A Supermodular Relaxation for Scheduling with Release Dates

Michel X. Goemans*

Dept. of Mathematics, Room 2-382, M.I.T., Cambridge, MA 02139.**

Abstract. We consider the scheduling problem of minimizing a weighted sum of completion times under release dates. We present a relaxation which is a supermodular polyhedron. We show that this relaxation is precisely the projection of a time-indexed relaxation introduced by Dyer and Wolsey.

1 Introduction

We consider the 1-machine scheduling problem with release dates and in which the objective is to minimize a weighted sum of completion times. In the classical scheduling notation, this corresponds to $1|r_j|\sum_j w_j C_j$.

We use the following notation. Job j has a processing time p_j, and a release date r_j. Its completion time in a schedule is given by C_j. Let $N = \{1, \cdots, n\}$ denote the set of jobs to be processed. For any subset S of jobs, let $p(S)$ denote the sum of the processing times of jobs in S, let $p^2(S)$ (not to be confused with $p(S)^2 = p(S) * p(S)$) denote $\sum_{j\in S} p_j^2$, and let $r(S) = \min_{j\in S} r_j$.

In this paper, we consider the scheduling problem $1|r_j|\sum_j w_j C_j$ from a polyhedral point-of-view. For a survey of the study of scheduling polyhedra, we refer the reader to Queyranne and Schulz [10]. Let P denote the convex hull of the set of feasible completion times over all possible schedules, i.e. $P = conv\{(C_1, \cdots, C_n)$: there exists a feasible schedule with completion time C_j for job $j\}$. When all jobs are available at time 0, i.e. $r_j = 0$ for all j, Queyranne [9] has completely characterized P in terms of linear inequalities:

$$P = \left\{ C : \sum_{j\in S} p_j C_j \geq h(S) \text{ for all } S \subseteq N \right\},$$

where $h(S) = \frac{1}{2}(p^2(S) + p(S)^2)$. Moreover, Queyranne has shown that $h(S)$ is supermodular, meaning that $h(S) + h(T) \leq h(S \cup T) + h(S \cap T)$ for all S and T. This implies that P is a supermodular polyhedron and that the greedy algorithm can be used to optimize over P when there are no release dates. For a discussion

* Part of this work was supported by NSF contract 9302476-CCR, a Sloan Foundation Fellowship, and ARPA Contract N00014-95-1-1246.

** Email:goemans@math.mit.edu

of supermodularity and related topics, see Lovász [5], Fujishige [3], or Nemhauser and Wolsey [6].

When there are release dates, one possible way to strengthen the supermodular constraints is to "shift" the entire schedule by the smallest release date in the set S. This implies that the following polyhedron is a relaxation of P

$$Q = \{C : \sum_{j \in S} p_j C_j \geq l(S) \text{ for all } S \subseteq N\},$$

where $l(S) = p(S)r(S) + \frac{1}{2}(p^2(S) + p(S)^2)$. However, it is easy to see that $l(S)$ is not necessarily supermodular. We may, however, construct an equivalent formulation of Q by replacing $l(S)$ by

$$g(S) = \max_{\{\text{Partitions } S_1,\cdots,S_k \text{ of } S\}} \sum_{i=1}^{k} l(S_i).$$

The function $g(S)$ is called the upper Dilworth truncation of $l(S)$ (see Lovász [5]). One of the results of this paper is to show that $g(S)$ is supermodular and therefore Q defines a supermodular polyhedron. This means that one can optimize over P by using the greedy algorithm, and we show how the greedy algorithm can be implemented in $O(n \log n)$ time. A very similar situation appears in [11]. We also give necessary and sufficient conditions under which this supermodular inequality defines a facet of P.

There are typically three types of scheduling formulations (see [10]); the first ones use the completion times as variables, the second ones have linear ordering variables indicating whether job j precedes job k in the schedule, and the third ones are time-indexed and have one variable for each job and each unit of time. Dyer and Wolsey [2] present several types of relaxations for $1|r_j|\sum_j w_j C_j$, the strongest ones being time-indexed. We actually show that one of their time-indexed relaxations is equivalent to Q; by projecting the time-indexed variables, one obtains Q. The way we derive time-indexed variables out of Q is quite interesting and may be useful for other types of scheduling problems. We should point out that the equivalence between these two relaxations gives another proof of the supermodularity of $g(S)$.

