S.L. van de Velde

Dual decomposition of single-machine scheduling problems

Department of Operations Research, Statistics, and System Theory Report BS-R9009 May

Dual Decomposition of Single-Machine Scheduling Problems

S.L. van de Velde
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam
The Netherlands

We propose a framework for Lagrangian duality theory for single-machine scheduling with minsum objectives, which induces a dual decomposition of such problems. The dual decomposition method offers promising opportunities for the design of approximative and enumerative algorithms. We develop the method through an analysis of the problem of minimizing the total weighted completion time subject to precedence constraints.

1980 Mathematics Subject Classification (Revision 1985): 90B35.
Keywords & Phrases: single-machine scheduling, Lagrangian relaxation, Lagrangian dual, dual decomposition, dynamic programming.
Note: This paper has been submitted for publication.

1. Introduction

In the 1980's we have witnessed successes of polyhedral combinatorics for a variety of combinatorial optimization problems. The traveling salesman problem is the subject of several such success stories (see e.g. Lawler et al. 1985). Machine scheduling is a recently discovered challenge for polyhedral techniques (see e.g. Balas 1985, Queyranne and Wang 1988). In view of this, there is renewed interest in the formulation of machine scheduling problems in terms of 0-1 programming. However, the number of variables and constraints needed is a heavy burden for bringing polyhedral techniques into effect. In spite of the initial optimism, polyhedral techniques have not yet been shown to be effective, let alone efficient, beyond the simplest and smallest single-machine scheduling problems.

Lagrangian relaxation, on the other hand, is a conventional technique for lower bound computation, that has proved its merits for many types of combinatorial optimization problems. An excellent introduction to Lagrangian relaxation theory and a survey of its applications has been given by Fisher (1981). In the area of machine scheduling, however, Lagrangian relaxation theory is still underdeveloped. The few publications fall into two classes, according to the mathematical formulation applied for the disjunctive constraints that reflect the capacities of the machines.

The first category covers machine scheduling problems that formulate the disjunctive constraints *explicitly*. These applications, however, suffer severely from the number of variables and constraints. Fisher (1976), for instance, introduces a pseudo-polynomial number of constraints that require that the machine processes at most one job in each unit-time interval. Hariri and Potts (1984) and Potts (1985) use decision variables x_{jk} that take the value 1 if job

J_j completes before job J_k, of which there are $O(n^2)$ if there are n jobs involved. In this case, however, there are $O(n^3)$ transitivity constraints needed. On the other hand, the Lagrangian relaxation of each of these formulations is provided with an iterative scheme, tailored after the characteristics of the problem at hand, to find multipliers that solve the Lagrangian dual to optimality or near-optimality.

The second class concerns formulations that present the disjunctive constraints *implicitly*, in which the job completion times are the decision variables. Examples are given by Hariri and Potts (1983) for minimizing total weighted job completion time in the presence of release dates, Potts and Van Wassenhove (1984) for minimizing total weighted completion time with deadlines, and Potts and Van Wassenhove (1985) for minimizing total weighted tardiness. In each of these applications, however, no scheme is proposed to solve the Lagrangian dual. Instead, a specific single pass method is proposed to make an intelligent choice for the multipliers. A notable exception is found in Van de Velde (1988), where the Lagrangian dual is solved for minimizing total completion time in the two-machine flow shop.

This paper is concerned with the second class of formulations for single-machine scheduling problems with minsum criteria. For this class, it is possible to develop a duality theory that serves as a framework for the design of both approximative and enumerative methods. In Section 2, we demonstrate this by considering the problem of minimizing total weighted completion time subject to precedence constraints. We consider its Lagrangian relaxation, and propose a method to solve the Lagrangian dual. In addition, we analyze the optimal dual solution and derive some useful properties that we exploit in the design of an approximation algorithm (Section 3) and in an attempt to decompose the primal problem (Section 4). Directions for extensions and future research are given in Section 5.

2. Single-machine scheduling

A single-machine job shop is described as follows. A set $\mathcal{J} = \{J_1, \ldots, J_n\}$ 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 a given positive uninterrupted processing time p_j. In addition, each job J_j has an associated weight w_j, that expresses its urgency relative to other jobs. Without loss of generality, we may 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. The scheduling objective we consider is the minimization of total weighted completion time $\Sigma_{j=1}^{n} w_j C_j$.

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

$$\sum_{j=1}^{n} w_j C_j \tag{P}$$

subject to

$$C_j \geqslant C_k + p_j \quad \text{or} \quad C_j \leqslant C_k - p_k \quad \text{for } j = 1, \ldots, n-1,\ k = j+1, \ldots, n, \tag{1}$$

$$C_j - p_j \geqslant 0 \qquad \text{for } j = 1, \ldots, n. \tag{2}$$

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

PROPOSITION 1. *Problem* (P) *is solved in* $O(n\log n)$ *time by Smith's shortest weighted processing time rule* (Smith, 1956), *which schedules the jobs in order of non-increasing ratios* w_j/p_j.

