TWO-DIMENSIONAL GANTT CHARTS AND A SCHEDULING ALGORITHM OF LAWLER

MICHEL X. GOEMANS* AND DAVID P. WILLIAMSON†

Abstract. In this note we give an alternate proof that a scheduling algorithm of Lawler [4, 5] finds the optimal solution for the scheduling problem $1|prec|\sum_j w_jC_j$ when the precedence constraints are series-parallel. We do this by using a linear programming formulation of $1|prec|\sum_j w_jC_j$ introduced by Queyranne and Wang [11]. Queyranne and Wang proved that their formulation completely describes the scheduling polyhedron in the case of series-parallel constraints; a by-product of our proof of correctness of Lawler's algorithm is an alternate proof of this fact. In the course of our proof it is helpful to use what might be called *two-dimensional Gantt charts*. We think these may find independent use, and to illustrate this we show that some recent work in the area becomes transparent using 2D Gantt charts.

Key words. scheduling, series-parallel, linear programming

AMS subject classifications. 90B35, 90C27, 68Q25, 90C05

1. Introduction.

We consider the problem of scheduling n jobs on a single machine. Each job j must be scheduled for p_j units of time, and only one job can be scheduled at any point in time. We only consider *non-preemptive* schedules, in which all p_j units of job j must be scheduled consecutively. Furthermore, the schedule must obey specified *precedence constraints*. Precedence constraints are given by a partial order on the jobs; if i precedes j in the partial order (denoted $i \to j$), then i must be completely processed before j can begin its processing. A (possibly negative) weight w_j is associated with job j, and our goal is to find a schedule which minimizes the sum $\sum_j w_jC_j$, where C_j is the time at which job j completes in the given schedule (the *completion time* of j). This scheduling problem is denoted $1|prec|\sum_j w_jC_j$ in the notation of Lawler et al. [6].

The general problem $1|prec|\sum_j w_jC_j$ was shown to be NP-complete by Lenstra and Rinnooy Kan [7] and Lawler [4]. Nevertheless, special cases are known to be polynomial-time solvable. In 1978, Lawler [4] gave an $O(n \log n)$ time algorithm for solving $1|prec|\sum_j w_jC_j$ when the given precedence constraints are *series-parallel*. Series-parallel precedence constraints can be defined inductively; the base elements are individual jobs. Given series-parallel constraints on sets of jobs S_1 and S_2, $S_1 \cap S_2 = \emptyset$, the *parallel composition* of S_1 and S_2 gives a partial order on $S_1 \cup S_2$ that maintains the orders on S_1 and S_2, and if $i \in S_1, j \in S_2$, then i and j are unordered. The *series composition* of S_1 and S_2, $S_1 \cap S_2 = \emptyset$, gives a partial order on $S_1 \cup S_2$ that maintains the orders on S_1 and S_2, and if $i \in S_1$ and $j \in S_2$, then i precedes j in the partial order. Often series-parallel constraints are given in terms of a binary *structure tree*, with the leaves denoting jobs and the internal nodes denoting either a parallel or series composition of the two corresponding subtrees.

In this note we give an alternate proof of the correctness of Lawler's algorithm. We

*Address: MIT, Room 2-351, Department of Mathematics, 77 Massachusetts Avenue, Cambridge, MA 02139. Part of this work was done while at CORE, Louvain-La-Neuve, Belgium. Email: goemans@math.mit.edu. Research supported in part by NSF contract 9623859-CCR, ARPA Contract N00014-95-1-1246, and a Sloan fellowship.

†Address: IBM T.J. Watson Research Center, Room 33-219, P.O. Box 218, Yorktown Heights, NY, 10598. Email: dpw@watson.ibm.com.

do this by using a linear programming formulation of $1|prec|\sum_j w_j C_j$ introduced by Queyranne and Wang [11]. Queyranne and Wang proved that their formulation completely describes the scheduling polyhedron in the case of series-parallel constraints; a by-product of our proof of correctness of Lawler's algorithm is an alternate proof of this fact. Other proofs of the correctness of Lawler's algorithm have been given (see [4, 10, 9], for example), but to the best of our knowledge ours is the first duality-based proof.

In the course of our proof it is helpful to use what might be called *two-dimensional Gantt charts*. Although 2D Gantt charts were introduced in a 1964 paper by Eastman, Even, and Isaacs [2], they seem to have become buried in the literature. We use these 2D Gantt charts to obtain geometric intuition into the dual of Queyranne and Wang's linear programming formulation. We think 2D Gantt charts may find further uses, and to illustrate this we show that a recent observation of Hall and Chudak [3], and a recent theorem of Margot, Queyranne, and Wang [8] (discovered independently by Chekuri and Motwani [1]) are transparent using 2D Gantt charts.

The remainder of the note is structured as follows. In Section 2 we introduce 2D Gantt charts and illustrate their usefulness. In Section 3 we review the LP formulation for $1|prec|\sum_j w_j C_j$. Finally, in Section 4 we turn to the proof of correctness of Lawler's algorithm.

2. Two-dimensional Gantt charts. We first need the following notation. Let $N = \{1, \ldots, n\}$ denote the set of all jobs. For any set S of jobs, let $w(S) = \sum_{j \in S} w_j$ and $p(S) = \sum_{j \in S} p_j$.

A traditional Gantt chart has a single dimension of processing time, in which jobs are represented as blocks of length p_j (See Figure 2.1).

