Using Fractional Primal-Dual to Schedule Split Intervals with Demands

Reuven Bar-Yehuda* Dror Rawitz†

April 9, 2004

Abstract

Given a limited *resource* (such as bandwidth or memory) we consider the problem of scheduling requests from clients that are given as groups of non-intersecting time intervals. Each request j is associated with a *demand* (the amount of resource required), d_j, a *t-interval*, which consists of up to t segments, for some $t \geq 1$, and a weight, $w(j)$. A schedule is a collection of requests. It is feasible if for every time instance s, the total demand of scheduled requests whose t-interval contains s does not exceed 1, the amount of resource available. Our goal is to find a feasible schedule that maximizes the total weight of scheduled requests. This problem generalizes many problems from the literature, and show up in a wide range of applications.

We present a $6t$-approximation algorithm for this problem that uses a novel extension of the primal-dual schema we call *fractional primal-dual*. A fractional primal-dual algorithm produces a primal solution x, and a dual solution y, whose value divided by r, the approximation ratio, bounds the weight of x. However, y is not a solution of the dual of an LP relaxation of the problem. The algorithm induces a new LP that has the same objective function as the original problem, but contains inequalities that may not be valid with respect to the original problem. y is a feasible solution of the dual of the new LP. x is r-approximate, since some optimal solution of an LP relaxation of the original problem is in the feasible set of this new LP.

We present a fractional local ratio interpretation of our $6t$-approximation algorithm. We also discuss the connection between fractional primal-dual and the *fractional local ratio technique*. Specifically, we show that the former is the primal-dual manifestation of latter.

*Department of Computer Science, Technion, Haifa 32000, Israel. E-mail: reuven@cs.technion.ac.il.

†Department of Electrical Engineering, Tel-Aviv University, Tel-Aviv 69978, Israel. E-mail: rawitz@eng.tau.ac.il.

1 Introduction

1.1 Problem Definition and Background

We consider the problem of scheduling jobs that are given as groups of non-intersecting intervals on the real line. Each job j is associated with a *t-interval*, which consists of up to t segments, for some $t \geq 1$, and a positive weight, $w(j)$. Each job requires the utilization of a given limited *resource*. The amount of resource available is fixed; we normalize it to unit size for convenience. The amount of resource required by job j, or the *demand* of j, is denoted by $d_j \in [0, 1]$. A *schedule* is a collection of jobs. It is *feasible* if for every $s \in \mathbb{R}$ the total demand of the jobs in the schedule whose t-interval contains s does not exceed 1. Our goal is to find a feasible schedule that maximizes the total weight of scheduled jobs.

The problem of scheduling t-intervals is NP-hard, even when $t = 1$, since it contains *knapsack* [12] as a special case in which all t-intervals intersect. The problem of scheduling t-intervals where all demands are equal to 1 was studied by Bar-Yehuda et al. [6]. Two jobs are said to be in *conflict* if any of their segments intersect. The objective in this special case is to schedule a subset of non-conflicting jobs whose total weight is maximum. In [6] Bar-Yehuda et al. formulate the unit-demand special case as the problem of finding *maximum weight independent set* (MWIS) in a *t-interval graph*. Note that when $t = 1$ this problem is equivalent to MWIS in interval graphs, and hence solvable in polynomial time. Bar-Yehuda et al. [6] presented a $2t$-approximation algorithm for MWIS in t-interval graphs that uses a new extension of the local ratio technique called *fractional local ratio*.

An interesting special case that was discussed in [6] is the family of t-union graphs, in which the segments associated with each job (or vertex) can be labeled in such a way that for any two jobs j_1 and j_2 the segment i_1 of j_1 and segment i_2 of j_2 do not intersect for every $1 \leq i_1, i_2 \leq t$ and $i_1 \neq i_2$. In this case the t segments can be viewed as intervals on orthogonal axes, corresponding to a t-dimensional box. Two boxes are in conflict if their projections on any of the t axes intersect. For example, in Figure 1, jobs 1 and 2 are in conflict, while jobs 1 and 3 are not. (Jobs 2 and 3 are also in conflict, but when arbitrary demands are allowed, they can be scheduled together, since the sum of their demands is 1.) Bar-Yehuda et al. [6] showed that the class of degree-3 graphs is contained in the class of t-union graphs. Since *maximum independent set* is APX-hard on degree-3 graphs [9, 18], MWIS on t-interval graphs is also APX-hard. They proved that the k-dimensional matching problem is equivalent to MWIS in the special class of k-union graphs of unit segments. They also pointed out that k-dimensional matching cannot be approximated within an $O(k/\log k)$ ratio unless P=NP [19], while the best known approximation ratio is $k/2 + \epsilon$, for any $\epsilon > 0$ [21]; and that 3-dimensional matching is APX-hard [23]. We note that it is NP-complete to determine whether a given graph is t-interval [25], or t-union [16], for any fixed $t \geq 2$. However, the hardness results remain true with respect to the problem of scheduling t-intervals with (or without) demands, since the constructions in [6] are made using t-interval scheduling instances.

