Open access • Journal Article • DOI:10.1287/IJOC.1040.0069

Near-Optimal Solutions of Large-Scale Single-Machine Scheduling Problems

— Source link

Pasquale Avella, Maurizio Boccia, Bernardo D'Auria

Institutions: University of Sannio, University of Salerno

Published on: 15 Apr 2005 - Informs Journal on Computing (INFORMS)

Topics: Job shop scheduling, Single-machine scheduling, Personal computer, Heuristic and Relaxation (approximation)

Near-Optimal Solutions of Large-Scale Single Machine Scheduling Problems

Pasquale Avella
RCOST - Research Center on Software Technology
Università del Sannio
Corso Garibaldi, 107
82100 Benevento, Italy
avella@unisannio.it

Maurizio Boccia, Bernardo D'Auria
Centro di Ricerca in Matematica Pura e Applicata
Università di Salerno
Via Ponte Don Melillo
84084 Fisciano (SA) - Italy
boccia@crmpa.unisa.it
dauria@diima.unisa.it

July 16, 2002

Abstract

Single Machine Scheduling Problems with Release Dates (SMSP) concern the optimal allocation of a set of jobs on a single machine, that cannot process more than one job at a time. Each job is ready for processing at a release date and it has to be processed without interruption. The goal is to minimize the total weighted completion time of the jobs.

In this paper the time-indexed formulation is considered and a new lagrangian heuristic is proposed, based on the observation that lagrangian relaxation of job constraints leads to a Weighted Stable Set problem on an Interval Graph. The problem is polinomially solvable by a dynamic programming algorithm.

Computational experience is reported, showing that instances up to 400 jobs and maximum processing time 50 (around 5 millions variables) are solved in less than 40 minutes on a Personal Computer, yielding duality gaps never exceeding 3.0%. We also test a set of hard instances, built to produce bad performances, where we yield duality gaps less than 5%.

Keywords: Scheduling, Lagrangian Relaxation, Interval Graphs.

1 Introduction

Let $J = \{1, 2, \ldots, n\}$ be a set of jobs to be scheduled on a single machine that can handle at most one job at a time. Each job $j \in J$ requires uninterrupted processing on the machine for a period of length p_j and it is ready for processing at the *release date* r_j. Both p_j and r_j are assumed to be integral. Let w_j be a weight associated with job j and let C_j denote its completion time. The *Single Machine Scheduling Problem with Release Dates (SMSP)* consists of finding a non-preemptive schedule minimizing the total weighted completion time $\sum_j w_j C_j$.

In scheduling theory this is a basic problem and it is denoted as $1|r_j|\sum w_j C_j$. Lenstra, Rinnooy Kan and Brucker [17] proved that $1|r_j|\sum w_j C_j$ is NP-hard even if $w_j = 1$ for all j. Another, more difficult [27], variant of the problem is $1|r_j|\sum w_j F_j$, where $F_j = C_j - r_j$.

Dyer and Wolsey [9] examined several formulations of $SMSP$, proving that the *time-indexed formulation*, introduced by Sousa and Wolsey [25], is the strongest. It is based on time-discretization over a time horizon T and has a binary variable x_{jt} to say that job $j \in J$ is completed at time t ($x_{jt} = 1$).

Crama and Spieksma [8] studied the polyhedral structure of the feasible solutions of the time-indexed formulation, when all processing times are equal. Van den Akker, van Hoesel and Savelsbergh [1] characterized all facet-defining inequalities with integral coefficients and right-hand-side 1 or 2.

Waterer, Johnson and Savelsbergh [29] established the equivalence between $SMSP$ and the Stable Set problem to derive polyhedral results.

For a more general study of polyhedral approaches to machine scheduling problems we refer the reader to the pioneering paper of Balas [4] and to the comprehensive surveys by Queyranne [21] and Queyranne and Schulz [22].

Belouadah, Posner and Potts [5] proposed a Branch-and-Bound algorithm based on a combinatorial lower bound.

Van den Akker, van Hoesel and Savelsbergh [1] proposed a Branch-and-Cut algorithm. In spite of the good quality of lower bounds, they cannot solve instances with more than 30 jobs and maximum processing time $p_{max} = 10$, due to the huge size of the time-indexed formulation.

