Mathematical Programming 70 (1995) 173–190

Stronger Lagrangian bounds by use of slack variables: applications to machine scheduling problems

J.A. Hoogeveen [a,*], S.L. van de Velde [b]

[a] *Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, Netherlands*

[b] *Department of Mechanical Engineering, University of Twente, P.O. Box 217, 7500 AE Enschede, Netherlands*

Received September 1992; revised manuscript received 27 July 1994

Abstract

Lagrangian relaxation is a powerful bounding technique that has been applied successfully to many $\mathcal{NP}$-hard combinatorial optimization problems. The basic idea is to see an $\mathcal{NP}$-hard problem as an "easy-to-solve" problem complicated by a number of "nasty" side constraints. We show that reformulating nasty inequality constraints as equalities by using slack variables leads to stronger lower bounds. The trick is widely applicable, but we focus on a broad class of machine scheduling problems for which it is particularly useful. We provide promising computational results for three problems belonging to this class for which Lagrangian bounds have appeared in the literature: the single-machine problem of minimizing total weighted completion time subject to precedence constraints, the two-machine flow-shop problem of minimizing total completion time, and the single-machine problem of minimizing total weighted tardiness.

Keywords: Lagrangian relaxation; Slack variables; Scheduling

1. Introduction

Lagrangian relaxation is a powerful bounding technique that has been successfully embedded in branch-and-bound algorithms for a gamut of $\mathcal{NP}$-hard combinatorial optimization problems. By now, it is already considered a conventional technique, dating back to the work by Held and Karp [10,11]. Excellent introductions to Lagrangian relaxation are given in [5,8,26]; an overview of its applications is given in [5,6].

* Corresponding author. e-mail: slam@win.tue.nl.

SSDI 0025-5610(94)00061-1

The main idea behind Lagrangian relaxation is to see an $\mathcal{NP}$-hard combinatorial optimization problem as an "easy-to-solve" problem complicated by a number of "nasty" side constraints. Consider the typical combinatorial optimization problem (P) of determining

$$z = \min\{cx \mid Ax \geqslant b,\ x \in X\},$$

where A is a given $m \times n$ matrix, b a given $m \times 1$ vector, and c a given $1 \times n$ vector; x is an $n \times 1$ vector of decision variables. It is assumed that X is a nonempty finite set and that X has some computationally convenient structure not shared by the entire problem; in this sense, $Ax \geqslant b$ are the nasty constraints. We remove them from the set of constraints and put them into the objective function, each weighted by a given non-negative Lagrangian multiplier; this is known as the *dualization* or *Lagrangian relaxation* of the nasty constraints. Using a given vector of Lagrangian multipliers $\lambda = (\lambda_1, \ldots, \lambda_m) \geqslant 0$, we obtain the *Lagrangian problem* (L_λ) of determining

$$L(\lambda) = \min\{(c - \lambda A)x + \lambda b \mid x \in X\}.$$

Clearly, we have that $L(\lambda) \leqslant z$ for any $\lambda \geqslant 0$. The application of Lagrangian relaxation is problem-specific, as it tries to exploit the underlying structure of the problem. Choosing an appropriate formulation of the problem, identifying the nasty constraints, and finding or approximating the vector λ^* of Lagrangian multipliers that solves the *Lagrangian dual problem* $\max\{L(\lambda) \mid \lambda \geqslant 0\}$ are traditionally seen as the key issues.

Lagrangian relaxation is an alternative bounding technique to *linear programming relaxation*, which proceeds by formulating an $\mathcal{NP}$-hard problem as an integer linear program, dropping the integrality conditions on the variables, and then solving the linear programming relaxation. The trade-off between the *quality* of the lower bounds and the *speed* in which they are computed is essential in designing branch-and-bound algorithms and therefore in choosing a lower bounding technique. As to quality, we have that, if the relaxations are applied to the same formulation, then $L(\lambda^*)$ is at least as large as the optimal solution value of the linear programming relaxation of problem (P). In fact, the bounds are equal if the Lagrangian problem (L_λ) is solvable as a linear program, i.e., if X is equal to the set of integral vectors of a system $Dx \geqslant e$ of linear inequalities in which the integrality constraints are redundant [8]. In this case, however, $L(\lambda^*)$ may be computed faster, since the easy-to-solve Lagrangian problem often allows a fast tailor-made algorithm. Another agreeable feature of Lagrangian relaxation is that solutions to the Lagrangian problem often induce good approximate solutions to the original problem; see, e.g., [12].

In this paper, we show that reformulating nasty inequalities as equalities by using slack variables can lead to better Lagrangian bounds. We work this idea out in Section 2. For the case that the Lagrangian problem possesses the integrality property, we give an interpretation of the slack variable approach in terms of linear programming theory in Section 3.

The slack variable approach is widely applicable, but we focus on a class of machine scheduling problems for which it is particularly useful. In Section 4, we identify this

class, and we apply the reformulation trick to three representative problems for which Lagrangian bounds have appeared in the literature: the single-machine problem of minimizing total weighted completion time, the two-machine flow-shop problem of minimizing total completion time, and the single-machine weighted tardiness problem. For each problem we obtain promising computational results in the sense that the new Lagrangian bound shows a significant improvement relative to the extra time needed to compute it. We draw some conclusions in Section 5.

2. Stronger Lagrangian bounds by slack variables

We now reformulate the nasty inequality constraints of problem (P) as equality constraints by use of a nonnegative vector y of slack variables. This gives

$$z = \min\{cx \mid Ax - y = b,\ x \in X,\ y \in Y\},$$

where $Y \subseteq \mathbb{R}^m_+$ is the unspecified set of all slack vectors that correspond to feasible solutions to problem (P). If we again dualize the nasty constraints with a given vector $\lambda \geqslant 0$, we get the Lagrangian problem (L'_λ) of finding

$$L'(\lambda) = \min\{(c - \lambda A)x + \lambda b + \lambda y \mid x \in X,\ y \in Y\}.$$

Since Y also depends on the constraints on x, problem (L'_λ) does not reveal any agreeable structure. Observe, however, that

$$\begin{aligned} L'(\lambda) &\geqslant \min\{(c - \lambda A)x + \lambda b \mid x \in X\} + \min\{\lambda y \mid y \in Y\} \\ &= L(\lambda) + \min\{\lambda y \mid y \in Y\}. \end{aligned}$$