This research was actually performed back in 1989 and was motivated by the preprints of the papers of Queyranne [9] and Dyer and Wolsey [2]. Very recently, Hall, Shmoys and Wein [4] used time-indexed formulations to derive the first approximation algorithms for several scheduling problems with a weighted sum of completion times as objective. Subsequently, Schulz [12] also showed that approximation algorithms for these problems can be obtained from relaxations based solely on completion times. This motivated us to write these results on the relationship between time-indexed relaxations and completion times relaxations for $1|r_j|\sum_j w_j C_j$.

2 Supermodular relaxation

Instead of computing the upper Dilworth truncation of $l(S)$, we will directly introduce the function $g(S)$ and then its relationship with $l(S)$ will become clear later on.

Let S be a subset of jobs. Consider the schedule in which we first schedule all jobs in S in order of non-decreasing release dates (ties are broken arbitrarily) and as early as possible, and then schedule all other jobs in any order. We refer to this schedule as the S-schedule (it is only well specified for the jobs in S, but this will not cause any problem). Some of the jobs in S will be scheduled at their release dates in the S-schedule, and the others will start strictly after their release dates. This leads to a partition of S into $S_1, \cdots, S_k$, such that (i) exactly one job in each S_i is scheduled at its release date, and (ii) all jobs in S_i are scheduled before the jobs in S_{i+1}. We refer to $S_1, \cdots, S_k$ as the *canonical decomposition* of S. Observe that the canonical decomposition does not depend on how ties are broken in the definition of the S-schedule when there are several jobs with the same release date. We will refer to the S-schedule corresponding to the canonical decomposition of S by a superscript ${}^{(S)}$.

For any subset S of jobs, let

$$g(S) = \sum_{j \in S} p_j C_j^{(S)}.$$

An expression of $g(S)$ in terms of its canonical decomposition will be given later. This will relate the function g to the function l discussed in the introduction.

Theorem 1. *For any S, the inequality*

$$\sum_{j \in S} p_j C_j \geq g(S) \tag{1}$$

is a valid inequality for P.

Proof. We need to prove that $z(C) = \sum_{j \in S} p_j C_j$ is minimized by the S-schedule. Consider any schedule and assume that the start time of some job in S can be decreased slightly without violating feasibility. Then $z(C)$ decreases, and therefore we can assume that all jobs in S are scheduled as early as possible. Furthermore, we can assume that all jobs not in S are scheduled after all the jobs in S, and therefore do not interfere with the jobs in S.

Now, if the jobs in S are not scheduled according to non-decreasing release dates then there exist two consecutive jobs j_1 and j_2 such that $C_{j_1} < C_{j_2}$ and $r_{j_1} > r_{j_2}$. By interchanging j_1 and j_2, we obtain a new feasible schedule with $C'_{j_2} = C_{j_2} - p_{j_1}$ and $C'_{j_1} = C_{j_1} + p_{j_2}$. Observe that $p_{j_1}C_{j_1} + p_{j_2}C_{j_2} = p_{j_1}C'_{j_1} + p_{j_2}C'_{j_2}$, and therefore $z(C) = z(C')$. Repeating this argument and the first part of the proof, we derive that the S-schedule must minimize $g(S)$.

Our next goal is to show that $g(S)$ is supermodular, but this will require some preliminaries. First, for any schedule, we define the indicator function $I_S(t)$ to be 1 at time t if some job in S is being processed at time t, and 0 otherwise. To avoid any confusion, if a job of S is being processed up to time t and no other job of S is started at time t, we let $I_S(t)$ to be 0. For simplicity, we denote $I_{\{j\}}(t)$ by $I_j(t)$. The indicator function will be very useful not only to prove supermodularity of $g(S)$ but also to relate our formulation to a time-indexed relaxation introduced by Dyer and Wolsey [2] (which in turn can be used to prove the supermodularity of $g(S)$ as well).

Here are two elementary properties of indicator functions.

