Mathematical Programming 69 (1995) 413–428

Dual decomposition of a single-machine scheduling problem

S.L. van de Velde

Department of Mechanical Engineering, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

Received 28 May 1990; revised manuscript received 22 September 1994

Abstract

We design a fast ascent direction algorithm for the Lagrangian dual problem of the single-machine scheduling problem of minimizing total weighted completion time subject to precedence constraints. We show that designing such an algorithm is relatively simple if a scheduling problem is formulated in terms of the job completion times rather than as an 0–1 linear program. Also, we show that upon termination of such an ascent direction algorithm we get a dual decomposition of the original problem, which can be exploited to develop approximative and enumerative approaches for it. Computational results exhibit that in our application the ascent direction leads to good Lagrangian lower and upper bounds.

Keywords: Machine scheduling; Lagrangian relaxation; Ascent direction method; Dual decomposition

1. Introduction

Lagrangian relaxation is already a conventional technique for lower bound computation, dating back to the work by Held and Karp [17,18] on the traveling salesman problem. Since then, it has shown its merits for a gamut of combinatorial optimization problems. Excellent introductions to Lagrangian relaxation theory are given by Geoffrion [12], Shapiro [33], and Fisher [8,9].

The underlying idea of Lagrangian relaxation is to see an NP-hard problem as an 'easy-to-solve' problem complicated by a number of 'nasty' side constraints. These nasty constraints are removed from the set of constraints, and put into the objective function, each weighted by a given Lagrangian multiplier. This manipulation gives *the Lagrangian problem* which is then an easy to solve problem and whose solution provides a lower bound for the original problem.

For any application, there are several issues to take care of, including dealing with *the Lagrangian dual problem.* This is the problem of finding the Lagrangian multipliers

SSDI 0025-5610(94)00085-9

that give the best Lagrangian lower bound. The subgradient method is frequently used to solve this problem, since it is always easy to implement. The downside of this iterative method, however, is that is does not produce a series of monotonically increasing lower bounds. It is known for its zigzagging in the beginning and slow convergence at the end (see, e.g., [19]).

So-called *ascent direction methods* do produce series of monotonically increasing lower bounds. These are iterative problem-specific approximation algorithms that exploit the structure of the problem and the formulation. An ascent direction algorithm is in general much faster than the subgradient method, but it cannot be guaranteed to produce lower bounds that are as good. Many success stories of Lagrangian relaxation are nonetheless attributed to ascent direction methods. They have shown to be successful for many combinatorial optimization problems, including plant location problems [3,6,14], the traveling salesman problem [2,4] the generalized assignment problem [10], and the set covering problem [11]. These applications indicate that the gain in speed over the subgradient method compensates the possible loss in lower quality more than sufficiently.

In spite of the abundance of machine scheduling problems, ascent direction algorithms have been used for them only by Hariri and Potts [16] and Potts [26]. Machine scheduling problems concern the scheduling of n jobs on machines of limited capacity and availability, generally so as to minimize some objective function of the job completion times. Much more than in other areas, the type of mathematical formulation employed is important for the design of ascent direction algorithms. The tricky issue is actually formulating the capacity constraints. There are essentially two ways to do this: either by means of 0–1 variables, which gives rise to an 0–1 linear program with many variables and constraints, or by 'logical' disjunctive constraints, which enables a compact formulation in terms of the job completion times C_j $(j = 1,\dots, n)$.

Integer linear programming formulations are predominantly cast in terms of *linear ordering* variables x_{jk} that take the value 1 if job J_j $(j = 1,\dots, n)$ completes before job J_k $(k = 1,\dots, n,\ k \neq j)$, and the value 0 otherwise, or in terms of *time-indexed* variables x_{jt} that take the value 1 if J_j $(j = 1,\dots, n)$ completes at time t, and the value 0 otherwise. Note that the second type of formulation requires a pseudopolynomial number of variables. As to the design of ascent direction algorithms, it seems that the first type of formulation is suitable for precedence-constrained scheduling problems only [16,26]. The second type of formulation does not seem to allow the kind of analysis required to develop fast ascent direction algorithms; the subgradient method is the only practical approach to deal with the Lagrangian dual problem. This gives generally very strong lower bounds, but the time needed to compute them is substantial [7]. For a specific single-machine scheduling problem, Dyer and Wolsey [5] show that Lagrangian bounds obtained from time-indexed formulations are stronger than those obtained from logical formulations in terms of the job completion times $C_1,\dots, C_n$. In many applications, however, logical formulations are easier to handle in that good Lagrangian lower bounds can quickly be obtained one way or the other, such as by elegant O(n) single-pass methods, as proposed by Hariri and Potts [15] and Potts and Van Wassenhove [27,28], by ascent direction algorithms, as we will show in this paper, or by more opportunistic approaches, as applied in [36].

In this paper, we design an ascent direction algorithm for the single-machine problem of minimizing total weighted completion time subject to precedence constraints. Our contribution is twofold. On the one hand, we show that it is relatively simple to design a fast and effective ascent direction algorithm if a problem is formulated in terms of the job completion times. On the other hand, we analyze the conditions under which the ascent direction algorithm terminates; we believe we are the first ever to do so. We assert that the kind of approach applies to about any classical machine scheduling problem when formulated in terms of the job completion times; for more details, refer to Van de Velde [37].

The organization of this paper is as follows. In Section 2, we develop the ascent direction algorithm and investigate the conditions under which the ascent direction algorithm terminates. It appears that upon termination we get a decomposition of the jobs into subsets; we call this a *dual decomposition.* In Section 3, we show how such a dual decomposition can be employed to find approximate solutions for the primal problem. Computational results exhibit the high quality of both the upper and lower bounds. Also, the time to compute the lower bound is almost negligible. In Section 4, we verify to what extent the dual decomposition concurs with a correct, primal decomposition, and point out how the dual decomposition can be of use for the design of enumerative methods. Section 5 concludes the paper.

