Resource Allocation with Time Intervals

Andreas Darmann ** Ulrich Pferschy * Joachim Schauer *

Abstract

We study a resource allocation problem where jobs have the following characteristics: Each job consumes some quantity of a bounded resource during a certain time interval and induces a given profit. We aim to select a subset of jobs with maximal total profit such that the total resource consumed at any point in time remains bounded by the given resource capacity.

While this case is trivially $\mathcal{NP}$-hard in general and polynomially solvable for uniform resource consumptions, our main result shows the $\mathcal{NP}$-hardness for the case of general resource consumptions but uniform profit values, i.e. for the case of maximizing the number of performed jobs. This result applies even for proper time intervals.

We also give a deterministic $(1/2-\varepsilon)$–approximation algorithm for the general problem on proper intervals improving upon the currently known 1/3 ratio for general intervals. Finally, we study the worst-case performance ratio of a simple greedy algorithm.

Keywords: resource allocation, proper intervals, unsplittable flow

1 Introduction

In this paper we investigate a *resource allocation problem* (RAP) where each task requires a certain amount of a limited resource but only during a certain time interval. Before and after this time interval the resource is not affected by this task. More formally, we are given a set of jobs $j \in \{1, ..., N\}$, each of them with a profit $p_j \in \mathbb{R}_0^+$ and a resource consumption (or weight) $w_j \in \mathbb{R}_0^+$ (we will also use the notation $w(j)$). Furthermore, to every job j a start time $s_j \in \mathbb{R}_0^+$ and an end time $t_j \in \mathbb{R}_0^+$, $t_j > s_j$, is assigned such that the resource consumption occurs only within the time interval $[s_j, t_j]$. The goal is to select a subset of jobs with maximal total profit such that the total resource consumption does not exceed the given resource capacity $W \in \mathbb{R}_0^+$ at any point in time.

An interesting application of this problem arises from the scheduling of tasks in the mission of a spacecraft described in a slightly different model in [18]. The individual projects which researchers wish to carry out during a space mission are strictly restricted to a time interval $[s_j, t_j]$ because of the spacecraft being in a certain configuration with respect to earth and other planetary bodies. Each project requires an amount w_j of a limited resource such as labor, workspace or energy. To choose from the large set of desirable projects each of them is assigned a priority value p_j. Maximizing the sum of priorities of all scheduled jobs under the given feasibility constraints amounts to solving an instance of RAP.

In the literature, this problem (or close relatives thereof) is also known as *bandwidth allocation problem* (cf. [11], where profits are given as weight times the interval length), *resource constrained scheduling* (cf. [23], where a generalization is considered in which each job is allowed to

**University of Graz, Institute of Public Economics, Universitaetsstr. 15, A-8010 Graz, Austria, andreas.darmann@uni-graz.at

*University of Graz, Department of Statistics and Operations Research, Universitaetsstr. 15, A-8010 Graz, Austria, {pferschy, joachim.schauer}@uni-graz.at

be positioned within a larger time interval) and *call admission control*, see [4] and [8] for further references. An exact algorithm for RAP based on a sophisticated ILP-formulation which is solved by column generation was recently developed by [9].

The generalization of RAP where the resource capacity varies over time was studied as *temporal knapsack problem* in [5] where exact algorithms were developed. A related problem with a geometric flavor, in which each job requires an *adjacent* block of the resource over its interval, was considered by several authors. In this case each job can be seen as requiring a rectangle whose length is determined by the time interval $[s_j, t_j]$ and whose width is given by w_j. [22] consider again the generalization with jobs allowed to be positioned in a larger interval while [13] consider the problem of minimizing W, such that all jobs can be processed, i.e. all rectangles can be packed into a strip of width W.

Moreover, RAP is also a well-known special case of the *unsplittable flow problem* (UFP). By [10] UFP consists of an n-vertex graph $G = (V, E)$ with edge capacities c_e and a set of k vertex pairs (terminals) $T = \{(s_i, t_i) \mid i = 1, \ldots, k\}$. Each pair (s_i, t_i) in T has a demand w_i and a profit p_i. The goal is to find the maximum profit subset of pairs from T, along with a path for each chosen pair, so that the entire demand for each such pair can be routed on its path while respecting the capacity constraints. The special case of UFP where G is a path and all the capacities c_e are uniform is known as *unsplittable flow problem on line graphs with uniform capacities* (UFPUC) (see [3]) and corresponds exactly to RAP with W equal to the uniform edge capacity. For a comprehensive overview of RAP (UFPUC) and UFP, the reader is referred to the papers [3] and [8].

Obviously, RAP is $\mathcal{NP}$-hard since the classical 0–1 knapsack problem can be understood as a special case of RAP where all time intervals are identical. In [2] a so-called quasi-PTAS is given for the (general) unsplittable flow problem on line graphs. This clearly also gives a quasi-PTAS for RAP. Nevertheless, the long standing question of whether there exists a PTAS for RAP is still open [8].

The trivial $\mathcal{NP}$-hardness of RAP gives rise to the study of two relevant special cases:

1. *uniform weights*: If all resource consumption values w_j are identical, the resource constraint reduces to a cardinality constraint. This version of the problem is well-known as an *interval scheduling problem* (see [21] for a survey and pointers to applications). RAP with uniform weights can easily be seen to be polynomially solvable (see e.g. [1], where an $O(N^2 \log N)$ algorithm is given, and [7] for a minimal cost flow model).

2. *uniform profits*: For the case where all profits p_j are identical, the complexity was open. It is the main result of our paper that RAP with uniform profits is $\mathcal{NP}$-hard, even if restricted to proper interval graphs (see Section 2).

Our result not only sharpens the obvious $\mathcal{NP}$-hardness result for the general RAP, but also proves that the existence of an FPTAS for RAP even on proper interval graphs with uniform profits (and thus also for RAP) can be ruled out, since due to uniform profits the regarded objective function value is polynomially bounded. Moreover, we believe that RAP with uniform profits is also an interesting combinatorial problem in its own right, since it combines the aspects of packing and scheduling problems in an intriguing way.

The intersection of the two cases with uniform profits and uniform weights was considered in [7] where a simple $O(N \log N)$ algorithm is given. However, for this case it was shown to be $\mathcal{NP}$-hard in [14] to minimize W such that all jobs can be performed. Maximizing the number of performed jobs with the additional restriction that only one job can be performed at a time (i.e. W equals the uniform weight) but with jobs allowed to be positioned within a larger time interval was considered in [18].

In [8] a deterministic $\frac{1}{3}$–approximation algorithm with running time $O(N^2 \log^2 N)$ and a randomized $(\frac{1}{2} - \varepsilon)$–approximation is given for RAP, thus improving upon earlier (and to some extend

more general) results in [23] and [4]. Our second result is a deterministic $(\frac{1}{2} - \varepsilon)$–approximation algorithm for the special case of RAP defined on proper interval graphs described in Section 3. This algorithm can easily be modified to an $\frac{1}{2}$–approximation algorithm for RAP defined on a proper interval graph with uniform profits, where the performance ratio is tight. Finally, in Section 4 we analyze the performance of a simple *greedy algorithm*, concluding that it also gives a tight $\frac{1}{2}$–approximation algorithm for the same special case, but can perform arbitrarily bad if the profit is not uniform or the intervals are not proper.

1.1 Problem formulations

Let $s_{\min} := \min_{1 \le j \le N} s_j$ and $t_{\max} := \max_{1 \le j \le N} t_j$. Then a straightforward ILP-formulation of RAP based on binary decision variables x_j for each job $j = 1, \ldots, N$, is given as follows.

Resource allocation problem (time interval formulation)

$$\text{maximize} \quad \sum_{j=1}^{N} p_j x_j$$

$$\begin{array}{lrcll} \text{s.t.} & \sum_{j:s_j \le t \le t_j} w_j x_j & \le & W & \forall\, t \in [s_{\min}, t_{\max}] \\ & x_j & \in & \{0,1\} & j = 1, ..., N. \end{array}$$

This straightforward time interval formulation requires pseudopolynomially many constraints. However, one can notice that the actual start and end times are not explicitly required in this problem. They only serve to model the overlap between jobs, while the length in time of such an overlap is irrelevant. Hence, we could map w.l.o.g. the sorted list of all start and end times to the discrete set $\{1, \ldots, 2N\}$ and thus bound the number of constraints by $2N$.

On the other hand, we will not apply such a transformation since it will turn out to be more useful to define RAP with the use of an interval graph.

Definition 1 *A set τ of intervals is called proper, if no interval of τ is contained in another interval of τ.*

Definition 2 *A graph $G = (V, E)$ is an interval graph if each vertex $v \in V$ can be assigned an interval on the real line, such that two intervals intersect iff the two corresponding vertices are adjacent. A representation of graph G by such intervals is called interval representation. A proper interval graph is an interval graph for which there exists an interval representation which is proper.*

For a given instance I of RAP we construct an interval graph $G = (V, E)$ in the following way: For each job j define a corresponding vertex v_j and make two vertices v_i and v_j adjacent, if the time intervals of the jobs i and j intersect. If the underlying instance I has the property of being described by *proper time intervals* then obviously the graph G is a proper interval graph.

In this paper we will represent the instance I by its corresponding interval graph, i.e., instead of looking at job j with specific profit, weight, and start and ending time, we consider the corresponding vertex v_j that gets assigned the same weight and profit as j. Note that the intersecting structure that was defined by the start and end times in I is only kept in the adjacency structure of the interval graph, which means that vertices that had an overlapping time interval are now adjacent. However, in order to formulate the resource allocation problem in terms of an interval graph, we state the following definition.

Definition 3 *A clique of a graph G is a complete subgraph of G. A maximal clique is a clique which is not properly contained in any other clique.*

Replacing the label v_j by j for $j = 1, \ldots, N$, the resource allocation problem RAP can also be formulated as follows.

Resource allocation problem (interval graph formulation)

$$\begin{array}{lrcll} \text{maximize} & \sum_{j=1}^{N} p_j x_j & & & \\ \text{s.t.} & \sum_{j \in C} w_j x_j & \leq & W & \text{for all maximal cliques } C \text{ of } G \\ & x_j & \in & \{0,1\} & j = 1, ..., N. \end{array} \tag{1}$$

For any subset of jobs $X \subseteq \{1, ..., N\}$ we denote its weight by $w(X) := \sum_{j \in X} w_j$ and its objective function value as $p(X) := \sum_{j \in X} p_j$. X is a feasible solution of I if the associated vector $x \in \{0,1\}^N$, defined by $x_j := 1$ if $j \in X$ and $x_j := 0$ otherwise, satisfies all constraints (1). An optimal feasible solution is denoted by X^*.

It is easy to see that there exists a one to one correspondence between feasible solutions of the interval graph representation of I and the time interval representation. Note that the number of maximal cliques of G and thus the number of constraints is bounded by N. This value is tight in sense that an instance where each job j overlaps only with jobs $j-1$ and $j+1$ induces an interval graph with $N-1$ maximal cliques. In what follows, we will use the interval graph formulation of RAP.