Lemma 2. *1. If $S_1, \cdots, S_k$ is a partition of S then $I_S(t) = \sum_{i=1}^k I_{S_i}(t)$. In particular, $I_S(t) = \sum_{j\in S} I_j(t)$.*

2. For any S, $\int_0^\infty I_S(t)dt = p(S)$.

Proof. 1. is obvious. 2. follows from the fact that the total processing time of jobs in S is $p(S)$.

The following lemma will be very useful for the analysis.

Lemma 3. *For any schedule and any set S,*

$$\sum_{j\in S} p_j C_j = \frac{1}{2}p^2(S) + \int_0^\infty t I_S(t)dt = \frac{1}{2}p^2(S) + \int_0^\infty \left[p(S) - \int_0^\tau I_S(t)dt\right] d\tau.$$

Note that the value $\sum_{j\in S} p_j C_j$ depends only on when the machine is processing a job of S, but does not depend on which specific job is being processed.

Proof. We first claim that, for any j,

$$p_j C_j = \frac{p_j^2}{2} + \int_0^\infty t I_j(t)dt.$$

If we then sum over all $j \in S$, we obtain the first equality by using Lemma 2, part 1.

To prove the claim, it suffices to observe that $I_j(t)$ is equal to 1 on the interval $[C_j - p_j, C_j)$ and 0 otherwise. As a result, the integral on the right-hand-side is equal to

$$\int_{C_j-p_j}^{C_j} t dt = \frac{C_j^2}{2} - \frac{(C_j-p_j)^2}{2} = p_j C_j - \frac{p_j^2}{2},$$

and the claim follows.

The second equality follows from simple calculus:

$$\begin{aligned}\int_0^\infty t I_S(t)dt &= \int_0^\infty \int_0^t I_S(t) d\tau dt \\ &= \int_0^\infty \int_\tau^\infty I_S(t) dt d\tau \\ &= \int_0^\infty \left[p(S) - \int_0^\tau I_S(t)dt\right] d\tau,\end{aligned}$$

using Lemma 2, part 2.

To prove the supermodularity of $g(S)$, we will only be dealing with the indicator functions corresponding to S-schedules for all S. Let $J_S(t)$ denote $I_S^{(S)}(t)$, namely the indicator function for the set S associated with the S-schedule. We refer to $J_S(t)$ as the canonical indicator function. Although this is not needed to prove supermodularity, we first give an expression for $g(S)$ in terms of its canonical decomposition.

Lemma 4. *Let $S_1, \cdots, S_k$ be the canonical decomposition of S. Then*

$$g(S) = \frac{1}{2}p^2(S) + \sum_{i=1}^{k} p(S_i)\left(r(S_i) + \frac{1}{2}p(S_i)\right) = \sum_{i=1}^{k} l(S_i).$$

Proof. By definition, $g(S) = \sum_{j\in S} p_j C_j^{(S)}$. For $j \in S_i$, the definition of the canonical decomposition implies that $C_j^{(S)} = C_j^{(S_i)}$, and thus

$$g(S) = \sum_{i=1}^{k} \sum_{j\in S_i} p_j C_j^{(S_i)}.$$

By Lemma 3,

$$\sum_{j\in S_i} p_j C_j^{(S_i)} = \frac{1}{2}p^2(S_i) + \int_0^\infty \left[p(S_i) - \int_0^\tau J_{S_i}(t)dt\right] d\tau.$$

But $J_{S_i}(t)$ is 1 if $t \in [r(S_i), r(S_i) + p(S_i))$ and 0 otherwise. Therefore,

$$\int_0^\infty \left[p(S_i) - \int_0^\tau J_{S_i}(t)dt\right] d\tau = p(S_i)\left[r(S_i) + \frac{p(S_i)}{2}\right],$$

which proves the result.

This implies that g is the upper Dilworth truncation of l (since there is a schedule achieving $g(S)$).

We now give some properties of the canonical indicator function.