We describe several practical applications in which the problem of scheduling t-intervals with demands arises. First, consider a multimedia-on-demand system with a limited bandwidth through which movies are broadcasted to clients (e.g., through a cable TV network). Each movie has a different bandwidth demand, which may depend on its quality. Each client requests a particular movie, and specifies the time at which he (or she) would like to start watching it. Moreover, each client specifies at which times he plans to take breaks. In the throughput maximization version of this problem, we aim to maximize the revenue by deciding which movies to broadcast, subject to the constraint that the bandwidth demands at any given moment can be supplied by the system. Another application is allocation of linear resources [17]. Requests for a *linear* resource can be

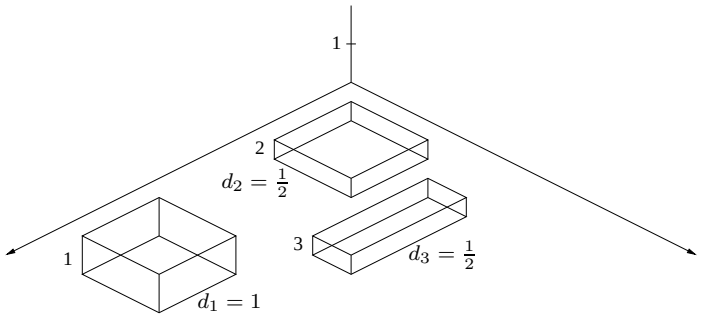

Figure 1: Scheduling 2-dimensional boxes with demands.

modeled as intervals on the line. (For example, a disk drive is a linear resource when requests are for contiguous blocks [24].) Consider a scenario in which the jobs are requests from several linear resources, and two jobs are in conflict if their requests on one of the resources overlap. (Note that this example corresponds to a t-union graph.) Our goal is to schedule as many jobs as possible such that the total demand from any linear resource at any given moment is not more than 1. The demand in this case can model transmission rate. For more details and applications, such as genomic sequence similarity [2], and computational geometry, see [6].

1.2 Our Results

In Section 2 we present a $6t$-approximation algorithm for the t-interval scheduling problem with demands that is based on a novel and non-standard extension of the *primal-dual schema* we call *fractional primal-dual*. As usual in primal-dual algorithms (see, e.g. [3]) our algorithm produces a primal solution x and a dual solution y, whose value divided by r, the approximation guarantee, bounds the weight of x. However, in contrast to other primal-dual algorithms, y is not a solution of the dual of an LP relaxation of the problem. Our algorithm induces a new LP that has the same objective function as the original problem, but contains inequalities that may not be valid with respect to the original problem. That is, the feasible set of this new LP is not the same as the feasible set of the LP-relaxation of the original problem. y is a feasible solution of the dual of the new LP. x is r-approximate, since we make sure that some optimal (fractional) solution x^* of the LP relaxation is in the feasible set of this new LP, and therefore the optimum of the new LP is not worse than the optimum of the LP relaxation. Furthermore, we show that x^* and the pair (x, y) satisfy a *fractional* version of the complementary slackness constraints. A description of the fractional primal-dual approach is given in Section 3.

We discuss the relation between fractional primal-dual and fractional local ratio in Section 4, and show that the former is the primal-dual manifestation of the latter. The connection between the fractional primal-dual and fractional local ratio is not unlike the connection between the two methods in their standard forms [7]. We also present a fractional local ratio interpretation of the $6t$-approximation algorithm for the problem of scheduling t-intervals with demands.

In Section 5 we consider the problem of scheduling t-intervals with demands, where the allocation is *contiguous* and the resource in *non-fungible*. In terms of the example in Figure 1, this means that we are required to pack the boxes without breaking them. A bi-criteria approximation algorithm is given. This algorithm returns $4t$-approximate solutions, that may need up to 4 times the given amount of resource.

1.3 Related Work

Several approximation frameworks that use the primal-dual schema were published in the last decade. Goemans and Williamson [14] presented a generic algorithm for a wide family of *network design* problems. They proposed a more general framework in [15]. A recent survey by Williamson [26] describes the primal-dual schema and several extensions of the primal-dual approach. Following [14], Bertsimas and Teo [10] proposed a primal-dual framework to design and analyze approximation algorithms for covering problems. As in [14, 15] this framework enforces the primal complementary slackness conditions while relaxing the dual conditions. However, in contrast to previous studies, Bertsimas and Teo [10] express each advancement step as the construction of a single valid inequality, and an increase of the corresponding dual variable (instead of an increase of several dual variables). Bar-Yehuda and Rawitz [7] presented a primal-dual framework that extends the one in [10]. They also showed that every advancement step of an approximation algorithm that uses the primal-dual schema (i.e., that enforces the primal conditions while relaxing the dual conditions) can be represented by a change in a single dual variable. In the analyses of both [10] and [7] it was convenient to define a new LP that contains the inequalities used by the algorithm, and to use it in the analysis. Since this new LP relaxes the original program, an r-approximation with respect to it is an r-approximation with respect to the original program.