To alleviate these difficulties, van den Akker, Hurkens and Savelsbergh [2] proposed a Branch-and-Price algorithm based on Dantzing-Wolfe decomposition, solving instances with 30 jobs and $p_{max} = 100$.

Approximation algorithms for $SMSP$ build a feasible schedule from the fractional solution of the LP-relaxation [**?**]. Surveys of these approaches are given in Schulz [24], Goemans [11], Goemans, Queyranne, Schulz, Skutella and Wang [12], Hall [14], Hall, Schulz, Shmoys and Wein [15].

Savelsbergh, Uma and Wein [23] and Uma [26] report computational experience with approximation algorithms for $1|r_j|\sum w_j Fj$ on 100 jobs instances.

A lagrangian heuristic based on the weaker *completion time* formulation of $SMSP$ has been proposed by van de Velde [28]. Mohring, Schulz, Stark and Uetz [19] [20] presented a lagrangian heuristic for the time-indexed formulation of the Project Scheduling Problem with Precendence Contraints, where the relaxed problem is a Weighted Stable Set Problem on a comparability graph, solved by min-cut computations. A Tabu Search and a Genetic Algorithm have been developed, respectively, by Laguna, Barnes and Glover [16] and by Chen, Elizandro, Liu and Miller [7].

In this paper we present a lagrangian heuristic for SMSP, based on the time-indexed formulation, which yields near-optimal solutions for large-scale instances.

The remainder of the paper is organized as follows. In section 2 we review the time-indexed formulation for SMSP. In section 3 we observe that lagrangian relaxation of the job constraints in the time-indexed formulation leads to a Weighted Stable Set problem on a graph $G(V, E)$ showing a special structure. We show that G is an *Interval Graph* [13] [10] and the polynomial algorithm described in Mannino and Oriolo [18] is adopted to determine the optimal solution of the relaxed problem (in [2] it is noted that capacity constraints define an Interval Matrix, which is known to be unimodular).

Since the integrality property holds for the relaxed problem, the maximum of the lagrangian relaxation coincides with the value of the LP relaxation. The lagrangian realxation is maximized through subgradient optimization. By some enhancements, we yield convergence to fairly good lower bounds, close ($\leq 0.3\%$) to the value of the LP-relaxation, in less than 250 iterations.

In section **??** we describe the upper bound heuristic which attempts to build a feasible solution by exploting the solution of the lagrangian relaxation.

In section 5, computational experience is reported, showing that the algorithm yields duality gaps never exceeding 3% for instances up to 400 jobs and maximum processing time 50 (about 5 millions variables), in less than 40 minutes on a Personal Computer. We also test a set of hard instances, built to produce bad lower bounds for the time-indexed formulation, where the algorithm yields duality gaps never exceeding 5%.

2 Time-indexed formulation

Let x_{jt} be a boolean variable which is 1 if job j is completed at time t, 0 otherwise. Let $c_{jt} = w_j t$ be the cost of completing job j at time t.

Let T denote the time horizon. We set $T = \sum_{j \in J} p_j + \max_{j \in J} r_j$ to ensure feasibility. Let $[t_1, t_2]$ denote the time interval between t_1 and t_2. For any $j \in J$ and $t \in [r_j + p_j - 1, T]$, let $\gamma(j, t) = max(r_j + p_j - 1, t - p_j + 1)$. The time-indexed formulation of $SMSP$ is:

$$
\begin{aligned}
& minimize \sum_{j \in J} \sum_{t \in [r_j + p_j - 1, T]} c_{jt} x_{jt} \\
& st \\
& \sum_{t \in [r_j + p_j - 1, T]} x_{jt} = 1, \quad j \in J \qquad (1)
\end{aligned}
$$

$$\sum_{j \in J} \sum_{s \in [\gamma(j,t),t]} x_{js} \leq 1, \quad t \in [1,T] \tag{2}$$
$$x_{jt} \in \{0,1\}, \quad j \in J, t \in [r_j + p_j - 1, T]$$

Job constraints (1) force each job to be processed. Capacity constraints (2) force the machine to process at most one job at a time.

Time-indexed formulation provides good quality lower bounds [1], at the cost of a huge number of variables and constraints: about 1 millions variables for an instance with 100 jobs and average processing time 10.