Various lower bounds have been proposed for the machine scheduling problem under consideration. Potts [26] proceeds from a formulation in terms of linear ordering variables and presents an ascent direction algorithm to compute a Lagrangian bound; this algorithm is time-consuming, however, requiring $\Omega(n^4)$ time per iteration. Queyranne and Wang [31] proceed from a logical formulation of the problem and obtain their bound by solving the problem as a linear program to which they add three types of facet-defining inequalities; see also Queyranne and Wang [30]. Hoogeveen and Van de Velde [20] present a general technique to improve Lagrangian lower bounds by use of slack variables. They present various applications, including the problem under consideration. As to the quality of the bounds, Queyranne and Wang [31] prove that Hoogeveen and Van de Velde's bound is no stronger than theirs. Also, Wolsey [38] and Queyranne and Schulz [29] show that the linear programming bound obtained from the linear ordering formulation, of which Potts's Lagrangian bound is an approximation from below, is no weaker than Queyranne and Wang's bound. As to the speed by which the bounds can be computed, we note that the linear programming bounds are time-consuming; in contrast, our ascent direction algorithm is fast.

2. Single-machine scheduling

A single-machine job shop is described as follows. A set $\mathscr{J} = \{J_1, \ldots, J_n\}$ of n independent jobs has to be scheduled on a single machine that can handle only one job at a time. The machine is continuously available from time zero onwards. Each job J_j $(j = 1, \ldots, n)$ requires processing during an uninterrupted period of a given positive length p_j. In addition, each job J_j has a positive weight w_j, expressing its urgency relative to other jobs. Without loss of generality, we assume that the processing times and weights are integral. A *schedule* is a specification of the job completion times, denoted by C_j $(j = 1, \ldots, n)$, such that the jobs do not overlap in their execution and such that $C_j - p_j \geqslant 0$ for each j. The objective is to find a feasible schedule that minimizes the total weighted completion time $\sum_{j=1}^{n} w_j C_j$.

This problem, hereafter referred to as problem (P), is formulated as follows. Determine job completion times that

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} w_j C_j \\ \text{subject to} \quad & C_j \geqslant C_k + p_j, \text{ or } C_j \leqslant C_k - p_k, \\ & \quad \text{for } j = 1, \ldots, n-1, \; k = j+1, \ldots, n, && (1) \\ & C_j \geqslant p_j, \quad \text{for } j = 1, \ldots, n. && (2) \end{aligned}$$

Conditions (1) ensure that the machine processes no more than one job at a time; conditions (2) reflect that the machine is available from time zero onwards.

Theorem 1. *Problem* (P) *is solved in* O(n log n) *time by Smith's ratio rule* [35], *which schedules the jobs in order of nonincreasing ratios* w_j/p_j.

This rule is easily validated through an interchange argument.

Now, suppose there are precedence constraints between the jobs. The precedence constraints are represented by an acyclic directed graph G with vertex set $\{J_1, \ldots, J_n\}$ and arc set A, which equals its transitive reduction. A path in G from J_j to J_k implies that J_j has to be executed before J_k; J_j is a *predecessor* of J_k, and J_k is a *successor* of J_j. In case there is an arc $(J_j, J_k) \in A$, then J_j is said to be an *immediate predecessor* of J_k; J_k is then an *immediate successor* of J_j. We define $\mathscr{P}_j$ and $\mathscr{S}_j$ as the set of immediate predecessors and immediate successors of J_j, respectively $(j = 1, \ldots, n)$. Following the notation of Graham et al. [13], we refer to the problem of minimizing $\sum_{j=1}^{n} w_j C_j$ subject to precedence constraints on a single machine as $1 | \text{prec} | \sum w_j C_j$.

For special classes of precedence constraints, the problem is still solvable in O(n log n) time; this is the case for tree-like precedence constraints [1,21,34] and for series-parallel precedence constraints [22]. For general precedence constraints the problem is $\mathscr{N}\mathscr{P}$-hard in the strong sense [22,24]. This justifies the development of approximative and enumerative algorithms. Morton and Dharan [25] propose several heuristics. Specifically, the so-called *tree-optimal heuristic*, which produces optimal solutions in case the precedence constraints take the form of a tree, generates high-quality solutions. Potts [26] presents a branch-and-bound algorithm that solves instances up to 100 jobs; he employs Lagrangian lower bounds obtained from a 0–1 linear programming formulation in terms of linear ordering variables. We formulate the precedence constraints in a concise manner as

$$C_k \geqslant C_j + p_k \quad \text{for each } (J_j, J_k) \in A. \tag{3}$$

The $1|\text{prec}|\Sigma w_j C_j$ problem can be regarded as an easy-to-solve problem complicated by conditions (3). Accordingly, we introduce a vector $\lambda \in \mathbb{R}^A$ that contains a Lagrangian multiplier $\lambda_{jk} \geqslant 0$ for each arc $(J_j, J_k) \in A$ and put the constraints (3), each weighted by its multiplier, into the objective function. For a given vector $\lambda \geqslant 0$, the Lagrangian relaxation problem, referred to as problem (L_λ), is to find $L(\lambda)$, which is the minimum of

$$\sum_{j=1}^{n} \left[\left(w_j + \sum_{J_k \in \mathscr{S}_j} \lambda_{jk} - \sum_{J_k \in \mathscr{P}_j} \lambda_{kj} \right) C_j + \sum_{J_k \in \mathscr{S}_j} \lambda_{jk} p_k \right]$$

subject to the machine capacity and availability conditions (1) and (2).

For $j = 1, \ldots, n$, let $w_j'(\lambda) = (w_j + \Sigma_{J_k \in S_j} \lambda_{jk} - \Sigma_{J_k \in P_j} \lambda_{kj})/p_j$; we call $w_j'(\lambda)$ the *relative weight* of job J_j. Using Smith's ratio rule, we solve problem (L_λ) by sequencing the jobs in order of nonincreasing relative weights. From standard Lagrangian theory, we know that $L(\lambda)$ is a lower bound for the $1|\text{prec}|\Sigma w_j C_j$ problem for any $\lambda \geqslant 0$. In this respect, we like to find the vector λ^* that induces the best Lagrangian lower bound. This is the Lagrangian dual problem of $1|\text{prec}|\Sigma w_j C_j$, referred to as problem (D):

$$\begin{aligned} &\text{maximize} && L(\lambda) \\ &\text{subject to} && \lambda_{jk} \geqslant 0 \text{ for each } (J_j, J_k) \in A. \end{aligned}$$

Theorem 2. *Problem* (D) *is solvable in time polynomial in n through the ellipsoid method.*

For a proof, see [37].