As pointed out by Williamson [26] several primal-dual algorithms were devised by first constructing a local ratio algorithm, and then transforming it into a primal-dual algorithm. Bafna et al. [1] extended the *local ratio technique* [5] in order to construct a 2-approximation algorithm for the *feedback vertex set* problem. This work and the 2-approximation from [8] were essential in the design of primal-dual approximation algorithms for *feedback vertex set* [11]. Bar-Noy et al. [3] used the local-ratio technique to develop a framework for resource allocation and scheduling problems. A primal-dual interpretation was given as well. This study was the first to present a local-ratio or primal-dual approximation algorithm for a natural maximization problem. Bar-Yehuda and Rawitz [7] proved that the two methods in their standard forms are equivalent. This equivalence is based on the fact that increasing a dual variable by ϵ is equivalent to subtracting the weight function obtained by multiplying the coefficients of the corresponding primal constraint by ϵ from the primal objective function.

As mentioned before, Bar-Yehuda et al. [6] have presented a new extension of local ratio, called *fractional local ratio*. They used this technique to design a $2t$-approximation algorithm for MWIS in t-interval graphs. The novelty of the fractional approach is that, in weight decomposition steps, the construction of a new weight function is based on an optimal solution to an LP relaxation of the original problem instance, and the analysis compares the weight of the solution returned to the weight of this optimal solution. Thus, the fractional local ratio technique is a combination of LP rounding and the standard local ratio approach. The fractional local ratio technique was also used by Lewin-Eytan et al. [22].

1.4 Definitions and Notation

Given a t-interval scheduling instance we denote the set of t-intervals or jobs by J. For $j \in J$, $N(j)$ is the set of jobs that are in conflict with j, and $N[j]$ is the set of jobs that are in conflict with j including j, i.e., $N[j] = N(j) \cup \{j\}$. (Recall that two jobs are in conflict if any of their segments intersect.) If I is one of the segments of j we write $I \in j$. For a segment $I \in j$, $R[I]$ contains every job k such that there exists a segment $I' \in k$ that contains the right endpoint of I (including j).

We denote the optimum value for a given problem instance by $\textsc{Opt}$. $\textsc{Opt}(\Pi)$ denotes the optimum of Π, which is usually a linear program. Given a schedule S, we denote by $w(S)$ the total

weight of S, i.e., $w(S) = \sum_{j \in S} w(j)$. For $r \geq 1$, a feasible schedule S is *r-approximate* if its weight is within a factor of r of the optimum. More specifically, if $w(S) \geq \textsc{Opt}/r$. An algorithm that always returns r-approximate solutions is said to achieve an *approximation factor* of r, and it is called an *r-approximation* algorithm.

Finally, throughout the paper we use the notation $w(j)$ to denote the weight of job j, and w_i to denote the ith weight function. For example, $w_1(j)$ is the weight of job j with respect to the weight function w_1.

2 A $6t$-approximation Algorithm

2.1 LP formulation

The problem of scheduling t-intervals with demands can be formalized as follows:

$$\begin{array}{lllll} \text{(IP)} & \max & \sum_{j \in J} w(j) x_j & & \\ & \text{s.t.} & \sum_{k \in R[I]} d_k x_k \leq 1 & \forall j \in J, \forall I \in j \\ & & x_j \in \{0,1\} & \forall j \in J \end{array}$$

where $x_j = 1$ if and only if j is in the schedule. The LP-relaxation that is denoted by (P) is obtained by replacing the integrality constraints by: $0 \leq x_j \leq 1$ for every $j \in J$.

Before presenting the algorithm we prove the following property of feasible solutions of (P).

Lemma 1 *Let x be a feasible solution of (P). Then, there exists a job ℓ such that $\sum_{j \in N[\ell]} d_j x_j \leq 2t$.*

Proof. In order to prove this, it is enough to prove that

$$\sum_k \sum_{j \in N[k]} d_k x_k \cdot d_j x_j = \sum_k d_k x_k \sum_{j \in N[k]} d_j x_j \leq 2t \cdot \sum_k d_k x_k$$

If j_1 and j_2 are in conflict then $j_1 \in N[j_2]$ and $j_2 \in N[j_1]$. Therefore, the term $d_{j_1} x_{j_1} \cdot d_{j_2} x_{j_2}$ is counted twice in the sum on the LHS for every $j_1, j_2 \in J$. Furthermore, if j_1 and j_2 are in conflict then either there exists a segment $I_1 \in j_1$ such that $j_2 \in R(I_1)$, or there exists a segment $I_2 \in j_2$ such that $j_1 \in R(I_2)$. Thus,

$$\sum_k \sum_{j \in N[k]} d_k x_k \cdot d_j x_j \leq 2 \cdot \sum_k \sum_{I \in k} \sum_{j \in R[I]} d_k x_k \cdot d_j x_j$$

Since x is a feasible solution of (P),

$$\sum_{j \in R[I]} d_k x_k \cdot d_j x_j = d_k x_k \sum_{j \in R[I]} d_j x_j \leq d_k x_k$$

Therefore,

$$\sum_k \sum_{j \in N[k]} d_k x_k \cdot d_j x_j \leq 2 \cdot \sum_k \sum_{I \in k} d_k x_k = 2t \cdot \sum_k d_k x_k$$

and we are done. ■

2.2 The Algorithm

To approximate the problem we first consider the following two special cases.

Special Case 1: All jobs are *wide*, i.e., $d_j > \frac{1}{2}$ for all j.