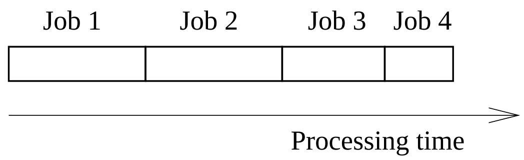

FIG. 2.1. *A Gantt chart.*

In a 2D Gantt chart, we introduce a second axis of weight. The chart starts at the point $(0, w(N))$ and ends at $(p(N), 0)$. Each job, say j, is represented by a rectangle of length p_j and height w_j whose position is defined by a startpoint and an endpoint. The startpoint (t, w) of a job is the endpoint of the previous job (or $(0, w(N))$ for the first job) while its endpoint is $(t + p_j, w - w_j)$. See Figure 2.2 for an illustration in which job 3 has a negative weight. The total amount of weight that has not been completed yet at any point in time — *the uncompleted work* — can be easily read off from the 2D Gantt chart: This corresponds to the bold line in the figure and it is composed of the upper side of each rectangle for the jobs of positive weight and of the lower side of each rectangle for the jobs of nonpositive weight. The value $\sum_j w_j C_j$ of a schedule can then be easily inferred from the 2D Gantt chart; it is simply the area below the uncompleted work line, as shaded in Figure 2.3. This is true even if there are negative weight jobs since $w_1 p_1 + w_2(p_1 + p_2) + \cdots + w_n(p_1 + p_2 + \cdots + p_n)$ can also be written as $p_1(w_1 + w_2 + \cdots + w_n) + p_2(w_2 + w_3 + \cdots + w_n) + \cdots + p_n w_n$. We will refer to this shaded area as the *area* of the schedule. Thus minimizing $\sum_j w_j C_j$ over non-preemptive schedules is equivalent to arranging the jobs as shown in the 2D Gantt chart so as to minimize its area. If we have negative weights and the chart of a given

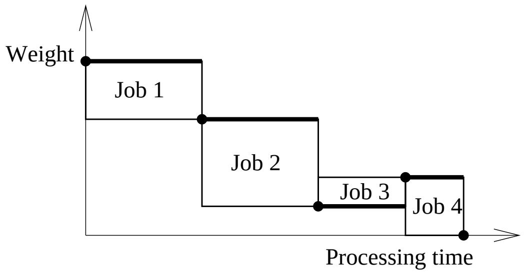

FIG. 2.2. *A 2D Gantt chart. Job 3 has a negative weight.*

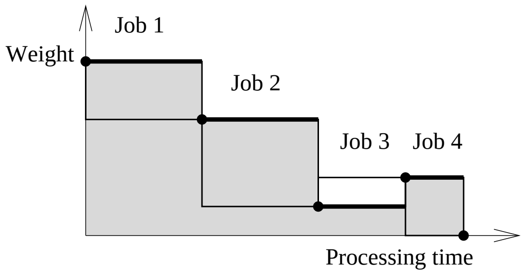

FIG. 2.3. *The area of a schedule.*

schedule goes below the x axis, then this means that, by postponing the processing of the jobs following that point, we can decrease the weighted sum of completion times arbitrarily and the problem is unbounded. We will see that this is the only possible situation to make the objective function unbounded; in particular, if the problem is bounded, then there will not be any idle time in an optimum schedule. As a result, we will only consider schedules without idle time.

For convenience, we will often draw a line from the startpoint to the endpoint of the rectangle representing a job. We will denote minus the slope of this line segment as $\rho(j) = w_j/p_j$, and will sometimes refer to this line segment as the *slope* of job j. See Figure 2.4. We will let $\rho(S)$ denote the slope of a set S of jobs, so that $\rho(S) = w(S)/p(S)$, where $w(S) = \sum_{j \in S} w_j$ and $p(S) = \sum_{j \in S} p_j$. The dotted line in the figure represents $\rho(S)$ for $S = \{1, 2, 3\}$. Observe that the area of a schedule is

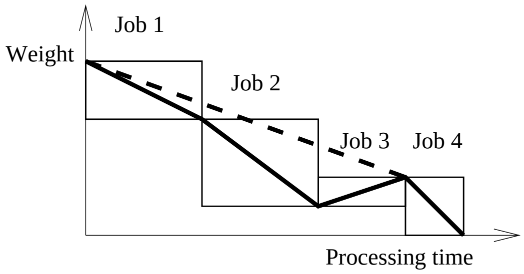

FIG. 2.4. *The slope of jobs, and of a set of jobs.*

equal to the area below its slopes plus $\sum_{j \in N} \frac{1}{2} w_j p_j$. The area below the slopes thus represents $\sum_{j \in N} w_j (C_j - \frac{1}{2} p_j) = \sum_{j \in N} w_j M_j$ where M_j denotes the mean busy time of job j; that is the mid-point between its start and its completion.

To show the usefulness of this concept, we illustrate two recently discovered facts about $1|prec|\sum_j w_j C_j$ using 2D Gantt charts. First, Hall and Chudak [3] made the following observation (see Von Arnim et al. [14] for the case $p_j = 1$). Given an instance $(p, w, \rightarrow)$ of $1|prec|\sum_j w_j C_j$, where $\rightarrow$ is the precedence relation, if one creates a new instance $(p', w', \rightarrow')$ by setting $p'_j = w_j$ and $w'_j = p_j$ for all j, and $j \rightarrow' i$ iff $i \rightarrow j$, then an optimal schedule for $(p', w', \rightarrow')$ is in the opposite order of an optimal schedule for $(p, w, \rightarrow)$ and has the same value. This follows quite simply by displaying any solution for $(p, w, \rightarrow)$ on a 2D Gantt chart and flipping the axes to obtain a solution of the same value for $(p', w', \rightarrow')$. See Figure 2.5 for an illustration.