In the remainder, we refer to the problem of finding

$$\mathrm{SV}(\lambda) = \min\{\lambda y \mid y \in Y\}$$

as the *slack variable problem*. Note that we have that $\mathrm{SV}(\lambda) > 0$ for all $\lambda > 0$ if the null vector is not in Y. The slack variable problem has the same computational complexity as problem (P), as it is alternatively formulated as $\min\{\lambda(Ax - b) \mid Ax \geqslant b,\ x \in X\}$. We must therefore try to compute a positive lower bound on $\mathrm{SV}(\lambda)$; if we succeed, then we add it to the traditional Lagrangian lower bound $L(\lambda)$. Using this alternative formulation of the slack variable problem and computing a lower bound on it by Lagrangian relaxation of the constraints $Ax \geqslant b$ will give no real improvement, as we will show next. Suppose we apply Lagrangian relaxation. Using a given vector $\mu \geqslant 0$ of Lagrangian multipliers, we get the Lagrangian problem of finding

$$F(\lambda, \mu) = \min\{(\lambda - \mu)Ax + (\mu - \lambda)b \mid x \in X\}.$$

For any $\lambda \geqslant 0$ and $\mu \geqslant 0$ we have that

$$\begin{aligned} L(\lambda) + F(\lambda, \mu) &= \min\{(c - \lambda A)x + \lambda b \mid x \in X\} \\ &\quad + \min\{(\lambda - \mu)Ax + (\mu - \lambda)b \mid x \in X\} \\ &\leqslant \min\{(c - \lambda A)x + \lambda b + (\lambda - \mu)Ax + (\mu - \lambda)b \mid x \in X\} \\ &= L(\mu), \end{aligned}$$

and consequently we have that $F(\lambda^*, \mu) \leqslant 0$ for all $\mu \geqslant 0$. Hence, we must develop alternative methods to find a positive lower bound on $SV(\lambda)$.

Beforehand, it can be expected that $SV(\lambda)$ is small relative to $L(\lambda)$. The effort to compute a lower bound on $SV(\lambda)$ must therefore be small relative to the effort to compute $L(\lambda)$ to make it worthwhile in a branch-and-bound algorithm.

For example, suppose that problem (P) is a general 0–1 linear programming problem. An obvious lower bound on the slack variable problem is then obtained by solving a series of m subset-sum problems of the type $\min\{a^{(i)}x - b_i \mid a^{(i)}x \geqslant b_i,\ x \in \{0, 1\}\}$, for $i = 1, \ldots, m$, where $a^{(i)}$ denotes the ith row of the matrix A. This follows from the observation that

$$\begin{aligned}\min\{\lambda y \mid y \in Y\} &= \min\{\lambda(Ax - b) \mid Ax \geqslant b,\ x \in X\} \\ &\geqslant \min\{\lambda(Ax - b) \mid Ax \geqslant b,\ x \in \{0, 1\}\} \\ &\geqslant \sum_{i=1}^{m} \lambda_i \min\{a^{(i)}x - b_i \mid a^{(i)}x \geqslant b_i, x \in \{0, 1\}\}.\end{aligned}$$

The subset-sum problem is $\mathcal{NP}$-hard in the ordinary sense, but in practice it is a relatively easy problem (see, e.g., [19]).

There exists a broad class of machine scheduling problems that can be formulated in such a way that the slack variables have strong intuitive interpretations. In Section 4, we show that these interpretations are helpful for developing lower bounding procedures for the slack variable problems.

3. Relation between the slack variable problem and valid inequalities

For the case that problem (P) is an integer linear program and the Lagrangian problem (L_λ) possesses the integrality property, we can explain in terms of linear programming theory where the bound improvement comes from. This explanation is based upon a suggestion by Müller [20].

Since the integrality conditions on x are assumed to be redundant, we may replace the set X by $X' = \{x \geqslant 0 \mid Cx \geqslant d\}$. For any $\lambda \geqslant 0$ we then have that

$$L(\lambda) = \min\{(c - \lambda A)x + \lambda b \mid Cx \geqslant d,\ x \geqslant 0\},$$

and since the Lagrangian problem can be solved as a linear program, that

$$\begin{aligned}L(\lambda) &= \max\{\min\{(c - \lambda A - \mu C)x + \lambda b + \mu d \mid x \geqslant 0\} \mid \mu \geqslant 0\} \\ &= \max\{\lambda b + \mu d \mid \mu \geqslant 0,\ c - \lambda A - \mu C \geqslant 0\} \\ &= \lambda b + \mu^* d,\end{aligned}$$

where μ^* is the vector of optimal values for the dual variables corresponding to the conditions $Cx \geqslant d$. Accordingly, (λ, μ^*) is a feasible but not necessary optimal dual solution to the linear programming relaxation of problem (P).

Now suppose that there exists an $m \times 1$ vector u and a real constant $b_0 > 0$ such that

$$uAx \geqslant ub + b_0, \quad \text{for all } x \in X \text{ with } Ax \geqslant b;$$

in other words, $uAx \geqslant ub + b_0$ is a *valid inequality* for problem (P). Consider the linear program (LP) of finding

$$v(\mathrm{LP}) = \min\{(c - \lambda A)x + \lambda b \mid Cx \geqslant d,\ x \geqslant 0,\ uAx \geqslant ub + b_0\}.$$

Obviously, we have that $L(\lambda) \leqslant v(\mathrm{LP}) \leqslant z$. By linear programming duality we have that

$$v(\mathrm{LP}) = \max\{\lambda b + \mu d + \rho(ub + b_0) \mid \mu \geqslant 0,\ \rho \geqslant 0,\ c - \lambda A - \mu C - \rho uA \geqslant 0\},$$

where ρ is the dual variable corresponding to the valid inequality $uAx \geqslant ub + b_0$. Note that for any $\rho \geqslant 0$ with $\lambda - \rho u \geqslant 0$, we have that

$$(\lambda - \rho u,\ \mu^*,\ \rho)$$

is a feasible dual solution to problem (LP). The feasible dual solutions of the type $(\lambda - \rho u,\ \mu^*,\ \rho)$ are interesting, since

$$\begin{aligned} D(\lambda, u) &= \max\{(\lambda - \rho u)b + \mu^* d + \rho(ub + b_0) \mid \rho \geqslant 0,\ \lambda - \rho u \geqslant 0\} \\ &= \max\{\lambda b + \mu^* d + \rho b_0 \mid \rho \geqslant 0,\ \lambda - \rho u \geqslant 0\} \\ &= L(\lambda) + \max\{\rho b_0 \mid \rho \geqslant 0,\ \lambda - \rho u \geqslant 0\} \geqslant L(\lambda). \end{aligned}$$

In fact, strict inequality holds if some $\rho > 0$ gives rise to a feasible dual solution.