Lemma 5 (Monotonicity). *For $S \subseteq T$, and for any t, $J_S(t) \leq J_T(t)$.*

Proof. Let $S_1, \cdots, S_k$ and $T_1, \cdots, T_l$ be the canonical decompositions of S and T respectively. In the S-schedule, the machine is busy from $r(S_i)$ to $r(S_i) + p(S_i)$ for every i. In the T-schedule, independently of whether a job is being processed right before $r(S_i)$, the machine must be continuously busy at least between $r(S_i)$ and $r(S_i) + p(S_i)$ since all the jobs in S_i were released on or after $r(S_i)$. In other words, whenever the S-schedule is processing a job in S, the T-schedule must be processing a job in T, proving the claim.

Lemma 6. *For $j \notin S$,*

$$\int_0^t \left[J_{S\cup\{j\}}(\tau) - J_S(\tau)\right] d\tau = \min\left(p_j, \int_{\min(r_j,t)}^t (1 - J_S(\tau))\, d\tau\right).$$

Proof. If $r_j > t$ then both sides are 0. So assume that $r_j \leq t$. The integral on the right-hand-side represents the amount of idle time between r_j and t in the S-schedule. In the $(S \cup \{j\})$-schedule, these periods of idle time will be used to process some job in $S \cup \{j\}$, up to p_j units. This proves the result.

A function f is submodular if for all S and T, $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. So, f is submodular iff $-f$ is supermodular. An alternate definition for submodularity (see, for example, [6]) is that for any $T \subset S$ and for any $j \notin T$, we have that

$$f(S \cup \{j\}) - f(S) \leq f(T \cup \{j\}) - f(T).$$

Let $f_S(t) = \displaystyle\int_0^t J_S(\tau) d\tau$.

Lemma 7. *For any t, $f_S(t)$ is submodular.*

Proof. By Lemma 6, for any $j \notin S$, we have that

$$f_{S\cup\{j\}}(t) - f_S(t) = \min\left(p_j, \int_{\min(r_j,t)}^t (1 - J_S(\tau) d\tau\right).$$

The same expression with S replaced by T can also be written.

Using Lemma 5, it then follows that

$$f_{S\cup\{j\}}(t) - f_S(t) \leq f_{T\cup\{j\}}(t) - f_T(t),$$

since the integrand on the right-hand-side is no greater for S than for T. This shows submodularity.

We are now ready to prove:

Theorem 8. *The function $g(S)$ is supermodular.*

Proof. From Lemma 3, $g(S)$ can be rewritten as:

$$g(S) = \frac{1}{2}p^2(S) + \int_0^\infty [p(S) - f_S(t)]\, d\tau.$$

Using the fact that $p(S)$ and $p^2(S)$ are modular (i.e. both supermodular and submodular) and the submodularity of $f_S(t)$ proved in Lemma 7, we derive the supermodularity of g.

We now consider the following linear program:

$$Z_R = \text{Min} \sum_{j \in N} w_j C_j$$

subject to:

$$(R) \qquad \sum_{j \in S} p_j C_j \geq g(S) \qquad S \subseteq N.$$

Because of Theorem 1, (R) is a valid relaxation for the scheduling problem $1|r_j|\sum_j w_j C_j$ and therefore Z_R is a lower bound on the optimum value Z^*. If we make a change of variables and replace $p_j C_j$ by x_j for all j, we can rewrite (R) as

$$Z_R = \text{Min} \sum_{j \in N} \frac{w_j}{p_j} x_j$$

subject to:

$$(R') \qquad \sum_{j \in S} x_j \geq g(S) \qquad S \subseteq N.$$

Because of Theorem 8, this is a *supermodular polyhedron* (see [3]). This implies that its optimum solution can be computed by the greedy algorithm. More precisely, first order the jobs in non-increasing order of w_j/p_j, i.e. assume that $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq \cdots \geq \frac{w_n}{p_n}$. Let $N_j = \{1, 2 \ldots, j\}$. Then, let $\tilde{x}_j = g(N_j) - g(N_{j-1})$ for $j = 1, \cdots, n$, where $g(\emptyset) = 0$. Then $\tilde{x}$ is an optimum solution for (R'). As a result, an optimum solution to (R) is given by $\tilde{C}_j = \frac{1}{p_j}(g(N_j) - g(N_{j-1}))$ for $j = 1, \cdots, n$, assuming that $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq \cdots \geq \frac{w_n}{p_n}$.