This rule is easily validated through an interchange argument.

Now, suppose there are precedence constraints between the jobs. Following the notation of Graham et al. (1979), we refer to the problem of minimizing $\Sigma w_j C_j$ subject to precedence constraints on a single machine as $1|prec|\Sigma w_j C_j$. 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 $\mathcal{P}_j$ and $\mathcal{S}_j$ as the set of immediate predecessors and immediate successors of J_j, respectively $(j=1,\ldots,n)$.

The presence of precedence contraints between jobs makes the problem $\mathcal{NP}$-hard (Lawler, 1978, Lenstra and Rinnooy Kan, 1978). This justifies the development of approximative and enumerative algorithms. Morton and Dharan (1978) proposed several heuristics, and Potts (1985) presented a branch-and-bound procedure, based on Lagrangian relaxation of a zero-one programming formulation of the problem. In contrast to Potts, we formulate the precedence constraints in a concise manner. Note that the precedence constraints imply

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

The $1|prec|\Sigma w_j C_j$ problem can be regarded as an easy-to-solve problem complicated by conditions (3). Therefore, 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 given multipliers, the Lagrangian relaxation problem, referred to as problem (L$_\lambda$), is to find $L(\lambda)$, which is the minimum of

$$\sum_{j=1}^{n}\Big(\big(w_j+\sum_{J_k\in\mathcal{S}_j}\lambda_{jk}-\sum_{J_k\in\mathcal{P}_j}\lambda_{kj}\big)C_j+\sum_{J_k\in\mathcal{S}_j}\lambda_{jk}p_k\Big), \tag{L$_\lambda$}$$

subject to conditions (1) and (2).

In analogy to Problem (P), Problem (L$_\lambda$) is evidently solved by sequencing the jobs in order of non-increasing values $(w_j+\Sigma_{J_k\in\mathcal{S}_j}\lambda_{jk}-\Sigma_{J_k\in\mathcal{P}_j}\lambda_{kj})/p_j$. In the remainder, we call these values the *relative weights* of the jobs. From standard Lagrangian theory, we know that $L(\lambda)$ is a lower bound for the $1|prec|\Sigma w_j C_j$ problem. In that respect, we are interested in finding values λ_{jk} that maximize $L(\lambda)$. This is the dual problem of $1|prec|\Sigma w_j C_j$, referred to as problem (D): maximize

$$L(\lambda) \tag{D}$$

subject to

$$\lambda_{jk}\geqslant 0 \quad \text{for each } (J_j,J_k)\in A.$$

Problem (D) can be transformed into the problem of maximizing a linear function subject to a finite number of linear constraints. There are at most $n!$ feasible sequences involved, and we can represent each of these by a vector of completion times $(C_1^t, C_2^t, \ldots, C_n^t)$, $t = 1, \ldots, T$, where $T \leqslant n!$; the superscript t refers to the t-th feasible schedule. Problem (D) is then equivalent to the following problem, referred to as problem (D′): maximize

$$v$$

subject to

$$v \leqslant \sum_{j=1}^{n} ((w_j + \textstyle\sum_{J_k \in \mathbb{S}_j} \lambda_{jk} - \sum_{J_k \in \mathcal{P}_j} \lambda_{kj})C_j^t + \sum_{J_k \in \mathbb{S}_j} \lambda_{jk} p_k) \quad \text{for } t = 1, \ldots, T,$$

$$\lambda_{jk} \geqslant 0 \qquad \text{for each } (J_j, J_k) \in A.$$

Proposition 2. *Problem* (D) *is solvable in time polynomial in n through Khachiyan's ellipsoid method* (Khachiyan, 1979).

Proof. Let $K = \{(v,\lambda) \in \mathbb{R} \times \mathbb{R}^A \mid v \leqslant \sum_{j=1}^{n} ((w_j + \sum_{J_k \in \mathbb{S}_j} \lambda_{jk} - \sum_{J_k \in \mathcal{P}_j} \lambda_{kj})C_j^t + \sum_{J_k \in \mathbb{S}_j} \lambda_{jk} p_k)$ for $t = 1, \ldots, T$, $\lambda_{jk} \geqslant 0$, for each $(J_j, J_k) \in A\}$. To prove the proposition, it suffices to show that the following *separation problem for K* is solvable in polynomial time (see Grötschel, Lovász, and Schrijver, 1981, and Padberg and Rao, 1982): given $(\bar{v}, \bar{\lambda}) \in \mathbb{Q} \times \mathbb{Q}^A$, decide whether $(\bar{v}, \bar{\lambda}) \in K$; if not, give a *separating hyperplane*, that is, an inequality $a^T\lambda + \alpha v \leqslant \beta$, such that