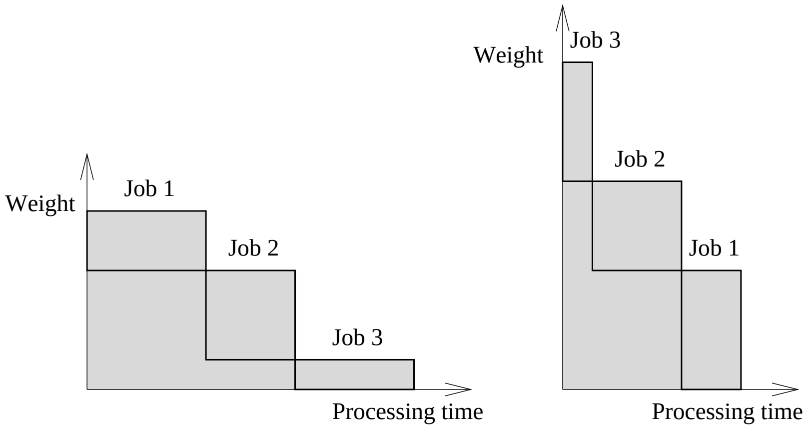

FIG. 2.5. *Illustration of the Hall-Chudak observation.*

Also, Margot et al. [8] recently showed the following (also discovered independently by Chekuri and Motwani [1]): suppose we have an instance of $1|prec|\sum_j w_j C_j$ such that all weights are nonnegative and for any *initial* set of jobs S (i.e. there exists a valid schedule in which all the jobs in S are scheduled before all jobs in $N - S$), $\rho(S) \leq \rho(N)$. Then any valid schedule comes within a factor of 2 of optimal. Given the condition $\rho(S) \leq \rho(N)$ for any initial set S, using a 2D Gantt chart it is easy to see that the slopes of the jobs will always remain above the slope of the entire set of jobs. Thus for any schedule $\sum_j w_j C_j \geq \frac{1}{2} w(N) p(N)$ since its area will always be at least this much. See Figure 2.6 for an illustration. But certainly $\sum_j w_j C_j \leq w(N)p(N)$, so that any schedule is no more than twice the optimal value.

This observation, together with a result of Sidney [12], leads to a 2-approximation algorithm for general instances of $1|prec|\sum_j w_j C_j$ as observed by Chekuri and Motwani [1]. Sidney [12] shows that if we let S to be an initial set of jobs of maximum ρ value then there exists an optimum solution to $1|prec|\sum_j w_j C_j$ for which the jobs in S are processed before any job in $N \setminus S$. This result can be shown using 2D Gantt charts, and this is left as a (non-trivial) exercise to the reader. Chekuri and Motwani [1] propose to find such a set S (which can be done using parametric maximum flow techniques), process these jobs first in any valid ordering and repeat the procedure with the remaining jobs in $N \setminus S$. Their previous observation together with the result of Sidney guarantees that the solution they construct is within a factor of 2 of the optimum.

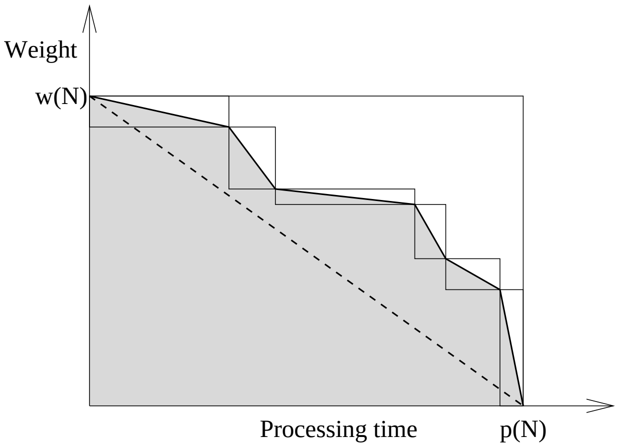

FIG. 2.6. *Illustration of the Margot et al. result.*

3. A linear program for $1|prec|\sum_j w_j C_j$ and some preliminaries. We now turn to proving that a scheduling algorithm of Lawler for $1|prec|\sum_j w_j C_j$ is optimal when the precedence constraints are series-parallel. The proof uses the following linear programming formulation of the problem:

$$\text{Min} \quad \sum_j w_j(M_j + p_j/2)$$

subject to:

$$\sum_{j\in S} p_j M_j \geq \frac{1}{2}p(S)^2 \qquad S \subseteq N \tag{3.1}$$

$$\frac{1}{p(B)}\sum_{j\in B} p_j M_j - \frac{1}{p(A)}\sum_{j\in A} p_j M_j \geq \frac{1}{2}(p(A)+p(B)) \quad A \subseteq N, B \subseteq N-A, A \to B, \tag{3.2}$$

where $A \to B$ means that the precedence constraints enforce that each job in A must be scheduled before each job in B. In this formulation, M_j represents the mean busy time of job j. By adopting the convention that $\emptyset \to S$ for any S, we could simply get rid of constraints (3.1). Queyranne and Wang [11] have shown that the completion time reformulations of constraints (3.1) and (3.2) completely describe the scheduling polyhedron when the precedence constraints are series-parallel (also shown by Von Arnim et al. [14] in the case $p_j = 1$ for all jobs). A by-product of our proof of correctness of Lawler's algorithm is an alternate proof of this fact.