The question of course is how to find such a u and b_0. If we now choose u to be $u = \lambda$, then the problem of determining

$$D(\lambda, \lambda) = L(\lambda) + \max\{\rho b_0 \mid \rho \geqslant 0,\ \lambda - \rho\lambda \geqslant 0\}$$

is solved by simply assigning $\rho = 1$, and hence, by the feasible dual solution $(0, \mu^*, 1)$. The dual solution value is then equal to

$$D(\lambda, \lambda) = L(\lambda) + b_0,$$

and we have then improved our Lagrangian lower bound by b_0. Of course, we would like to specify a b_0 as large as possible. If $u = \lambda$, then the valid inequality is alternatively written as

$$\lambda(Ax - b) \geqslant b_0,$$

and hence the largest feasible value of b_0 is found by determining

$$\mathrm{SV}(\lambda) = \min\{\lambda(Ax - b) \mid x \in X,\ Ax \geqslant b\}.$$

This is exactly the slack variable problem we introduced in the previous section. As pointed out there, this problem is $\mathcal{NP}$-hard, and computing a lower bound on $\mathrm{SV}(\lambda)$ by Lagrangian relaxation is not meaningful. This derivation shows that the slack variable approach is also useful for polyhedral approaches for $\mathcal{NP}$-hard problems, if we can compute a positive lower bound on the slack variable problem.

4. Application to machine scheduling problems

The usual setting of a job shop is as follows. There are m machines available for executing a set of n jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$. Each job J_j, $j = 1, \ldots, n$, consists of an

ordered list of operations, each of which requires processing during a certain uninterrupted period of time on some machine. Each machine can process at most one job at a time and is continuously available from time 0 onwards. A job can be processed by at most one machine at a time. Each J_j is available for processing during a prespecified period: it becomes available at its release date r_j and must be completed at its deadline $\bar{d}_j$. A *schedule* specifies for each job when and by which machine it is executed. Also, there may be precedence constraints between the jobs, i.e., for each job a number of jobs may have been specified that have to precede this job in any feasible schedule. In addition, each J_j may have a weight w_j, expressing its importance, and a due date d_j, at which it ideally should be completed; the weights and due dates are typically used to define the objective function, which is usually a function of the job completion times C_j, $j = 1, \ldots, n$. Such a job-shop situation gives rise to a myriad of problems; for an overview, see, e.g., [16].

An elementary single-machine problem that fits in with the description is the following: schedule n independent jobs, each consisting of one operation and having $r_j = 0$ and $\bar{d}_j = \infty$, so as to minimize total weighted completion time $\sum_{j=1}^{n} w_j C_j$. This problem is solvable in polynomial time: sequence the jobs in order of nonincreasing ratios w_j/p_j, where p_j denotes the processing time of J_j, and process them consecutively from time 0 to time $\sum_{j=1}^{n} p_j$. This priority rule is easily validated by a simple interchange argument [28].

In this paper, we show the merits of the slack variable problem for the class of $\mathcal{NP}$-hard single-machine and multiple-machine scheduling problems for which the Lagrangian problem is solvable through Smith's ratio rule or an analogon of it. This broad class contains essentially three types of problems; in the subsequent subsections, we consider a problem of each type. The first type concerns the single-machine problems of minimizing $\sum_{j=1}^{n} w_j C_j$ subject to nasty constraints on the job completion times, including general release dates, deadlines and precedence constraints; as an example, we examine in Section 4.1 the problem of minimizing total weighted completion time subject to precedence constraints. The second type concerns the multiple-machine problems; as an example of this type, we consider in Section 4.2 the two-machine flow-shop problem of minimizing total completion time. The last type concerns problems with step-wise linear objective functions of the job completion times; as an example, we examine in Section 4.3 the single-machine problem of minimizing total weighted tardiness.

In our formulations of these problems, the completion times of the operations of the jobs are the decision variables. Such compact formulations, requiring only $O(n)$ decision variables, give weaker Lagrangian bounds than the formulations based upon time-discretization, in which 0–1 variables x_{jt} are introduced that indicate whether J_j is executed at time t or not [4]. Formulations of the latter type, however, require a pseudo-polynomial number of variables and constraints; the time and space required to solve the associated Lagrangian problems limit the applicability considerably; see e.g., [29].

In the next subsections, we provide computational evidence that the so-called weak formulations can be significantly strengthened by use of slack variables.

4.1. Scheduling with precedence constraints

In this subsection, we consider the single-machine problem of minimizing total weighted completion time $\sum_{j=1}^{n} w_j C_j$ subject to precedence constraints. The precedence constraints are represented by an acyclic precedence graph G with vertex set $\{J_1, \ldots, J_n\}$ and arc set A, which equals its transitive reduction; i.e., no arc in A can be removed on the basis of transitivity. 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 $(J_j, J_k) \in A$, 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, $j = 1, \ldots, n$, in A.

For special classes of precedence constraints, including tree-like and series-parallel precedence constraints, the problem is still solvable in $O(n \log n)$ time [2,13,15,27], but the general problem is $\mathscr{NP}$-hard in the strong sense [15,17].

Potts [21] presents a branch-and-bound algorithm that solves instances up to 100 jobs. He employs a Lagrangian lower bound obtained from a 0–1 linear programming formulation of the problem, in which variables x_{jk} are introduced that indicate whether J_j is executed before J_k or not. It requires, however, $\Omega(n^4)$ time to compute this bound. Strong lower bounds are also proposed by Van de Velde [32] and by Queyranne and Wang [25]. Van de Velde's bound is based upon Lagrangian relaxation, and we will show here that this bound can be improved significantly at the cost of little additional computational effort. Queyranne and Wang obtain their bound by solving the problem as a linear program to which they add two types of facet-defining inequalities. At the end of this section, we discuss the relationship between their bound and ours.

We proceed from the following formulation of the problem: determine job completion times $C_1, \ldots, C_n$ that minimize

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

subject to

$$C_k \geqslant C_j + p_k, \quad \text{for each } (J_j, J_k) \in A, \tag{1}$$

$$\text{the machine capacity and availability constraints.} \tag{2}$$

Conditions (1), stipulating the precedence constraints, are regarded as the nasty constraints. Accordingly, we introduce a vector $\lambda \in \mathbb{R}_+^A$ that contains a Lagrangian multiplier λ_{jk} for each arc $(J_j, J_k) \in A$ and put the constraints (1), each weighted by its multiplier, into the objective function. For a given vector $\lambda \geqslant 0$, the Lagrangian relaxation problem 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 conditions (2).

For $j = 1, \ldots, n$, let $w_j'(\lambda) = (w_j + \sum_{J_k \in \mathscr{S}_j} \lambda_{jk} - \sum_{J_k \in \mathscr{P}_j} \lambda_{kj})/p_j$; we call $w_j'(\lambda)$ the *relative weight* of job J_j. Using Smith's ratio rule, we solve the Lagrangian problem by sequencing the jobs in order of nonincreasing relative weights. Van de Velde [32] presents a fast iterative ascent direction algorithm, running in $O(I \mid A \mid + n \log n)$ time, where I denotes the number of iterations, to approximate the vector λ^* of Lagrangian multipliers that solves the Lagrangian dual problem.