$$(v, \lambda) \in K \Rightarrow a^T\lambda + \alpha v \leqslant \beta, \text{ and}$$

$$a^T\bar{\lambda} + \alpha\bar{v} > \beta.$$

Observe now that for given $(\bar{v}, \bar{\lambda})$ we determine the value $L(\bar{\lambda})$ and the corresponding vector $(\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_n)$ of job completion times by solving problem $(\mathrm{L}_{\bar{\lambda}})$, which can be done in $O(n \log n)$ time. If $L(\bar{\lambda}) \geqslant \bar{v}$, then $(\bar{v}, \bar{\lambda}) \in K$; if $L(\bar{\lambda}) < \bar{v}$, then $(\bar{v}, \bar{\lambda}) \notin K$, and

$$\bar{v} > \sum_{j=1}^{n} ((w_j + \sum_{J_k \in \mathbb{S}_j} \bar{\lambda}_{jk} - \sum_{J_k \in \mathcal{P}_j} \bar{\lambda}_{kj})\bar{C}_j + \sum_{J_k \in \mathbb{S}_j} \bar{\lambda}_{jk} p_k),$$

is a separating hyperplane. □

Nonetheless, we propose a different method to solve the dual problem. It follows from the transformation that the function $F: \lambda \rightarrow L(\lambda)$ is a concave step-wise linear function in λ. Hence, problem (D) is a convex programming problem that can be solved through an *ascent direction* technique. For given values of the Lagrangian multipliers, we need to determine which component to perturb, i.e. the direction, and by how much, i.e. the step size, in order to increase the value $L(\lambda)$. We have solved the dual problem if no such direction exists. The next algorithm is an ascent direction method that solves problem (D).

Ascent Direction Algorithm

Step 0. Set $\lambda_{jk}=0$ for each $(J_j,J_k)\in A$, solve (L_λ), and compute the job completion times.

Step 1. For each $(J_j,J_k)\in A$, put $\Delta=0$ and do the following:

if $C_j>C_k$, though $(J_j,J_k)\in A$, compute Δ such that

$$(w_j+\Delta+\sum_{J_l\in\mathcal{S}_j}\lambda_{jl}-\sum_{J_l\in\mathcal{P}_j}\lambda_{lj})/p_j=(w_k-\Delta+\sum_{J_l\in\mathcal{S}_k}\lambda_{kl}-\sum_{J_l\in\mathcal{P}_k}\lambda_{lk})/p_k;$$

if $C_j<C_k$, $(J_j,J_k)\in A$, and $\lambda_{jk}>0$, compute the largest value Δ such that

$$\lambda_{jk}-\Delta\geqslant 0,\ \text{and}$$

$$(w_j-\Delta+\sum_{J_l\in\mathcal{S}_j}\lambda_{jl}-\sum_{J_l\in\mathcal{P}_j}\lambda_{lj})/p_j\geqslant(w_k+\Delta+\sum_{J_l\in\mathcal{S}_k}\lambda_{kl}-\sum_{J_l\in\mathcal{P}_k}\lambda_{lk})/p_k.$$

In case $\Delta>0$, adjust λ_{jk} as follows: if $C_j>C_k$, increase λ_{jk} by Δ; if $C_j<C_k$, decrease λ_{jk} by Δ. Accordingly, solve (L_λ) and compute the job completion times.

Step 2. If no multiplier adjustment has taken place, then stop: we have attained the optimal solution. Otherwise, go to Step 1.

Proposition 3. *The procedure described above is an ascent direction method that solves problem* (D) *in a finite number of steps.*

Proof. First, suppose that $C_j>C_k$ though $(J_j,J_k)\in A$. Let $\Delta>0$ be the step size that is computed as prescribed in the ascent direction algorithm. In addition, let λ^1 and λ^2 denote the Lagrangian multiplier before and after the adjustment of λ_{jk}, respectively. Hence, the two vectors differ only in one component; for this component we have $\lambda_{jk}^2=\lambda_{jk}^1+\Delta$. This means that the relative weight for J_j is increased, the relative weight for J_k is decreased, and the relative weights for the remaining jobs stay the same with respect to problem (L_{λ^1}). Let the optimal sequence associated with problem (L_{λ^1}) be $(J_1,\ldots,J_{k-1},J_k,J_{k+1},\ldots,J_{j-1},J_j,J_{j+1},\ldots,J_n)$; we have reindexed the jobs according to increasing completion times. Consequently, the optimal schedule for problem (L_{λ^2}) can be written as $(J_1,\ldots,J_{k-1},J_{k+1},\ldots,J_l,J_j,J_k,J_{l+1},\ldots,J_{j-1},J_{j+1},\ldots,J_n)$, for some job J_l with $k+1\leqslant l\leqslant j-1$. We will now demonstrate that $L(\lambda^2)>L(\lambda^1)$. The vector $(C_1,\ldots,C_n)$ refers to the job completion times in the first schedule; the job completion times for the second schedule can be expressed in terms of this vector and the job processing times. In addition, we let for brevity $\mu_i=w_i+\Sigma_{J_h\in\mathcal{S}_i}\lambda_{ih}^1-\Sigma_{J_h\in\mathcal{P}_i}\lambda_{hi}^1$ for each i, $i=1,\ldots,n$. Then, we have