In practice, the ellipsoid method is too slow to be of use. We develop a quick ascent direction algorithm to approximate the optimal solution of problem (D). The notion of *directional derivative* plays a central role in ascent direction algorithms. The directional derivative of the function L at λ is defined as

$$L_u(\lambda) = \lim_{\epsilon \downarrow 0} \frac{L(\lambda + \epsilon u) - L(\lambda)}{\epsilon}$$

for any vector $u \in \mathbb{R}^A$. Hence, λ is optimal if and only if

$$L_u(\lambda) \leqslant 0, \quad \text{for all } u \in \mathbb{R}^A.$$

If $L_{\bar{u}}(\lambda) > 0$ for some $\bar{u} \in \mathbb{R}^A$, then $\bar{u}$ is called an *ascent direction* of L at λ: we get an improved lower bound by moving some scalar step size Δ along $\bar{u}$. In general, it is difficult to compute directional derivatives. However, it is easy to compute them for the *primitive* vectors. A vector u is called primitive if $u_{jk} = 0$ for all (J_j, J_k) but one. Hence, there are at most $2|A|$ different primitive directional derivatives at any λ.

First, we derive expressions for the primitive directional derivatives. In an optimal solution for the Lagrangian problem, the position of J_j depends on its relative weight: the larger its relative weight, the smaller its completion time. If its weight is tied, then its position also depends on the way ties are settled. Let $C_j^+(\lambda)$ denote the earliest possible completion time of J_j in an optimal schedule for problem (L_λ); let $C_j^-(\lambda)$ denote the latest possible completion time of J_j in an optimal schedule for problem (L_λ). Increasing λ_{jk} by a specific $\epsilon > 0$ will increase the relative weight of J_j from $w_j'(\lambda)$ to $w_j'(\lambda) + \epsilon/p_j$; simultaneously, it will decrease the relative weight of J_k from $w_k'(\lambda)$ to $w_k'(\lambda) - \epsilon/p_k$. It is possible to choose $\epsilon > 0$ small enough to ensure that at least one optimal schedule for problem (L_λ) remains optimal (see [37]). In such an optimal schedule, J_j must be completed on time $C_j^+(\lambda)$ and J_k must be completed on time $C_k^-(\lambda)$. Increasing λ_{jk} by such a small ϵ affects the Lagrangian objective value by $\epsilon[C_j^+(\lambda) - C_k^-(\lambda) + p_k]$. From this, we derive that the primitive directional derivative for increasing λ_{jk} at λ, denoted by $l_{jk}^+(\lambda)$, is

$$l_{jk}^+(\lambda) = C_j^+(\lambda) - C_k^-(\lambda) + p_k \quad \text{for each } (J_j, J_k) \in A.$$

If $l_{jk}^+(\lambda) > 0$, then increasing λ_{jk} is an ascent direction: we get an improved objective value by moving along this direction. The sign of each $l_{jk}^+(\lambda)$ is determined in constant time. Note that for each arc $(J_j, J_k) \in A$, we have

$$C_j^+(\lambda) > C_k^-(\lambda) + p_k \Leftrightarrow w_j'(\lambda) < w_k'(\lambda),$$

hence, $l_{jk}^+(\lambda) > 0 \Leftrightarrow w_j'(\lambda) < w_k'(\lambda)$. In a similar fashion, we find that $l_{jk}^-(\lambda)$, the primitive directional derivative for decreasing λ_{jk} at each λ with $\lambda_{jk} > 0$, is

$$l_{jk}^-(\lambda) = C_k^+(\lambda) - C_j^-(\lambda) - p_k \quad \text{for each } (J_j, J_k) \in A.$$

If $l_{jk}^-(\lambda) > 0$, then decreasing λ_{jk} is an ascent direction: we get an improved objective value by moving along this direction.

Given an ascent direction, we invariably move by the step size that maximizes the increment to the objective value. If $l_{jk}^+(\lambda) > 0$, then the increment is maximized by moving to the first point where increasing λ_{jk} is no longer an ascent direction. At this point, the relative weights of J_j and J_k are equal. Hence, the required step size is the value Δ for which

$$w_j'(\lambda) + \Delta/p_j = w_k'(\lambda) - \Delta/p_k,$$

it is determined in constant time. Consider now the case $l_{jk}^-(\lambda) > 0$. To ensure that the Lagrangian vector remains nonnegative, we impose the condition that $\Delta \leqslant \lambda_{jk}$. If this condition is not restrictive, then we move to the first point where decreasing λ_{jk} is no longer an ascent direction. If it is restrictive, then we take the step size as large as possible. Hence, the step size that maximizes the increment of the objective value is computed as the largest value $\Delta \leqslant \lambda_{jk}$ for which

$$w_j'(\lambda) - \Delta/p_j \geqslant w_k'(\lambda) + \Delta/p_k.$$

Eventually, termination occurs at some $\bar{\lambda}$ at which no ascent direction exists any more. Later on, we will analyze the termination conditions. We first give a step-wise description of the ascent direction algorithm.

Ascent Direction Algorithm

Step 0. Set $\lambda_{jk} = 0$ for each $(J_j, J_k) \in A$, and compute the relative weights $w'_j(\lambda)$.

Step 1. For each $(J_j, J_k) \in A$, do the following:

(a) If $w'_j(\lambda) < w'_k(\lambda)$, then compute the step size

$$\Delta = \left[w'_k(\lambda) - w'_j(\lambda)\right] p_j p_k/(p_j + p_k).$$

Put $\lambda_{jk} \leftarrow \lambda_{jk} + \Delta$, and update $w'_j(\lambda)$ and $w'_k(\lambda)$.

(b) If $w'_j(\lambda) > w'_k(\lambda)$, then compute the step size

$$\Delta = \min\left\{\lambda_{jk}, \left[w'_j(\lambda) - w'_k(\lambda)\right] p_j p_k/(p_j + p_k)\right\}.$$

Put $\lambda_{jk} \leftarrow \lambda_{jk} - \Delta$, and update $w'_j(\lambda)$ and $w'_k(\lambda)$.

Step 2. If no multiplier adjustment has taken place, then compute $L(\lambda)$ and stop; otherwise, return to Step 1.