2 Resource Allocation with uniform profits

In this section we show that the resource allocation problem RAP is $\mathcal{NP}$-hard even if restricted to jobs that are assigned uniform profits (w.l.o.g. $p_j = 1$, $j = 1, \ldots, N$) and proper intervals. The proof of this $\mathcal{NP}$-hardness result consists in a reduction from the following variant of the partition problem.

Definition 4 ***Ordered partition***

GIVEN: *A multiset $B := \{b_1, b_2, ..., b_{2n-1}, b_{2n}\}$ of $2n$ integers such that $b_{2i} < b_{2i-1}$ for all i, $1 \leq i \leq n$.*

QUESTION: *Is there a subset $B' \subset B$ such that*

1. *B' contains exactly one of $\{b_{2i-1}, b_{2i}\}$ for all i, $1 \leq i \leq n$, and*
2. *$\sum_{b_i \in B'} b_i = \sum_{b_i \in B \setminus B'} b_i$?*

Since the variant of the above problem in which the requirement $b_{2i} < b_{2i-1}$ is omitted is $\mathcal{NP}$-complete [16], it immediately follows that ordered partition is $\mathcal{NP}$-complete.

Theorem 1 *[16] Ordered partition is $\mathcal{NP}$-complete.*

It can easily be seen that ordered partition is $\mathcal{NP}$-complete both for even n and odd n; for the sake of brevity we omit the proof of this simple observation. Since for the rest of the paper we assume n to be even, we state the following lemma.

Lemma 2 *Ordered partition is $\mathcal{NP}$-complete if n is even.*

2.1 The instance R of RAP

Given an even instance OP of Ordered Partition with a set of integers $B := \{b_1, b_2, ..., b_{2n-1}, b_{2n}\}$ such that $b_{2i} < b_{2i-1}$ for all i, $1 \leq i \leq n$, we will construct an instance R of RAP and state some of its basic features in this section.

2.1.1 Idea of the construction

The idea beyond the instance R is as follows. After specifying an appropriate resource bound W, we introduce the jobs a_i and the jobs $\bar{a}_i$ each of which we associate with the integer b_i of Ordered Partition, $1 \leq i \leq 2n$. Next we introduce two sorts of dummy jobs. Having introduced these jobs, there are $3n$ maximal cliques in R: The cliques $C_2, C_4, ..., C_{2n}$, the cliques $I_2, I_4, ..., I_{2n}$ and the cliques $\bar{C}_2, \bar{C}_4, ..., \bar{C}_{2n}$. The dummy jobs are assigned appropriate resource consumptions such that the resource constraints on the maximal cliques guarantee several structural properties of optimal solutions of R.
First, the construction allows to establish an upper bound U for the objective function value of solutions of R. Next, the maximal cliques C_{2i} are established in such a way that an optimal solution X^* of R whose objective function value meets the bound U contains exactly one element of $\{a_{2n-2i+2}, a_{2n-2i+1}\}$. Analogously, the cliques $\bar{C}_{2i}$ are designed in order to make sure that exactly one element of $\{\bar{a}_{2i}, \bar{a}_{2i-1}\}$ is contained in such a solution X^*. The cliques I_{2i} finally serve the purpose that a_{2i} and $\bar{a}_{2i}$ are not both contained in X^*. These properties are shown in Section 2.2
In Section 2.3 we show that from the resource constraint on the cliques C_{2n} and $\bar{C}_{2n}$ it then follows that such an optimal solution X^* yields the desired subset $B' \subset B$ of instance OP. The construction of B' is based on the idea of identifying $a_i \in X^* \Leftrightarrow b_i \in B'$. On the other hand, the construction of an optimal solution X^* with $p(X^*) = U$ from a given solution B' of OP can be done in a straightforward way by use of the above equivalence and inserting all the dummy jobs in X^*.

We will now give a detailed specification of instance R.

2.1.2 Resource bound W and the jobs in R

Let $M := \sum_{i=1}^{2n} b_i$ and

$$W := M(\frac{3}{2}n^2 + n + 1) \ .$$

First we introduce $2n$ jobs a_i, $1 \leq i \leq 2n$, and $2n$ jobs $\bar{a}_i$, $1 \leq i \leq 2n$. The resource consumptions of these jobs are

$$w(a_{2i+1}) = b_{2i+1} + (2n - 2i)M \text{ and } w(a_{2i+2}) = b_{2i+2} + (2n - 2i)M$$

resp.

$$w(\bar{a}_{2i+1}) = b_{2i+1} + (2i + 2)M \text{ and } w(\bar{a}_{2i+2}) = b_{2i+2} + (2i + 2)M \ ,$$

for $0 \leq i \leq n - 1$. Note that this implies

$$w(a_{2i+1}) > w(a_{2i+2}) \text{ and } w(\bar{a}_{2i+1}) > w(\bar{a}_{2i+2}) \tag{2}$$

for all $i \in \{0, ..., n-1\}$. For illustration reasons the resource consumptions of these jobs are given in Table 1.

Next we introduce the following types of sets of dummy jobs: the sets D_{2i} for all i, $\frac{n}{2} + 2 \leq i \leq n$, the sets $\bar{D}_{2i}$, for all i, $1 \leq i \leq \frac{n}{2} - 1$, and the sets E_{2i} and $\bar{E}_{2i}$, $1 \leq i \leq n$. The resource

Resource consumptions of jobs a_i	Resource consumptions of jobs $\bar{a}_i$
$w(a_1) = b_1 + 2nM$	$w(\bar{a}_1) = b_1 + 2M$
$w(a_2) = b_2 + 2nM$	$w(\bar{a}_2) = b_2 + 2M$
$w(a_3) = b_3 + (2n-2)M$	$w(\bar{a}_3) = b_3 + 4M$
$w(a_4) = b_4 + (2n-2)M$	$w(\bar{a}_4) = b_4 + 4M$
$\vdots$	$\vdots$
$w(a_{n-1}) = b_{n-1} + (n+2)M$	$w(\bar{a}_{n-1}) = b_{n-1} + nM$
$w(a_n) = b_n + (n+2)M$	$w(\bar{a}_n) = b_n + nM$
$w(a_{n+1}) = b_{n+1} + nM$	$w(\bar{a}_{n+1}) = b_{n+1} + (n+2)M$
$w(a_{n+2}) = b_{n+2} + nM$	$w(\bar{a}_{n+2}) = b_{n+2} + (n+2)M$
$\vdots$	$\vdots$
$w(a_{2n-1}) = b_{2n-1} + 2M$	$w(\bar{a}_{2n-1}) = b_{2n-1} + 2nM$
$w(a_{2n}) = b_{2n} + 2M$	$w(\bar{a}_{2n}) = b_{2n} + 2nM$

Table 1: Resource consumptions of the jobs a_i and $\bar{a}_i$, $1 \leq i \leq 2n$

consumption of each of the jobs in these job sets equals $\frac{M}{2}$. The number of jobs in the dummy job sets $D_{n+4}, ..., D_{2n}$ is given by

$$|D_{n+2+2i}| = 8i - 4 \ ,$$

for $2i \in \{2, ..., n-2\}$. The cardinalities of the dummy job sets $\bar{D}_2, ..., \bar{D}_{n-2}$ are given by

$$|\bar{D}_{n-2i}| = 8i - 4 \ ,$$

for $2i \in \{2, ..., n-2\}$.
The cardinalities of the dummy job sets E_{2i} and $\bar{E}_{2i}$ are as follows:

$$|\bar{E}_{2i}| = |E_{2i}| = \begin{cases} 4n - 4i & \text{if } 2i \in \{2, ..., n-2\} \\ 4i + 4 & \text{if } 2i \in \{n, ..., 2n-4\} \\ 4n - 1 & \text{if } 2i = 2n-2 \\ 4n - 3 & \text{if } 2i = 2n \end{cases}$$

Table 2 and 3 display the starting and end times of the jobs. The time interval $[a, b]$ of a job set F means that the jobs in this job set have the time interval $[a + \frac{i}{|F|}, b + \frac{i}{|F|}]$, $i \in \{0, 1, ..., |F| - 1\}$.
Note that, for $2i \in \{2, ..., n-2\}$, we get

$$\begin{aligned} w(a_{2i+2}) - w(\bar{a}_{2i+2}) &= (2n - 4i - 2)M \\ &= (4(n-2i) - 4)\tfrac{M}{2} \\ &= w(\bar{D}_{n-(n-2i)}) \end{aligned}$$

Thus,

$$w(a_{2i+2}) - w(\bar{a}_{2i+2}) = w(\bar{D}_{2i}) \text{ for } 2i \in \{2, ..., n-2\} \tag{3}$$

and analogously

$$w(\bar{a}_{2i-2}) - w(a_{2i-2}) = w(D_{2i}) \text{ for } 2i \in \{n+4, ..., 2n\} \ . \tag{4}$$

Finally, we assign profit 1 to each of the jobs in instance R and denote the total number of dummy jobs by h.

Table 2: Start and end times of the jobs a_i and the job sets E_i and D_i

Jobs E_i		Jobs a_i		Job sets D_i	
Job	[start, end]	Job	[start, end]	Job	[start, end]
		a_{2n}	$[n+8, 9n+8]$		
E_2	$[1, n+8]$	a_{2n-1}	$[n+4, 9n+4]$		
		a_{2n-2}	$[n+16, 9n+16]$		
E_4	$[2, n+16]$	a_{2n-3}	$[n+12, 9n+12]$	D_{2n}	$[n+10, 9n+10]$
				D_{2n-2}	$[n+18, 9n+18]$
	$\vdots$		$\vdots$		$\vdots$
		$a_{2n-2i+2}$	$[n+8i, 9n+8i]$		
E_{2i}	$[i, n+8i]$	$a_{2n-2i+1}$	$[n+8i-4, 9n+8i-4]$	$D_{2n-2i+4}$	$[n+8i-6, 9n+8i-6]$
	$\vdots$		$\vdots$		$\vdots$
		a_{n+2}	$[5n, 13n]$		
E_n	$[n/2, 5n]$	a_{n+1}	$[5n-4, 13n-4]$	D_{n+4}	$[5n-6, 13n-6]$
		a_4	$[9n-8, 17n-8]$		
E_{2n-2}	$[n-1, 9n-8]$	a_3	$[9n-12, 17n-12]$		
		a_2	$[9n, 17n]$		
E_{2n}	$[n, 9n]$	a_1	$[9n-4, 17n-4]$		