Since the nasty constraints are inequalities, there is room for improving the Lagrangian lower bound. Introducing a nonnegative vector $y \in Y$ of slack variables y_{jk}, where Y denotes the set of all slack vectors that correspond to a feasible solution to the primal problem, we rewrite the precedence constraints as

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

Note that y_{jk} is the waiting time of J_k after its virtual release by J_j, i.e., after C_j. If J_j has two immediate successors J_k and J_l, then one of these cannot be started immediately after J_j has been finished. As a result, y_{jk} or y_{jl} will be positive. Similarly, if J_l has two immediate predecessors J_j and J_k, then y_{jl} or y_{kl} will be positive. Hence, the null vector is in Y only if each job has no more than one predecessor and one successor, in which case the problem is solvable in polynomial time. We now compute a lower bound on the slack variable problem $\min\{\sum_{(J_j, J_k) \in A} \lambda_{jk} y_{jk} \mid y \in Y\}$.

Define $\mathscr{U}$ and $\mathscr{V}$ as the set of jobs with at least two successors and two predecessors, respectively. For any $J_u \in \mathscr{U}$, consider the problem

$$F'(\lambda, u) = \min\left\{ \sum_{J_j \in \mathscr{S}_u} \lambda_{uj} y_{uj} \mid y \in Y \right\}.$$

The variables y_{uj} may be interpreted as the start times of a feasible schedule restricted to the job set $\mathscr{S}_u$ and subject to no precedence constraints. It follows that $F'(\lambda, u)$ is obtained by using Smith's ratio rule. Namely, if $\mathscr{S}_u = \{J_{j_1}, J_{j_2}, \ldots, J_{j_q}\}$ with

$$\frac{\lambda_{uj_1}}{p_{j_1}} \geqslant \frac{\lambda_{uj_2}}{p_{j_2}} \geqslant \cdots \geqslant \frac{\lambda_{uj_q}}{p_{j_q}},$$

then

$$F'(\lambda, u) = \lambda_{uj_1} 0 + \lambda_{uj_2} p_{j_1} + \cdots + \lambda_{uj_q}(p_{j_1} + \cdots + p_{j_{q-1}}).$$

For any $J_v \in \mathscr{V}$, consider the problem

$$F''(\lambda, v) = \min\left\{ \sum_{J_j \in \mathscr{P}_v} \lambda_{jv} y_{jv} \mid y \in Y \right\}.$$

The variables y_{jv} may be interpreted as the start times of a feasible schedule restricted to the job set $\mathscr{P}_v$ and subject to no precedence constraints. Therefore, if $\mathscr{P}_v = \{J_{k_1}, J_{k_2}, \ldots, J_{k_r}\}$, we have that

$$F''(\lambda, v) = \lambda_{k_1 v} 0 + \lambda_{k_2 v} p_{k_1} + \cdots + \lambda_{k_q v}(p_{k_1} + \cdots + p_{k_{r-1}}),$$

where

$$\frac{\lambda_{k_1 v}}{p_{k_1}} \geqslant \frac{\lambda_{k_2 v}}{p_{k_2}} \geqslant \cdots \geqslant \frac{\lambda_{k_r v}}{p_{k_r}}.$$

Theorem 1. *For any $\lambda \geqslant 0$, we have that*

$$\sum_{J_u \in \mathscr{U}} F'(\lambda, u) + \sum_{J_v \in \mathscr{V}} F''(\lambda, v)$$

is a valid lower bound for the slack variable problem.

Proof. Consider an arbitrary feasible schedule σ, and let y be the corresponding slack vector. For each pair of jobs J_j and J_k with J_k scheduled after J_j, define $\mathscr{S}_{jk}$ as the set of jobs scheduled in-between, define $T_{jk}(\mathscr{S}_j)$ as the sum of the processing times of the jobs in $\mathscr{S}_{jk}$ that are not in $\mathscr{S}_j$, and let $T_{jk}(\mathscr{P}_k)$ be the sum of the processing times of the jobs in $\mathscr{S}_{jk}$ that are not in $\mathscr{P}_k$. We have that

$$F'(\lambda, u) \leqslant \sum_{J_j \in \mathscr{S}_u} \lambda_{uj}\left[y_{uj} - T_{uj}(\mathscr{S}_u)\right]$$

and

$$F''(\lambda, v) \leqslant \sum_{J_j \in \mathscr{P}_v} \lambda_{jv}\left[y_{jv} - T_{jv}(\mathscr{P}_v)\right].$$

Summing over all $J_u \in \mathscr{U}$ and all $J_u \in \mathscr{V}$ yields the inequality

$$\begin{aligned}\sum_{J_u \in \mathscr{U}} F'(\lambda, u) + \sum_{J_v \in \mathscr{V}} F''(\lambda, v) \leqslant & \sum_{J_u \in \mathscr{U}} \sum_{J_j \in \mathscr{S}_u} \lambda_{uj} y_{uj} + \sum_{J_v \in \mathscr{V}} \sum_{J_j \in \mathscr{P}_v} \lambda_{jv} y_{jv} \\ & - \sum_{J_u \in \mathscr{U}} \sum_{J_j \in \mathscr{S}_u} \lambda_{uj} T_{uj}(\mathscr{S}_u) \\ & - \sum_{J_v \in \mathscr{V}} \sum_{J_j \in \mathscr{P}_v} \lambda_{jv} T_{jv}(\mathscr{P}_v).\end{aligned}$$

The first two terms of the right-hand side sum up to

$$\sum_{(J_k, J_l) \in A} \lambda_{jk} y_{jk} + \sum_{(J_u, J_v) \in \bar{A}} \lambda_{uv} y_{uv},$$

where $\bar{A}$ is the subset of A containing all arcs (J_u, J_v) with $J_u \in \mathscr{U}$ and $J_v \in \mathscr{V}$. Hence, we are done if we prove that

$$\sum_{(J_u, J_v) \in \bar{A}} \lambda_{uv} y_{uv} \leqslant \sum_{J_u \in \mathscr{U}} \sum_{J_j \in \mathscr{S}_u} \lambda_{uj} T_{uj}(\mathscr{S}_u) + \sum_{J_v \in \mathscr{V}} \sum_{J_j \in \mathscr{P}_v} \lambda_{jv} T_{jv}(\mathscr{P}_v).$$