Let I be the number of times that Step 1 is executed. The ascent direction algorithm runs then in $O(I\,|A| + n \log n)$ time. Since we cannot bound I by a polynomial in n and $|A|$, the ascent direction algorithm is presumably not a polynomial-time method. In practice, however, the algorithm is very fast and produces very good approximate solutions.

Theorem 3. *The ascent direction algorithm described above generates a series of monotonically increasing lower bounds for problem* (P).

Proof. Given an arbitrary $\lambda \geqslant 0$, we first assume $w'_j(\lambda) < w'_k(\lambda)$; hence, $l^+_{jk}(\lambda) = C^+_j(\lambda) - C^-_k(\lambda) + p_k > 0$, and increasing λ_{jk} is an ascent direction. We reindex the jobs according to nonincreasing relative weights, settling all ties arbitrarily except for J_j and J_k: we give J_j the smallest index possible and J_k the largest index possible. Let $C_1, \ldots, C_n$ be the job completion times for the sequence $(J_1, \ldots, J_n)$; note that $C_j = C^+_j(\lambda)$ and $C_k = C^-_k(\lambda)$. Hence, in more detail, the schedule under consideration is $(J_1, \ldots, J_{k-1}, J_k, J_{k+1}, \ldots, J_{j-1}\ J_j, J_{j+1}, \ldots, J_n)$. Let Δ be the step size as dictated, and let $\bar\lambda$ denote the vector of Lagrangian multipliers after increase of λ_{jk} by Δ. Since λ and $\bar\lambda$ differ only in one component, the relative weights for all jobs but J_j and J_k remain the same. An optimal schedule for problem $(\mathrm{L}_{\bar\lambda})$ is then $(J_1, \ldots, J_{k-1}, J_{k+1}, \ldots, J_l, J_j, J_k, J_{l+1}, \ldots, J_{j-1}, J_{j+1}, \ldots, J_n)$, for some J_l with $k+1 \leqslant l \leqslant j-1$; the job completion times for this schedule can conveniently be expressed in terms of $C_1, \ldots, C_n$. We now demonstrate that $L(\bar\lambda) > L(\lambda)$; it is essentially a matter of writing out. For brevity, we let $\mu_i = w_i + \sum_{J_h \in \mathscr{S}_i} \lambda_{ih} - \sum_{J_h \in \mathscr{P}_i} \lambda_{hi}$ for each i $(i = 1, \ldots, n)$. We have

$$\begin{aligned} L(\bar\lambda) = {} & \sum_{i=1}^{k-1} \mu_i C_i + \sum_{i=j+1}^{n} \mu_i C_i + \sum_{i=k+1}^{l} \mu_i (C_i - p_k) + \sum_{i=l+1}^{j-1} \mu_i (C_i + p_j) \\ & + (\mu_k - \Delta)\left[C^-_k(\lambda) + p_j + \sum_{i=k+1}^{l} p_i\right] \\ & + (\mu_j + \Delta)\left[C^+_j(\lambda) - p_k - \sum_{i=l+1}^{j-1} p_i\right] \end{aligned}$$

$$+\sum_{i=1}^{n}\sum_{J_h\in\mathscr{S}_i}\lambda_{ih}p_h+\Delta p_k$$

$$=L(\lambda)+\sum_{i=l+1}^{j-1}(\mu_ip_j-\mu_jp_i)+\sum_{i=k+1}^{l}(\mu_kp_i-\mu_ip_k)+\mu_kp_j-\mu_jp_k$$

$$+\Delta\left[\left(C_j^+(\lambda)-p_k-\sum_{i=l+1}^{j-1}p_i\right)-\left(C_k^-(\lambda)+p_j+\sum_{i=k+1}^{l}p_i\right)\right]+\Delta p_k.$$

Since J_j and J_k are adjacent in the second schedule, we have that

$$\left[C_j^+(\lambda)-p_k-\sum_{i=l+1}^{j-1}p_i\right]-\left[C_k^-(\lambda)+p_j+\sum_{i=k+1}^{l}p_i\right]=-p_k.$$

This implies that

$$L(\bar{\lambda})=L(\lambda)+\sum_{i=l+1}^{j-1}(\mu_ip_j-\mu_jp_i)+\sum_{i=k+1}^{l}(\mu_kp_i-\mu_ip_k)+\mu_kp_j-\mu_jp_k$$

$$=L(\lambda)+\sum_{i=k+1}^{l}(\mu_k/p_k-\mu_i/p_i)p_ip_k+\sum_{i=l+1}^{j-1}(\mu_i/p_i-\mu_j/p_j)p_ip_j$$

$$+(\mu_k/p_k-\mu_j/p_j)p_jp_k$$

$$=L(\lambda)+\sum_{i=k+1}^{l}\left[w_k'(\lambda)-w_i'(\lambda)\right]p_ip_k+\sum_{i=l+1}^{j-1}\left[w_i'(\lambda)-w_j'(\lambda)\right]p_ip_j$$

$$+\left[w_k'(\lambda)-w_j'(\lambda)\right]p_jp_k.$$

Since $w_k'(\lambda)>w_j'(\lambda)$, $w_i'(\lambda)<w_k'(\lambda)$ for each i $(i=k+1,\dots,l)$, and $w_i'(\lambda)>w_j'(\lambda)$ for each i $(i=l+1,\dots,j-1)$, we find that $L(\bar{\lambda})>L(\lambda)$.

The analysis for the case $l_{jk}^-(\lambda)>0$ proceeds in a similar fashion. □

Consider the 10-job example from Potts [26] for which the processing times, weights, and precedence graph are given in Table 1 and Fig. 1.