3 Lagrangian Relaxation

The lagrangian function $\Theta(\lambda)$ obtained by relaxing job constraints (1) is:

$$\begin{aligned}
\Theta(\lambda) \; = \; min \; & \sum_{j \in J} \sum_{t \in [r_j+p_j-1,T]} (c_{jt} - \lambda_j) x_{jt} \; + \sum_{j \in J} \lambda_j \\
st & \\
& \sum_{j \in J} \sum_{s \in [\gamma(j,t),t]} x_{js} \leq 1, \quad t \in [1,T] \\
& x_{jt} \in \{0,1\}, \quad j \in J, t \in [r_j + p_j - 1, T]
\end{aligned}$$

We associate with $\Theta(\lambda)$ the *intersection graph* $G(V,E)$, having a node jt, with cost $c_{jt} - \lambda_j$, for each variable x_{jt} in the formulation, i.e. $V = \{jt : j \in J, t \in [r_j + p_j - 1, T]\}$. Let is and jt be two nodes of V. The edge (is, jt) belongs to E if the variables x_{is} and x_{jt} appear in the same machine constraint (2), i.e. $(is, jt) \in E$ if either (i) $s > t$ and $t > s - p_i + 1$ or (ii) $t > s$ and $s > t - p_j + 1$.

The intersection graph G has a special structure, as noted in [2]. By associating the interval $[t - p_j + 1, t]$ with each node jt, we have that two nodes of V are adjacent if and only if their corresponding intervals intersect. It follows that the intersection graph G is an *interval graph* [13] and, for a given set $\bar{\lambda}$ of multipliers, $\Theta(\bar{\lambda})$ is equivalent to a *Weighted Stable Set Problem* on the Interval Graph G. Let $X(\bar{\lambda})$ denote the solution of such problem. In what follows we will refer to the $X(\lambda)$ as the *lagrangean solutions*.

3.1 Computing $\Theta(\lambda)$

For an interval graph [10] [18] there always exists an ordering $\{v_1, v_2, \ldots, v_n\}$ of V with the following property:

Property 3.1 *Let $1 \leq q < r < s \leq |V|$. If $(v_q, v_s) \in E$, then $(v_r, v_s) \in E$.*

Given an ordering of V satisfying property (3.1), the *Weighted Stable Set Problem* on G is polynomially solvable by a dynamic programming algorithm.

Let $k \leq |V|$ and let $\alpha_W(k)$ denote the maximum weighted stable set for the subgraph induced by the nodes $\{v_1, v_2, \ldots, v_k\}$. Let $c(i)$ be the cost of the node i and let $\delta(i)$ be the largest index of a node prior to i which is not adjacent to i. The dynamic programming algorithm is defined by the following recursion:

$$\alpha_W(i) = max(\alpha_W(i-1), \alpha_W(\delta(i)) + c(i)) \tag{3}$$

To compute $\Theta(\lambda)$ by the recursion (3), we only need to exhibit an ordering of V satisfying property 3.1.

We define an ordering $\{j_1, j_2, \ldots, j_{|J|}\}$ of the jobs and then we order the nodes of V as

$$\{j_1, 1, j_2, 1, \ldots, j_{|J|}, 1, j_1, 2, j_2, 2, \ldots, j_{|J|}, 2, \ldots, j_1, t, j_2, t, \ldots, j_{|J|}, t, \ldots, j_1, T, j_2, T, \ldots, j_{|J|}, T\}$$

We prove that this ordering satisfies property 3.1.

Lemma 3.1 *Let $1 \leq q < r < s \leq |V|$ and let $j_q t_q, j_r t_r, j_s t_s$ be, respectively, the nodes whose indices are q, r, s. If $(j_q t_q, j_s t_s) \in E$, then $(j_r t_r, j_s t_s) \in E$.*

Proof. If $(j_q t_q, j_s t_s) \in E$, then $t_q \in [\gamma(j_s, t_s), t_s]$. Since $q < r < s$ we have $t_q \leq t_r \leq t_s$ and $t_r \in [\gamma(j_s, t_s), t_s]$. It follows that $(j_r t_r, j_s t_s) \in E$. □

3.2 Subgradient Optimization

The lagrangian function $\Theta(\lambda)$ is maximized through the subgradient method, which recursively updates multipliers according to the formula:

$$\lambda_{k+1} = \lambda_k + sg \tag{4}$$

where s is the step-size and g is the subgradient. For SMSP, the generic j^{th} component of the subgradient is:

$$g_j = \sum_{t \in [r_j + p_j - 1, T]} x_{jt} - 1 \tag{5}$$

The step-size s is computed as:

$$s = \varphi \frac{Z_{UB} - \Theta(\lambda^*)}{\|g\|^2} \tag{6}$$

where Z_{UB} is an upper bound to the optimal value of $\Theta(\lambda)$, $\Theta(\lambda^*)$ is the best value of $\Theta(\lambda)$ found so far and φ is a parameter modified according to some rules.