To that end, consider an arbitrary arc $(J_u, J_v) \in \bar{A}$. We have that $y_{uv} = \sum_{J_j \in \mathscr{S}_{uv}} p_j = \sum_{J_j \in \mathscr{S}_{uv} \cap \mathscr{S}_u} p_j + T_{uv}(\mathscr{S}_u)$ and that $y_{uv} = \sum_{J_j \in \mathscr{S}_{uv} \cap \mathscr{P}_v} p_j + T_{uv}(\mathscr{P}_v)$. Since A equals its transitive reduction, the sets $\mathscr{S}_u$ and $\mathscr{P}_v$ are disjoint, implying that the sets $\mathscr{S}_{uv} \cap \mathscr{S}_u$

and $\mathscr{S}_{uv} \cap \mathscr{P}_v$ are disjoint. Hence, we have that

$$\begin{aligned} 2y_{uv} &= T_{uv}(\mathscr{S}_u) + T_{uv}(\mathscr{P}_v) + \sum_{J_j \in \mathscr{S}_{uv} \cap \mathscr{S}_u} p_j + \sum_{J_j \in \mathscr{S}_{uv} \cap \mathscr{P}_v} p_j \\ &= T_{uv}(\mathscr{S}_u) + T_{uv}(\mathscr{P}_v) + \sum_{J_j \in \mathscr{S}_{uv} \cap \{\mathscr{S}_u \cup \mathscr{P}_v\}} p_j \\ &\leqslant T_{uv}(\mathscr{S}_u) + T_{uv}(\mathscr{P}_v) + y_{uv}, \end{aligned}$$

implying that for each arc $(J_u, J_v) \in \bar{A}$ we have that

$$y_{uv} \leqslant T_{uv}(\mathscr{S}_u) + T_{uv}(\mathscr{P}_v).$$

Using this inequality, we obtain

$$\begin{aligned} \sum_{(J_u, J_v) \in \bar{A}} \lambda_{uv} y_{uv} &\leqslant \sum_{(J_u, J_v) \in \bar{A}} \lambda_{uv} [T_{uv}(\mathscr{S}_u) + T_{uv}(\mathscr{P}_v)] \\ &\leqslant \sum_{J_u \in \mathscr{U}} \sum_{J_j \in \mathscr{S}_u} \lambda_{uj} T_{uj}(\mathscr{S}_u) + \sum_{J_v \in \mathscr{V}} \sum_{J_j \in \mathscr{P}_v} \lambda_{jv} T_{jv}(\mathscr{P}_v). \end{aligned}$$ □

Note that this lower bound on the slack variable problem is computed in $O(n \log n)$ time. We evaluated its merits in the following way. For any instance of the problem, we first applied Van de Velde's ascent direction method to find a good Lagrangian multiplier $\bar{\lambda}$ and its corresponding lower bound $L(\bar{\lambda})$; we then computed the improved lower bound $\mathrm{LB}(\bar{\lambda}) = L(\bar{\lambda}) + \sum_{J_u \in \mathscr{U}} F'(\bar{\lambda}, u) + \sum_{J_v \in \mathscr{V}} F''(\bar{\lambda}, v)$. We feel that finding a good approximate solution $\hat{\lambda}$ for the problem $\max\{\mathrm{LB}(\lambda) \mid \lambda \geqslant 0\}$ and comparing it with $L(\bar{\lambda})$ is not meaningful. This problem has not the same agreeable properties as the original Lagrangian dual problem $\max\{L(\lambda) \mid \lambda \geqslant 0\}$, which makes it much more expensive to find such a $\hat{\lambda}$. We tested $\mathrm{LB}(\bar{\lambda})$ against $L(\bar{\lambda})$ on instances that were generated in the way Potts [21] proposes; i.e., the processing times were drawn from the discrete uniform distribution [1, 100] and the weights were drawn from the discrete uniform distribution [1, 10]. The precedence graph was induced by the probability p by which

Table 1
Computational results for $n = 40$

p	Median value of $100 \times L(\bar{\lambda})/z$	Median value of $100 \times \mathrm{LB}(\bar{\lambda})/z$
0.01	100.00	100.00
0.02	100.00	100.00
0.04	99.87	100.00
0.06	99.77	99.95
0.08	99.66	99.89
0.10	99.37	99.72
0.15	98.72	99.23
0.20	98.34	98.90
0.30	97.65	98.41
0.50	97.22	98.15

each arc (J_j, J_k) with $j < k$ was included. The resulting graph was then replaced by its transitive reduction.

In Table 1, we show the computational results for $n = 40$ for various values of p. The entries show the median values of $100 \times L(\bar{\lambda})/z$ and $100 \times \mathrm{LB}(\bar{\lambda})/z$ over 40 instances, where z denotes the optimal solution value. Table 1 shows that reformulating the problem by using slack variables gives a considerable reduction of the duality gap for any value of p. Potts [21] reports that the value $p = 0.15$ generates the most difficult class of instances. For small values of p, the duality gap is small, since few constraints are involved. As can be expected, the duality gap is considerably bigger for large values of p; nonetheless, large values of p generate the relatively easiest instances, because the solution space is not so big.

Our branch-and-bound algorithm is too rudimentary to provide complete results for $n \geqslant 40$. Our limited computational experience with larger instances, however, indicates that the results for $n = 40$ are typical, since the performance of $L(\bar{\lambda})$ and $\mathrm{LB}(\bar{\lambda})$ does not significantly vary with n.

We now comment on the relationship between our bound and the polyhedral bound by Queyranne and Wang [25]. Queyranne and Wang proceed from the same formulation as we did. They first add the so-called *parallel* inequalities. This class of inequalities forms a complete description of the facets of the polytope of the feasible solutions for the $1 \| \sum_{j=1}^{n} w_j C_j$ problem; see [24,33]. In fact, these inequalities are just the explicit rendering of condition (2), that is, the machine capacity and availability constraints. Queyranne [23] gives an $O(n \log n)$ separation algorithm for these inequalities. Accordingly, the Lagrangian problem (L_λ) is solvable as a linear program and hence the best Lagrangian bound $L(\lambda^*)$ is equal to the linear programming bound lifted by these parallel inequalities. Then, the so-called *simple-series* inequalities are added. The simple-series inequalities along with the parallel inequalities form a complete description of the facets of the polytope of the feasible solutions for the problems $F'(\lambda, u)$ and $F''(\lambda, v)$. Queyranne and Wang [25] give an $O(n^2)$ separation algorithm for these inequalities and proved that the best improved Lagrangian bound $\max\{\mathrm{LB}(\lambda) \mid \lambda \geqslant 0\}$, which can be found by the subgradient method, is equal to the polyhedral bound. Accordingly, the bound $\mathrm{LB}(\bar{\lambda})$ is generally weaker than the polyhedral bound. It is computed much faster, however.

4.2. Scheduling multi-operation jobs

In this section, we consider the following two-machine flow-shop problem. There are two machines, M_1 and M_2, each handling no more than one job at a time and continuously available from time 0 onwards. There is a set of n jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$, each consisting of a chain of two operations. The hth operation of J_j has to be executed on M_h during a positive uninterrupted period of time p_{hj}, $h = 1, 2$, $j = 1, \ldots, n$; hence, each job passes first through M_1, then through M_2. A job can be executed by at most one machine at a time, implying that the operations of the same job may not overlap in their execution. A schedule specifies a completion time C_{hj} for the hth operation,

$h = 1, 2$, of each J_j, $j = 1, \ldots, n$, such that the above conditions are met. The completion time of J_j is then simply C_{2j}.