If we put $\lambda_{jk}=0$ for all $(J_j, J_k)\in A$, then an optimal schedule is $(J_3, J_{10}, J_4, J_9, J_7, J_6, J_2, J_5, J_8, J_1)$ with total cost 1055. The same schedule and lower bound are obtained by disregarding the precedence constraints and solving $1\,\|\,\Sigma w_jC_j$. The schedule is not feasible for the original problem; for instance, J_{10} is executed before J_6 although $(J_6, J_{10})\in A$. Since $w_6'(\lambda)<w_{10}'(\lambda)$, increasing $\lambda_{6,10}$ is an ascent direction. The

Table 1
Processing times and weights

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
p_j	6	9	1	3	9	5	7	7	6	2
w_j	2	5	9	6	5	4	9	3	8	5

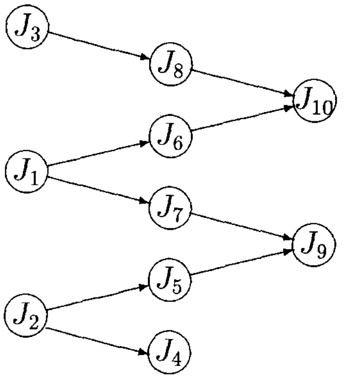

Fig. 1. Precedence graph.

appropriate step size is $\Delta = \frac{17}{7}$, giving $\lambda_{6,10} = \frac{17}{7}$. We have that $(J_3, J_4, J_9, J_7, J_6, J_{10}, J_2, J_5, J_8, J_1)$ is an optimal schedule for the new Lagrangian problem with value $L(\lambda) = 1106$. Proceeding along these lines, we get the value $L(\lambda) = 1526.69$ upon termination. Potts' procedure, requiring $\Omega(n^4)$ time, proceduces the lower bound 1519; the upper bound generated by the tree-optimal heuristic is 1530. The duality gap is therefore no more than 3.

In the remainder, we let $\bar{\lambda}$ denote the vector of Lagrangian multipliers upon termination of the ascent direction method. Using the termination conditions that all primitive directional derivatives are nonpositive, we derive some properties for $\bar{\lambda}$ and for the optimal solutions of problem $(\mathrm{L}_{\bar{\lambda}})$. These properties are important for the development of approximation and optimization algorithms for $1 \,|\, \mathrm{prec} \,|\, \Sigma w_j C_j$.

Definition 1. The job set $\mathscr{B} \subseteq \mathscr{J}$ is called a *block* for a given $\lambda \geqslant 0$ if

$$w_j'(\lambda) = c \quad \text{for each } J_j \in \mathscr{B},$$

where c is some positive real constant.

In any optimal schedule to problem (L_λ), the jobs in a block are interchangeable without affecting the Lagrangian objective value $L(\lambda)$. For any given $\lambda \geqslant 0$, the job set $\mathscr{J}$ is decomposed into $B(\lambda)$ blocks $\mathscr{B}_1, \ldots, \mathscr{B}_{B(\lambda)}$, indexed such that

$$w_j'(\lambda) = c_b \quad \text{for each } J_j \in \mathscr{B}_b,\ b = 1, \ldots, B(\lambda),$$

with $c_1 > \cdots > c_{B(\lambda)} > 0$.

Theorem 4. *Any vector λ satisfying the termination conditions induces a decomposition of $\mathscr{J}$ into $B(\lambda)$ blocks $\mathscr{B}_1, \ldots, \mathscr{B}_{B(\lambda)}$ such that, if $(J_j, J_k) \in A$ and $J_k \in \mathscr{B}_b$, then*

$$J_j \in \mathscr{B}_1 \cup \cdots \cup \mathscr{B}_b,$$

and

$$\lambda_{jk} = 0 \quad \text{if } J_j \in \mathscr{B}_1 \cup \cdots \cup \mathscr{B}_{b-1}.$$

Proof. If one of these claims were not true, then we could identify an ascent direction, contradicting the assumption that the termination conditions are satisfied. □

We call a decomposition induced by a vector that satisfies the termination conditions a *dual decomposition.* Both λ^* and $\bar{\lambda}$ induce dual decompositions. For our example, the dual decomposition induced by $\bar{\lambda}$ consists of three blocks: $\mathscr{B}_1 = \{J_3\}$, $\mathscr{B}_2 = \{J_2, J_4\}$, and $\mathscr{B}_3 = \{J_1, J_5, J_6, J_7, J_8, J_9, J_{10}\}$, with $c_1 = 9$, $c_2 = \frac{11}{12}$, and $c_3 = \frac{6}{7}$, respectively.

3. Approximation

We present an approximation algorithm that exploits the agreeable structure of the dual decomposition induced by $\bar{\lambda}$. For $b = 1, \ldots, B_{\bar{\lambda}}$, let σ_b be a feasible sequence for the jobs in $\mathscr{B}_b$. From Theorem 4, we derive the following.

Corollary 1. *The sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{B(\bar{\lambda})})$ is feasible for the original problem.*

If each σ_b is optimal for the $1|\text{prec}|\Sigma_{J_j \in \mathscr{B}_b} w_j C_j$ problem ($b = 1, \ldots, B(\bar{\lambda})$), then we have the best such σ. From a theoretical point of view, each $1|\text{prec}|\Sigma_{J_j \in \mathscr{B}_b} w_j C_j$ problem is as hard as the original problem; from a practical point of view, each problem is simpler, because it is of a smaller dimension. Dynamic programming in tandem with a compact labeling scheme [23,32] solves small instances quickly. If the size of a block is too large for the application of dynamic programming, then we resort to the tree-optimal heuristic to find an approximate solution. However, even if the dual decomposition is induced by λ^* and σ is composed of optimal subsequences, then we still have no guarantee that σ is an optimal sequence; all optimal sequences may have been excluded by the dual decomposition.

For the example, the optimal sequences for the first two blocks are trivial: $\sigma_1 = (J_3)$, and $\sigma_2 = (J_2, J_4)$; using dynamic programming, we find $\sigma_3 = (J_1, J_7, J_5, J_9, J_6, J_8, J_{10})$; the tree-optimal heuristic gives the same sequence. We obtain $\sigma = (J_3, J_2, J_4, J_1, J_7, J_5, J_9, J_6, J_8, J_{10})$ with total cost 1530.

We tested the approximation algorithm and the tree-optimal heuristic on problems with 20, 30, ..., 100 jobs. The processing times were drawn from the uniform distribution [1, 100]; the weights were generated from the uniform distribution [1, 10]. The precedence graph was induced by the probability P with which each arc (J_j, J_k) with $j < k$ was included. The graph obtained in this way was then stripped down to its transitive reduction. We generated problems for $P = 0.01$, 0.02, 0.04, 0.06, 0.08, 0.10, 0.15, 0.20, 0.30, and 0.50. For each combination of n and P we generated five problems; hence, 45 problems were generated for each value P. This procedure parallels Potts' procedure to generate instances. Furthermore, we solved each subproblem to optimality if less than 15000 labels were needed; otherwise, we used the tree-optimal heuristic.