$$\begin{aligned}L(\lambda^2)=&\sum_{i=1}^{k-1}\mu_iC_i+\sum_{i=j+1}^{n}\mu_iC_i+\sum_{i=k+1}^{l}\mu_i(C_i-p_k)+\sum_{i=l+1}^{j-1}\mu_i(C_i+p_j)+\\&(\mu_k-\Delta)(C_k+p_j+\sum_{i=k+1}^{l}p_i)+(\mu_j+\Delta)(C_j-p_k-\sum_{i=l+1}^{j-1}p_i)+\sum_{i=1}^{n}\sum_{J_h\in\mathcal{S}_i}\lambda_{ih}^1p_h+\Delta p_k\\=&L(\lambda^1)+\sum_{i=l+1}^{j-1}(\mu_ip_j-\mu_jp_i)+\sum_{i=k+1}^{l}(\mu_kp_i-\mu_ip_k)+\mu_kp_j-\mu_jp_k+\\&\Delta\left[(C_j-p_k-\sum_{i=l+1}^{j-1}p_i)-(C_k+p_j+\sum_{i=k+1}^{l}p_i)\right]+\Delta p_k.\end{aligned}$$

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

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

This implies that

$$L(\lambda^2) = L(\lambda^1) + \sum_{i=l+1}^{j-1} (\mu_i p_j - \mu_j p_i) + \sum_{i=k+1}^{l} (\mu_k p_i - \mu_i p_k) + \mu_k p_j - \mu_j p_k$$

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

Since $\mu_k/p_k > \mu_j/p_j$, $\mu_i/p_i < \mu_k/p_k$ for each i, $i = k+1, \ldots, l$, and $\mu_i/p_i > \mu_j/p_j$ for each i, $i = l+1, \ldots, j-1$, we have established that $L(\lambda^2) > L(\lambda^1)$. Suppose now that $C_j < C_k$, $(J_j, J_k) \in A$, and $\lambda_{jk} > 0$. If now Δ, computed as described, is greater than 0, then we can perform a similar analysis as above to show that $L(\lambda^2) > L(\lambda^1)$. For that reason, this part of the proof is omitted. Finally, the ascent direction method solves problem (D) in a finite number of steps, since the problem (D) is a linear convex programming problem. □

Notice that

$$C_j \geqslant C_k \Leftrightarrow (w_j + \sum_{J_l \in \mathcal{S}_j} \lambda_{jl} - \sum_{J_l \in \mathcal{P}_j} \lambda_{lj})/p_j \leqslant (w_k + \sum_{J_l \in \mathcal{S}_k} \lambda_{kl} - \sum_{J_l \in \mathcal{P}_k} \lambda_{lk})/p_k.$$

For each arc $(J_j, J_k) \in A$, we need only constant time to determine the ascent direction, if there exists one. Hence, there is no need to solve (L_λ) and update the completion times in Step 1 of the ascent direction procedure. Since the associated step size is computed in constant time, too, it takes $O(|A|)$ time to verify whether a given set of multipliers solves the dual problem (D). If we have attained the optimal set, then we need $O(n \log n)$ time to compute an optimal dual schedule, the corresponding job completion times, and the optimal dual objective value. Hence, the algorithm runs in $O(I \cdot |A| + n \log n)$ time, where I is the number of iterations. Since I cannot be bounded by a polynomial in n and $|A|$, the ascent direction method is presumably not polynomial. However, from an empirical point of view, it is a fast simplex-like algorithm in the sense that it traverses the edges of the polytope to find the optimal solution.

Consider the 10-job example that is taken from Potts (1985) for which the processing times, weights, and precedence graph are found in Table 1 and Figure 1, respectively.

	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

TABLE 1.

If there were no precedence constraints, which concurs with the problem (L_λ) with λ equal to the null vector, the optimal schedule is $(J_3, J_{10}, J_4, J_9, J_7, J_6, J_2, J_5, J_8, J_1)$, having total cost

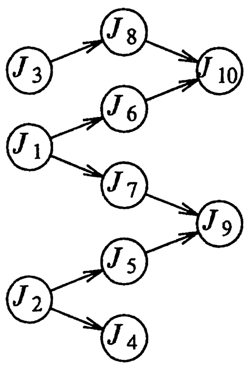

FIGURE 1. Precedence graph.