We consider the problem of minimizing $\sum_{j=1}^{n} C_{2j}$, which is $\mathcal{NP}$-hard in the strong sense [7]. It is well known that for this problem it suffices to optimize over all *permutation schedules* with no machine idle time on M_1 before the execution of jobs; see e.g., [3]; a permutation schedule is a schedule in which every machine has the same job sequence. The best optimization algorithm is due to Van de Velde [31] and solves instances up to twenty jobs. This branch-and-bound algorithm employs a Lagrangian lower bound that is obtained from the following formulation. Determine completion times C_{hj} that minimize

$$\sum_{j=1}^{n} C_{2j}$$

subject to

the precedence constraints between the operations of the jobs, (3)

the capacity and availability constraints of the machines, (4)

and to the condition that

the C_{hj} form a permutation schedule with no idle time on M_1. (5)

Condition (5) is redundant for the primal problem, but it is not redundant for the Lagrangian problem and the slack variable problem. The conditions (3) are formulated as

$$C_{2j} \geqslant C_{1j} + p_{2j}, \quad \text{for } j = 1, \ldots, n. \tag{3'}$$

A nonnegative vector of multipliers $\lambda = (\lambda_1, \ldots, \lambda_n)$ is introduced for dualizing conditions (3'). This gives the Lagrangian problem of finding $L(\lambda)$, which is the minimum of

$$\sum_{j=1}^{n} \left[\lambda_j C_{1j} + (1 - \lambda_j) C_{2j} + \lambda_j p_{2j} \right]$$

subject to conditions (4) and (5). In order to prevent that $L(\lambda)$ becomes arbitrarily small, we require that $\lambda \leqslant 1$.

This Lagrangian problem is a linear ordering problem, which in general is $\mathcal{NP}$-hard. In this application, however, it is solvable in polynomial time if we choose $\lambda_j = c$ for each j, $j = 1, \ldots, n$, for some constant $0 \leqslant c \leqslant 1$. The Lagrangian problem is then solved by an analogon of Smith's rule which schedules the jobs in order of nondecreasing values $cp_{1j} + (1 - c)p_{2j}$.

Let $L(c)$ denote the optimal solution value of the Lagrangian problem with $\lambda_j = c$ for each j, $j = 1, \ldots, n$. The restricted Lagrangian dual problem $\max\{L(c) \mid 0 \leqslant c \leqslant 1\}$ is solvable in $O(n^2 \log n)$ time, but a linear-time approximation algorithm based upon binary search over the interval [0, 1] performs sufficiently well. For details, we refer to [31].

We now formulate conditions (3') as equalities. Introducing a nonnegative vector $y = (y_1, \ldots, y_n) \in Y$ of slack variables, we write

$$C_{2j} = C_{1j} + p_{2j} + y_j, \quad \text{for } j = 1, \ldots, n,$$

where Y denotes the set of all slack vectors corresponding to schedules that are feasible with respect to the conditions (3)–(5). This leads to the slack variable problem $\min\{cy \mid y \in Y\}$.

We compute a lower bound on the slack variable problem as follows. Note that y_j can be interpreted as the time that J_j spends waiting in a buffer between the two machines. Let a_{ij} denote the minimal waiting time of J_j if it is immediately preceded by J_i. Since idle time on M_1 is not allowed, we have that $a_{ij} = \max\{p_{2i} - p_{1j}, 0\}$. For any permutation schedule π, we have that

$$\sum_{j=1}^{n} y_{\pi(j)} \geqslant \sum_{j=1}^{n-1} a_{\pi(j),\pi(j+1)},$$

where $\pi(j)$ denotes the index of the job that occupies the jth position in π. Hence, the optimal solution value of the problem

$$\min\left\{ c \sum_{j=1}^{n-1} a_{\pi(j),\pi(j+1)} \mid \pi \in \Pi \right\},$$

where Π is the set containing all permutations of $\{1, \ldots, n\}$, is a lower bound on the slack variable problem. This is a *Hamiltonian path* problem with distances $a_{ij} = \max\{p_{2i} - p_{1j}, 0\}$ for $i, j = 1, \ldots, n$. In general, the Hamiltonian path problem is $\mathcal{NP}$-hard in the strong sense; this problem, however, is solvable in $O(n \log n)$ time, since it can be transformed into the so-called Maximal Traveling Salesman Problem, for which Van Dal et al. [30] show that it reduces to the Gilmore–Gomory Traveling Salesman Problem [9]. The transformation is achieved by adding a dummy job J_0 with processing times $p_{1,0} = \infty$ and $p_{2,0} = \max\{t_2 - t_1, 0\}$, where t_h, $h = 1, 2$ denotes the earliest time that M_h becomes available for processing.

We implement the improved Lagrangian lower bound in the following way. Note that the solution to this Hamiltonian path problem does not depend on the Lagrangian multiplier c. We therefore address the Hamiltonian problem first, and actually we compute a lower bound on the Hamiltonian path problem rather than solving it to optimality. The optimization algorithm proceeds by *matching* and *patching*. We only did the former; this takes $O(n)$ time once the jobs are prearranged and gives a strong lower bound, say, B. For any c, we thus have that the improved Lagrangian bound

Table 2
Computational results for the flow-shop problem

n	Median value of $100 \times L(c^*)/z$	Median value of $100 \times \mathrm{LB}(\bar{c})/z$
10	96.35	96.78
15	96.89	97.19
20	96.76	96.99
25	96.81	97.29
30	96.54	96.79

$LB(c) = L(c) + cB$. Subsequently, we conduct binary search over the interval [0, 1] to find the best such bound $LB(\bar{c})$.

We tested the quality of the improved lower bound $LB(\bar{c})$ against the best "old" bound $L(c^*)$, where c^* is also obtained by binary search over the interval [0, 1], on instances with 10, 15, 20, 25 and 30 jobs. For each value of n, we generated 40 instances. The processing times for each job were drawn from the discrete uniform distribution [1, 10].

In Table 2, we show the median values of $100 \times L(c^*)/z$ and $100 \times LB(\bar{c})/z$, where z denotes the optimal solution value. We note that if a branch-and-bound algorithm is used with a *forward sequencing branching rule*, then the value $p_{2,0}$ tends to increase if we go down the search tree; such an increase has a positive effect on the performance of $LB(\bar{c})$. A forward sequencing branching rule builds a search tree in which nodes at level k correspond to an initial partial sequence in which jobs are assigned to the first k positions.

4.3. The total weighted tardiness problem

The setting for the total weighted tardiness problem is as follows. A set of n independent jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$ has to be scheduled on a single machine that can handle no more than one job at a time. The machine and the jobs are assumed to be continuously available from time zero onwards. Each J_j, $j = 1, \ldots, n$, requires processing during an uninterrupted processing period of a given length p_j, has a weight w_j, and a due date d_j by which it ideally should have been completed. Given a schedule, the tardiness of J_j is defined as $T_j = \max\{C_j - d_j, 0\}$. The objective is to find a schedule that minimizes total weighted tardiness $\sum_{j=1}^{n} w_j T_j$.

