, Inder S. Gopal, and Shay Kutten. Efficient on-line call control algorithms. *Journal of Algorithms*, 23(1):180–194, April 1997.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[Kie91] H. A. Kierstead. A polynomial time approximation algorithm for dynamic storage allocation. *Disccrete Mathematics*, 88:231–237, 1991.

[Knu73] D.E. Knuth. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 2nd Edition.* Addison-Wesley, 1973.

[LLKS93] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin, editors, *Handbooks in Operations Research and Management Science Vol. 4:Logistics of Production and Inventory.* North Holland, 1993.

[Slu89] M. Slusarek. A coloring algorithm for interval graphs. In *Proc. of the 14th Mathematical Foundations of Computer Science*, pages 471–480, 1989.

[Spi99] F.C.R. Spieksma. On the approximability of an interval scheduling problem. *Journal of Scheduling*, 2:215–2227, 1999.