To see that the constraints (3.1) are valid, choose any $S \subseteq N$; suppose $S = \{1, \ldots, k\}$. If the jobs $1, \ldots, k$ are the first jobs scheduled, then simple algebra shows that $\sum_{j\in S} p_j M_j = \frac{1}{2}p(S)^2$ (notice that this does not depend on the ordering of the jobs). In any valid schedule the sum is at least as large, and hence inequality (3.1) is valid. More generally, if the jobs in S are continuously scheduled between s and $t = s + p(S)$ then $\sum_{j\in S} p_j M_j = \left(s + \frac{1}{2}p(S)\right)p(S) = \left(t - \frac{1}{2}p(S)\right)p(S)$. Hence, in any valid schedule in which the jobs in S are scheduled (not necessarily continuously) between s and t, we have

$$\left(s + \frac{1}{2}p(S)\right)p(S) \leq \sum_{j\in S} p_j M_j \leq \left(t - \frac{1}{2}p(S)\right)p(S). \tag{3.3}$$

To see that the constraints (3.2) are valid, choose any A and B obeying the conditions. From (3.3), we derive that $\frac{1}{p(B)}\sum_{j\in B} p_j M_j \geq \tau + \frac{1}{2}p(B)$ and $\frac{1}{p(A)}\sum_{j\in A} p_j M_j \leq \tau - \frac{1}{2}p(A)$, where τ is any time between the completion of A and B. Subtracting these two inequalities gives (3.2). Note that an inequality of the form (3.2) is tight exactly when all jobs in A are processed from time $\tau - p(A)$ to τ (for some τ), followed by jobs in B processed from time τ to time $\tau + p(B)$.

To prove the optimality of Lawler's algorithm, we will construct a feasible solution to the dual of the linear program above:

$$\text{Max} \quad \frac{1}{2}\sum_S p(S)^2 y_S + \frac{1}{2}\sum_{A,B}(p(A)+p(B))z_{A,B} + \frac{1}{2}\sum_j w_j p_j$$

subject to:

$$\begin{aligned} &\sum_{S:j\in S} y_S + \sum_{A,B:j\in B}\frac{1}{p(B)}z_{A,B} - \sum_{A,B:j\in A}\frac{1}{p(A)}z_{A,B} = \rho(j) && \forall j \\ &y_S \geq 0 && \forall S \subseteq N \\ &z_{A,B} \geq 0 && \forall A \subseteq N, B \subseteq N - A, A \to B. \end{aligned}$$

We will show that our dual solution obeys the complementary slackness conditions with respect to the M_j constructed by Lawler's algorithm. Assuming that all $p_j > 0$, notice that if we have a solution M and (y,z) that obey the complementary slackness relations, then by the observations above, if $y_S > 0$, then the jobs in S must all appear at the beginning of the schedule, and if $z_{A,B} > 0$, then the jobs in A and B are all scheduled together, with the jobs in A appearing immediately before those in B.

As a warm-up exercise, suppose that $\rho(1) \geq \rho(2) \geq \cdots \geq \rho(n) \geq 0$, and the schedule $1,\ldots,n$ is compatible with the precedence constraints. Notice that we require that the minimum slope $\rho(n)$ be nonnegative; otherwise the problem is unbounded since we can postpone the processing of this last job arbitrarily. Then the dual solution

$$\begin{aligned} y_{\{1\}} &= \rho(1) - \rho(2) \\ y_{\{1,2\}} &= \rho(2) - \rho(3) \\ &\vdots \\ y_{\{1,\ldots,n-1\}} &= \rho(n-1) - \rho(n) \\ y_{\{1,\ldots,n\}} &= \rho(n) \end{aligned}$$

and all other variables set to zero is feasible and obeys the complementary slackness conditions, by the discussion above. Hence the schedule $1,\ldots,n$ is optimal. Notice that this gives an alternate proof of Smith's rule [13], which states that scheduling jobs in order of non-increasing ρ value gives the optimal schedule for problem instances without precedence constraints.

We now consider this simple case from the perspective of 2D Gantt charts. We observe that in this case, the diagonals of the jobs in a 2D Gantt chart representation of the sequence $1,2,\ldots,n$ form a piecewise-linear convex function (see Figure 3.1). Note also that the area of the schedule can be expressed as $\sum_{j\in N}\frac{1}{2}w_j p_j$ plus the area below the slopes of the jobs. This area can be decomposed into n triangles, shown as shaded regions in Figure 3.1. These triangles are formed by extending the slope of each job to the y axis. We associate with job j the triangle formed between the line from the slope of job j and that of the slope of job $j+1$ (where we use the y

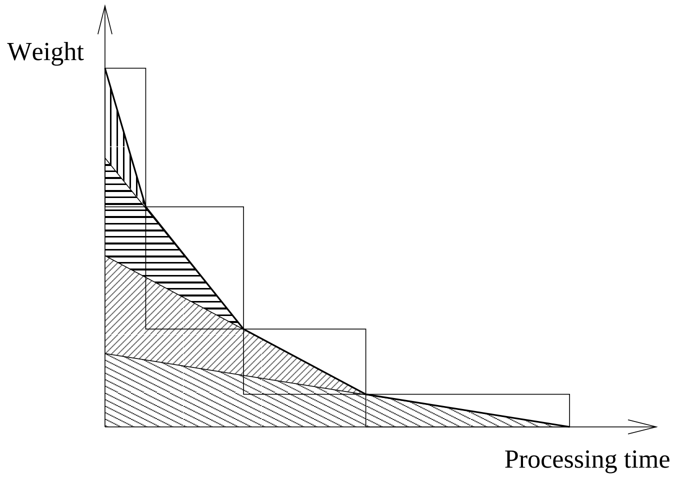

FIG. 3.1. *Illustration of Smith's rule. Shaded areas indicate the n triangles that account for the area of the schedule (after $\frac{1}{2}\sum_j w_j p_j$ is included).*

axis of slope $\rho(n+1) = 0$ for job $n+1$). Then the "base" of the jth triangle on the y axis is $p(\{1,\ldots,j\})(\rho(j) - \rho(j+1))$, and its height is $p(\{1,\ldots,j\})$, so that its area is exactly $\frac{1}{2}p(\{1,\ldots,j\})^2(\rho(j) - \rho(j+1)) = \frac{1}{2}p(\{1,\ldots,j\})^2 y_{\{1,\ldots,j\}}$, using the dual solution of the preceding paragraph. Then the total area of the schedule is $\frac{1}{2}\sum_S p(S)^2 y_S + \frac{1}{2}\sum_j w_j p_j$, or exactly the value of the dual objective function.