The total weighted tardiness problem is $\mathcal{NP}$-hard in the strong sense [14,18]. It is a challenging machine scheduling problem for which many optimization algorithms have been developed; see [1] for a survey. The best branch-and-bound algorithm is due to Potts and Van Wassenhove [22] and solves instances up to 40 jobs. They use the following formulation to obtain a Lagrangian lower bound. Determine job tardinesses $T_1, \ldots, T_n$ and job completion times $C_1, \ldots, C_n$ that minimize

$$\sum_{j=1}^{n} w_j T_j$$

subject to

$$T_j \geqslant C_j - d_j, \quad \text{for } j = 1, \ldots, n, \tag{6}$$

$$T_j \geqslant 0, \quad \text{for } j = 1, \ldots, n, \tag{7}$$

$$\text{the capacity and availability constraints of the machine.} \tag{8}$$

Conditions (6) and (7) reflect the definition of job tardiness. Using a vector $\lambda = (\lambda_1, \ldots, \lambda_n) \geqslant 0$ of Lagrangian multipliers to dualize the conditions (6), Potts and

Van Wassenhove [22] obtain the following Lagrangian problem: determine the value $L(\lambda)$, which is the minimum of

$$\sum_{j=1}^{n} (w_j - \lambda_j)T_j + \sum_{j=1}^{n} \lambda_j(C_j - d_j)$$

subject to the conditions (7) and (8).

Since the conditions (7) affect only the first component of the Lagrangian objective function and the conditions (8) only the second one, the Lagrangian problem decomposes into two subproblems. If $w_j - \lambda_j \leqslant 0$ for some j, $j = 1, \ldots, n$, then we get $T_j = \infty$, resulting in $L(\lambda) = -\infty$. We therefore require that $w_j - \lambda_j \geqslant 0$ for each j, $j = 1, \ldots, n$, and we minimize the first component by setting $T_j = 0$ for each j. The second component reduces to the problem of minimizing $\sum_{j=1}^{n} \lambda_j C_j$, which is simply solved by scheduling the jobs in order of nondecreasing values λ_j / p_j in the interval $[0, \sum_{j=1}^{n} p_j]$.

Potts and Van Wassenhove propose a linear-time heuristic algorithm to set the Lagrangian multipliers. This algorithm generally gives a very good approximation of the Lagrangian dual solution value, but for some classes of instances the Lagrangian dual solution value is quite a weak lower bound. We propose an improvement by using slack variables. We introduce a nonnegative vector $y = (y_1, \ldots, y_n) \in Y$ of slack variables, where Y is the set of all slack vectors corresponding to feasible schedules, and rewrite the conditions (6) as

$$T_j = C_j - d_j + y_j, \quad \text{for } j = 1, \ldots, n.$$

Note that y_j is equivalent to the *earliness* of J_j, which in machine scheduling theory is defined as $E_j = \max\{d_j - C_j, 0\}$. If we dualize the equality conditions, we get the following Lagrangian problem: find $L'(\lambda)$, which is the minimum of

$$\sum_{j=1}^{n} (w_j - \lambda_j)T_j + \sum_{j=1}^{n} \lambda_j E_j + \sum_{j=1}^{n} \lambda_j(C_j - d_j)$$

subject to (7), (8) and $(E_1, \ldots, E_n) \in Y$. Since the term $\sum_{j=1}^{n} (w_j - \lambda_j)T_j$ does not contribute to the bound $L(\lambda)$, we include it in the slack variable problem; as we will see, we gain by it, since in no nontrivial problem we can have that both $T_j = 0$ and $E_j = 0$ for all j. This gives the *modified* slack variable problem of minimizing

$$\sum_{j=1}^{n} \left[(w_j - \lambda_j)T_j + \lambda_j E_j\right]$$

subject to (7), (8) and $(E_1, \ldots, E_n) \in Y$.

No tardiness or earliness penalties are incurred if it is feasible to execute each J_j in the interval $[d_j - p_j, d_j]$; in this case, however, the original problem would be trivial. Suppose that there are jobs J_k and J_l with $0 < \lambda_k < w_k$ and $0 < \lambda_l < w_l$ for which the ideal execution intervals $[d_k - p_k, d_k]$ and $[d_l - p_l, d_l]$ overlap. We call J_k and J_l *conflicting* jobs, since J_k or J_l will be early or tardy. The minimum penalty to settle the conflict is obtained by evaluating four options: executing J_k and J_l in the interval $[d_k - p_k, d_k + p_l]$ with J_k before J_l, scheduling J_k and J_l in the interval $[d_l - p_l -$

Table 3
Computational results for difficult instances

n	Median value of $100\times L(\bar{\lambda})/z$	Median value of $100\times \mathrm{LB}'(\bar{\lambda})/z$	Median value of $100\times \mathrm{LB}''(\bar{\lambda})/z$
20	62.37	67.47	69.92
30	58.53	62.35	64.84
40	59.14	63.65	65.93

p_k, d_l] with J_k before J_l, scheduling J_k and J_l in the interval $[d_l - p_l, d_l + p_k]$ with J_l before J_k, and scheduling J_k and J_l in the interval $[d_k - p_k - p_l, d_k]$ with J_l before J_k. All other options are dominated by these four. The minimum penalty is readily computed, since the cost of scheduling J_k before J_l is equal to $\min\{\lambda_k, w_l - \lambda_l\}(d_k - d_l - p_l)$, and the minimum cost of scheduling J_l before J_k is equal to $\min\{\lambda_l, w_k - \lambda_k\}(d_l - d_k - p_k)$.

We compute a lower bound for the modified slack variable problem as follows. First, we arrange the jobs in nondecreasing order of the due dates, and renumber them accordingly. Then, we identify pairs of adjacent conflicting jobs; no job may appear in more than one pair. Finally, we compute for each pair the minimum penalty to settle the conflict. The sum of these penalties is a lower bound on the optimal solution value of the modified slack variable problem; adding this sum to $L(\lambda)$ gives the improved Lagrangian lower bound $\mathrm{LB}'(\lambda)$. Like Potts and Van Wassenhove's lower bound, $\mathrm{LB}'(\lambda)$ is computed in $O(n)$ time if the jobs are prearranged. In a similar fashion, we can compute a lower bound on the slack variable problem by specifying triples of adjacent conflicting jobs. To compute the minimum penalty for such a triple, we need to evaluate twelve options. This gives rise to the alternative improved Lagrangian lower bound $\mathrm{LB}''(\lambda)$, which is also computed in $O(n)$ time. Note that $\mathrm{LB}''(\lambda)$ does not dominate $\mathrm{LB}'(\lambda)$, but $\mathrm{LB}''(\lambda)$ will usually be greater in case of many conflicts.