Computing $\tilde{C}_j$ for all j can be done in $O(n \log n)$ time, although we will need some data structures for this purpose. First, we can sort the $\frac{w_j}{p_j}$ in $O(n \log n)$ time. Then we need to be able to compute $g(N_j)$ efficiently for all j. A naive implementation would take $O(n^2)$ time. Here is a brief sketch of how to implement this computation in $O(n \log n)$ time. For a given j, we keep track of the disjoint intervals (corresponding to the canonical decomposition of N_j) in which $J_{N_j}(t) = 1$. For this purpose, we use a balanced binary search tree (such as a red-black tree, see [1]). When considering $j+1$, we start by finding the first interval I whose right endpoint is greater than r_{j+1}. This requires one SEARCH operation (for the terminology see [1]). The addition of job $j+1$ will either create a new interval (if $r_{j+1} + p_{j+1}$ is no greater than the left endpoint of I) or will cause several consecutive intervals, say q, to merge (because we need p_{j+1} units of idle time to accommodate job $j+1$). This can be determined by performing at most q SUCCESSOR operations, and then performing either one INSERT operation or $q-1$ DELETION operations. This gives $O(q)$ operations, each taking $O(\log n)$ time. However, the number of intervals decreases by $q-1$, and therefore this can be charged to the time when these intervals were created. This shows that maintaining the intervals over all j takes $O(n \log n)$ time. As we update these intervals, it is very easy to compute $g(N_{j+1}) - g(N_j)$ in $O(q)$ time.

3 Equivalence with a relaxation of Dyer and Wolsey

In [2], Dyer and Wolsey introduce 5 different relaxations of $1|r_j|\sum_j w_jC_j$ that they denote by $(A), (B), (C), (D)$ and (E). They show that their values satisfy $Z_A \le Z_B \le Z_C \le Z_D \le Z_E$, i.e. E is the strongest one. Relaxation (D) is a time-indexed formulation using two types of variables: $y_{j\tau} = 1$ if job j is being processed at time τ, and t_j represents the start time of job j. For simplicity, we add p_j to t_j and replace the resulting expression by C_j; this gives an equivalent relaxation.

$$
\begin{array}{llll}
 & Z_D = \text{Min} \sum_j w_j C_j & & \\
 & \quad \text{subject to:} & & \\
(D) & \sum_j y_{j\tau} \le 1 & \tau = 0, 1, \cdots, T & \\
 & \sum_\tau y_{j\tau} = p_j & j \in N & (2) \\
 & C_j = \frac{p_j}{2} + \frac{1}{p_j} \sum_\tau \left(\tau + \frac{1}{2}\right) y_{j\tau} & j \in N & \\
 & 0 \le y_{j\tau} & j \in N, \tau = r_j, r_j+1, \cdots, T, &
\end{array}
$$

where T is an upper bound on the makespan of any schedule. In this relaxation, the release dates are assumed to be integral, and we will keep this assumption for the rest of this paper. Because of the equalities (2), the expression for C_j can take many different equivalent forms; we selected the easiest for the forthcoming analysis. Observe that the number of variables of this formulation is only pseudo-polynomial.

Theorem 9. $Z_D = Z_R$.

Proof. The most interesting part of the proof is to show that $Z_D \le Z_R$. Consider the optimal solution $\tilde{C}$ given by the greedy algorithm for (R). We will construct variables $y_{j\tau}$ such that $\tilde{C}$ and y are feasible in (D), showing that $Z_D \le Z_R$.