In our computational experience we initialize Z_{UB} by a greedy algorithm. Parameter φ is usually initialized to 2 and it is halved after the lower bound has not improved for a given number (say $20 \div 30$) of iterations. For $SMSP$ we adopted a more effective updating strategy: we divide φ by 1.01 at every iteration, independently from the behavior of $\Theta(\lambda)$.

Moreover we used subgradient deflection [6] to improve convergence. At the generic iteration k, we compute the direction d_k as:

$$d_k = \frac{g_k + 0.3d_{k-1} + 0.1d_{k-2}}{1.4} \tag{7}$$

where g_k is the current subgradient vector at the generic iteration k, and d_{k-1} and d_{k-2} are the directions used in the last two iterations.

With these settings, the subgradient method converges in less than 250 iterations to fairly good lower bounds, very close (less than 0.3%) to the LP-relaxation of the time-indexed formulation as shown in Table 1 for a set of 100 job instances of $1|r_j|\sum w_jC_j$.

name	$njob$	p_max	$LgLB$	$LPLB$
T100-10-1	100	10	193200.9	193787.3
T100-20-1	100	20	420310.1	421543.3
T100-30-1	100	30	519683.7	521673.1
T100-40-1	100	40	687569.1	689853.7

Table 1: Lagrangian and LP lower bounds for 100 job instances of $1|r_j|\sum w_jC_j$.

4 The upper bound heuristic

We compute upper bounds for $SMSP$ by completing the lagrangian solutions $X(\lambda)$ associated with the optimal (or near-optimal) lagrangian multipliers to make them feasible. The approach has revealed very effective and robust for large instances too.

With each solution $X(\lambda)$ we associate the *ordered list* of jobs $J(\lambda)$, containing a job h for each $x_{ht} = 1$, where $h \prec k$ if $x_{ht_h} = 1$, $x_{kt_k} = 1$ and $t_k > t_h$.

Since the job constraints (1) are relaxed, the same job may appear in $J(\lambda)$ more than once. We refer to such jobs as the *repeated jobs*. On the other hand it is possible that some jobs do not belong to $J(\lambda)$. We refer to such jobs as the *missing jobs*. The solution $X(\lambda)$ is *feasible* if it contains neither repeated nor missing jobs.

We say that λ is *near-optimal* if $\theta(\lambda)$ is close to its maximum. Preliminary computational experience showed that the following properties hold for $J(\lambda)$, when λ is near-optimal:

a) repeated jobs rarely occur;

b) the number of missing jobs is about $10 \div 15\%$ of the total number of jobs;

c) The jobs in $J(\lambda)$ are in the same order as in the optimal solution.

Example 1 *We show a lagrangian solution $X(\lambda)$ and the associated list $J(\lambda)$ for a 20 job instance.*

$$\begin{array}{llllll} X(\lambda): & x_{0,36} = 1 & x_{1,98} = 1 & x_{3,22} = 1 & x_{4,46} = 1 & x_{5,82} = 1 \\ & x_{7,108} = 1 & x_{8,10} = 1 & x_{9,64} = 1 & x_{9,72} = 1 & x_{10,56} = 1 \\ & x_{11,30} = 1 & x_{12,26} = 1 & x_{12,40} = 1 & x_{15,45} = 1 & x_{17,12} = 1 \end{array}$$