Special Case 2: All jobs are *narrow*, i.e., $d_j \leq \frac{1}{2}$ for all j.

In the case of wide jobs the problem reduces to the special case in which all jobs have demand 1 since no pair of conflicting jobs may be scheduled together. Thus, it can be approximated using the $2t$-approximation algorithm from [6]. In the sequel we present a $4t$-approximation algorithm for narrow jobs. To solve the problem in the general case we solve it separately for the narrow jobs, and for the wide jobs, and return the solution of greater weight. Since either the optimum of the narrow jobs is at least $\frac{2}{3}$ of the optimum (for the original problem), or the optimum for the wide jobs is at least $\frac{1}{3}$ of the optimum, the schedule returned is $6t$-approximate.

2.3 A $4t$-approximation Algorithm for Narrow Jobs

The first step in our algorithm is to obtain an optimal solution of (P), denoted by x^*. The second step is given below.

Algorithm FPD(J, w, x^*)

1. $i \leftarrow 1$
2. $J_1 \leftarrow J$
3. While $J_i \neq \emptyset$ do:
4. Let $\ell_i \in J_i$ be a job minimizing $\sum_{j \in N[\ell] \cap J_i} d_j x^*_j$
5. Implicitly construct Inequality i:
   $$(1 - d_{\ell_i}) z_{\ell_i} + \sum_{j \in N(\ell_i) \cap J_i} d_j z_j \leq 1 - 2d_{\ell_i} + 2t$$
6. Increase y_i until
   $(1 - d_{\ell_i}) y_i + \sum_{k \,:\, \ell_i \in N(\ell_k) \cap J_k} d_{\ell_i} y_k = w(\ell_i)$
7. $J_{i+1} \leftarrow \left\{ j \notin \{\ell_1, \ldots, \ell_i\} \,:\, \sum_{k \,:\, j \in N(\ell_k) \cap J_k} d_j y_k < w(j) \right\}$
8. $i \leftarrow i + 1$
9. $S \leftarrow \emptyset$
10. While $i > 1$ do:
11. $i \leftarrow i - 1$
12. If $S \cup \{\ell_i\}$ is a feasible solution do: $S \leftarrow S \cup \{\ell_i\}$
13. Return S

Due to Lines 9-12 it is evident that Algorithm **FPD** returns a feasible schedule. It remains to show that this schedule is $4t$-approximate.

Let (P′) be the LP that contains the inequalities constructed by the algorithm:

$$\begin{aligned} (\mathrm{P}') \quad \max \quad & \sum_j w(j) z_j & & \\ \text{s.t.} \quad & (1 - d_{\ell_i}) z_{\ell_i} + \sum_{j \in N(\ell_i) \cap J_i} d_j z_j \leq c_i & & \forall i \in \{1, \ldots, m\} \\ & z_j \geq 0 & & \forall j \end{aligned}$$

where m is the last iteration of the algorithm, and $c_i = 1 - 2d_{\ell_i} + 2t$. The dual of (P′) is:

$$
\begin{array}{lll}
(\mathrm{D}') \quad \min & \displaystyle\sum_{i=1}^{m} c_i y_i & \\
\text{s.t.} & (1-d_{\ell_i})y_i + \displaystyle\sum_{k\,:\,\ell_i \in N(\ell_k)\cap J_k} d_{\ell_i} y_k \geq w(\ell_i) & \forall i \in \{1,\ldots,m\} \\
& \displaystyle\sum_{k\,:\,j \in N(\ell_k)\cap J_k} d_j y_k \geq w(j) & \forall j \notin \{\ell_1,\ldots,\ell_m\} \\
& y_i \geq 0 & \forall i \in \{1,\ldots,m\}
\end{array}
$$

For the analysis, let x denote the incidence vector of the schedule S returned by the algorithm. Also, let y be the vector constructed by the algorithm. We show that: (1) y is a feasible solution of (D′), (2) x^* is a feasible solution of (P′), and (3) $w \cdot x \geq c \cdot y/4t$. When putting it all together we get that: $w \cdot x \geq c \cdot y/4t \geq w \cdot x^*/4t = \textsc{Opt}(\mathrm{P})/4t \geq \textsc{Opt}(\mathrm{IP})/4t$, which means that x is $4t$-approximate.

To see that y is a feasible solution of (D′) observe that by the termination condition of the first while loop (Line 3) all the constraints in (D′) are satisfied. Next we show that x^* is a feasible solution of (P′). To that end we use Lemma 1. Consider Inequality i. Let x^i be the projection of x^* on J_i. That is,

$$
x^i_j = \begin{cases} x^*_j, & j \in J_i \\ 0, & \text{otherwise.} \end{cases}
$$

Clearly, x^i is a feasible solution of (P). Thus, by Lemma 1 we know that $\sum_{j \in N[\ell_i]\cap J_i} d_j x^i_j \leq 2t$. And, therefore,

$$
\begin{aligned}
(1-d_{\ell_i})x^*_{\ell_i} + \sum_{j \in N(\ell_i)\cap J_i} d_j x^*_j &= (1-d_{\ell_i})x^i_{\ell_i} + \sum_{j \in N(\ell_i)\cap J_i} d_j x^i_j \\
&= (1-2d_{\ell_i})x^i_{\ell_i} + \sum_{j \in N[\ell_i]\cap J_i} d_j x^i_j \\
&\leq 1 - 2d_{\ell_i} + 2t\ .
\end{aligned}
$$

Since x^* satisfies the inequalities in (P′), we conclude that x^* is a feasible solution of (P′), and that $w \cdot x^* \leq c \cdot y$.