1055. This schedule is not feasible for the original problem; for instance, J_{10} is executed before J_6 though $(J_6, J_{10}) \in A$. Accordingly, we compute Δ as prescribed in the ascent direction algorithm; this gives $\Delta = 17/7$, and hence we put $\lambda_{6,10} = 17/7$. This yields $(J_3, J_4, J_9, J_7, J_6, J_{10}, J_2, J_5, J_8, J_1)$ as an optimal schedule for the current Lagrangian problem with $L(\lambda) = 1106$. Proceeding along these lines, we find that the optimal value for problem (D) is 1526.69. In contrast, Potts' procedure, which requires at least $\Omega(n^4)$ time, produces 1519 as a lower bound, while the so-called tree-optimal-heuristic developed by Morton and Dharan generates a schedule with cost 1530, which is the optimal value. Hence, the duality gap is only 3.31.

DEFINITION. The job set $\mathcal{B} \subseteq \mathcal{J}$ is called a *block* if

$$(w_j + \sum_{J_k \in \mathcal{S}_j} \lambda_{jk} - \sum_{J_k \in \mathcal{P}_j} \lambda_{kj})/p_j = c \quad \text{for each } J_j \in \mathcal{B},$$

where c is some positive real constant.

Note that the jobs in a block may be interchanged without affecting the dual objective value. For any set of Lagrangian multipliers, the job set $\mathcal{J}$ is decomposed into B blocks $\mathcal{B}_1, \ldots, \mathcal{B}_B$, indexed such that

$$(w_j - \sum_{J_k \in \mathcal{S}_j} \lambda_{jk} + \sum_{J_k \in \mathcal{P}_j} \lambda_{kj})/p_j = c_b, \quad \text{for each } J_j \in \mathcal{B}_b,$$

with $c_1 > \cdots > c_B > 0$. The blocks that are associated with the vector of optimal Lagrangian multipliers (denoted by λ^*) display an agreeable structure.

PROPOSITION 4. *The optimal Lagrangian multipliers decompose the job set $\mathcal{J}$ into B blocks $\mathcal{B}_1, \ldots, \mathcal{B}_B$, such that if $(J_j, J_k) \in A$ and $J_k \in \mathcal{B}_b$, then:*

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

$$\lambda^*_{jk} = 0 \quad \text{if } J_j \in \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_{b-1}.$$

PROOF. If one of these claims were not true, then we could still identify an ascent direction. This contradicts the optimality of the vector λ^*. □

In our example, we obtain three blocks: $\mathcal{B}_1 = \{J_3\}$, $\mathcal{B}_2 = \{J_2, J_4\}$, and $\mathcal{B}_3 = \{J_1, J_5, J_6, J_7, J_8, J_9, J_{10}\}$, with $c_1 = 9$, $c_2 = 11/12$, and $c_3 = 6/7$, respectively.

3. APPROXIMATION

For the optimal Lagrangian multipliers, the decomposition of the job set $\mathcal{J}$ into blocks $\mathcal{B}_1, \ldots, \mathcal{B}_B$ and the properties possessed by these blocks are useful for developing an attractive approximation algorithm. We refer to this decomposition of the job set as the *dual decomposition*. In addition, we speak of *primal decomposition*, if the set of jobs can be decomposed into subsets such that the $1|prec|\Sigma w_j C_j$ problem can be solved to optimality by minimizing total weighted completion time subject to precedence constraints for each subset separately. The issue is how to find such a primal decomposition, since there is no guarantee that the dual decomposition concurs with a primal decomposition. However, the dual decomposition *suggests* a primal decomposition; in this section, we develop an approximation algorithm that is based upon this decomposition. In Section 4, we investigate to which extent the dual decomposition really coincides with a primal decomposition.

Recall that the optimal Lagrangian multipliers decompose the set of jobs into blocks such that for each $(J_j, J_k) \in A$ we have

$$J_k \in \mathcal{B}_b \Rightarrow J_j \in \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_b.$$

If S_b denotes a feasible sequence for the jobs in block $\mathcal{B}_b$, then the schedule S obtained as $S = (S_1, S_2, \ldots, S_B)$ is a feasible solution to the overall problem. Note that we would obtain an approximate solution that is at least as good if for each $\mathcal{B}_b$, S_b would be a sequence for the jobs in $\mathcal{B}_b$ that minimizes the total weighted completion time subject to the precedence constraints that exist between the elements of $\mathcal{B}_b$. Of course, minimizing $\Sigma w_j C_j$ for the block $\mathcal{B}_b$ is as hard as the original problem, but it is of a smaller dimension. For small instances of $1|prec|\Sigma w_j C_j$, a dynamic programming based algorithm with a compact labeling scheme as suggested by Schrage and Baker (1978) and Lawler (1979) is highly efficient. In the approximation algorithm we propose, we minimize $\Sigma w_j C_j$ for a block if less than 15000 labels are needed. Otherwise, we take our resort to the tree-optimal-heuristic developed by Morton and Dharan to generate a feasible sequence for this block. This heuristic has proven to be a very effective, but rather time-consuming approximation algorithm (Morton and Dharan, 1978, Potts, 1985). Our hope is then that the dual decomposition concurs with a primal decomposition, or at least leads up to a promising decomposition, and that the blocks are amenable to dynamic programming.