We tested the improved lower bounds $\mathrm{LB}'(\bar{\lambda})$ and $\mathrm{LB}''(\bar{\lambda})$ against the traditional Lagrangian lower bound $L(\bar{\lambda})$, where $\bar{\lambda}$ is the Lagrangian multiplier obtained by Potts and Van Wassenhove's algorithm, on instances with 20, 30 and 40 jobs that were generated in the same way as Potts and Van Wassenhove generated theirs. The processing times were generated from the discrete uniform distribution [10, 100], and the weights were generated from the discrete uniform distribution [1, 10]. The due dates were generated from the discrete uniform distribution $[P(1 - T - \frac{1}{2}R), P(1 - T + \frac{1}{2}R)]$,

Table 4
Computational results for instances of average difficulty

n	Median value of $100\times L(\bar{\lambda})/z$	Median value of $100\times \mathrm{LB}'(\bar{\lambda})/z$	Median value of $100\times \mathrm{LB}''(\bar{\lambda})/z$
20	92.27	92.69	94.42
30	92.63	93.31	93.70
40	91.70	92.24	92.67

where $P = \sum_{j=1}^{n} p_j$, and where R and T are parameters. For both parameters, we considered the values 0.2, 0.4, 0.6, 0.8 and 1.0. In Table 3, we show the results for $T = 0.4$ and $R = 0.6$; this choice generates the class of instances for which Potts and Van Wassenhove's lower bound has its worst performance. Table 3 shows that the improvement we achieve is substantial; it suggests that it may be worthwhile to consider a more sophisticated lower bound procedure for the slack variable problem. In Table 4, we give the results for instances of average difficulty, generated by choosing $T = 0.2$ and $R = 0.8$.

5. Conclusions

We have shown that better Lagrangian bounds can be obtained by addressing the slack variable problem that results from reformulating nasty inequality constraints as equalities. In each application, the computation of the improved Lagrangian lower bound proceeded in two phases. In the first phase, we dealt with the Lagrangian dual problem and computed the traditional Lagrangian lower bound. The Lagrangian multiplier found here served as input for the second phase, in which we computed a lower bound on the slack variable problem. The improved Lagrangian lower bound was then set equal to the traditional bound plus the bound on the slack variable problem. In this way, we attained for each application significant improvements.

The main conclusion of this paper is that the slack variable problem deserves to be investigated if Lagrangian relaxation is used to compute bounds.

Acknowledgements

The authors like to thank Maurice Queyranne and the referees for their constructive comments.

References

[1] T.S. Abdul-Razaq, C.N. Potts and L.N. Van Wassenhove, "A survey of algorithms for the single machine total weighted tardiness scheduling problem," *Discrete Applied Mathematics* 26 (1990) 235–253.

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

[3] R.W. Conway, W.L. Maxwell and L.W. Miller, *Theory of Scheduling* (Addison-Wesley, Reading, MA, 1967).

[4] 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.

[5] M.L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Management Science* 27 (1981) 1–18.

[6] M.L. Fisher, "An applications oriented guide to Lagrangian relaxation," *Interfaces* 15 (1985) 10–21.

[7] M.R. Garey, D.S. Johnson and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research* 13 (1976) 330–348.

[8] A.M. Geoffrion, "Lagrangian relaxation and its uses in integer programming," *Mathematical Programming Study* 2 (1974) 82–114.

[9] P.C. Gilmore and R.E. Gomory, "Sequencing a one state-variable machine: a solvable case of the traveling salesman problem," *Operations Research* 12 (1964) 655–679.
[10] M. Held and R.M. Karp, "The traveling-salesman problem and minimum spanning trees," *Operations Research* 18 (1970) 1138–1162.
[11] M. Held and R.M. Karp, "The traveling-salesman problem and minimum spanning trees: Part II," *Mathematical Programming* 1 (1971) 6–25.
[12] J.A. Hoogeveen, H. Oosterhout and S.L. van de Velde, "New lower and upper bounds for scheduling around a small common due date," *Operations Research* 42 (1994) 102–110.
[13] W.A. Horn, "Single-machine job sequencing with treelike precedence ordering and linear delay penalties," *SIAM Journal of Applied Mathematics* 23 (1972) 189–202.
[14] E.L. Lawler, "A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness," *Annals of Discrete Mathematics* 1 (1977) 331–342.
[15] E.L. Lawler, "Sequencing jobs to minimize total weighted completion time subject to precedence constraints," *Annals of Discrete Mathematics* 2 (1978) 75–90.
[16] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, "Sequencing and scheduling: algorithms and complexity," in: S.C. Graves, A.H.G. Rinnooy Kan and P. Zipkin, eds., *Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory* (North-Holland, Amsterdam, 1993) pp. 445–522.
[17] J.K. Lenstra and A.H.G. Rinnooy Kan, "Complexity of scheduling under precedence constraints," *Operations Research* 26 (1978) 22–35.
[18] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, "Complexity of machine scheduling problems," *Annals of Discrete Mathematics* 1 (1977) 343–362.
[19] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations* (Wiley, Chichester, 1990).
[20] R. Müller, Personal communication (1993).
[21] 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.
[22] 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.
[23] M. Queyranne, "Structure of a simple scheduling polyhedron," *Mathematical Programming* 58 (1993) 263–286.
[24] M. Queyranne and Y. Wang, "Single-machine scheduling polyhedra with precedence constraints," *Mathematics of Operations Research* 16 (1991) 1–20.
[25] M. Queyranne and Y. Wang, "A cutting plane procedure for precedence-constrained single machine scheduling," Working Paper, University of British Columbia, Vancouver, Canada (1991).
[26] J.F. Shapiro, "A survey of Lagrangian techniques for discrete optimization," *Annals of Discrete Mathematics* 5 (1979) 113–138.
[27] J.B. Sidney, "Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs," *Operations Research* 23 (1975) 283–298.
[28] W.E. Smith, "Various optimizers for single-stage production," *Naval Research Logistics Quarterly* 3 (1956) 59–66.
[29] J.P. Sousa and L.A. Wolsey, "A time indexed formulation of non-preemptive single machine scheduling problems," *Mathematical Programming* 54 (1992) 353–367.
[30] R. van Dal, J.A.A. van der Veen and G. Sierksma, "Small and large TSP: Two polynomially solvable cases of the traveling salesman problem," *European Journal of Operational Research* 69 (1993) 107–120.
[31] 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.
[32] S.L. van de Velde, "Dual decomposition of machine scheduling problems" in: Machine scheduling and Lagrangian relaxation, Ph.D. Thesis, (Chapter 2) CWI, Amsterdam (1991); an earlier version appeared in: *Proceedings of the First Conference on Integer Programming and Combinatorial Optimization*, University of Waterloo (Waterloo, 1990) pp. 495–507.
[33] A. von Arnim, U. Faigle and R. Schrader, "The permutahedron of series-parallel posets," *Discrete Applied Mathematics* 28 (1990) 3–9.