Jobs $\bar{a}_i$		Jobs $\bar{D}_i$		Jobs $\bar{E}_i$	
Job	[start, end]	Job	[start, end]	Job	[start, end]
$\bar{a}_{2n}$	$[9n+7, 17n+7]$			$\bar{E}_{2n}$	$[17n+7, 25n+7]$
$\bar{a}_{2n-1}$	$[9n+13, 17n+13]$				
$\bar{a}_{2n-2}$	$[9n+15, 17n+15]$			$\bar{E}_{2n-2}$	$[17n+15, 25n+15]$
$\bar{a}_{2n-3}$	$[9n+21, 17n+21]$				
$\dots$	$\dots$			$\dots$	$\dots$
$\bar{a}_{n+2}$	$[13n-1, 21n-1]$			$\bar{E}_{n+2}$	$[21n-1, 29n-1]$
$\bar{a}_{n+1}$	$[13n+5, 21n+5]$				
$\dots$	$\dots$			$\dots$	$\dots$
$\bar{a}_{n-2}$	$[13n-17, 21n-17]$	$\bar{D}_{n-2}$	$[13n-18, 21n-18]$	$\bar{E}_{n-2}$	$[21n-17, 29n-17]$
$\bar{a}_{n-3}$	$[13n+21, 21n+21]$				
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$\bar{a}_{2i}$	$[17n-8i+7, 25n+8i+7]$	$\bar{D}_{2i}$	$[17n-8i+6, 25n+8i+6]$	$\bar{E}_{2i}$	$[25n+8i+7, 33n+8i+7]$
$\bar{a}_{2i-1}$	$[17n-8i+13, 25n+8i+13]$				
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$\bar{a}_4$	$[17n-9, 25n-9]$	$\bar{D}_4$	$[17n-10, 25n-10]$	$\bar{E}_4$	$[25n-9.5, 33n-9.5]$
$\bar{a}_3$	$[17n-3, 25n-3]$				
$\bar{a}_2$	$[17n-1, 25n-1]$	$\bar{D}_2$	$[17n-2, 25n-2]$	$\bar{E}_2$	$[25n-1.5, 33n-1.5]$
$\bar{a}_1$	$[17n+5, 25n+5]$				

Table 3: Start and end times of the jobs $\bar{a}_i$ resp. the job sets $\bar{D}_i$ and $\bar{E}_i$

clique	C_{2i}	I_{2i}	C_{2i}
time	$n+8i$	$17n-8i+8$	$25n+8i+7$

Table 4: The time positions of the maximal cliques in R

Index	Clique	Dummy jobs in the clique	Non-dummy jobs in the clique
$2i \in \{2, 4, \dots, n\}$	C_{2i}	$E_{2i}, E_{2i+2}, \dots, E_{2n}$ $D_{2n-2i+4}, D_{2n-2i+6}, \dots, D_{2n}$	$a_{2n-2i+1}, a_{2n-2i+2}, \dots, a_{2n}$
$2i \in \{n+2, \dots, 2n\}$	C_{2i}	$E_{2i}, E_{2i+2}, \dots, E_{2n}$ $D_{n+4}, D_{n+6}, \dots, D_{2n}$	$a_{2n-2i+1}, a_{2n-2i+2}, \dots, a_{2n}$
$2i \in \{n+2, \dots, 2n\}$	I_{2i}	$D_{n+4}, D_{n+6}, \dots, D_{2i}$	$a_1, a_2, \dots, a_{2i-2}, a_{2i}$ $\bar{a}_{2i}, \bar{a}_{2i+1}, \dots, \bar{a}_{2n}$
$2i \in \{2, 4, \dots, n\}$	I_{2i}	$\bar{D}_{2i}, \bar{D}_{2i+2}, \dots, \bar{D}_{n-2}$	$a_1, a_2, \dots, a_{2i-2}, a_{2i}$ $\bar{a}_{2i}, \bar{a}_{2i+1}, \dots, \bar{a}_{2n}$
$2i \in \{n, \dots, 2n\}$	$\bar{C}_{2i}$	$\bar{E}_{2i}, \bar{E}_{2i+2}, \dots, \bar{E}_{2n}$ $\bar{D}_2, \bar{D}_4, \dots, \bar{D}_{n-2}$	$\bar{a}_1, \bar{a}_2, \dots, \bar{a}_{2i}$
$2i \in \{2, 4, \dots, n-2\}$	$\bar{C}_{2i}$	$\bar{E}_{2i}, \bar{E}_{2i+2}, \dots, \bar{E}_{2n}$ $\bar{D}_2, \bar{D}_4, \dots, \bar{D}_{2i-2}$	$\bar{a}_1, \bar{a}_2, \dots, \bar{a}_{2i}$

Table 5: The maximal cliques in R with their respective jobs and job sets

2.1.3 The maximal cliques in R

We denote the $3n$ maximal cliques of instance R by $C_2, C_4, ...C_{2n}$, $I_2, I_4, ..., I_{2n}$ and $\bar{C}_2, \bar{C}_4, ..., \bar{C}_{2n}$. In Table 4 the times that constitute these cliques are shown. The description of the cliques according to the jobs they contain is given in Table 5. Note that among these cliques, none is a subset of another. Thus, in order to show that these cliques are maximal and that they are the only maximal cliques of the instance it suffices to show that there is no other clique containing one of these $3n$ cliques. Now observe the following facts (going from left to right in Figure 1).

- It is obvious that C_2 is maximal. Further, C_2 is the first maximal clique. Clearly no clique $C \neq C_{2i}$ in between C_{2i} and C_{2i+2} can contain C_{2i} since C does not contain E_{2i}. However, C contains $E_{2i+2}, ..., E_{2n}$, $D_{2n-2i+6}, ..., D_{2n}$ and $a_{2n-2i+1}, ..., a_{2n}$. In addition, it may contain $D_{2n-2i+4}$ and $a_{2n-2i-1}$. Since all these job sets and jobs are contained in the clique C_{2i+2}, the clique C is contained in C_{2i+2}.
  Hence, C_{2i} is maximal for all $i \in \{1, ..., n-1\}$ and no maximal clique exists in between C_{2i} and C_{2i+2}, $i = 1, ..., n-1$. Since by definition no clique $C \neq C_{2n}$ in between C_{2n} and I_{2n} can contain E_{2n}, it follows that C_{2n} is maximal as well.

- Considering the transition from C_{2n} to I_{2n}, note that the job set E_{2n} and the job a_{2n-1} disappear. Noting that $\bar{a}_{2n}$ is the only job $j \in I_{2n}$ that is not contained in C_{2n} it is obvious that each clique $C \neq C_{2n}$ in between C_{2n} and I_{2n} is contained in I_{2n}.
  Furthermore, any job that "starts" in between I_{2i} and I_{2i-2} is contained in the clique I_{2i-2}. Due to these observations there are no maximal cliques in between I_{2i} and I_{2i-2}, and the cliques I_{2i} are maximal, $2 \leq i \leq n$.
  Since no clique $C \neq I_2$ in between I_2 and $\bar{C}_{2n}$ contains a_2, also I_2 is a maximal clique.

- Analogously to the above observations, it follows that every job that "starts" in between I_2 and $\bar{C}_{2n}$ is contained in $\bar{C}_{2n}$, and every job that "starts" in between $\bar{C}_{2i}$ and $\bar{C}_{2i-2}$ is contained in $\bar{C}_{2i-2}$, $i = 1, ..., n-1$. Furthermore, it is clear that $\bar{C}_2$ is the last maximal clique. As a consequence, all the cliques $C_2, C_4, C_6, ...C_{2n}$, $I_2, I_4, ..., I_{2n}$ and $\bar{C}_2, \bar{C}_4, \bar{C}_6, ..., \bar{C}_{2n}$ are maximal cliques, and besides these cliques no maximal clique exists.

2.1.4 Preliminary definitions and observations

In course of this paper the following definitions will be useful.

Definition 5 *Let X be a feasible solution of R and C a maximal clique in R. Then we define*

- *the set $K_{2j}(X)$ of jobs of $\{a_{2j}, a_{2j-1}\}$ that are contained in X. Formally, $K_{2j}(X) := \{a_{2j}, a_{2j-1}\} \cap X$.*
- *the set $A(X) := \{a_i | a_i \in X,\ 1 \leq i \leq 2n\}$ of jobs a_i contained in X, $1 \leq i \leq 2n$.*
- *the set $B_{2j}(X) := \{a_k | a_k \in (X \cap C_{2j})\}$ of jobs $a_k \in C_{2j}$ that are contained in X.*
- *the total resource consumption of clique C with respect to X by $r_X(C) := \sum_{a \in C \cap X} w(a)$.*
- *the set $J(X)$ of indices $j \in \{1, ..., n\}$ such that both elements of the set $\{a_{2n-2j+2}, a_{2n-2j+1}\}$ are contained in X. Formally spoken, $J(X) := \{j \in \{1, ..., n\} | \, |K_{2n-2j+1}(X)| = 2\} = \{j \in \{1, ..., n\} | a_{2n-2j+2} \in X$ and $a_{2n-2j+1} \in X\}$.*
- *$m := \min J(X^*)$ and $J'(X^*) := \{j < m | \, |K_{2n-2j+2}(X^*)| = 0,\ j \geq 1\}$.*

The sets $\bar{K}_{2j}(X)$, $\bar{A}(X)$, $\bar{B}_{2j}(X)$ are defined analogously by replacing each job a_i with job $\bar{a}_i$ and each clique C_{2j} by $\bar{C}_{2j}$ respectively. For formal reasons however, the set $\bar{J}(X)$ is defined by $\bar{J}(X) := \{j \in \{1, ..., n\} | \bar{a}_{2j} \in X$ and $\bar{a}_{2j-1} \in X\}$. With $\bar{m} := \min \bar{J}(X^*)$ we define $\bar{J}'(X^*) := \{j < m | \, |\bar{K}_{2j}(X^*)| = 0,\ j \geq 1\}$.

An important feature of instance R is the total resource consumption of the dummy jobs contained in the maximal cliques. The following proposition summarizes the main resource consumptions; since the proposition is a result of simple algebra, the proof is omitted here for the sake of brevity.

Proposition 1 *For instance R, the following statements hold:*

1. *The total resource consumption of the dummy jobs $D_{n+4}, D_{n+6}, ..., D_{2n}$ sums up to $M(\frac{n^2}{2} - 2n + 2)$.*
2. *For $1 \leq i \leq n-1$, the total resource consumption of the dummy jobs contained in the clique C_{2i} equals $\delta(C_{2i}) := M(\frac{3}{2}n^2 + n - \sum_{j=1}^{i} 2j)$.*
   The total resource consumption of the dummy jobs contained in the clique C_{2n} equals $\delta(C_{2n}) := M[(\frac{3}{2}n^2 + n - \sum_{j=1}^{n} 2j) + \frac{1}{2}]$
3. *The total resource consumption of the dummy jobs $\bar{D}_2, \bar{D}_4, ..., \bar{D}_{n-2}$ sums up to $M(\frac{n^2}{2} - 2n + 2)$.*
4. *For $1 \leq i \leq n-1$, the resource consumption of the dummy jobs in the clique $\bar{C}_{2i}$ equals $\delta(\bar{C}_{2i}) := M(\frac{3}{2}n^2 + n - \sum_{j=1}^{i} 2j)$.*
   The total resource consumption of the dummy jobs contained in the clique $\bar{C}_{2n}$ equals $\delta(\bar{C}_{2n}) := M[(\frac{3}{2}n^2 + n - \sum_{j=1}^{n} 2j) + \frac{1}{2}]$

2.2 Properties of solutions of R

In this section the key features of optimal solutions of R used for proving the $\mathcal{NP}$-hardness result in Section 2.3 are presented. Here the following lemma turns out to be useful.