Finally, we now turn to prove that $w \cdot x \geq c \cdot y/4t$.

$$
\begin{aligned}
\sum_j w(j)x_j &= \sum_i x_{\ell_i}\left((1-d_{\ell_i})y_i + \sum_{k\,:\,\ell_i \in N(\ell_k)\cap J_k} d_{\ell_i} y_k\right) + \sum_{j \notin \{\ell_1,\ldots,\ell_m\}} x_j \sum_{k\,:\,j \in N(\ell_k)\cap J_k} d_j y_k && (1)\\
&= \sum_i y_i\left((1-d_{\ell_i})x_{\ell_i} + \sum_{j \in N(\ell_i)\cap J_i} d_j x_j\right) && (2)\\
&\geq \sum_i y_i(1-d_{\ell_i}) && (3)\\
&= \sum_i (1-2d_{\ell_i}+2t)\cdot y_i \frac{1-d_{\ell_i}}{1-2d_{\ell_i}+2t} \\
&\geq \frac{1}{4t}\cdot \sum_i c_i y_i && (4)
\end{aligned}
$$

where (1) is true since $x_j = 1$ if $j = \ell_i$ for some i and the corresponding constraint is tight, and otherwise $x_j = 0$ (due to Line 6); (2) is due to a change in the summation order; (3) stems from the fact that either $\ell_i \in S$, or the total demand of jobs in $N(\ell_i) \cap J_i \cap S$ is more than $1 - d_{\ell_i}$ since otherwise ℓ_i would have been added to S (in Line 12); and (4) is due to the fact that the function $f(z) = \frac{1-2z+2t}{1-z}$ is an increasing function for $0 \leq z \leq \frac{1}{2}$ and $t \geq 1$, and that $f(\frac{1}{2}) = 4t$.

3 Using Fractional Primal-Dual

This section is written in terms of maximization problems. Similar arguments can be made in the minimization case.

Consider the following linear program and its dual:

$$\begin{array}{llll} \text{(P)} & \max & \sum_{j=1}^{n} w(j)x_j & \\ & \text{s.t.} & \sum_{j=1}^{n} a_{ij}x_j \leq c_i & \forall i \in \{1,\ldots,m\} \\ & & x_j \geq 0 & \forall j \in \{1,\ldots,n\} \end{array} \qquad \begin{array}{llll} \text{(D)} & \min & \sum_{i=1}^{n} c_i y_i & \\ & \text{s.t.} & \sum_{i=1}^{n} a_{ij}y_j \geq w(j) & \forall j \in \{1,\ldots,n\} \\ & & y_i \geq 0 & \forall i \in \{1,\ldots,m\} \end{array}$$

A primal-dual r-approximation algorithm is an algorithm that constructs an integral primal solution x and (possibly implicitly) a dual solution y, such that x and y satisfy $w \cdot x \geq c \cdot y/r$. It follows, by weak duality, that x is r-approximate. One way to find such a pair of primal and dual solutions is to focus on pairs (x, y) satisfying the following *relaxed complementary slackness* conditions (for an appropriately chosen r):

$$\begin{array}{lllll} \text{Primal:} & \forall j,\ x_j > 0 & \Rightarrow & \sum_{i=1}^{m} a_{ij}y_i = w(j). \\ \text{Relaxed Dual:} & \forall i,\ y_i > 0 & \Rightarrow & c_i/r \leq \sum_{j=1}^{n} a_{ij}x_j \leq c_i. \end{array}$$

These conditions imply that x is r-approximate since

$$\sum_{j=1}^{n} w(j)x_j = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij}y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij}x_j \right) y_i \geq \frac{1}{r} \cdot \sum_{i=1}^{m} c_i y_i.$$

Typically, such an algorithm begins with the solutions $x = 0$ and $y = 0$. It then iteratively increases dual variables until y becomes feasible. Afterwards, it adds as many elements that correspond to tight dual constraints to x as possible, ending with a primal/dual pair satisfying the relaxed conditions. This design method is commonly referred to as the *primal-dual schema*.

Our algorithm deviates from the standard primal-dual approach. First, the algorithm is based on an optimal fractional solution, denoted by x^*. Another difference is that our algorithm constructs new primal constraints during execution that are not necessarily valid for the original problem instance. That is, a feasible solution may not satisfy these constraints. However, we make sure that x^* satisfies them. Note that the construction of new primal constraints during execution was previously used by Bertsimas and Teo [10], and consequently by Bar-Yehuda and Rawitz [7]. However, in both papers the algorithms construct constraints that are valid with respect to the original problem instance. The constraints that are constructed by our algorithm induce a new linear program (P′) and a dual program (D′). It produces a primal solution x for the original problem instance, and a solution y for (D′), whose value divided by r bounds the weight of x. Since x^* is in the feasible set of (P′), y serves as an upper bound to the weight of x^*. Therefore, x is r-approximate.