$x_{18,35} = 1 \quad x_{19,92} = 1$

$$J(\lambda) = \{8, 17, 3, 12, 11, 18, 0, 12, 15, 4, 10, 9, 9, 5, 19, 1, 7\}$$

In $J(\lambda)$, jobs 9 and 12 are repeated jobs, while jobs 2, 6, 13, 14, 16 are missing jobs. The optimal solution for the same instance is:

$$\begin{array}{ll} X_{opt}: & x_{0,36} = 1 \quad x_{1,100} = 1 \quad x_{2,26} = 1 \quad x_{3,22} = 1 \quad x_{4,46} = 1 \\ & x_{5,79} = 1 \quad x_{6,57} = 1 \quad x_{7,110} = 1 \quad x_{8,10} = 1 \quad x_{9,69} = 1 \quad x_{10,56} = 1 \\ & x_{11,30} = 1 \quad x_{12,40} = 1 \quad x_{13,84} = 1 \quad x_{14,61} = 1 \quad x_{15,45} = 1 \quad x_{16,81} = 1 \\ & x_{17,12} = 1 \quad x_{18,35} = 1 \quad x_{19,94} = 1 \end{array}$$

$$J_{opt} = \{8, 17, 3, 2, 11, 18, 0, 12, 15, 4, 10, 6, 14, 9, 5, 16, 13, 19, 1, 7\}$$ □

Given a near-optimal λ, the heuristic aims at making the lagrangian solution $X(\lambda)$ feasible by removing repeated jobs and then inserting each missing job at the 'right place'. For each missing job j we define a *confidence interval* $[h(j), k(j)]$, where $h(j)$ and $k(j)$ are two jobs in $J(\lambda)$. To define $h(j)$ and $k(j)$, we look at another near-optimal set of multipliers μ, whose list $J(\mu)$ contains the missing job j: $h(j)$ and $k(j)$ are respectively the jobs that precede and follow j in $J(\lambda) \cap J(\mu)$.

The rationale of this choice is property c): we guess that the jobs in $J(\lambda)$ and $J(\mu)$ are in the same order as in the optimal solution and that $J(\mu)$ can suggest where the missing job should be placed in an optimal sequence.

We construct a feasible solution by inserting each missing job in an average point of $[h(j), k(j)]$. The feasible solution is then improved by running a simple interchange algorithm over each confidence interval.

For the sake of clarity below we summarize the main steps of the heuristic:

i) Let λ be a near-optimal set of lagrangian multipliers and let $X(\lambda)$ and $J(\lambda)$ define the lagrangian solution.

ii) For each repeated job j in $J(\lambda)$, we delete all its occurrences but the latest.

iii) For each missing job j, we look at another near-optimal set of lagrangian multipliers μ to define the confidence interval $[h(j), k(j)]$. We add the job j to $J(\lambda)$, by placing j in an average point of $[h(j), k(j)]$.

vi) The current solution is improved by running a simple interchange algorithm over each confidence interval.

Example 2 *For the same instance of example 1, let μ_1 and μ_2 be two other sets of near-optimal lagrangian multipliers and let $J(\mu_1)$ and $J(\mu_2)$ be, respectively, the ordered lists associated with $X(\mu_1)$ and $X(\mu_2)$.*

$$J(\lambda) = \{8, 17, 3, 12, 11, 18, 0, 15, 4, 10, 9, 5, 19, 1, 7\}$$

$$J(\mu_1) = \{8, 17, 3, 2, 12, 11, 18, 0, 15, 4, 14, 9, 5, 16, 19, 1, 7\}$$

$$J(\mu_2) = \{8, 17, 3, 12, 11, 18, 0, 15, 4, 6, 14, 9, 5, 16, 13, 19, 1, 7\}$$

For each missing job in $J(\lambda)$, $J(\mu_1)$ and $J(\mu_2)$ provide the following confidence intervals:

$$2 \rightarrow [3, 12]; \quad 6 \rightarrow [4, 9]; \quad 13 \rightarrow [5, 19]; \quad 14 \rightarrow [6, 9]; \quad 16 \rightarrow [5, 13].$$

To get a feasible solution J_{feas}, we complete $J(\lambda)$ by inserting each missing job in an average point of its confidence interval:

$$J_{feas} = \{8, 17, 3, 11, 2, 18, 0, 12, 15, 4, 10, 14, 6, 9, 5, 16, 13, 19, 1, 7\}$$

□

The heuristic can be applied iteratively, choosing different sets of near-optimal multipliers λ and μ. It runs fast and results very effective in computing high quality upper bounds for large scale instances as from computational results reported in the next section.