Lemma 3 *Let X be a feasible solution of R. Let $J(X) \neq \emptyset$. Let $m := \min J(X)$. For some l, $1 \leq l < m$, let neither $a_{2n-2l+2}$ nor $a_{2n-2l+1}$ be contained in X. Then the solution $x_1 := X \cup \{a_{2n-2l+1}\} \setminus \{a_{2n-2m+1}\}$ is a feasible solution of R.*

Proof. In order to guarantee the feasibility of x_1 we need to show that the resource constraint is satisfied for all the maximal cliques. Obviously the constraint holds for all the cliques that do not contain $a_{2n-2l+1}$. Thus the constraint is trivially satisfied for all the cliques $C_{2l'}$ with $1 \leq l' < l$. Note that due to the definition of m, for all $j < m$, $j \geq 1$, the jobs $a_{2n-2j+2}$ and $a_{2n-2j+1}$ cannot both be contained in X. Now consider the clique C_{2l}. Additionally, X does not contain $a_{2n-2l+2}$. Thus for all $j < m$, $j \geq 1$, we get that the solution x_1 does contain at most one of $\{a_{2n-2j+2}, a_{2n-2j+1}\}$. Observing that $j < m$ implies $j < n$ and that $\sum_{a_k \in B_{2j}(x_1)} a_k < M$ holds we get

$$\begin{array}{rcl} r_{x_1}(C_{2j}) & \leq & \delta(C_{2j}) + \sum_{a_k \in B_{2j}(x_1)} a_k \\ & < & M[(\frac{3}{2}n^2 + n) - \sum_{k=1}^{j} 2k)] + M(\sum_{k=1}^{j} 2k) + M \\ & = & M(\frac{3}{2}n^2 + n + 1) \\ & = & W \end{array} \tag{5}$$

and hence the constraint is satisfied for all the cliques C_{2j} with $j < m$ and $j \geq 1$.
For the cliques C_{2j} with $j \geq m$, $j \leq n$, it is sufficient to show that $a_{2n-2m+1} > a_{2n-2l+1}$ holds. This follows directly from the definition of M and the fact that $m > l$:

$$a_{2n-2m+1} = b_{2n-2m+1} + 2mM > b_{2n-2l+1} + 2lM = a_{2n-2l+1}$$

Analogously, x_1 satisfies the resource constraint for all the cliques I_{2i}, $1 \leq i \leq n$, since each of these cliques that contains $a_{2n-2l+1}$ contains $a_{2n-2m+1}$ as well. Finally, the resource constraints for the cliques $\bar{C}_{2i}$, $1 \leq i \leq n$, are trivially satisfied because neither $a_{2n-2l+1}$ nor $a_{2n-2m+1}$ is contained in any of these cliques. □

Lemma 4 *Let X be a feasible solution of R. Let $J(X) \neq \emptyset$. Let $m := \min J(X)$. If there are two dummy jobs $b, c \in C_{2m}$ that are not contained in X, then $x_1 := X \cup \{b, c\} \setminus \{a_{2n-2m+1}\}$ is a feasible solution of R.*

Proof. Note that $w(b) = w(c) = \frac{M}{2}$ implies $w(b) + w(c) < w(a_{2n-2m+1})$. Thus, the weight constraint of x_1 is obviously satisfied for all cliques that contain $a_{2n-2m+1}$. It remains to show that the resource constraint is satisfied for the cliques that do not contain $a_{2n-2m+1}$ but contain at least one element of $\{b, c\}$; i.e. for the cliques C_{2k}, $1 \leq k < m$. If $m = 1$ there is nothing to show. Let $m > 1$. Recall that due to the definition of m, for all $j \leq m$, $j \geq 1$, we get $|K_{2n-2j+2}(X)| = |\{a_{2n-2j+2}, a_{2n-2j+1}\} \cap X| \leq 1$. Thus, for the resource consumption of the clique C_{2j} we get $r_{x_1}(C_{2j}) < W$ analogous to (5). □

With Lemma 3 and 4 it can be proven that any optimal solution of R contains at most n jobs of $\{a_1, a_2, ..., a_{2n}\}$ and at most n jobs of $\{\bar{a}_1, \bar{a}_2, ..., \bar{a}_{2n}\}$. This key feature is stated in the following lemma.

Lemma 5 *Let X^* be an optimal solution of R. Then $|A(X^*)| \leq n$ and $|\bar{A}(X^*)| \leq n$.*

Proof. We give the proof for $|A(X^*)| \leq n$, the proof for $|\bar{A}(X^*)| \leq n$ follows analogously. Assume

$$|A(X^*)| \geq n + 1 \ . \tag{6}$$

Hence for some j^*, $1 \leq j^* \leq n$, both a_{2n-2j^*+2} and a_{2n-2j^*+1} must be contained in X^*. This implies that the set

$$J(X^*) = \{j \in \{1, ..., n\} \,|\, |K_{2n-2j+2}(X^*)| = 2\}$$

is non-empty. Let $m := \min J(X^*)$. Obviously, for all $j < m$, $j \geq 1$, we have $|K_{2n-2j+2}(X^*)| \leq 1$. Now we distinguish the following cases.
Case 1. All the dummy jobs are contained in X^*.
Case 1a. $|J'(X^*)| = 0$. I.e., for all $j < m$, $j \geq 1$, we have $|K_{2n-2j+2}(X^*)| = 1$.
Consider the clique C_{2m}. For this clique, the constraint on the resource consumption is

$$\delta(C_{2m}) + \sum_{a_k \in B_{2m}(X^*)} a_k \leq W \tag{7}$$

However, with $m \geq 1$ we get

$$\begin{array}{rcl} \delta(C_{2m}) + \sum_{a_k \in B_{2m}(X^*)} a_k & > & \\ M(\frac{3}{2}n^2 + n - \sum_{j=1}^{m} 2j) + M(\sum_{j=1}^{m} 2j + 2m) & = & \\ M(\frac{3}{2}n^2 + n + 2m) & > & \\ M(\frac{3}{2}n^2 + n + 1) & = & W \end{array} \tag{8}$$

contradicting to (7). Thus we must have $|J'(X^*)| \geq 1$.

Case 1b. $|J'(X^*)| \geq 1$. For some $l \in J'(X^*)$ let $y_1 := X^* \cup \{a_{2n-2l+1}\} \setminus \{a_{2n-2m+1}\}$. y_1 is a feasible solution because of Lemma 3. Additionally we have $p(y_1) = p(X^*)$, $|A(y_1)| = |A(X^*)| \geq n + 1$ and $|J'(y_1)| = |J'(X^*)| - 1$.
Thus, for every optimal solution X with $|A(X)| \geq n+1$ and $|J'(X)| \geq 1$ we can create an optimal solution y with $|A(y)| \geq n + 1$ and $|J'(y)| = |J'(X)| - 1$. Hence, by iterating we finally find an optimal solution y with $|J'(y)| = 0$. However, Case 1a shows that this is not possible. Therefore, not all dummy jobs can be contained in X^*.

Case 2. There exists a dummy job that is not contained in X^*.
Case 2a. $|J'(X^*)| = 0$. I.e., for all $j < m$, $j \geq 1$, we have $|K_{2n-2j+2}(X^*)| = 1$. Let z be the

number of dummy jobs contained in C_{2m} that are elements of X^*.
If $m < n$, then we get from the weight constraint for the clique C_{2m}

$$\begin{array}{rcl} W & \geq & r_{X^*}(C_{2m}) \\ & \geq & \sum_{a_k \in B_{2m}(X^*)} a_k + z\frac{M}{2} \\ & = & M(\sum_{j=1}^{m} 2j + 2m) + z\frac{M}{2} \\ & = & M[m(m+1) + 2m + \frac{z}{2})] \\ & = & M[m(m+3) + \frac{z}{2}] \end{array}$$

Thus,

$$\begin{array}{rcl} & z & \leq & 2\frac{W - M(m(m+3))}{M} \\ \Leftrightarrow & z & \leq & 2[\frac{3}{2}n^2 + n + 1 - m(m+3)] \end{array}$$

However, from statement (2) in Proposition 1 we know that in the clique C_{2m} there are $2(\frac{3}{2}n^2 + n - \sum_{j=1}^{m} 2j)$ dummies contained. This implies that there are at least

$$\begin{array}{rcl} 2[(-\sum_{j=1}^{m} 2j) - 1 + m(m+3)] & = \\ 2[-m(m+1) - 1 + m(m+3)] & = \\ 4m - 2 \end{array}$$

dummy jobs in C_{2m} that are not contained in X^*. Since $m \geq 1$, this means that there exist 2 dummies $b, c \in C_{2m}$ that are not contained in X^*. Due to Lemma 4 the solution $X := X^* \cup \{b, c\} \setminus \{a_{2n-2m+1}\}$ is feasible. However, $p(X) = p(X^*) + 1$ in contradiction to the optimality of X^*. Thus we must have $|J'(X^*)| \geq 1$.

Case 2b. $|J'(X^*)| \geq 1$. With $l = \min J'(X^*)$ the solution $y_1 := X^* \cup \{a_{2n-2l+1}\} \setminus \{a_{2n-2m+1}\}$ is feasible due to Lemma 3. We have $|A(y_1)| = |A(X^*)| \geq n + 1$ and thus $J(y_1) \neq \emptyset$. Furthermore, $p(y_1) = p(X^*)$ and $|J'(y_1)| = |J'(X^*)| - 1$. As in Case 1 either $|J'(y_1)| = 0$ or we can proceed to find an optimal solution y with $|J'(y)| = 0$. Either way we end up in Case 2a which leads to a contradiction. □

Lemma 5 immediately yields that the value of an optimal solution is at most $2n + h$ (recall that h is the number of dummy jobs in R). This is stated in Corollary 6.

Corollary 6 *Let X^* be an optimal solution of R. Then $p(X^*) \leq 2n + h$.*

From the proof of Lemma 5 we also get the following corollary.

Corollary 7 *Let X^* be an optimal solution of R that contains all dummy jobs. Then the following statements hold:*

1. *If $J(X^*) \neq \emptyset$, then $|J'(X^*)| \geq 2$.*
2. *Let $\bar{J}(X^*) \neq \emptyset$, then $|\bar{J}'(X^*)| \geq 2$.*

Proof. We prove statement 1, statement 2 follows in an analogous manner.
Assume $J'(X^*) = \emptyset$. Then we get $r_{X^*}(C_{2m}) > W$ exactly as in (8). This violates the resource constraint for the clique C_{2m}.
Next, assume that $|J'(X^*)| = 1$. Let $l \in J'(X^*)$. Then we get due to $l < m$

$$\begin{array}{rcl} r_{X^*}(C_{2m}) & = \\ \delta(C_{2m}) + \sum_{a_k \in B_{2m}(X^*)} a_k & > \\ M(\frac{3}{2}n^2 + n - \sum_{j=1}^{m} 2j) + M(\sum_{j=1}^{m} 2j + 2m - 2l) & = \\ M(\frac{3}{2}n^2 + n + 2(m-l)) & > \\ M(\frac{3}{2}n^2 + n + 1) & = & W \end{array}$$

which again contradicts the resource constraint for C_{2m}. □

With the use of Lemma 5 and Corollary 7 the following important property of an optimal solution X^* of R can be derived: if $p(X^*) = 2n+h$, then for all $j \in \{1, ..., n\}$ exactly one job of $\{a_{2j}, a_{2j-1}\}$ (and exactly one of $\{\bar{a}_{2j}, \bar{a}_{2j-1}\}$) must be contained in X^*. This property is stated in Lemma 8 below and constitutes the key feature used in the proof of our $\mathcal{NP}$-hardness result in Section 2.3.