4. Lawler's algorithm. We now turn to Lawler's algorithm. Lawler's algorithm works bottom-up on the series-parallel structure tree of the precedence constraints. We give the algorithm below, but first we will try to give some intuition on how the algorithm and our dual construction works. One perspective is that as much as possible it tries to follow Smith's rule, and makes sure that the jobs in the current subtree can be maintained in order of non-increasing ρ, so that the dual solution given above proves the optimality of the schedule. This may not always be possible, so in some cases two or more jobs are replaced by a *composite* job. The composite jobs may be composed themselves of composite jobs. The weight of a composite job of a set S of jobs is $w(S)$ and its processing time is $p(S)$, so that $\rho(S) = w(S)/p(S)$. A sequence of the jobs forming the composite job is given, so that the effect of scheduling the $p(S)$ time units of the composite job is that the jobs in S are scheduled together in the given sequence.

To be more specific, suppose we have a parallel composition of two sets of jobs S_1 and S_2, such that the ρ values of the jobs in S_i can be scheduled in non-increasing order. Then the jobs in $S = S_1 \cup S_2$ can be scheduled in non-increasing ρ order. (We should stress that, when scheduling in non-increasing order of ρ values, the slopes of the jobs form a convex function which starts to increase once we start processing the jobs with negative ρ value; see Figure 4.1 for an illustration.) However, if we have a series composition of two sets of jobs S_1 and S_2, then it is possible that some job in S_2 has ρ value greater than some job in S_1. In this case, we create a composite job c formed of a certain number of jobs of S_1 with lowest ρ values and a certain number of jobs of S_2 with greatest ρ value. The crucial property to achieve is that the slope of the composite job c is no greater than the ρ value of any remaining

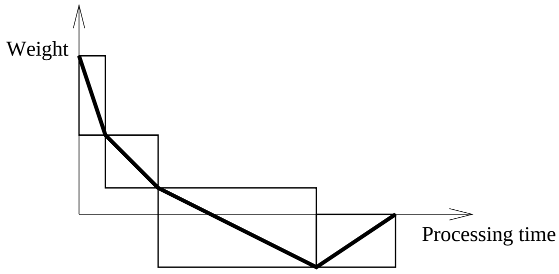

FIG. 4.1. *Slopes in each S_i are non-increasing before performing a series or parallel composition.*

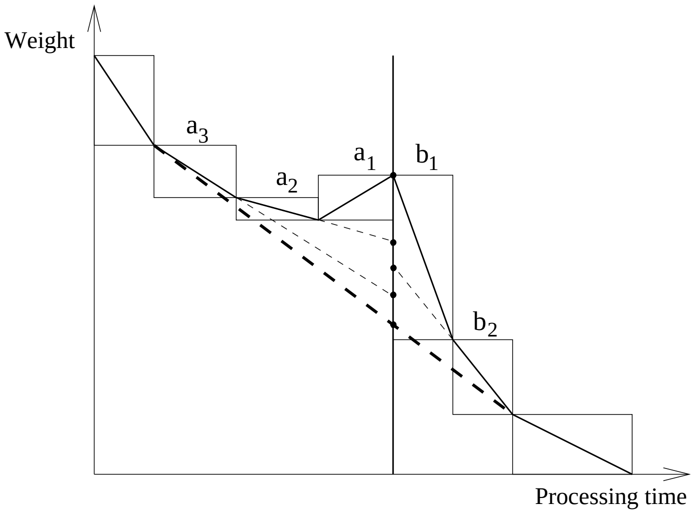

FIG. 4.2. *An illustration of a composite job c resulting from a series composition of two sets of jobs. The slope $\rho(c)$ is shown as the bold dashed line.*

job in S_1 and no less than the ρ value of any remaining job in S_2. The scheduling of job c is then understood to mean that the jobs taken from S_1 are scheduled in order of non-increasing ρ, followed immediately by the jobs taken from S_2 in order of non-increasing ρ.

There is a unique way of performing this aggregation of jobs, as one can easily see from the 2D Gantt chart. Consider, for example, the situation in Figure 4.2. In this case, there is the series composition of four jobs in S_1 (the four jobs to the left of the vertical line, which we will call the *dividing line*) and three jobs in S_2. Note that both S_1 and S_2 can be scheduled according to Smith's rule, so that the slopes of the jobs in each form a piecewise-linear convex function. A composite job c is formed from a_3, a_2, a_1, b_1, b_2. In general, the composite job is formed by the jobs whose slopes do not participate in the lower envelope of the slopes in S_1 and in S_2.