Assume that $\frac{w_1}{p_1} \ge \frac{w_2}{p_2} \ge \cdots \ge \frac{w_n}{p_n}$. Define $y_{j\tau} = J_{N_j}(\tau) - J_{N_{j-1}}(\tau)$ for every job j and every integral τ. We claim that $\tilde{C}$ and y are feasible in (D). First, $y_{j\tau} \ge 0$ because of Lemma 5. Moreover, $y_{j\tau}$ as defined will be 0 if $\tau < r_j$ or if $\tau \ge C_j^{(N_j)}$ and thus certainly if $\tau \ge T$. Equality (2) follows from Lemma 2 part 2, and the fact that $J_S(\tau)$ is constant over $[t, t+1)$ for any integer t:

$$
\begin{aligned}
\sum_\tau y_{j\tau} &= \sum_\tau \left(J_{N_j}(\tau) - J_{N_{j-1}}(\tau)\right) \\
&= \int_0^\infty \left(J_{N_j}(\tau) - J_{N_{j-1}}(\tau)\right) d\tau = p(N_j) - p(N_{j-1}) = p_j.
\end{aligned}
$$

Furthermore, we have that

$$\sum_j y_{j\tau} = \sum_{j=1}^{n} \left(J_{N_j}(\tau) - J_{N_{j-1}}(\tau)\right) = J_N(\tau) \leq 1,$$

for all τ. Finally, we need to verify the expression for $\tilde{C}_j$:

$$\begin{aligned}
\frac{p_j}{2} + \frac{1}{p_j}\sum_\tau (\tau + \frac{1}{2}) y_{j\tau} &= \frac{p_j}{2} + \frac{1}{p_j}\sum_\tau \int_\tau^{\tau+1} t\left(J_{N_j}(t) - J_{N_{j-1}}(t)\right) dt \\
&= \frac{p_j}{2} + \frac{1}{p_j}\int_0^\infty t\left(J_{N_j}(t) - J_{N_{j-1}}(t)\right) dt \\
&= \frac{p_j}{2} + \frac{1}{p_j}\left(g(N_j) - \frac{1}{2}p^2(N_j) - g(N_{j-1}) + \frac{1}{2}p^2(N_{j-1})\right) \\
&= \frac{1}{p_j}\left(g(N_j) - g(N_{j-1})\right) \\
&= \tilde{C}_j,
\end{aligned}$$

the third equality following from Lemma 3.

The second part of the proof is to show that $Z_D \geq Z_R$. For this purpose, we show that if y and C are feasible in (D) then C is also feasible in (R). Consider any set S. Then,

$$\begin{aligned}
\sum_{j\in S} p_j C_j &= \sum_{j \in S} p_j \left[\frac{p_j}{2} + \frac{1}{p_j}\sum_\tau \left(\tau + \frac{1}{2}\right) y_{j\tau}\right] \\
&= \frac{p^2(S)}{2} + \sum_{j\in S}\sum_\tau \left(\tau + \frac{1}{2}\right) y_{j\tau} \\
&= \frac{p^2(S)}{2} + \sum_\tau \left(\tau + \frac{1}{2}\right) \sum_{j\in S} y_{j\tau}.
\end{aligned}$$

Now, to y, we can easily associate a preemptive schedule: between τ and $\tau + 1$, $y_{j\tau}$ units of time are used to process job j. Because of the constraints on y in (D), every job is processed for a total of p_j units of time and the machine never works on two jobs at the same time. Let I' be the indicator function for this preemptive schedule. Then the above expression can be rewritten as:

$$\sum_{j\in S} p_j C_j = \frac{p^2(S)}{2} + \int_0^\infty t I'_S(t) dt.$$

But, the integral only depends on the times at which the preemptive schedule is busy processing some job in S, but not on which job is being processed at any such time. As a result, the expression on the right is minimized by a non-preemptive schedule. But, over the non-preemptive schedules, the expression is minimized for the S-schedule (see Lemma 3 and Theorem 1). This shows that $\sum_{j\in S} p_j C_j \geq g(S)$.

The proof actually shows a stronger result. When T is ∞, the projection of relaxation (D) onto the space of the C_j's is precisely (R).

From the proof, we can also derive a stronger result for the problem $1|r_j, p_j = 1|\sum_j w_j C_j$ with unit processing times. Indeed, the preemptive schedule derived in the first part of the proof is not preemptive if the processing times are all equal to 1. This shows that the value Z_R is precisely equal to the optimum value in the case of unit processing times, and that the feasible region defined by (R) is precisely equal to P. This was already known; see Queyranne and Schulz [11].