Let $\text{P}' = \max\{w \cdot z : Ax \leq c\}$. When using fractional primal-dual the solutions x^*, x, y produced by the algorithm satisfy the following *fractional relaxed* complementary slackness conditions:

$$\begin{array}{lllll} \text{Primal:} & \forall j,\ x_j > 0 & \Rightarrow & \sum_{i=1}^{m} a_{ij}y_i = w(j) \\ \text{Relaxed Dual:} & \forall i,\ y_i > 0 & \Rightarrow & c_i/r \leq \sum_{j=1}^{n} a_{ij}x_j \ \text{ and } \ \sum_{j=1}^{n} a_{ij}x_j^* \leq c_i \end{array}$$

Note that y_i is actually positive for every i, since (P′) contains inequalities that were used by the algorithm. Since x^*, x, y satisfy the above conditions we get that:

$$w \cdot x = \sum_{j=1}^{n} w(j)x_j = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \geq \sum_{i=1}^{m} \frac{1}{r}\ c_i y_i = \frac{1}{r}\ c \cdot y$$

which this means that x is r-approximate, since

$$c \cdot y \geq \textsc{Opt}(\mathrm{P}') \geq w \cdot x^* = \textsc{Opt}(\mathrm{P}) \geq \textsc{Opt}\ .$$

4 Fractional Local Ratio

In this section we present a fractional local ratio interpretation of the $4t$-approximation algorithm for narrow jobs, and study the connection between fractional primal-dual and fractional local-ratio. We start with a brief description of the fractional local ratio technique.

A typical local ratio algorithm is recursive, and it constructs, in each recursive call, a new weight function w_1. In essence, a local ratio analysis consists of comparing, at each level of the recursion, the solution found in that level to an optimal solution for the problem instance passed to that level, where the comparison is made with respect to w_1 and with respect to $w - w_1$. Thus, in each level of the recursion there are potentially two optima (one with respect to w and one with respect to $w - w_1$) against which the solution is compared, and in addition, different optima are used at different recursion levels. The *fractional local ratio* paradigm takes a different approach. It uses a single solution x^* to the *original* problem instance as the yardstick against which all intermediate solutions (at all levels of the recursion) are compared. In fact, x^* is not even feasible for the original problem instance but rather for a relaxation of it. Typically, x^* will be an optimal fractional solution to an LP relaxation.

The fractional local ratio technique is based on a *fractional* version of the Local Ratio Theorem [5, 4]. We precede the statement of the theorem with the following useful definition.

Definition 1 *Let $r \geq 1$ and let $w \in \mathbb{R}^n$ be a weight function. Let x and x^* be vectors in $\mathbb{R}^n$. x is said to be* r-approximate relative to x^* *(with respect to w) if $w \cdot x \geq (w \cdot x^*)/r$.*

The proof of the next theorem is immediate.

Theorem 1 (Fractional Local Ratio [6]) *Let $w, w_1, w_2 \in \mathbb{R}^n$ be weight functions such that $w = w_1 + w_2$. Let x^* and x be vectors in $\mathbb{R}^n$ such that x is r-approximate relative to x^* with respect to w_1 and with respect to w_2. Then, x is r-approximate relative to x^* with respect to w as well.*

4.1 A Fractional Local Ratio Interpretation of the $4t$-approximation Algorithm

Let x^* be an optimal fractional solution. We now run the recursive algorithm described next to obtain a feasible schedule. The algorithm contains problem-size reduction steps, as do local ratio algorithms for packing problems (e.g., [3]). The initial call is $\mathbf{FLR}(J, w)$.

Algorithm FLR(J, w)

1. If $J = \emptyset$, return $\emptyset$
2. Let ℓ be a job minimizing $\sum_{j \in N[\ell]} d_j x^*_j$
3. $\epsilon \leftarrow w(\ell)/(1 - d_\ell)$
4. Define the weight functions $w_1(j) = \epsilon \cdot \begin{cases} 1 - d_\ell & j = \ell, \\ d_j & j \in N(\ell), \\ 0 & \text{otherwise,} \end{cases}$
   and $w_2 = w - w_1$
5. Let J^+ be the set of positive weighted jobs
6. $S' \leftarrow \mathbf{FLR}(J^+, w_2)$
7. If $S' \cup \{\ell\}$ is a feasible solution do:
8. Return $S = S' \cup \{\ell\}$
9. Else:
10. Return $S = S'$

For the analysis, let x denote the incidence vector of the schedule S returned by the algorithm. We assume that x (and w) is of size n, where n is the number of jobs in the original problem instance. This way we can compare x to x^*. We claim that S is $4t$-approximate relative to x^*, namely, that $w \cdot x \geq \frac{1}{4t}\, w \cdot x^*$. The proof is by induction on the recursion. In the base case ($J = \emptyset$) we have $S = \emptyset$, and therefore $w \cdot x = 0$. Since the weights in the recursive base are non-positive we get that $w \cdot x^* \leq 0$. Thus, $w \cdot x \geq w \cdot x^*$. For the inductive step, let x' be the incidence vector of S' (obtained in Line 6). By the inductive hypothesis x' is $4t$-approximate relative to x^* and with respect to w_2. That is, $w_2 \cdot x' \geq \frac{1}{4t}\, w_2 \cdot x^*$. Moreover, the weight of ℓ with respect to w_2 is zero, and therefore $w_2 \cdot x = w_2 \cdot x'$. This means that x is also $4t$-approximate relative to x^* and with respect to w_2. Next, we show that $w_1 \cdot x \geq \frac{1}{4t} w_1 \cdot x^*$. This completes the proof since this means that by the Fractional Local Ratio Theorem x is $4t$-approximate relative to x^* and with respect to w.