In the example, the optimal sequences for the first two blocks are trivial: $S_1 = (J_3)$, and $S_2 = (J_2, J_4)$; by dynamic programming (with maximum label 68), we find $S_3 = (J_1, J_7, J_5, J_9, J_6, J_8, J_{10})$, the same sequence as we would have found with the tree-optimal-heuristic. By aggregation of the three sequences, we obtain $S = (J_3, J_2, J_4, J_1, J_7, J_5, J_9, J_6, J_8, J_{10})$ with total cost 1530.

We tested the approximation algorithm 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 generated in this way was then subsequently squashed to its transitive reduction. We generated problems for $P = 0.001$, 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, for each value of P, we generated 45 problems. This procedure parallels Potts' procedure to generate instances.

In Table 2, we present the computational results. As Potts already pointed out, the relative difficulty of an instance is more related to the density of the precedence matrix than to the number of jobs n. Therefore, we have classified the results according to the value P rather than the number of jobs. For each combination of P, we give the proportional deviation between upper bound and lower bound for both the tree-optimal-heuristic and the dual-decomposition approach. Within brackets we indicate for how many problems (out of 45) the upper bound equalled the lower bound, that is, for how many problems we touched upon an optimal solution. As can be seen from this table, the dual-decomposition approach outperforms the tree-optimal-heuristic approach on the average for any problem class. In addition, we found that out of 450 instances, the tree-optimal-heuristic produced only 16 better solutions than the dual-decomposition based algorithm, each of which was only marginally better.

Potts points out that both small and large values for P tend to generate relatively easy problems. For small values there are only few precedence constraints involved, for large values most disjunctive constraints seem to be settled. For small values, this claim is unconditionally supported by our results: the duality gap is very small. As Potts also reports that the tree-optimal-heuristic is a robust procedure for varying values of P, it must be that the duality gap widens if P increases, rather than that the dual decomposition approach subsides.

P	Tree-Opt-Heuristic	Dual Decomposition
0.001	0.007 (42)	0.007 (42)
0.02	0.074 (15)	0.069 (15)
0.04	0.516 (8)	0.248 (10)
0.06	1.214 (2)	0.675 (2)
0.08	1.446 (1)	1.076 (1)
0.10	2.040 (2)	1.518 (4)
0.15	2.252 (0)	2.024 (0)
0.20	2.551 (0)	2.113 (2)
0.30	4.111 (0)	3.733 (2)
0.50	4.334 (2)	4.116 (3)

TABLE 2. Experimental results.

The tree-optimal-heuristic requires $O(n^3)$ time, and is therefore sensitive to instances with a large number of jobs. The running time of the dual decompositon approximation algorithm mainly depends on the number of calls on the dynamic programming algorithm and the maximum label number. We have coded both algorithms in the computer language C; all experiments were conducted on a Compaq 386/20 Personal Computer. For $n \leqslant 40$, the tree-optimal-heuristic generally needed a few seconds at the most. Though for these values for n our approximation required only slightly more computation time on the average, there were occasional peaks as a result of a relatively high labels in the dynamic programming subroutine. For $n \geqslant 60$, we found out that the tree-optimal-heuristic needed about twice or thrice as much computation time as the dual-decomposition algorithm; even the peaks for the latter remained below the average of the former.

4. Primal decomposition

The dual decomposition only suggests a primal decomposition; however, there are instances that demonstrate that the dual decomposition may eliminate the optimal solution. In this section, we derive sufficient conditions to establish to which extent the dual composition coincides with a primal decomposition. If the dual decomposition excludes any optimal solution, then there must be at least two jobs belonging to different blocks for which the processing order should be reversed. Consider the jobs $J_j \in \mathfrak{B}_b$ and $J_k \in \mathfrak{B}_{b+m}$ $(m > 0)$ for which there is no path in A from J_j to J_k. Suppose that J_k precedes J_j in any optimal schedule. This would imply that the arc (J_k, J_j) could be included in the arc set A without impunity. Let now $\lambda^*(k,j)$ denote the optimal Lagrangian multiplier for problem (D_{kj}), which is the dual problem with respect to the set A augmented with the arc (J_k, J_j). Accordingly, $L(\lambda^*(k,j))$ is the optimal objective value for the problem (D_{kj}). Since J_j and J_k belong to different blocks, we must have that $L(\lambda^*(k,j)) > L(\lambda^*)$. If $L(\lambda^*(k,j)) > UB - 1$, where UB is the currently best solution to the $1|prec|\Sigma w_j C_j$ problem in hand, then we know that in any solution with cost less than UB, if such a solution exists, J_j must be executed before J_k. Based upon this observation, we have the following result.