In Lawler's algorithm, for a series composition of S_1 and S_2, the composite job was obtained by repeatedly combining two jobs at a time. More precisely, let j_1 be the job in S_1 that minimizes $\rho(j)$, and let j_2 be the job in S_2 that maximizes $\rho(j)$. If $\rho(j_1) \geq \rho(j_2)$, then $S = S_1 \cup S_2$ can be scheduled in non-increasing ρ order, and we stop. Otherwise, we remove j_1 from S_1 and j_2 from S_2, and form a composite job $c = (j_1, j_2)$. As long as there is a job j in S_1 whose ρ value is lower than $\rho(c)$ or a job

j in S_2 whose ρ value is greater than $\rho(c)$, we remove j from S_1 or S_2 and add it to the composite job c. When this terminates, $\rho(j) \geq \rho(c)$ for all jobs remaining in S_1 (if any), and $\rho(c) \geq \rho(j)$ for any job j remaining in S_2 (if any).

When performing a series composition of S_1 and S_2 which concatenates jobs $a_k, a_{k-1}, \ldots, a_1 \in S_1$ (where $\rho(a_i) \geq \rho(a_{i-1})$) and jobs $b_1, b_2, \ldots, b_\ell \in S_2$ (where $\rho(b_i) \geq \rho(b_{i+1})$) into a single job c, we define the z variables in such a way that the ρ value of a_i effectively increases to $\rho(c)$ and the ρ value of b_j effectively decreases to $\rho(c)$. More formally, we construct dual variables $z_{A,B} \geq 0$ such that $z_{A,B} > 0$ only if $A \subseteq S_1$, $B \subseteq S_2$, the jobs in A appear immediately before B in the schedule, and such that for all a_i,

$$\rho(c) = \rho(a_i) + \sum_{A,B: a_i \in A} \frac{1}{p(A)} z_{A,B},$$

and for all jobs b_j

$$\rho(c) = \rho(b_j) - \sum_{A,B: b_j \in B} \frac{1}{p(B)} z_{A,B}.$$

All the jobs composing c look now identical in terms of ρ values. Later on, if we set a variable $z_{A,B}$, or even y_S, involving the composite job c, we simply need to replace c by the set of jobs (or by the set of composite jobs and proceed recursively) to get the appropriate dual solution. In particular, if an a_i or b_j is itself a composite job formed in a previous series composition, applying the argument recursively ensures that all the (original) jobs that compose c look identical in terms of ρ values. In the process of defining $z_{A,B}$, we need to ensure that the contribution of these newly defined variables to the dual objective function corresponds exactly to the area between the slope of the composite job c and the slopes of the jobs composing c (the area shaded on Figure 4.5); this is the area of the schedule which is lost by replacing the jobs composing c into a single composite job. If we can show that these newly defined variables account for this lost area, then by induction it suffices to show that, at the end, after processing the root of the series-parallel structure tree, we can account for the area below the resulting schedule, and this will follow from our discussion of Smith's rule from Section 3.

The z variables are defined as follows. Let $A_i = \{a_1, \ldots, a_i\}$ for $i \leq k$ and $B_j = \{b_1, \ldots, b_j\}$ for $j \leq \ell$. We set $z = 0$ and give the procedure shown in Figure 4.3 for computing $z_{i,j} \equiv z_{A_i,B_j}$, where we use the convention that $A_{k+1} \equiv A_k$, $B_{\ell+1} \equiv B_\ell$, $\rho(a_{k+1}) \equiv \rho(b_{\ell+1}) \equiv \rho(A_k \cup B_\ell)$. As stated above, we need to show that $z \geq 0$, that for job a_i

$$\rho(A_k \cup B_\ell) - \sum_{r,s: a_i \in A_r} \frac{1}{p(A_r)} z_{r,s} = \rho(a_i),$$

and that for job b_j

$$\rho(A_k \cup B_\ell) + \sum_{r,s: b_j \in B_s} \frac{1}{p(B_s)} z_{r,s} = \rho(b_j).$$

Before we formally prove the correctness of these values for $z_{i,j}$, we use 2D Gantt charts to give some intuition of where these values come from. In the situation

```
for i ← 1 to k
  α_i ← ρ(a_{i+1})p(A_{i+1}) − w(A_{i+1})
for j ← 1 to ℓ
  β_j ← w(B_{j+1}) − ρ(b_{j+1})p(B_{j+1})
i ← 1
j ← 1
x ← 0
while i ≤ k and j ≤ ℓ
  if α_i < β_j
    z_{i,j} ← α_i − x
    x ← α_i
    i ← i + 1
  else
    z_{i,j} ← β_j − x
    x ← β_j
    j ← j + 1
```

FIG. 4.3. *Procedure for computing* $z_{i,j}$

depicted in Figure 4.2, we extend the slopes of a_3,a_2, and b_2 to intersect the dividing line: these extensions are shown as dashed lines. If we think of the origin of the dividing line as the point at which a_1 and b_1 touch, then α_i is the point on the dividing line at which the slope extended from job a_{i+1} touches the dividing line. Similarly, β_i is the point on the dividing line at which the slope extended from job b_{i+1} touches the dividing line. See Figure 4.4 for a blowup of the dividing line. Then the $z_{i,j}$ are computed by walking down the dividing line from the origin to the next point on the line: $z_{i,j}$ is the difference between the current point and the last one, and i gets incremented if an α is encountered, whereas j gets incremented if a β is encountered. Now observe that $\frac{1}{2}\sum_{i,j}(p(A_i)+p(B_j))z_{i,j}$ expresses exactly the area above the slope of composite job c but underneath the slopes of the jobs a_i and b_i, as needed. To see this, observe that the shaded area in Figure 4.4 is the area of two triangles, each with base $z_{2,1} = \beta_1 - \alpha_1$, one of height $p(A_2)$ and one of height $p(B_1)$, for a total area of $\frac{1}{2}z_{2,1}(p(A_2)+p(B_1))$. The area between the lower envelope and the slopes of the jobs can be expressed as the sum of pairs of triangles like this pair, as shown in Figure 4.5.