It remains to show that x is $4t$-approximate relative to x^* and with respect to w_1.

$$\begin{aligned} w_1 \cdot x^* &= \sum_j w_1(j) x^*_j \\ &= \epsilon(1 - d_\ell) x_\ell + \sum_{j \in N(\ell)} \epsilon d_j x^*_j \\ &= \epsilon(1 - 2d_\ell) x_\ell + \epsilon \sum_{j \in N[\ell]} d_j x^*_j \\ &\leq \epsilon(1 - 2d_\ell + 2t) \qquad (5) \end{aligned}$$

where (5) is due to Lemma 1. On the other hand, we show that $w_1 \cdot x \geq \epsilon(1 - d_\ell)$. If ℓ is added to S (in Line 8) then this is obviously true. Otherwise, if ℓ is not added to S, then the total demand of jobs in $N(\ell) \cap S$ is more than $1 - d_\ell$, since otherwise ℓ would have been added to S. Therefore, $w_1 \cdot x \geq \epsilon(1 - d_\ell)$. Now, x is $4t$-approximate relative to x^* and with respect to w_1, since

$$\frac{w_1 \cdot x^*}{w_1 \cdot x} \leq \frac{1 - 2d_\ell + 2t}{1 - d_\ell} \leq 4t \qquad (6)$$

where (6) is due to the fact that the function $f(z) = \frac{1-2z+2t}{1-z}$ is an increasing function for $0 \leq z \leq \frac{1}{2}$ and $t \geq 1$, and that $f(\frac{1}{2}) = 4t$.

4.2 Connection to Fractional Local Ratio

In [7] Bar-Yehuda and Rawitz have shown that the equivalence between the primal-dual schema and local ratio technique in their standard forms are equivalent. This equivalence is based on the fact that increasing a dual variable by ϵ is equivalent to subtracting the weight function obtained by multiplying the coefficients of the corresponding primal constraint by ϵ from the primal objective function. A similar equivalence exists between both fractional methods. For example, the weight function that is used in the ith recursive call of Algorithm **FLR** is equal to the vector of coefficients of the inequality that was constructed in the ith iteration of Algorithm **FPD** multiplied by y_i. Generally, in each recursive call of a fractional local ratio algorithm we utilize a weight function w_1 such that $w_1 \cdot x^* \leq r\ w_1 \cdot x$. Implicitly, this means that there exists some c_i such that $w_1 \cdot x \geq c_i/r$ and $w_1 \cdot x^* \leq c_i$. Thus, the inequality $w_1 \cdot z \leq c_i$ can be used by a fractional primal-dual algorithm. (Note that we do not need to know the value of c_i.) For the other direction, an inequality $\alpha \cdot z \leq \beta$ corresponds to the weight function $\epsilon \cdot \alpha$ where ϵ is the value of the corresponding dual variable.

5 Contiguous Allocation of a Non-Fungible Resource

Until now we have assumed that our resource can be divided between several jobs, and the only requirement was that the sum of demands at any given point $s \in \mathbb{R}$ is bounded by 1. In other words, we have assumed that a job receives a portion of the resource, but the identity of this portion is subject to change. For example, in a multimedia-on-demand system, where the resource is the bandwidth of the system, the identity of the bandwidth allocated to a specific film is irrelevant. Such a resource is called *fungible*. On the other hand, there are scenarios in which a specific portion of the resource is allocated to each job. For example, memory allocation in a multi-threaded programming environment. Such resources are called *non-fungible*. In cases we are able to allocate several fractions of the resource instead of a single *contiguous* portion, non-fungible resources can be treated as though they were fungible. However, in some cases, such as memory allocation, the allocation of the resource must by contiguous. See [3] for more details.

In this section we consider the problem of scheduling t-intervals with demands in the context of contiguous allocation of a non-fungible resource. In [3] Bar-Noy et al. presented approximation algorithms for several resource allocation problems, where the allocation is *contiguous* and the resource is *non-fungible*. We use similar ideas in order to devise a bi-criteria approximation algorithm that produces $4t$-approximate solutions by allowing the use of 4 times the amount of resource available.

As we did in the Section 2.2 we examine the two special cases of wide jobs and narrow jobs. In the case of wide jobs the problem reduces to the unit-demand case since no pair of conflicting jobs may be scheduled together. (In this case there is no difference between fungible and non-fungible resources.) Thus, it can be approximated using the $2t$-approximation algorithm from [6]. We present a bi-criteria approximation algorithm for narrow jobs that returns $4t$-approximate solutions that use 3 times the amount of resource. Therefore, to solve the problem on general instances we solve it separately for the narrow jobs, and for the wide jobs, and combine the solutions. The result is a $4t$-approximate solution that uses 4 times the amount of resource.