Table 2
Experimental results; for each value of P and for either approximation algorithm, we present the average value (upper bound)/$L(\bar{\lambda})$ and the number of times out of 45 that the upper bound equaled the lower bound

P	Tree-optimal heuristic		Dual decomposition	
	UB/$L(\bar{\lambda})$	# opt	UB/$L(\bar{\lambda})$	# opt
0.01	1.00007	42	1.00007	42
0.02	1.00074	15	1.00069	15
0.04	1.00516	8	1.00248	10
0.06	1.01122	2	1.00584	2
0.08	1.01303	1	1.00934	1
0.10	1.01731	2	1.01211	4
0.15	1.01993	0	1.01765	0
0.20	1.02050	0	1.01614	2
0.30	1.02447	0	1.02075	2
0.50	1.02831	2	1.02616	3

Potts points out that the relative difficulty of an instance depends on $|A|$ rather than on n. We have therefore classified the results according to the value P. For each P, we present the average value (upper bound)/$L(\bar{\lambda})$ for both approaches (see Table 2). The columns "# opt" indicate for how many problems out of 45 the upper bound equaled the lower bound; this figure gives the number of times we found a provably optimal solution. On the average, the dual-decomposition algorithm outperforms the tree-optimal heuristic approach for any problem class. For the 450 instances altogether, the tree-optimal heuristic produced only 16 better solutions; moreover, each of these was only marginally better.

We have coded both algorithms in the computer language C; all experiments were conducted on a Compaq-386/20 Personal Computer. The time to compute $L(\bar{\lambda})$ is virtually negligible. The tree-optimal heuristic requires $O(n|A|)$ time and is sensitive to instances with many precedence constraints. The running time of the dual decomposition approximation algorithm mainly depends on the number of calls on the dynamic programming procedure and the maximum label number. For $n \leqslant 40$, the tree-optimal heuristic needed a few seconds at most. On the average, our approximation required only slightly more time; there were occasional peaks, however, due to high labels in the dynamic programming subroutine. For $n \geqslant 60$, the tree-optimal heuristic needs about twice or three times as much computation time as the dual decomposition algorithm although it needs never more than 1 minute; even the peaks of the dual decomposition algorithm remain then below the average of the tree-optimal heuristic.

Potts also points out that small and large values of P generate relatively easy problems. For small P, only few precedence constraints are involved; for large P, most disjunctive constraints are settled. Our results support the claim for small P: the duality gap is very small. Since the optimal-tree heuristic generates good approximate solutions for all values of P [26], there are two possible explanations for the growth of the gap between upper and lower bound for larger values of P. It may be that the ascent direction method produces worse approximate solutions in case P is large; it is more likely, however, that the duality gap is an increasing function of P.

4. Primal decomposition

For $b = 1, \dots, B$, let σ_b^* denote an optimal sequence for the problem $1|\text{prec}|\Sigma_{J_j \in \mathscr{J}_b} w_j C_j$, where $\mathscr{J}_b \subseteq \mathscr{J}$. A decomposition of the job set $\mathscr{J}$ into B mutually disjoint subsets $\mathscr{J}_1, \dots, \mathscr{J}_B$ is said to be a *primal decomposition* if the sequence $\sigma = \sigma_1^*, \dots, \sigma_B^*$ is optimal for the original $1|\text{prec}|\Sigma w_j C_j$ problem. We already mentioned that a dual decomposition may exclude all optimal sequences; a dual decomposition only suggests a primal decomposition. In this section, we try to establish to what extent a dual decomposition coincides with a primal decomposition.

If a dual decomposition excludes all optimal solutions, then there are at least two jobs belonging to different blocks with no path in A between them for which the processing order should be reversed. Suppose $J_j \in \mathscr{B}_b$ and $J_k \in \mathscr{B}_{b+m}$ ($m > 0$) are such jobs. In all feasible sequences obtained by the dual decomposition approach, J_j precedes J_k; but in all optimal sequences, J_k precedes J_j. Hence, the arc (J_k, J_j) can be added to the arc set A with impunity. Let problem $(\text{L}_\lambda(k, j))$ be the Lagrangian problem for the arc set $A \cup (J_k, J_j)$, and let $\lambda(k, j) \geqslant 0$ be a vector of Lagrangian multipliers. Since the arc (J_k, J_j) does not exclude the optimal solution, $L(\lambda(k, j))$ is still a lower bound on the optimal solution, for any $\lambda(k, j) \geqslant 0$.

This observation gives rise to the following result. Let $\mathscr{B}_1, \dots, \mathscr{B}_B$ be the blocks of some dual decomposition, and let UB be an upper bound for the $1|\text{prec}|\Sigma_{j=1}^n w_j C_j$ problem.

Theorem 5. *If there are two jobs $J_j \in \mathscr{B}_b$ and $J_k \in \mathscr{B}_{b+m}$ ($b = 1, \dots, B-1$, $m = 1, \dots, B-b$) with no path in A between them for which*

$$L(\lambda(k, j)) > \text{UB} - 1, \tag{4}$$

then J_j precedes J_k in any optimal solution for the $1|\text{prec}|\Sigma w_j C_j$ *problem.*

If (4) holds for all such J_j and J_k, then the decomposition is a primal decomposition; in fact, due to transitivity, it is sufficient that (4) holds for specific J_j and J_k only.

Corollary 2. *If for each pair of jobs $J_j \in \mathscr{B}_b$ and $J_k \in \mathscr{B}_{b+m}$ ($b = 1, \dots, B-1$, $m = 1, \dots, B-b$) such that*

(1) *there is no path in A from J_j to J_k,*

(2) *J_j has no successors in $\mathscr{B}_b \cup \cdots \cup \mathscr{B}_{b+m-1}$, and*

(3) *J_k has no predecessors in $\mathscr{B}_{b+1} \cup \cdots \cup \mathscr{B}_{b+m}$,*

we have that $L(\lambda(k, j)) > \text{UB} - 1$ for some $\lambda(k, j) \geqslant 0$, then the dual decomposition is a primal decomposition.