We should point out that Dyer and Wolsey [2] indicate that, after elimination of the variables C_j, (D) becomes a transportation problem. This has several consequences. First, we could have restricted our attention to *extreme* points of this transportation problem in the second part of the proof above, which have the property that $y_{j\tau} \in \{0,1\}$. This would have avoided dealing with fractional y's.

Also, as indicated in [2], the relaxation (D) can be solved in $O(n \log n)$ time by exploiting the special cost structure of this transportation problem. Dyer and Wolsey refer the reader to Posner [8] for details. This gives the same time bound as the one we obtained for (R).

Finally, the fact that (D) becomes a transportation problem after elimination of the C_j's can be used to give an alternate proof of the supermodularity of g. Clearly,

$$g(S) = \min \left\{ \sum_{j\in S} p_j C_j \text{ subject to } \sum_{j \in T} p_j C_j \geq g(T) \text{ for all } T \subseteq N \right\}.$$

By Theorem 9, $g(S)$ can then be expressed by the following transportation problem:

$$\begin{aligned} g(S) \;&= \frac{p^2(S)}{2} + \text{Min} \sum_{j\in S} \sum_{\tau} \left(\tau + \frac{1}{2}\right) y_{j\tau} \\ &\text{subject to:} \\ &\qquad \sum_j y_{j\tau} \leq 1 \qquad\qquad \tau = 0, 1, \cdots, T \\ &\qquad \sum_\tau y_{j\tau} = p_j \qquad\qquad j \in S \\ &\qquad 0 \leq y_{j\tau} \qquad\qquad j \in S, \tau = r_j, r_j + 1, \cdots, T. \end{aligned}$$

From an interpretation by Nemhauser et al. [7] of a result of Shapley [13], it then follows that the value of the above transportation problem as a function of S is supermodular.

This also shows that the value of Z_R (or Z_D) as a function of the set S of jobs is supermodular. But this follows from the fact that Z_R is obtained by optimizing over a supermodular polyhedron.

4 Additional results

The separation problem for the inequalities (1) can be solved efficiently. Indeed, we only need to separate over $\sum_{j\in S} p_j C_j \geq l(S)$, and this can be done by trying all n possible values for $r(S)$ and then applying a separation routine of Queyranne [9] for the problem without release dates. The overall separation routine can be implemented in $O(n^2)$ time.

We can also give necessary and sufficient conditions under which the inequalities (1) define facets of P. We should point out that the condition that all the release dates in S should be identical is not necessary as claimed in [11]. We say that S is inseparable if it is its own canonical decomposition. Certainly, the condition that S is inseparable is necessary for the inequality (1) to define a facet. For a given inseparable set S, we say that a schedule is S-tight if $I_S(t) = 1$ for all $t \in [r(S), r(S) + p(S))$. The proof of the following theorem is omitted.

Theorem 10. *Let $S = \{1, \cdots, i\}$ be an inseparable set with $r_1 \leq r_2 \leq \cdots \leq r_i$. Then $\sum_{j\in S} p_j C_j \geq g(S)$ defines a facet of P if and only if, for every $j \in \{2, \cdots, i\}$, there exists an S-tight schedule for which j precedes a job k with $k < j$.*

Proof. We start with the "only if" part. If, for some j, there is no S-tight schedule for which j precedes some job k with $k < j$ then we claim that every S-tight schedule starts with the jobs $\{1, 2, \cdots, j-1\}$ (in any order). Indeed, assume there is a schedule in which a job $l > j-1$ precedes a job $k \leq j-1$. Then if we schedule j just in front of l and continue the schedule as before, we obtain a feasible schedule (since $r_l \geq r_j$). This gives a S-schedule violating the hypothesis (since j precedes k). Hence, every S-tight schedule is also $\{1, \cdots, j-1\}$-tight and therefore the set of feasible schedules satisfying (1) at equality are included in the face defined by the set $\{1, \cdots, j-1\}$.