We now turn to the formal proof of the correctness of the dual variables. To show that $z \geq 0$, note that by construction $z_{i,j}$ is either $\alpha_i - \alpha_{i-1}$, $\beta_j - \beta_{j-1}$, $\alpha_i - \beta_{j-1}$, or $\beta_j - \alpha_{i-1}$. In the last two cases, it follows immediately that $z_{i,j} \geq 0$: in the previous iteration of the program (when x was set to β_{j-1} (resp. α_{i-1})), it was the case that $\alpha_i \geq \beta_{j-1}$ (resp. $\beta_j > \alpha_{i-1}$). In the first case,

$$\begin{aligned}
\alpha_i - \alpha_{i-1} &= \rho(a_{i+1})p(A_{i+1}) - \rho(a_i)p(A_i) + w(A_i) - w(A_{i+1}) \\
&= p(A_i)(\rho(a_{i+1}) - \rho(a_i)) + \rho(a_{i+1})p(a_{i+1}) - w(a_{i+1}) \\
&= p(A_i)(\rho(a_{i+1}) - \rho(a_i)) \\
&\geq 0.
\end{aligned}$$

In the second case,

$$\begin{aligned}
\beta_j - \beta_{j-1} &= w(B_{j+1}) - w(B_j) + \rho(b_j)p(B_j) - \rho(b_{j+1})p(B_{j+1}) \\
&= p(B_j)(\rho(b_j) - \rho(b_{j+1})) - \rho(b_{j+1})p(b_{j+1}) + w(b_{j+1})
\end{aligned}$$

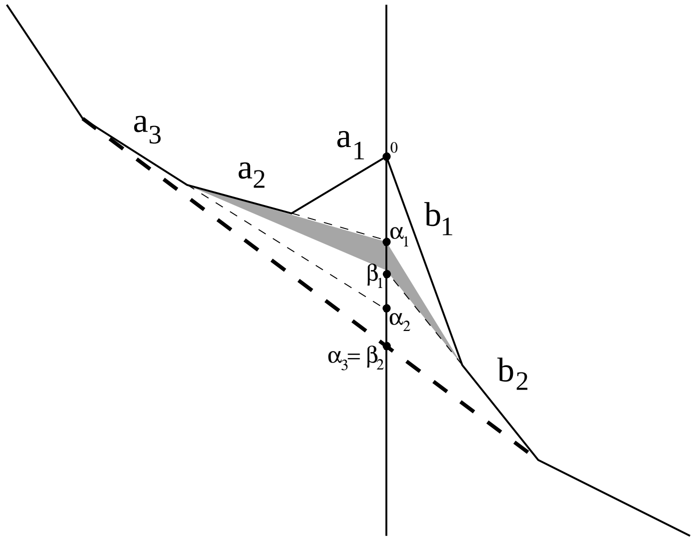

FIG. 4.4. *A blowup of the dividing line, and intersections of it with extensions of slopes of the a_i and b_i, resulting in the α_{i+1} and β_{i+1}. By construction $z_{2,1} = \beta_1 - \alpha_1$ and the gray shaded area is the sum of two triangles of base $z_{2,1}$ and heights $p(A_2)$ and $p(B_1)$ respectively.*

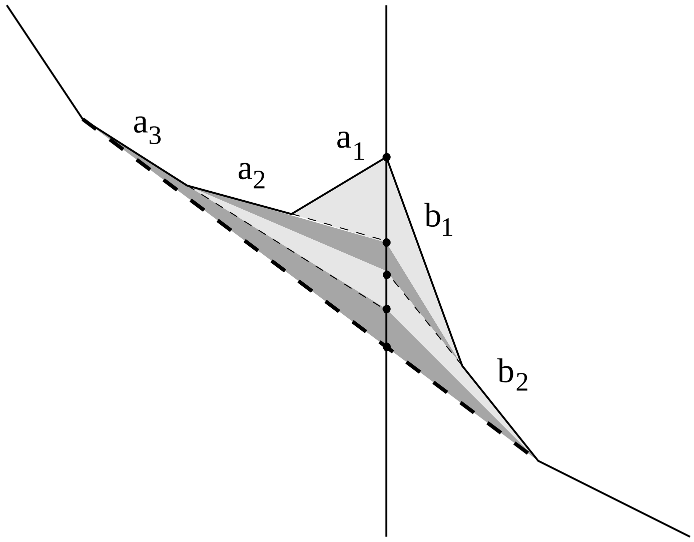

FIG. 4.5. *An illustration showing that the area between the slope of a composite job and its constituent jobs can be expressed as the sum of the area of pairs of triangles.*

$$
\begin{aligned}
&= p(B_j)(\rho(b_j) - \rho(b_{j+1})) \\
&\geq 0.
\end{aligned}
$$

To see that

$$\rho(A_k \cup B_\ell) - \sum_{r,s:a_i \in A_r} \frac{1}{p(A_r)} z_{r,s} = \rho(a_i)$$

for job a_i, first observe that by construction

$$\sum_{r,s:a_i\in A_r} \frac{1}{p(A_r)} z_{r,s} = \sum_{r:a_i\in A_r} \frac{1}{p(A_r)} \sum_s z_{r,s} = \sum_{r=i}^{k} \frac{1}{p(A_r)}(\alpha_r - \alpha_{r-1}).$$