Lemma 8 *Let X^* be an optimal solution of R with $p(X^*) = 2n + h$. Then the following statements hold:*

1. $|A(X^*)| = n$ *and* $|\bar{A}(X^*)| = n$.
2. $|K_{2j}(X^*)| = 1$ *and* $|\bar{K}_{2j}(X^*)| = 1$ *for all* j, $1 \leq j \leq n$.

Proof. The first statement follows directly from Lemma 5. We show the second statement for $K_{2n-2j+2}(X^*)$, $1 \leq j \leq n$ – the proof for $\bar{K}_{2j}(X^*)$, $1 \leq j \leq n$ is analogous.
Assume that for some $j \in \{1, ..., n\}$ we have $|K_{2n-2j+2}(X^*)| \neq 1$. Since $|A(X^*)| = n$ this implies that the set $J(X^*)$ is non-empty. Let $m := \min J(X^*)$. Corollary 7 implies that $|J'(X^*)| = |\{j < m | |K_{2n-2j+2}(X^*)| = 0, j \geq 1\}| \geq 2$ holds. Now let $l := \min J'(X^*)$. Then due to Lemma 3 $y_1 := X^* \cup \{a_{2n-2l+1}\} \setminus \{a_{2n-2m+1}\}$ is a feasible solution of R. Note that

$$|J(y_1)| = |J(X^*)| - 1 \ .$$

Further, we have $|J'(y_1)| = |J'(X^*)| - 1 \geq 1$. This fact and the equation $|A(y_1)| = |A(X^*)| = n$ imply $|J(y_1)| \geq 1$. Applying Corollary 7 again yields $|J'(y_1)| \geq 2$. Thus, for every optimal solution X with and $|A(X)| = n$ and $|K_{2j}(X)| \neq 1$ for some j we can find another optimal solution y with $|A(y)| = n$ and $|J(y)| = |J(X)| - 1$. Hence, iteration leads to an optimal solution y with $|A(y)| = n$ and $|J(y)| = 1$. However, Corollary 7 yields $|J'(y)| \geq 2$ and thus $|A(y)| < n$.□

Another useful structural property of an optimal solution X^* of R with $p(X^*) = 2n + h$ is stated in the next lemma: the jobs a_{2i} and $\bar{a}_{2i}$ cannot be simultaneously contained in such a solution X^*, for all i, $1 \leq i \leq n$.

Lemma 9 *Let X^* be an optimal solution of R with $p(X^*) = 2n + h$. Then $a_{2i} \in X^* \Rightarrow \bar{a}_{2i} \notin X^*$ for all i, $1 \leq i \leq n$.*

Proof. Statement 1 of Lemma 8 implies that all dummy jobs must be contained in X^*. We give the proof for $\frac{n}{2} + 1 \leq i \leq n$, the proof for $1 \leq i \leq \frac{n}{2}$ follows in analogous manner.
Let $a_{2i} \in X^*$ for some i, $\frac{n}{2} + 1 \leq i \leq n$. Assume $\bar{a}_{2i} \in X^*$. Recall that the clique I_{2i} contains

- the jobs a_j with $1 \leq j \leq 2i$, $j \neq 2i - 1$
- the jobs $\bar{a}_j$ with $j \geq 2i$ and
- the dummy job sets D_{2j} with $2j \in \{n + 4, n + 6, ..., 2i\}$ (see Table 5).

Statement 2 of Lemma 8 implies that for all j, either a_{2j} or a_{2j-1} is contained in X^*. By construction, $w(a_{2j}) < w(a_{2j-1})$ and $w(\bar{a}_{2j}) < w(\bar{a}_{2j-1})$ for all j (see (2)). Furthermore, recall that $w(\bar{a}_j) > w(a_j)$ for $j \geq n + 1$. This yields for the resource constraint for clique I_{2i}:

$$\begin{aligned} r_{X^*}(I_{2i}) \quad \geq \quad & \textstyle\sum_{j=1}^{i-1} w(a_{2j}) + w(a_{2i}) + w(\bar{a}_{2i}) \\ & + \textstyle\sum_{j=i+1}^{n} w(\bar{a}_{2j}) + \sum_{j=\frac{n}{2}+2}^{i} w(D_{2j}) \end{aligned}$$

From (4) we know that $w(\bar{a}_{2j}) - w(a_{2j}) = w(D_{2j+2})$ for all $n + 2 \leq 2j \leq 2n - 2$. Thus replacing, for $n + 2 \leq 2j \leq 2n - 2$, each term $w(\bar{a}_{2j})$ by $w(a_{2j}) + w(D_{2j+2})$ we get

$$\begin{aligned} r_{X^*}(I_{2i}) \quad \geq \quad & \textstyle\sum_{j=1}^{i-1} w(a_{2j}) + w(a_{2i}) + w(a_{2i}) + w(D_{2i+2}) + w(\bar{a}_{2n}) \\ & + \textstyle\sum_{j=i+1}^{n-1} w(a_{2j}) + \sum_{j=i+1}^{n-1} w(D_{2j+2}) + \sum_{j=\frac{n}{2}+2}^{i} w(D_{2j}) \\ = \quad & w(a_{2i}) + w(\bar{a}_{2n}) + \textstyle\sum_{j=1}^{n-1} w(a_{2j}) + \sum_{j=\frac{n}{2}+2}^{n} w(D_{2j}) \end{aligned}$$

Recall that $w(a_{2i}) > 2M$, $w(\bar{a}_{2n}) > 2nM$, and $\sum_{j=1}^{n-1} w(a_{2j}) > M \sum_{j=2}^{n} 2j$ hold. Together with Statement 1 of Proposition 1 this yields

$$\begin{array}{rcl} r_{X^*}(I_{2i}) & > & (2n+2)M + M[n(n+1)-2] + M(\frac{n^2}{2} - 2n + 2) \\ & = & M(\frac{3}{2}n^2 + n + 2) \\ & > & W \ , \end{array}$$

violating the resource constraint of I_{2i}. Thus, $a_{2i} \in X^*$ implies $\bar{a}_{2i} \notin X^*$ for all $\frac{n}{2} + 1 \leq i \leq n$.□

Lemma 10 *Let X^* be an optimal solution of R with $p(X^*) = 2n + h$. Then $\sum_{i:a_i \in X^*} b_i \leq \frac{M}{2}$ and $\sum_{i:\bar{a}_i \in X^*} b_i \leq \frac{M}{2}$.*

Proof. In order to show that $\sum_{i:a_i \in X^*} b_i \leq \frac{M}{2}$ holds, we consider the clique C_{2n}. Because of Statement 2 in Lemma 8 we know that for all $1 \leq i \leq n$, exactly one element of $\{a_{2j}, a_{2j-1}\}$ is contained in X^*. Because of statement 1 in Lemma 8 all dummy jobs must be contained in X^*. Thus, with Statement 2 of Proposition 1 we get

$$\begin{array}{rcl} r_{X^*}(C_{2n}) & = & M \sum_{j=1}^{n} 2j + \sum_{i:a_i \in X^*} b_i + \delta(C_{2n}) \\ & = & \sum_{i:a_i \in X^*} b_i + M[\sum_{j=1}^{n} 2j + (\frac{3}{2}n^2 + n - \sum_{j=1}^{n} 2j) + \frac{1}{2}] \\ & = & \sum_{i:a_i \in X^*} b_i + M(\frac{3}{2}n^2 + n + \frac{1}{2}) \end{array}$$

Hence, $r_{X^*}(C_{2n}) \leq W$ yields

$$\begin{array}{rcl} \sum_{i:a_i \in X^*} b_i + M(\frac{3}{2}n^2 + n + \frac{1}{2}) & \leq & M(\frac{3}{2}n^2 + n + 1) \\ \Leftrightarrow \qquad \sum_{i:a_i \in X^*} b_i & \leq & \frac{M}{2} \end{array}$$

Analogously it follows that $\sum_{i:\bar{a}_i \in X^*} b_i \leq \frac{M}{2}$ holds. □

Remark. The above lemma states that summing up the b_i over all indices i such that $a_i \in X^*$ (resp. $\bar{a}_i \in X^*$) does not exceed the bound $\frac{M}{2}$. Thus, it clearly indicates the idea used to derive a feasible solution of the ordered partition instance OP from an optimal solution X^* of R with $p(X^*) = 2n + h$ in the proof of the $\mathcal{NP}$-hardness result in the next section.

2.3 The $\mathcal{NP}$-hardness result

In this section we show that the resource allocation problem with uniform profits is $\mathcal{NP}$-hard even if the interval graph is proper. The proof of the $\mathcal{NP}$-hardness consists of a transformation of an arbitrary instance OP of the ordered partition problem with an even number n of integer-pairs (see Definition 4) to the decision problem corresponding to the resource allocation problem.

Given such an instance OP, we constructed an instance R of the resource-allocation problem in Section 2.1. With the structural properties of an optimal solution of R presented in Section 2.2, we will now proof our first main result.

Theorem 11 *The resource allocation problem RAP is $\mathcal{NP}$-hard, even if restricted to uniform profits and proper interval graphs.*

We prove the theorem by showing that OP is a "YES"-instance of ordered partition if and only if there is a solution X^* of R with $p(X^*) \geq 2n + h$.

Sketch of the proof of Theorem 11. The "if"-part of this equivalence is proven in a straightforward way. Given a solution B' of OP, simply construct a solution X^* of R by

1. letting $a_i \in X^* \Leftrightarrow b_i \in B'$ and $\bar{a}_i \in X^* \Leftrightarrow b_i \in B \setminus B'$ and

2. inserting all the dummy jobs in X^*.

With simple algebra the feasibility of X^* is shown and the optimality of X^* follows directly from Corollary 6.

The "only-if"-part is proven as follows. Given a solution X^* of R with $p(X) = 2n + h$, we construct the sets B_1 and $\bar{B}_1$ by letting $b_i \in B_1 \Leftrightarrow a_i \in X^*$ and $b_i \in \bar{B}_1 \Leftrightarrow \bar{a}_i \in X^*$. Lemma 8 and Lemma 10 ensure that B_1 is a solution of OP if B_1 and $\bar{B}_1$ are disjoint. Thus, the final part of the proof consists in showing that B_1 and $\bar{B}_1$ are disjoint by simply using the definition of the ordered partition problem.

Proof of Theorem 11.