For the "if" part, assume that any feasible schedule satisfying (1) at equality also satisfy $\sum_l a_l C_l = b$. For every j, consider an S-tight schedule in which j precedes k with $k < j$. If, in any schedule, we interchange two consecutive jobs such that the release date of the first one is at least the release date of the second one, then the schedule remains feasible. This means that, by the interchange of consecutive jobs (and possibly redefining k), we can assume that in our schedule j immediately precedes k. Now, if we interchange j and k, C_j increases by p_k and C_k decreases by p_j. But the equality $\sum_i a_i C_i = b$ must still be satisfied, implying that $a_k p_j = a_j p_k$, or $\frac{a_j}{p_j} = \frac{a_k}{p_k}$. Since, for every $2 \leq j \leq i$, we can find such a k with $k < j$, this proves that $\frac{a_1}{p_1} = \frac{a_2}{p_2} = \cdots = \frac{a_i}{p_i}$. Moreover, any job $j > i$ can be delayed arbitrarily, implying that $a_j = 0$ for $j > i$. This therefore shows that the equality must be $\sum_{j\in S} p_j C_j = g(S)$.

For example, the condition of the theorem is satisfied if $r_1 + p(S_{j-2}) \geq r_j$ for $j = 2, \cdots, i-1$.

The results in this paper can also be translated to the problem with deadlines instead of release dates by simply inverting the time. For the problem

$1|d_j|\sum w_jC_j$, we can therefore introduce valid inequalities of the form $\sum_{j\in S} p_jC_j \leq k(S)$ for all S. If we now have simultaneously release dates and deadlines, we can consider the relaxation defined by $k(S) \geq \sum_{j\in S} p_jC_j \geq g(S)$ for all S. However, in the case of unit processing times $(1|r_j, d_j, p_j = 1|\sum w_jC_j)$, these inequalities are not sufficient to describe the convex hull of feasible completion times. This was also observed in [11]. The problem however is simply an assignment problem and can therefore be solved efficiently. Moreover, from Hall's theorem, we can easily derive that this scheduling problem is feasible if and only if $k(S) \geq g(S)$ for all S.

Acknowledgments

The author would like to thank Andreas Schulz for stimulating discussions and for convincing him to write these results. The results presented here resulted from discussions many years ago with Leslie Hall and Tim Magee on the papers of Maurice Queyranne and of Martin Dyer and Laurence Wolsey.

References

1. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to algorithms*, McGraw Hill (1990).
2. M.E. Dyer and L.A. Wolsey, "Formulating the single machine sequencing problem with release dates as a mixed integer program", *Discrete Applied Mathematics*, **26**, 255–270 (1990).
3. S. Fujishige, "Submodular functions and optimization", *Annals of Discrete Mathematics*, **47**, North-Holland (1991).
4. L.A. Hall , D.B. Shmoys and J. Wein, "Scheduling to minimize average completion time: Off-line and on-line algorithms", *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, 142–151, 1996.
5. L. Lovász, "Submodular functions and convexity", in: A. Bachem, M. Grötschel and B. Korte (eds.), *Mathematical Programming: The State of the Art, Bonn, 1982*, Springer, Berlin, 235–257, 1983.
6. G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York (1988).
7. G.L. Nemhauser, L.A. Wolsey and M.L. Fisher, "An analysis of approximations for maximizing submodular set functions — I", *Mathematical Programing*, **14**, 265–294 (1978).
8. M.E. Posner, "A sequencing problem with release dates and clustered jobs", *Management Science*, **32**, 731–738 (1986).
9. M. Queyranne, "Structure of a simple scheduling polyhedron", *Mathematical Programming*, **58**, 263–285 (1993).
10. M. Queyranne and A. S. Schulz, "Polyhedral approaches to machine scheduling", Preprint 408/1994, Department of Mathematics, Technical University of Berlin, Berlin, Germany, 1994.
11. M. Queyranne and A. S. Schulz, "Scheduling units jobs with compatible release dates on parallel machines with nonstationary speeds", Proceedings of the 4th Integer Programming and Combinatorial Optimization Conference, E. Balas and

J. Clausen, eds., Lecture Notes in Computer Science, **920**, Springer-Verlag, 307–320 (1995).
12. A.S. Schulz, "Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds", these proceedings, 1996.
13. L.S. Shapley, "Complements and substitutes in the optimal assignment problem", *Naval Research Logistics Quaterly*, **9**, 45–48 (1962).