Accordingly, if the dual decomposition induces a primal decomposition and if UB is associated with the sequence that is composed of optimal subsequences, then UB is the optimal solution value of the $1|\text{prec}|\Sigma w_j C_j$ problem.

Corollary 3. *If for some block $\mathscr{B}_b$, each pair of jobs $J_j \in \mathscr{B}_b$ and $J_k \in \mathscr{B}_{b+m}$ ($m = 1, \dots, B - b$) such that*

(1) *there is no path in A from J_j to J_k,*

(2) *J_j has no successors in $\mathscr{B}_b \cup \cdots \cup \mathscr{B}_{b+m-1}$, and*

(3) *J_k has no predecessors in $\mathscr{B}_{b+1} \cup \cdots \cup \mathscr{B}_{b+m}$,*

satisfies $L(\lambda(k, j)) > \mathrm{UB} - 1$ for some $\lambda(k, j) \geqslant 0$, then the subsets $\mathscr{B}_1 \cup \cdots \cup \mathscr{B}_b$ and $\mathscr{B}_{b+1} \cup \cdots \cup \mathscr{B}_B$ constitute a primal decomposition of $\mathscr{J}$.

In this case, we say that the dual decomposition *partly* concurs with a primal decomposition. Whether we succeed to establish that a dual decomposition partly or completely concurs with a primal decomposition depends on the quality of the lower bounds $L(\lambda(k, j))$. From this point of view, we like to have available the vector of optimal Lagrangian multipliers for problem $(\mathrm{L}_\lambda(k, j))$; let $\lambda^*(k, j)$ denote this vector. Of course, $\lambda^*(k, j)$ is as difficult to find as the vector λ^*. However, an ascent direction method to approximate $\lambda^*(k, j)$ is readily available: we apply the direction method for problem (D), adjusted for the additional arc (J_k, J_j), using as initial vector $\lambda(k, j)^{(0)}$ obtained as $\lambda(k, j)^{(0)}_{ih} = \bar{\lambda}_{ih}$ for each $(J_i, J_h) \in A$ and $\lambda(k, j)^{(0)}_{kj} = 0$. We note that $L(\lambda(k, j)^{(0)}) = L(\bar{\lambda})$. At $\lambda(k, j)^{(0)}$, all primitive directional derivatives are nonpositive but one: we have $l^+_{kj}(\lambda(k, j)^{(0)}) > 0$; increasing $\lambda(k, j)_{kj}$ is an ascent direction. If J_j and J_k belong to blocks that are far apart from each other, then the Lagrangian lower bound corresponding with the point where the sign of this directional derivative changes may already exceed UB − 1. This Lagrangian lower bound is conveniently computed; this is stipulated in the next theorem, where $p(\mathscr{B}_b)$ is defined as $p(\mathscr{B}_b) = \sum_{J_i \in \mathscr{B}_b} p_i$.

Theorem 6. *If there are two jobs $J_j \in \mathscr{B}_b$ and $J_k \in \mathscr{B}_{b+m}$ ($b = 1, \dots, B(\bar{\lambda}) - 1$, $m = 1, \dots, B(\bar{\lambda}) - b$) for which there is no path in A from J_j to J_k such that*

$$L(\bar{\lambda}) + (c_b - c_{b+m})p_j p_k + \sum_{i=b+1}^{l} (c_b - c_i) p(\mathscr{B}_i) p_j + \sum_{i=l+1}^{b+m-1} (c_i - c_{b+m}) p(\mathscr{B}_i) p_k \tag{5}$$

exceeds UB − 1, *where l is the largest index with $c_l \geqslant (p_j c_b + p_k c_{b+m})/(p_k + p_j)$, then J_j precedes J_k in all optimal solutions for the $1|\mathrm{prec}|\sum w_j C_j$ problem.*

Proof. The validation of this proposition requires the same logic applied in the proof of Theorem 3. □

If (5) does not hold, then we run the ascent direction algorithm until no ascent directions can be found any more; upon termination, we get the vector $\bar{\lambda}(k, j)$.

We now work out the effects of these propositions on our example. According to Corollary 3, we need consider only the pairs (J_3, J_2) and (J_3, J_4) in order to decompose the jobs into $\mathscr{B}_1$ on the one hand and $\mathscr{B}_2 \cup \mathscr{B}_3$ on the other hand. Including (J_2, J_3) in A, we get, using (5), that $L(\lambda^*) + (c_1 - c_2)p_2 p_3 = 1526.69 + (9 - \frac{11}{12}) \times 1$

$\times 9 > 1529$; we conclude that J_3 precedes J_2 in any schedule with cost less than 1530. Similarly, if we include (J_4, J_3) in A, then, according to Theorem 6 (where we have $l = 1$), we must verify if

$$L(\bar{\lambda}) + (c_1 - c_3)p_3 p_4 + (c_2 - c_3)p(\mathscr{B}_2)p_4 > \mathrm{UB} - 1.$$

This is so; hence, J_3 must precede J_4, implying that we may decompose the job set into subsets $\mathscr{B}_1$ and $\mathscr{B}_2 \cup \mathscr{B}_3$. We must consider the pairs (J_4, J_1), (J_4, J_5), and (J_4, J_8) to separate the blocks $\mathscr{B}_2$ and $\mathscr{B}_3$. More than one iteration in the ascent direction procedure is required. Since $L(\bar{\lambda}(1, 4))$, $L(\bar{\lambda}(5, 4))$, and $L(\bar{\lambda}(8, 4))$ exceed 1529, we conclude that the dual decomposition concurs with a primal decomposition. Furthermore, the schedule with value 1530 is optimal, since it was obtained from the optimal sequences for the individual blocks.

The theorems and corollaries presented in this section are applicable in a preprocessing phase in conjunction with any existing branch-and-bound algorithm. Their main purpose is to derive additional precedence constraints and to decompose the problem primally in order to reduce the size of the search tree.

5. Conclusions

We have developed a fast ascent direction algorithm for a scheduling problem formulated in terms of the job completion times. We also have shown that upon termination of the algorithm we get a dual decomposition of the original problem, which can be used for designing approximative and enumerative algorithms.