"$\Rightarrow$": Let $B' \subset B$ be a solution of OP. Thus,

$$\sum_{b_i \in B'} b_i = \sum_{b_i \in B \setminus B'} b_i = \frac{M}{2} \ , \tag{9}$$

$|B'| = n$ and, for $1 \leq j \leq n$,

$$b_{2j} \in B' \Leftrightarrow b_{2j-1} \in B \setminus B' \ . \tag{10}$$

Let H denote the set of all dummy jobs in R. Let $X^* := \{a_i | b_i \in B', 1 \leq i \leq 2n\} \cup \{\bar{a}_i | b_i \in B \setminus B'\} \cup H$. Obviously, $p(X^*) = 2n + h$. Furthermore note that

$$a_i \in X^* \Leftrightarrow \bar{a}_i \notin X^* \tag{11}$$

for all $1 \leq i \leq 2n$. It remains to show is that X^* is a feasible solution of R.
(10) implies $K_{2j}(X^*) = 1$ and $\bar{K}_{2j}(X^*) = 1$ for all $1 \leq j \leq n$. Thus, for the cliques C_{2j}, $1 \leq j \leq n$, we get

$$\begin{array}{rcl} r_{X^*}(C_{2j}) & = & \delta(C_{2j}) + \sum_{a_k \in B_{2j}(X^*)} w(a_k) \\ & = & \delta(C_{2j}) + M \sum_{k=1}^{j} 2k + \sum_{k:a_k \in B_{2j}(X^*)} b_k \end{array}$$

Hence, for $1 \leq j < n$, (9) and (10) together with Proposition 1 yield

$$\begin{array}{rcl} r_{X^*}(C_{2j}) & < & M[(\frac{3}{2}n^2 + n - \sum_{k=1}^{j} 2k)] + M \sum_{k=1}^{j} 2k + \frac{M}{2} \\ & = & M[(\frac{3}{2}n^2 + n + \frac{1}{2}) \\ & < & W \end{array}$$

For clique C_{2n} we get

$$\begin{array}{rcl} r_{X^*}(C_{2n}) & = & M[(\frac{3}{2}n^2 + n - \sum_{k=1}^{n} 2k) + \frac{1}{2}] + M \sum_{k=1}^{n} 2k + \frac{M}{2} \\ & = & M[(\frac{3}{2}n^2 + n + 1) \\ & = & W \end{array}$$

Thus, the resource constraint holds for all cliques C_{2j}, $1 \leq j \leq n$. Analogously it follows that the resource constraint is satisfied for all cliques $\bar{C}_{2j}$, $1 \leq j \leq n$.
Next consider the clique I_{2n}. Recall that Clique I_{2n} contains

- the jobs a_i, $1 \leq i \leq 2n$, $i \neq 2n - 1$,
- the job $\bar{a}_{2n}$ and
- all the dummy job sets $D_{n+4}, D_{n+6}, ..., D_{2n}$.

Note that (11) implies either $a_{2n} \in X^*$ or $\bar{a}_{2n} \in X^*$. Furthermore recall that $w(a_{2i}) < w(a_{2i-1})$ (resp. $w(\bar{a}_{2i}) < w(\bar{a}_{2i-1})$) for $1 \leq i \leq n$ and $w(\bar{a}_{2i}) > w(a_{2i})$ for $2i \in \{n+4, n+6, ..., 2n\}$ (see Table 1). Thus, with $\sum_{i=1}^{n} b_{2i} < M$ we get

$$\begin{array}{rcc} r_{X^*}(I_{2n}) & \leq & \sum_{i=\frac{n}{2}+2}^{n} w(D_{2i}) + w(\bar{a}_{2n}) + \sum_{i=1}^{n-1} w(a_{2i-1}) \\ & < & \alpha := \sum_{i=\frac{n}{2}+2}^{n} w(D_{2i}) + w(\bar{a}_{2n-1}) + \sum_{i=1}^{n-1} w(a_{2i-1}) \\ & = & M(\frac{n^2}{2} - 2n + 2) + (b_{2n} + 2nM) + \sum_{i=1}^{n-1} b_{2i-1} + M \sum_{i=2}^{n} 2i \\ & < & M(\frac{n^2}{2} - 2n + 2) + \sum_{i=1}^{n} b_{2i-1} + 2nM + Mn(n+1) - 2M \\ & = & M(\frac{3}{2}n^2 + n) + \sum_{i=1}^{n} b_{2i-1} \\ & < & W \end{array}$$

Hence, for the clique I_{2n} the resource constraint is satisfied.
Now clique I_{2n-2} contains the jobs

- the jobs a_i, $1 \leq i \leq 2n-2$, $i \neq 2n-2$, the job a_{2n-2}
- the jobs $\bar{a}_{2n-2}$, $\bar{a}_{2n-1}$, $\bar{a}_{2n}$ and
- all the dummy job sets $D_{n+4}, D_{n+6}, ..., D_{2n-2}$.

Analogous to above we get

$$r_{X^*}(I_{2n-2}) \leq \sum_{i=\frac{n}{2}+2}^{n} w(D_{2i}) - w(D_{2n}) + w(\bar{a}_{2n-1}) + w(\bar{a}_{2n-2}) + \sum_{i=1}^{n-2} w(a_{2i-1}) \ . \quad (12)$$

Now recall that $w(\bar{a}_{2i-2}) - w(a_{2i-2}) = w(D_{2i})$ for $2i \in \{n+4, ..., 2n\}$ (stated in (3)). Substituting $w(D_{2n}) = w(\bar{a}_{2n-2}) - w(a_{2n-2})$ in (12) yields

$$\begin{array}{rcc} r_{X^*}(I_{2n-2}) & \leq & \sum_{i=\frac{n}{2}+2}^{n} w(D_{2i}) - w(\bar{a}_{2n-2}) + w(a_{2n-2}) + w(\bar{a}_{2n-1}) + w(\bar{a}_{2n-2}) + \sum_{i=1}^{n-2} w(a_{2i-1}) \\ & < & \sum_{i=\frac{n}{2}+2}^{n} w(D_{2i}) + w(\bar{a}_{2n-1}) + \sum_{i=1}^{n-1} w(a_{2i-1}) \\ & = & \alpha \\ & < & W \end{array}$$

Analogously, we get $r_{X^*}(I_{2i}) < \alpha < W$ for all $2i \in \{n+2, ..., 2n\}$.
For $2i \in \{2, ..., n\}$, starting with I_2 and using the equality $w(a_{2j-2}) - w(\bar{a}_{2j-2}) = D_{2j}$ for $2j \in \{4, ..., n\}$ (stated in (3)) the inequality $r_{X^*}(I_{2i}) < W$ can be shown in an analogous manner. Thus, since $p(X^*) = 2n + h$, X^* is a feasible and optimal solution of R.

"$\Leftarrow$": Let X^* be a feasible solution of R with $p(X^*) = 2n + h$. Obviously, X^* is an optimal solution due to Corollary 6.
Let $B_1 := \{b_i | a_i \in X^*, 1 \leq i \leq 2n\}$ and $\bar{B}_1 := \{b_i | \bar{a}_i \in X^*, 1 \leq i \leq 2n\}$. As stated in Lemma 10,

$$\sum_{i:b_i \in B_1} b_i \leq \frac{M}{2} \text{ and } \sum_{i:b_i \in \bar{B}_1} b_i \leq \frac{M}{2} \quad (13)$$

hold.
Due to Lemma 8 $|A(X^*)| = n$ (resp. $|\bar{A}(X^*)| = n$) and thus $|B_1| = n$ (resp. $|\bar{B}_1| = n$). Note that the fact that exactly one element of $\{a_{2j}, a_{2j-1}\}$ (resp. $\{\bar{a}_{2j}, \bar{a}_{2j-1}\}$) is contained in X^*(see Lemma 8) implies that exactly one element of $\{b_{2j}, b_{2j-1}\}$ is contained in B_1 (resp. $\bar{B}_1$), for $1 \leq j \leq n$.
If $B_1 \cap \bar{B}_1 = \emptyset$ then $B \setminus B_1 = \bar{B}_1$. Together with (13) this implies that B_1 is a solution of OP. Thus, it remains to show that $B_1 \cap \bar{B}_1 = \emptyset$ holds.
Assume $B_1 \cap \bar{B}_1 \neq \emptyset$. Because of Lemma 9 we know that $b_i \in (B_1 \cap \bar{B}_1)$ implies $i = 2j - 1$ for some $1 \leq j \leq n$. Furthermore note that because of Lemma 8

$$b_{2j-1} \in (B_1 \cap \bar{B}_1) \Rightarrow b_{2j} \notin B_1, \ b_{2j} \notin \bar{B}_1 \quad (14)$$

for all $1 \leq j \leq n$. Now create the set B_2 from B_1 by replacing every $b_{2j-1} \in (B_1 \cap \bar{B}_1)$ with b_{2j}, $1 \leq j \leq n$. Formally spoken,

$$B_2 := (B_1 \setminus \bar{B}_1) \cup \{b_{2j} | b_{2j-1} \in (B_1 \cap \bar{B}_1),\ 1 \leq j \leq n\} \ .$$

Note that (14) yields $B_2 \cap \bar{B}_1 = \emptyset$. Obviously, $|B_2| = n$ because of $|B_1| = n$. Thus we must have $B \setminus B_2 = \bar{B}_1$ and hence

$$\sum_{i=1}^{2n} b_i = \sum_{i:b_i \in B_2} b_i + \sum_{i:b_i \in \bar{B}_1} b_i \ . \tag{15}$$

Since by definition $b_{2j} < b_{2j-1}$ for $1 \leq j \leq n$, we get

$$\sum_{i:b_i \in B_2} b_i \quad < \quad \sum_{i:b_i \in B_1} b_i \quad \leq \quad \frac{M}{2} \tag{16}$$

where the last inequality was stated in (13). However, inserting (13) and (16) in (15) yields

$$\begin{array}{rcl} \sum_{i=1}^{2n} b_i & = & \sum_{i:b_i \in B_2} b_i + \sum_{i:b_i \in \bar{B}_1} b_i \\ & < & \frac{M}{2} + \frac{M}{2} \\ & = & M \end{array}$$

which contradicts the definition of M. Thus, $B_1 \cap \bar{B}_1 = \emptyset$ which completes the proof. □

3 A $(\frac{1}{2} - \varepsilon)$–Approximation Algorithm

In this section we present an approximation algorithm with worst-case performance guarantee $\frac{1}{2} - \varepsilon$ for instances I of the resource allocation problem RAP in which the time intervals of all jobs are proper intervals. This algorithm is based on the clique path representation of the proper interval graph which has a one-to-one correspondence to instance I. We first give the formal definition of chordal graphs and clique trees and cite some facts from the literature needed to prove the correctness of our algorithm.

Definition 6 *A graph $G = (V, E)$ is called chordal graph, if it does not contain induced cycles other than triangles [12].*

Lemma 12 *Every interval graph is chordal [17].*

Definition 7 *A clique tree $T = (\mathcal{K}, \mathcal{E})$ of a chordal graph G is a tree that has all the maximal cliques C of G as vertices and for each vertex $v \in G$ all the cliques C containing v induce a subtree in T [6].*

Definition 8 *A subset $S \subset V$ is called* separator of G *if there are two vertices a and b in one component of G, such that after the removal of S from V a and b are in different components of $G - S$.*

Lemma 13 *Let T be a clique tree of the chordal graph G and let C_1 and C_2 be two maximal cliques adjacent in T. Then $C_1 \cap C_2$ is a minimal separator w.r.t. G for all $a \in C_1 \setminus C_2$ and $b \in C_2 \setminus C_1$ [15].*

Theorem 14 *Let G be a connected chordal graph. Then G is a proper interval graph if and only if G has a unique clique tree and this tree is a path P such that for every subsequence (C_1, C_2, C_3) of P either $C_1 \cap C_3 = \emptyset$ or $C_2 \subseteq (C_1 \cup C_3)$ [19, Cor.10].*

For algorithmic purposes it is important that the clique tree of a chordal graph can be computed using $O(n + m)$ time and space [15] where m denotes the number of edges and n the number of vertices in G.