Thus we have

$$\begin{aligned}
\rho(A_k \cup B_\ell) - \sum_{r,s:a_i\in A_p} \frac{1}{p(A_r)} z_{r,s} &= \rho(A_k \cup B_\ell) - \sum_{r=i}^{k} \frac{1}{p(A_r)}(\alpha_r - \alpha_{r-1}) \\
&= \rho(A_k \cup B_\ell) - \sum_{r=i}^{k} \frac{1}{p(A_r)}(p(A_r)(\rho(a_{r+1}) - \rho(a_r))) \\
&= \rho(A_k \cup B_\ell) - (\rho(A_k \cup B_\ell) - \rho(a_i)) \\
&= \rho(a_i).
\end{aligned}$$

Showing that

$$\rho(A_k \cup B_\ell) + \sum_{r,s:b_j\in B_s} \frac{1}{p(B_s)} z_{r,s} = \rho(b_j)$$

for job b_j is similar. First we observe that

$$\sum_{r,s:b_j\in B_s} \frac{1}{p(B_s)} z_{r,s} = \sum_{s=j}^{\ell} \frac{1}{p(B_s)}(\beta_s - \beta_{s-1}),$$

so that

$$\begin{aligned}
\rho(A_k \cup B_\ell) + \sum_{r,s:b_j\in B_s} \frac{1}{p(B_s)} z_{r,s} &= \rho(A_k \cup B_\ell) + \sum_{s=j}^{\ell} \frac{1}{p(B_s)}(\beta_s - \beta_{s-1}) \\
&= \rho(A_k \cup B_\ell) + \sum_{s=j}^{\ell} \frac{1}{p(B_s)}(p(B_s)(\rho(b_s) - \rho(b_{s+1}))) \\
&= \rho(A_k \cup B_\ell) + (\rho(b_j) - \rho(A_k \cup B_\ell)) \\
&= \rho(b_j).
\end{aligned}$$

After processing the root of the series-parallel structure tree, we are left with the final set of jobs $c_1, c_2, \ldots, c_k$ returned by the algorithm (some of them possibly composite jobs), with $\rho(c_1) \geq \rho(c_2) \geq \cdots \geq \rho(c_k)$, and the jobs scheduled as ordered. Let $J(c_i)$ denote the set of all the actual jobs contained in the job c_i. There are two possibilities. Either $\rho(c_k) \geq 0$ or $\rho(c_k) < 0$. In the first case, we can set

$$\begin{aligned}
y_{J(c_1)} &= \rho(c_1) - \rho(c_2) \\
y_{J(c_1)\cup J(c_2)} &= \rho(c_2) - \rho(c_3) \\
&\vdots \\
y_{\cup_{i=1}^{k-1} J(c_i)} &= \rho(c_{k-1}) - \rho(c_k) \\
y_{\cup_{i=1}^{k} J(c_i)} &= \rho(c_k),
\end{aligned}$$

so that complementary slackness is obeyed for the y variables with respect to the schedule, and so that for any given actual job $j \in J(c_i)$, $\sum_{S:j\in S} y_S = \rho(c_i)$. We have therefore derived that the schedule is optimum both for the problem and for the linear programming formulation of Queyranne and Wang [11].

In the second case, i.e. when $\rho(c_k) < 0$, we can postpone the processing of the jobs $J(c_k)$ and make the schedule of arbitrarily negative objective function value. Since any feasible schedule leads to a feasible primal solution for the LP formulation we are considering, this LP is also unbounded.

In summary, we have just shown that if there is an optimum schedule then this schedule provides an optimum solution to the LP formulation and if there is none (and the problem is unbounded) the LP is also unbounded. We have therefore simultaneously given a proof of correctness of Lawler's algorithm and an alternate proof of the polyhedral result of Queyranne and Wang [11].

Acknowledgments. We thank David Shmoys for mentioning this problem to us and pointing out reference [2]. We also thank Jan Karel Lenstra for publicizing the problem in his talk at the 1997 International Symposium on Mathematical Programming, and Gideon Weiss for convincing us to modify slightly our definition of 2D Gantt charts. Finally, we thank an anonymous referee for several very useful comments.

REFERENCES

[1] C. Chekuri and R. Motwani. Precedence constrained scheduling to minimize weighted completion time on a single machine. Technical Note STAN-CS-TN-97-58, Department of Computer Science, Stanford University, 1997.

[2] W. L. Eastman, S. Even, and I. M. Isaacs. Bounds for the optimal scheduling of n jobs on m processors. *Management Science*, 11:268–279, 1964.

[3] L. Hall and F. Chudak. Personal communication, 1997.

[4] E. L. Lawler. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics*, 2:75–90, 1978.

[5] E. L. Lawler and J. K. Lenstra. Machine scheduling with precedence constraints. In I. Rival, editor, *Ordered Sets*, pages 655–675. D. Reidel Publishing, 1982.

[6] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. Graves, A. Rinnooy Kan, and P. Zipkin, editors, *Handbooks in Operations Research and Management Science, Vol 4., Logistics of Production and Inventory*, pages 445–522. North Holland, 1993.

[7] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26:22–35, 1978.

[8] F. Margot, M. Queyranne, and Y. Wang. Personal communication, 1997.

[9] C. L. Monma. *Properties and efficient algorithms for certain classes of sequencing problems.* PhD thesis, Cornell University, School of Operations Research and Industrial Engineering, 1978.

[10] C. L. Monma and J. B. Sidney. Sequencing with series-parallel precedence constraints. *Mathematics of Operations Research*, 4:215–224, 1979.

[11] M. Queyranne and Y. Wang. Single-machine scheduling polyhedra with precedence constraints. *Mathematics of Operations Research*, 16:1–20, 1991.

[12] J. B. Sidney. Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Operations Research*, 23:283–298, 1975.

[13] W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

[14] A. Von Arnim, U. Faigle, and R. Schrader. The permutahedron of series-parallel posets. *Discrete Applied Mathematics*, 28:3–9, 1990.