We claim that about any scheduling problem for which the Lagrangian relaxation problem reduces to an analog of problem (P), that is, a problem solvable by Smith's ratio rule, can be dealt with in the same spirit. For more details, we refer to [37]. The strength of the lower bound and the effectiveness of the dual decomposition approach, however, depend on the structure of the problem and the nature of the dualized constraints.

Acknowledgements

The author likes to thank Bert Gerards, Han Hoogeveen and Jan Karel Lenstra for their helpful suggestions.

References

[1] D. Adolphson and T.C. Hu, "Optimal linear ordering," *SIAM Journal of Applied Mathematics* 25 (1973) 403–423.

[2] E. Balas and N. Christofides, "A restricted Lagrangean approach to the traveling salesman problem," *Mathematical Programming* 21 (1981) 19–46.

[3] O. Bilde and S. Krarup, "Sharp lower bounds and efficient algorithms for the simple plant location problem," *Annals of Discrete Mathematics* 1 (1977) 79–88.
[4] N. Christofides, "The shortest hamiltonian chain of a graph," *SIAM Journal of Applied Mathematics* 19 (1970) 689–696.
[5] 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 (1990) 255–270.
[6] D. Erlenkotter, "A dual-based procedure for uncapacitated facility location," *Operations Research* 21 (1978) 1114–1127.
[7] M.L. Fisher, "A dual algorithm for the one-machine scheduling problem," *Mathematical Programming* 11 (1976) 229–251.
[8] M.L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Management Science* 27 (1981) 1–18.
[9] M.L. Fisher, "An application oriented guide to Lagrangian relaxation," *Interfaces* 15 (1985) 10–21.
[10] M.L. Fisher, R. Jaikumar and L.N. van Wassenhove, "A multiplier adjustment method for the generalized assignment problem," *Management Science* 32 (1986) 1098–1103.
[11] M.L. Fisher and P. Kedia, "Optimal solution of set covering/partitioning problems using dual heuristics," *Management Science* 36 (1990) 674–688.
[12] A.M. Geoffrion, "Lagrangian relaxation and its uses in integer programming," *Mathematical Programming Study* 2 (1974) 82–114.
[13] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics* 5 (1979) 287–326.
[14] M. Guignard and K. Spielberg, "A direct dual method of the mixed plant location problem with some side constraints," *Mathematical Programming* 17 (1979) 198–228.
[15] A.M.A. Hariri and C.N. Potts, "An algorithm for single-machine sequencing with release dates to minimize total weighted completion time," *Discrete Applied Mathematics* 5 (1983) 99–109.
[16] A.M.A. Hariri and C.N. Potts, "Algorithms for two-machine flow-shop sequencing with precedence constraints," *European Journal of Operational Research* 17 (1984) 238–248.
[17] M. Held and R.M. Karp, "The traveling salesman problem and minimum spanning trees," *Operations Research* 18 (1970) 1138–1162.
[18] M. Held and R.M. Karp, "The traveling salesman problem and minimum spanning trees: Part II," *Mathematical Programming* 1 (1971) 6–25.
[19] M. Held, P. Wolfe and H. Crowder, "Validation of subgradient optimization," *Mathematical Programming* 6 (1974) 62–88.
[20] J.A. Hoogeveen and S.L. van de Velde, "Stronger Lagrangian bounds by use of slack variables: applications to machine scheduling problems," *Mathematical Programming* 70 (1995), to appear.
[21] W.A. Horn, "Single-machine job sequencing with treelike precedence ordering and linear delay penalties," *SIAM Journal of Applied Mathematics* 23 (1972) 189–202.
[22] E.L. Lawler, "Sequencing jobs to minimize total weighted completion time subject to precedence constraints," *Annals of Discrete Mathematics* 2 (1978) 75–90.
[23] E.L. Lawler, "Efficient implementation of dynamic programming algorithms for sequencing problems," Report BW 106, Centre for Mathematics and Computer Science (Amsterdam, 1979).
[24] J.K. Lenstra and A.H.G. Rinnooy Kan, "Complexity of scheduling under precedence constraints," *Operations Research* 26 (1978) 22–35.
[25] T.E. Morton and B.G. Dharan, "Algoristics for single-machine sequencing with precedence constraints," *Management Science* 24 (1978) 1011–1020.
[26] C.N. Potts, "A Lagrangean based branch-and-bound algorithm for single machine sequencing with precedence constraints to minimize total weighted completion time," *Management Science* 31 (1985) 1300–1311.
[27] C.N. Potts and L.N. van Wassenhove, "An algorithm for single-machine sequencing with deadlines to minimize total weighted completion time," *European Journal of Operational Research* 33 (1983) 363–377.
[28] C.N. Potts and L.N. van Wassenhove, "A branch and bound algorithm for the total weighted tardiness problem," *Operations Research* 33 (1985) 363–377.
[29] M. Queyranne and A.S. Schulz, "Polyhedral approaches to machine scheduling," Working paper, Technische Universität Berlin (Berlin, 1994).

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

[31] M. Queyranne and Y. Wang, "A cutting plane procedure for precedence constrained single-machine scheduling," Working paper, University of British Columbia (Vancouver, 1991).

[32] L. Schrage and K.R. Baker, "Dynamic programming solution of sequencing problems with precedence constraints," *Operations Research* 26 (1978) 444–449.

[33] J.F. Shapiro, "A survey of Lagrangian techniques for discrete optimization," *Annals of Discrete Mathematics* 5 (1974) 113–138.

[34] J.B. Sidney, "Decomposition algorithms for single-machine sequencing with predecence relations and deferral costs," *Operations Research* 23 (1975) 283–298.

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

[36] S.L. van de Velde, "Minimizing the sum of the job completion times in the two-machine flow shop by Lagrangian relaxation," *Annals of Operations Research* 26 (1990) 257–268.

[37] S.L. van de Velde, "Machine scheduling and Lagrangian relaxation," Ph.D. Thesis, CWI (Amsterdam, 1991).

[38] L.A. Wolsey, "Formulating single-machine scheduling polyhedra with precedence constraints," in: J. Gabszewicsz, J.-F. Richard and L.A. Wolsey, eds., *Economic Decision-Making: Games, Econometrics and Optimization* (North-Holland, Amsterdam, 1990).