3.1 The Algorithm

Let I be an instance of RAP with proper intervals and $P = (\mathcal{K}, \mathcal{E})$ the clique path of the corresponding proper interval graph G. Let $C_1, \ldots, C_\ell$ be the ℓ maximal cliques of G ordered such that C_i is adjacent to C_{i-1} and C_{i+1} in P for $i = 2, \ldots, \ell - 1$. For a subgraph $\bar{G}$ of G let $N(\bar{G})$ denote the neighborhood of $\bar{G}$.

Definition 9 *Two subgraphs C_1, C_2 of G are called* unconnected *if $N(C_1) \cap C_2 = \emptyset$ and $C_1 \cap N(C_2) = \emptyset$.*

Lemma 15 *Let K' be a set of cliques of G which are pairwise unconnected. Then the resource allocation problem restricted to the items contained in the cliques in K' and the constraints these cliques impose can be decomposed into $|K'|$ independent knapsack problems.*

Proof. Follows immediately from the definition of unconnectedness. □

In our algorithm we will construct two sets K_1, K_2 of cliques such that their union contains each vertex of G exactly once. The cliques in each of the two subsets will be pairwise unconnected. Solving the problem separately for each of these subsets and taking the better of the two solutions yields a $\frac{1}{2}-$approximation algorithm. In polynomial time we can only perform an $\varepsilon-$approximation scheme for every knapsack problems which decreases the performance ratio by ε.

Let $C_{jk} := C_j \backslash C_k$ (seen as in induced subgraph of G). The following algorithms gets as input the clique path representation $C_1, \ldots, C_\ell$ of an instance I.

```
Algorithm Clique Partitioning :
    K_1 := ∅; K_2 := ∅; C_{ℓ+1} := ∅
    c := 0; i := 1
    while i < ℓ
        c := c + 1
        if c is odd
            K_1 := K_1 ∪ {C_i}
        else
            K_2 := K_2 ∪ {C_i}
        find j > i with C_i ∩ C_j ≠ ∅  ∧  C_i ∩ C_{j+1} = ∅   (*)
        C_j := C_{ji}
        i := j
    end while
    if c is odd
        K_1 := K_1 ∪ {C_i}
    else
        K_2 := K_2 ∪ {C_i}
```

Lemma 16 *Every vertex of G is contained in exactly one clique of $K_1 \cup K_2$.*

Proof. Every vertex $v \in G$ is contained in at least one maximal clique. For all cliques C_k which are skipped in line (*) there is $C_i \cap C_k \neq \emptyset$ and $C_i \cap C_{k+1} \neq \emptyset$. It follows from Theorem 14 that $C_k \subseteq C_i \cup C_{k+1}$. Hence, no vertices are lost by skipping C_k. (For general interval graphs that are not proper, this would not be the case when applying this algorithm.)

After adding a clique C_i to some K_ℓ its items are removed from C_j. Since the clique tree is a path and $C_i \cap C_{j+1} = \emptyset$ the items of C_i can not appear in any other cliques of K_1 or K_2.

It is obvious from the construction of the algorithm that all elements of K_ℓ are cliques. □

Lemma 17 *The cliques in each set K_1, K_2 are pairwise unconnected.*

Proof. Consider an arbitrary clique $C' \in K_1$ and an arbitrary vertex $v \in C'$. Then we have to show that for all vertices $u \in C''$ for some $C'' \in K_1$, $C'' \neq C'$, there is no edge (v, u) in G.

For simplicity we assume that the cliques $C' = C_{ba}$, $C_{cb} \in K_2$, $C'' = C_{dc}$ are generated in this order. (If C'' is generated further away from C' the following argument holds a fortiori.)

Applying Lemma 13 for the corresponding maximal cliques we know that $C_c \cap C_d$ separates C_{cd} from $C_{dc} = C''$. If $v \in C_b \cap C_c$ and hence $v \in C_{cd}$ (clearly $C_b \cap C_d = \emptyset$ by construction of the algorithm and the path property), this means that v is separated from u. Otherwise there is $v \in C_{bc}$ and we assume that there is an edge $(v, u) \in E$. By the clique property every vertex in $C_b \cap C_c \subseteq C_{cd}$ (which is non-empty by construction) has an edge to v and thus via the edge (v, u) a path to $u \in C_{dc}$ in contradiction to the above separation property.

The same argument works for K_2. □

Theorem 18 *Let z^* denote the optimal solution value of I. Then there is a polynomial time algorithm with solution value z^A such that for proper interval graphs $z^A \geq (\frac{1}{2} - \varepsilon)\, z^*$ for every fixed $\varepsilon > 0$.*

Proof. After executing Clique Partitioning we solve the standard knapsack problems defined by each of the cliques in K_1 and K_2. By Lemma 16 every item appears in exactly one of these problems. By Lemma 15 and 17 taking the union of solutions over all cliques in K_ℓ yields optimal solutions z^*_ℓ, $\ell = 1, 2$, for the instance restricted to the subgraph induced by the vertices in the cliques of K_ℓ.

Since every feasible solution of I must fulfill the weight constraints for all cliques, we have $z^* \leq z^*_1 + z^*_2$. Hence, taking $z^A := \max\{z^*_1, z^*_2\}$ immediately yields a $\frac{1}{2}-$approximation algorithm.

To avoid the pseudopolynomial running time for solving the at most n occurring knapsack problems to optimality we apply an FPTAS (cf. [20]) instead and thus obtain a polynomial running time. Summing up the solutions of the FPTAS with performance guarantee $(1 - \delta)$ we get approximate solution values $z^A_\ell \geq (1 - \delta) z^*_\ell$.

Putting things together we have $z^* \leq \frac{z^A_1}{(1-\delta)} + \frac{z^A_2}{(1-\delta)}$ and again taking the maximum $(1 - \delta) z^* \leq 2 \max\{z^A_1, z^A_2\}$. Choosing $\delta \leq 2\,\varepsilon$ yields the statement of the theorem. □

Corollary 19 *There is a polynomial time algorithm with performance guarantee $\frac{1}{2}$ for the resource allocation problem with proper interval graphs and uniform profits.*

Proof. This result follows immediately since the separate knapsack subproblems with uniform profits can be solved to optimality by sorting the jobs in increasing order of weights. □

A simple example with only three jobs shows that the performance ratio of Corollary 19 is tight.

Example:

j	1	2	3
p_j	1	1	1
w_j	W	W	W
s_j	1	2	4
t_j	3	5	6

There are two maximal cliques $C_1 = \{1, 2\}$ and $C_2 = \{2, 3\}$. Algorithm Clique Partitioning yields $K_1 = \{C_1\}$ and $K_2 = \{C_{21}\}$. The approximation algorithm computes as solutions item 1 (or

item 2) for z_1^A and item 3 for z_2^A and outputs one of them as maximum. The optimal solution packs both items 1 and 3 which yields a $\frac{1}{2}-$approximation.

Replacing items 1 and 3 by an instance for the knapsack problem where the $(1-\varepsilon)-$ approximation of the FPTAS is tight it can be shown that also the bound of Theorem 18 is tight.

In practice the algorithm could also be modified in such a way that a maximum independent set of the path $(P_{11}\ P_{21}\ P_{12}\ldots)$ is calculated. The weight of the vertices of this path is derived from an appropriate knapsack algorithm for the subgraphs. Clearly this modification may improve the solution and is definitely not worse than the value of the approach described in the algorithm. However this modification cannot guarantee a better worst-case performance guarantee.

4 A Simple Greedy Algorithm

In this section we study the performance of a simple greedy algorithm for the resource allocation problem with proper intervals and uniform profits as considered in Section 2. Since all jobs have the same profit it is natural to sort the jobs in increasing order of weights and try to add them to the current solution in this order if feasibility is preserved. It will turn out that this straightforward algorithm Greedy and has a tight worst-case ratio of 1/2. This does not improve upon the result of Corollary 19. However, we believe that the analysis of such an intuitive approach, which also yields an improved running time, is interesting in its own right.

Algorithm Greedy:

Sort the jobs in increasing order of weights
$GS := \emptyset$ *set of jobs selected by Greedy*
for $i = 1$ to N do
  if $GS \cup \{i\}$ is feasible then *for every maximal clique*
    $GS := GS \cup \{i\}$
end for
$z^G := |GS|$ *z^G is the value of the greedy solution*

Let z^* denote the optimal solution of a resource allocation problem with proper intervals and profits $p_j = 1$ for $j = 1, \ldots, N$.

Theorem 20

$$z^G \geq \frac{1}{2}\, z^*$$

Proof. Let X^* be the optimal set of jobs, i.e. $z^* = |X^*|$ and J be the set of all jobs. The principle of the proof is very simple: We start with the set GS and remove iteratively jobs from $GS \setminus X^*$ and add jobs from $X^* \setminus GS$ until we have completely transformed GS into X^*. It will be shown that whenever we remove a job *at most* two jobs can be added, which suffices to show the claim. This transformation can be done by the following hypothetical algorithm to be executed after performing Greedy:

Proof Construction:

Eliminate all jobs $J \setminus (GS \cup X^*)$ from consideration
$S := GS$
while $S \neq X^*$ do
  $j_m := \arg\max\{w_j \mid j \in S \setminus X^*\}$
  $S := S \setminus \{j_m\}$
  for all $i \in X^* \setminus S$ in increasing order of weights do
    if $S \cup \{i\}$ is feasible then
      $S := S \cup \{i\}$
  end for
end while

The algorithm starts with $S = GS$ and terminates with $S = X^*$. If jobs are added to S in an iteration, the most recently removed job j_m will be called *split job*.

Claim 1: After removing j_m, only jobs with weight $\geq w_{j_m}$ can be added to S.

This claim follows from the greedy strategy of the algorithm: Assume that a job $j' \notin S$ with $w_{j'} < w_{j_m}$ can be added to S. Then job j' would have been packed also by Greedy before packing j_m since at this point the set of packed jobs was a strict subset of the current set S (note that jobs are removed from S in decreasing order of weight).

Claim: 2 After removing j_m, at most two jobs from $X^* \setminus S$ can be added.

Assume in contrary to the claim that three jobs $i_1, i_2, i_3 \in X^* \setminus S$ are added after removing j_m. Note that due to Claim 1 we have $w_{i_1} \geq w_{j_m}, \ldots, w_{i_3} \geq w_{j_m}$. W.l.o.g. we assume $s_{i_1} < s_{i_2} < s_{i_3}$.

Let

$$W_t := \sum_{\substack{j \in S \\ t \in I_j}} w_j$$

denote the weight of the jobs currently in S directly after the removal of j_m (i.e. before the possible addition of jobs) whose intervals contain t.