Consider the following scheduling problem. We are given a set of jobs. Each job j is associated with an interval (not a t-interval) during which the job will be executed, and the amount of resource required for the job, d_j, where $0 \leq d_j \leq 1$. Our goal is to schedule *all* jobs while using minimal amount of resource. Clearly, one cannot hope to obtain a schedule that uses less than D^* amount of resource, where D^* is the maximum, taken over all times instances s, of the sum of the demands at time s. This problem is known to be NP-hard [12], but can be approximated by a constant factor.

Specifically, Gergov [13] presented a "blow-up" algorithm that outputs schedules that use $3 \cdot D^*$ amount of resource. A nice property of this algorithm is that the schedules it produces use three unit size resources, and not a single resource of size 3. (Assume that each resource is a unit-size strip. The algorithm returns schedules in which each job is fully contained in one of the strips.) We use this algorithm to construct a schedule of t-intervals. Let S be the schedule obtained by Algorithm **FPD**. Now, given the t-intervals in S we run the blow-up algorithm, and find a new schedule S' that uses $3 \cdot D^*$ amount of resource, where D^* is defined with respect to S, and hence $D^* \leq 1$. Note that from the stand point of the blow-up algorithm any segment in a t-interval is a different job, and therefore our solution may assign different segments of the same t-split to different resources. (Recall that there are three.)

Acknowledgments

We thank Ari Freund and Guy Even for helpful discussions.

References

[1] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

[2] V. Bafna, B. Narayanan, and R. Ravi. Nonoverlapping local alignments (weighted independent sets of axis parallel rectangles). *Discreate Applied Mathematics*, 71:41–53, 1996. Special issue on Computational Molecular Biolagy.

[3] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Shieber. A unified approach to approximating resource allocation and schedualing. *Journal of the ACM*, 48(5):1069–1090, 2001.

[4] R. Bar-Yehuda. One for the price of two: A unified approach for approximating covering problems. *Algorithmica*, 27(2):131–144, 2000.

[5] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.

[6] R. Bar-Yehuda, M. M. Halldórsson, J. Naor, H. Shachnai, and I. Shapira. Scheduling split intervals. In *13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 732–741, 2002.

[7] R. Bar-Yehuda and D. Rawitz. On the equivalence between the primal-dual schema and the local ratio technique. *In preparation*, 2004. An extended abstract containing some of these results appeared in APPROX'01.

[8] A. Becker and D. Geiger. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83(1):167–188, 1996.

[9] P. Berman and T. Fujito. Approximating independent sets in degree 3 graphs. In *4th Workshop on Algorithms and Data Structures*, volume 995 of *LNCS*, pages 449–460, 1995.

[10] D. Bertsimas and C. Teo. From valid inequalities to heuristics: A unified view of primal-dual approximation algorithms in covering problems. *Operations Research*, 46(4):503–514, 1998.

[11] F. A. Chudak, M. X. Goemans, D. S. Hochbaum, and D. P. Williamson. A primal-dual interpretation of recent 2-approximation algorithms for the feedback vertex set problem in undirected graphs. *Operations Research Letters*, 22:111–118, 1998.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[13] J. Gergov. Algorithms for compile-time memory optimization. In *10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 907–908, 1999.

[14] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[15] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In Hochbaum [20], chapter 4.

[16] A. Gyárfás and D. B. West. Multitrack interval graphs. In *26th SE Intl. Conf. Graph Th. Comb. Comput.*, volume 109 of *Congr. Numer.*, pages 109–116, 1995.

[17] M. M. Halldórsson, S. Rajagopalan, H. Shachnai, and A. Tomkins. Shceduling multiple resources. Manuscript, 1999.

[18] M. M. Halldórsson and K. Yoshihara. Greedy approximations of independent sets in low degree graphs. In *6th Annual International Symposium on Algorithms And Computation*, volume 1004 of *LNCS*, pages 152–161, 1995.

[19] E. Hazan, S. Safra, and O. Schwartz. On the hardness of approximating k-dimensional matching. In *6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, volume 2764 of *LNCS*, pages 83–97, 2003.

[20] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problem*. PWS Publishing Company, 1997.

[21] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68–72, 1989.

[22] L. Lewin-Eytan, J. Naor, and A. Orda. Routing and admission control in networks with advance reservations. In *5th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, volume 2462 of *LNCS*, pages 215–228, 2002.

[23] C. H. Papadimitriou and M. Yannakakis. Optimization, approximization and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[24] D. Rotem. Analysis of disk arm movement for large sequential reads. In *Proceedings of the Eleventh ACM Symposium on Principles of Database Systems*, pages 47–54, 1992.

[25] D. B. West and D. B. Shmoys. Recognizing graphs with fixed interval number is NP-complete. *Discrete Applied Mathematics*, 8:295–305, 1984.

[26] D. P. Williamson. The primal dual method for approximation algorithms. *Mathematical Programming*, 91(3):447–478, 2002.