Proposition 5. *If for each pair of jobs $J_j \in \mathfrak{B}_b$ and $J_k \in \mathfrak{B}_{b+m}$, $b = 1, \ldots, B-1$, $m = 1, \ldots, B-b$, such that*

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

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

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

we have that $L(\lambda^*(k,j)) > UB - 1$, then the dual decompositon is a primal decomposition.

Accordingly, if the dual decomposition induces a primal decomposition in the sense of Proposition 5 and if UB is associated with the schedule that is composed of the optimal schedules for each block separately, then UB is the optimal solution value to the $1|prec|\Sigma w_j C_j$ problem.

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

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

(*ii*) *J_j has no successors scheduled in $\mathfrak{B}_b \cup \cdots \cup \mathfrak{B}_{b+m-1}$, and* *(iii)* J_k *has no predecessors scheduled in* $\mathfrak{B}_{b+1} \cup \cdots \cup \mathfrak{B}_{b+m}$,

satisfies $L(\lambda^*(k,j)) > UB - 1$, *then the set* $\mathcal{J}$ *can be primally decomposed into the subsets* $\mathfrak{B}_1 \cup \cdots \cup \mathfrak{B}_b$ *and* $\mathfrak{B}_{b+1} \cup \cdots \cup \mathfrak{B}_B$.

In this case, we say that the dual decomposition concurs *partly* with a primal decomposition. From the next proposition, it follows that $\lambda^*(k,j)$ is conveniently computed from λ^*.

PROPOSITION 6. *We have that*

$$\lambda^*(k,j)_{ih} = \lambda^*_{ih} \textit{ if either } J_i \textit{ or } J_h \notin \mathfrak{B}_b \cup \cdots \cup \mathfrak{B}_{b+m}.$$

PROOF. The introduction of the arc (J_k,J_j) only affects the jobs in the blocks $\mathfrak{B}_b, \ldots, \mathfrak{B}_{b+m}$. Hence, we can only have that $\lambda^*(k,j)_{ih} \neq \lambda^*_{ih}$ if both J_i and $J_h \in \mathfrak{B}_b \cup \cdots \cup \mathfrak{B}_{b+m}$. □

Nonetheless, the verification of Proposition 5 demands the solution of $O(n^2)$ reoptimization problems. It is conceivable that if J_j and J_k belong to blocks that lie far apart from each other, there is no need to go through the entire reoptimization procedure. It may suffice to perform only the first step in the ascent direction procedure. This very first step can be conveniently computed; this is stipulated in the next proposition, where P_b is defined as $P_b = \Sigma_{J_i \in \mathfrak{B}_b} p_i$.

PROPOSITION 7. *If there are two jobs* $J_j \in \mathfrak{B}_b$ *and* $J_k \in \mathfrak{B}_{b+m}$, $b = 1, \ldots, B-1$, $m = 1, \ldots, B-b$ *such that there is no path in* A *from* J_j *to* J_k *for which*

$$L(\lambda^*) + (c_b - c_{b+m})p_j p_k + \sum_{i=b+1}^{l} (c_b - c_i)P_i p_j + \sum_{i=l+1}^{b+m-1} (c_i - c_{b+m})P_i p_k > UB - 1,$$

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 any optimal solution to the* $1|prec|\Sigma w_j C_j$ *problem.*

PROOF. The validation of this proposition, which requires the same logic as applied in the proof of Proposition 3, is left to the reader. □

We now work out the effects of these propositions on our example. To decompose the jobs into $\mathfrak{B}_1$ on one side and $\mathfrak{B}_2 \cup \mathfrak{B}_3$ on the other, we need consider only the pairs (J_3,J_2) and (J_3,J_1) according to Corollary 1. If we include (J_2,J_3) in A, then it suffices to execute only one step in the reoptimization procedure. The application of Proposition 7 yields that $L(\lambda^*)+(c_1-c_2)p_2p_3 = 1526.69+(9-11/12)\cdot 1\cdot 9 > 1529$, as a result of which we conclude that J_3 precedes J_2 in any schedule with cost less than 1530. Similarly, if we include (J_1,J_3) in A, then, according to Proposition 7 (where we have $l=1$), we must check if

$$L(\lambda^*) + (c_1 - c_3)p_1p_3 + (c_2 - c_3)P_2p_1 > UB - 1.$$

It is easy verifiable that this is true; hence, J_3 must precede J_1, which implies that we may decompose the job set into subsets $\mathfrak{B}_1$ and $\mathfrak{B}_2 \cup \mathfrak{B}_3$. We must consider the pairs (J_4,J_1), (J_4,J_5), and (J_4,J_8) to separate the blocks $\mathfrak{B}_2$ and $\mathfrak{B}_3$. For the associated reoptimization problems more than one step is required. Since $L(\lambda^*(1,4))$, $L(\lambda^*(5,4))$, and $L(\lambda^*(8,4))$ are greater than 1529, we conclude that the dual decomposition induces a complete primal decomposition. Furthermore, as the schedule with value 1530 was obtained from the optimal sequences for the individual blocks, it must be an optimal schedule.