In the previous iteration of our hypothetical algorithm before removing j_m all these three jobs were tested for possible inclusion but must have failed the feasibility test. Hence, there must exist three points in time $q_\ell \in [s_{j_m}, t_{j_m}]$, $\ell = 1, 2, 3$, such that

$$W_{q_\ell} + w_{j_m} + w_{i_\ell} > W, \ \ell = 1, 2, 3. \tag{17}$$

Case 1: $s_{i_1} < s_{j_m}$

Case 1.1: $s_{i_2} < s_{j_m}$

Since $t_{i_2} > t_{i_1}$ there must be $q_1 \in I_{i_2} \cap I_{j_m}$ in this case. From (17) we get the violated feasibility constraint at time q_1 as $W_{q_1} + w_{i_1} + w_{i_2} > W$.

Case 1.2: $s_{i_2} > s_{j_m}$

Now we have $t_{i_2} > t_{j_m}$ and hence $q_3 \in I_{i_2} \cap I_{j_m}$. (17) yields the violated constraint $W_{q_3} + w_{i_2} + w_{i_3} > W$ at time t_3.

Case 2: $s_{i_1} > s_{j_m}$

There is $t_{i_1} > t_{j_m}$ and in analogy to Case 1.2 we have a violated constraint at time q_2 given by $W_{q_2} + w_{i_1} + w_{i_2} > W$.

Summarizing, we have shown that every removed split job allows the addition of only one or two jobs (namely in Case 1.2) not chosen by Greedy which completes the proof. □

The following example shows that the bound of Theorem 20 is tight.

Example: Consider the following instance with $N = 3k$ jobs where $\bar{\varepsilon}$ small enough (e.g., $\bar{\varepsilon} = \frac{1}{6(k+1)}$) and some $\varepsilon > 0$:

j	$1, \ldots, k$	$k+1, \ldots, 2k$	$2k+1, \ldots, 3k$
w_j	$\frac{W}{k}$	$\frac{W}{k} - \varepsilon$	$\frac{W}{k}$
s_j	$1 - (j\bar{\varepsilon})$	$2 - (j\bar{\varepsilon})$	$4 - (j\bar{\varepsilon})$
t_j	$3 + (j\bar{\varepsilon})$	$5 + (j\bar{\varepsilon})$	$6 + (j\bar{\varepsilon})$

Greedy selects $GS = \{k+1, \ldots, 2k\}$ with $z^G = k$ whereas the optimal solution consists of the complement with $z^* = 2k$.

It is easy to show that the 1/2−approximation ratio of Greedy requires both proper intervals and uniform profits. The following easy example shows that Greedy can perform arbitrarily bad for uniform profits and non-proper intervals.

Example: We are given $N = k + 1$ jobs:

	$j = 1$	$j = 2, \ldots, n + 1$
w_j	$W - 1$	W
s_j	1	$2j - 1$
t_j	$2n + 2$	$2j$

Greedy selects $GS = \{1\}$ while the optimal solution consists of $X^* = \{2, \ldots, k + 1\}$ and hence $z^* = k \cdot z^G$.

As can be expected Greedy can perform also arbitrarily bad for general profits even on proper intervals as shown by the following completely trivial example with only two jobs.

Example:

j	1	2
p_j	1	M
w_j	$W - 1$	W
s_j	1	2
t_j	3	4

Greedy selects job 1 with $z^G = 1$, while the optimal solution consists of job 2 with $z^* = M$.

The time complexity of Greedy is determined by $O(N \log N)$ for sorting and N feasibility checks. These can be done trivially in $O(N)$ time which yields a total running time of $O(N^2)$. However, a more involved representation of the current weight for each of the $2N$ starting and end points with a binary search tree permits a feasibility check in $O(\log N)$ time and can also be updated within the same time complexity. Thus an overall running time complexity of $O(N \log N)$ for Greedy can be obtained.

Each subtree of this search tree represents an interval containing all points in time from the leftmost to the rightmost leaf node. In each node we store the range of this interval of the emanating subtree, the weight of all selected jobs that cover the whole interval and the maximal *additional weight* for any point of the interval. The details of searching and updating this data structure require some more elaboration which is outside the scope of this paper.

Acknowledgments: We would like to thank Gaia Nicosia (Università Rome Tre) for introducing us to the resource allocation problem and pointing out several references.

References

[1] E.M. Arkin and E.B. Silverberg. Scheduling jobs with fixed start and end times. *Discrete Appl. Math.*, 18(1):1–8, 1987.

[2] N. Bansal, A. Chakrabarti, A. Epstein, and B. Schieber. A quasi-ptas for unsplittable flow on line graphs. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 721–729. ACM, 2006.

[3] N. Bansal, Z. Friggstad, R. Khandekar, and M.R. Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. In *SODA '09: Proceedings of the Twentieth Annual ACM - SIAM Symposium on Discrete Algorithms*, pages 702–709. SIAM, 2009.

[4] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48:1069–1090, 2001.

[5] M. Bartlett, A.M. Frisch, Y. Hamadi, I. Miguel, A. Tarim, and C. Unsworth. The temporal knapsack problem and its solution. In *CPAIOR Proceedings*, volume 3524 of *LNCS*, pages 34–48. Springer, 2005.

[6] J.R.S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In A. George, J.R. Gilbert, and J.H.U. Liu, editors, *Graph Theory and Sparse Matrix Computations*, pages 1–29. Springer, 1993.

[7] K.I. Bouzina and H. Emmons. Interval scheduling on identical machines. *Journal of Global Optimization*, 9:353–362, 1996.

[8] G. Calinescu, A. Chakrabarti, H. Karloff, and Y. Rabani. Improved approximation algorithms for resource allocation. In *9th International Integer Programming and Combinatorial Optimization Conference*, volume 2337 of *LNCS*, pages 401–414. Springer, 2002.

[9] A. Caprara, F. Furini, and E. Malaguti. Practical solution of the resource allocation problem, 2009. working paper, DEIS, University of Bologna.

[10] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47(1):53–78, 2007.

[11] B. Chen, R. Hassin, and M. Tzur. Allocation of bandwidth and storage. *IIE Transactions*, 34:501–507, 2002.

[12] R. Diestel. *Graph Theory*. Springer, 3 edition, 2006.

[13] C.W. Duin and E. van der Sluis. On the complexity of adjacent resource scheduling. *Journal of Scheduling*, 9(1):49–62, 2006.

[14] M. Fischetti, S. Martello, and P. Toth. The fixed job schedule problem with work-time constraints. *Operations Research*, 37:395–403, 1989.

[15] P. Galinier, M. Habib, and C. Paul. Chordal graphs and their clique graphs. In *WG '95: Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 1017 of *LNCS*, pages 358–371. Springer, 1995.

[16] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[17] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., 2004.

[18] N.G. Hall and M.J. Magazine. Maximizing the value of a space mission. *European Journal of Operational Research*, 78:224–241, 1994.

[19] L. Ibarra. The clique-separator graph for chordal graphs. *Discrete Appl. Math.*, 157(8):1737–1749, 2009.

[20] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[21] A.W.J. Kolen, J.K. Lenstra, C.H. Papadimitriou, and F.C.R. Spieksma. Interval scheduling: A survey. *Naval Research Logistics*, 54(5):530–543, 2007.

[22] S. Leonardi, A. Marchetti-Spaccamela, and A Vitaletti. Approximation algorithms for bandwidth and storage allocation problems under real time constraints. In *Foundations of Software Technology and Theoretical Computer Science*, volume 1974 of *LNCS*, pages 409–420. Springer, 2000.

[23] C.A. Phillips, R.N. Uma, and J. Wein. Off-line admission control for general scheduling problems. *Journal of Scheduling*, 3:365–381, 2000.

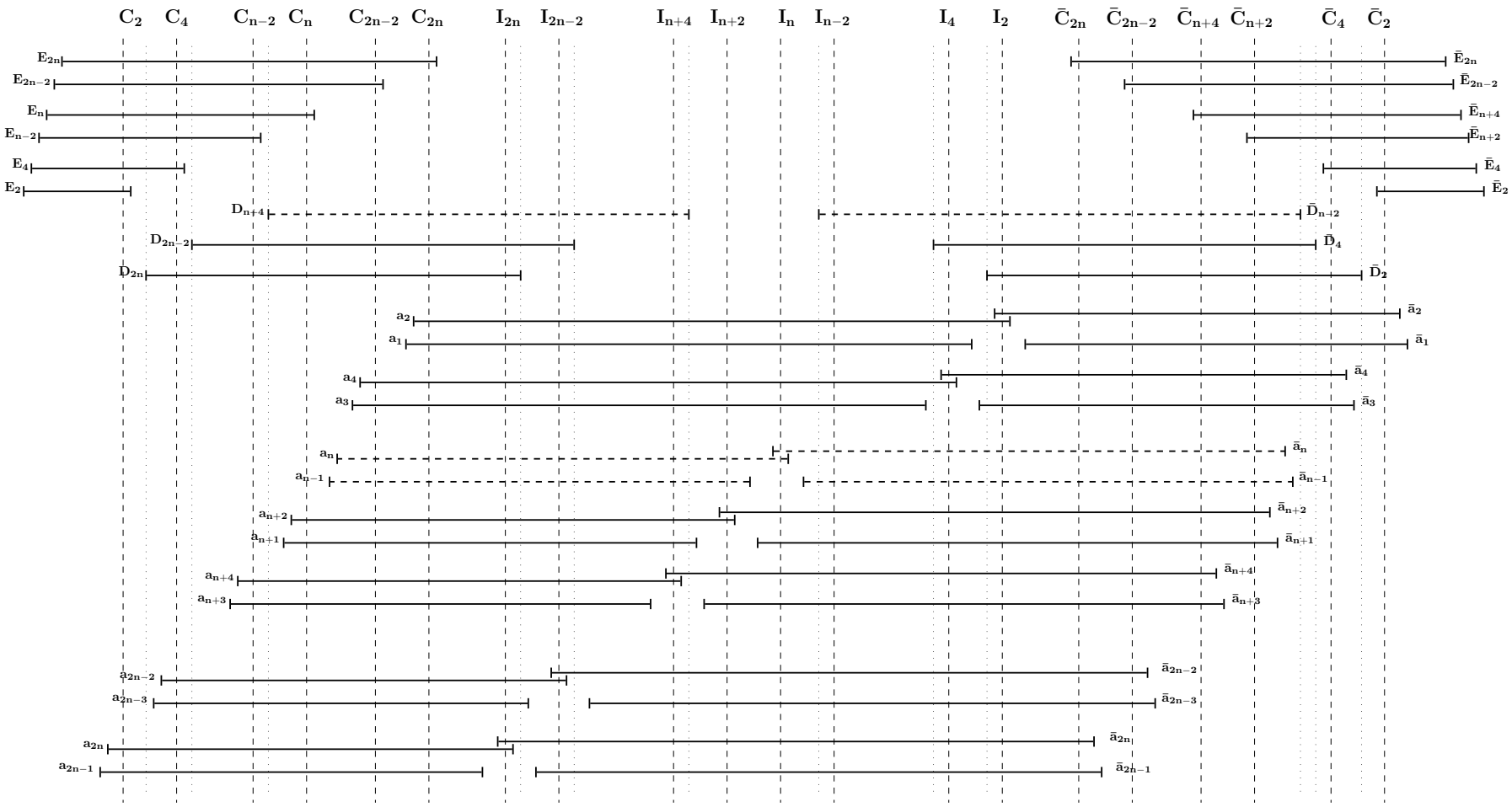

Figure 1: Interval graph and maximal cliques of instance R