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

5. Conclusions

Dual theories similar to the one discussed here for the $1|prec|\Sigma w_j C_j$ problem can be developed for other single-machine scheduling problems with minsum criteria functions provided that the Lagrangian problem reduces to problem (P). However, the effectiveness of the dual decomposition approach depends on the nature of the relaxed constraints. In this respect, the two most eye-catching and likely applications are the problems of minimizing the total weighted completion time subject to deadlines and minimizing the total weighted tardiness. For the latter problem, for instance, it would be interesting to see to which extent the multiplier adjustment method presented by Potts and Van Wassenhove (1985) could be integrated with or replaced by our approach. In addition, the various approximation algorithms for this problem evaluated by Potts and Van Wassenhove (1988) ask for a comparison with the dual decomposition method.

Acknowledgment

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

References

E. Balas (1985). On the facial structure of scheduling polyhedra. *Mathematical Programming Study 24*, 179-218.

M.L. Fisher (1976). A dual algorithm for the one-machine scheduling problem. *Mathematical programming 11*, 229-251.

M.L. Fisher (1981). The Lagrangian relaxation method for solving integer programming problems. *Management Science 27*, 1-18.

R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G Rinnooy Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics 5*, 287-326.

M. Grötschel, L. Lovász, and A. Schrijver (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica 1*, 169-197. [Corrigendum (1984): *Combinatorica 4*, 291-295.]

A.M.A. Hariri and C.N. Potts (1983). An algorithm for single machine sequencing with release dates to minimize total weighted completion time. *Discrete Applied Mathematics 5*, 99-109.

A.M.A. Hariri and C.N. Potts (1984). Algorithms for two-machine flow-shop sequencing with precedence constraints. *European Journal of Operational Research 17*, 238-248.

L.G. KHACHIYAN (1979). A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR 244*, 1093-1096 (English translation: *Soviet Mathematics Doklady 20*, 191-194).

E.L. LAWLER (1978). Sequencing jobs to minimize total weighted subject to precedence constraints. *Annals of Discrete Mathematics 2*, 75-90.

E.L. LAWLER (1979). *Efficient Implementation of Dynamic Programming Algorithms for Sequencing Problems*, Report BW 106, Centre for Mathematics and Computer Science, Amsterdam.

E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN AND D.B. SHMOYS (eds.) (1985). *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization*, Wiley, Chichester.

J.K. LENSTRA AND A.H.G. RINNOOY KAN (1978). Complexity of scheduling under precedence constraints. *Operations Research 26*, 22-35.

T.E. MORTON AND B.G. DHARAN (1978). Algoristics for single-machine sequencing with precedence contraints. *Management Science 24*, 1011-1020.

M.W. PADBERG AND M.R. RAO (1982). Odd minimum cut-sets and *b*-matchings. *Mathematics and Operations Research 7*, 67-80.

C.N. POTTS (1985). A Lagrangean based branch-and-bound algorithm for single machine sequencing with precedence constraints to minimize total weighted completion time. *Management Science 31*, 1300-1311.

C.N. POTTS AND L.N. VAN WASSENHOVE (1983). An algorithm for single machine sequencing with deadlines to minimize total weighted completion time. *European Journal of Operational Research 33*, 363-377.

C.N. POTTS AND L.N. VAN WASSENHOVE (1985). A branch and bound algorithm for the total weighted tardiness problem. *Operations Research 33*, 363-377.

C.N. POTTS AND L.N. VAN WASSENHOVE (1988). *Single machine tardiness sequencing heuristics*, Report 8906/A, Erasmus University, Rotterdam.

M. QUEYRANNE AND Y. WANG (1988). *Single machine scheduling polyhedra with precedence constraints*, Working Paper No. 88-MSC-017, University of British Columbia, Vancouver.

L. SCHRAGE AND K.R. BAKER (1978). Dynamic programming solution of sequencing problems with precedence constraints. *Operations Research 26*, 444-449.

W.E. SMITH (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly 3*, 59-66.

S.L. VAN DE VELDE (1990). Minimizing the sum of the job completion times in the two-machine flow shop by Lagrangian relaxation. To appear in *Annals of Operations Research.*