5 Computational Experience

The algorithm has been tested on a rich set of instances. The test bed consists of two classes of instances, named, respectively, *Optimal* and *Hard* and generated as in Savelsbergh, Uma and Wein [23] and Uma [26].

Optimal instances have been randomly generated according to the following procedure:

a) weights w_j are generated from U[1,20], i.e. they are uniformly distributed in the interval [1,20].

b) release dates r_j are generated from U[0, $\frac{1}{2}\sum_{j=1}^{n} p_j$], where n is the number of jobs.

c) processing times p_j are generated from U[1,p_{max}].

The *Hard* test bed was designed to produce bad lower bounds from the time-indexed formulation. These instances have few very large jobs and a large number of tiny jobs that are released regularly at small intervals. The generation procedure of the *Hard* instances is obtained by replacing the step c) with the following:

c') Processing times p_j may assume only the values 1 or 50. Size 1 is generated on average 9 times more frequently than the other.

Optimal instances are organized into 5 groups, each containing problems with the same number of jobs. The sizes (number of jobs) here considered are 75, 100, 200, 300 and 400. Each groups contains 25 instances, 5 for each choice of p_{max} (= 10, 20, 30, 40, 50).

Hard instances are organized into 5 groups containing respectively instances of 5, 100, 200, 300 and 400 jobs. Each class contains 5 instances.

The code has been written on Visual C++ 6.0. All the computations were carried out on a Compaq PC (Pentium IV-1.8 Ghz CPU, 256Mb RAM).

The outcomes for the *Optimal* instances of $1|r_j|\sum w_j C_j$ are reported in tables 2-6. Tables 7-10 report on *Optimal* instances of $1|r_j|\sum w_j F_j$.

In each table, columns $Name$, $njob$, p_{max} and $nvar$ show, respectively, the name of the instance, the number of jobs, the maximum processing time and the number of variables in the time-indexed formulation. Columns LB, UB and *Time* report, respectively, the lagrangian lower bound, the upper bound and the total CPU time. Column *%gap* shows the duality gap, computed as $\frac{UB-LB}{UB} \cdot 100$.

For 75 job instances of $1|r_j|\sum w_j C_j$, we also report computational experience with a MIP solver. Columns *2% Opt* and *Cplex time* of Table 2, report, respectively, on the value of a feasible solution providing a gap $\leq 2\%$ and on the (much larger) time spent by the Cplex 7.0 to find it.

For larger instances, the lagrangian heuristic determines good upper bounds, providing duality gaps never exceeding 3.0% in few minutes for 100, 200 and 300 instances and in less than 40 minutes for 400 job instances, whose time-indexed formulation contains a huge number of variables (more then 5 millions when $p_{max} = 50$) and constraints.

Instances of $1|r_j|\sum w_j F_j$ have revealed more difficult and some tuning of our heuristic was necessary. Particularly we had to slow down the convergence of the subgradient method, by setting the reduction parameter of φ (see section 3.2) to 1.007 instead of 1.01.

The outcomes for the *Hard* instances are reported in tables 11 and 12. We note that the duality gap is larger than for the *Optimal* instances. This can be easily explained by observing (Table 11) that for these instances the quality of the lower bound yielded from the time-indexed formulation, reported in the column *LP-LB*, is poor. Nevertheless the lagrangian heuristic confirms to be robust, since the upper bound is very close (less than 1%) to the value of the optimal solution computed by Cplex 7.0, reported in the column *Opt*.

Acknowledgements The authors wish to thank C. Mannino, P. Nobili, G. Oriolo, A. Sassano, A. S. Schultz and R.N. Uma, for the helpful discussions and the many suggestions provided.

Name	njob	p_{max}	nvar	LB	UB	$\%gap$	Time (Secs.)	2% opt solution	Cplex Time
C75-10-1	75	10	38336	116642.6	117339	0.5	2.0	117834	62.7
C75-10-2	75	10	40198	135056.5	137767	1.9	2.2	136457	299.9
C75-10-3	75	10	39723	133982.6	136571	1.9	2.1	137431	233.8
C75-10-4	75	10	36793	100083.5	101264	1.2	1.9	101101	158.3
C75-10-5	75	10	37964	110965.1	112472	1.3	2.0	112377	80.6
C75-20-1	75	20	69399	196545.1	199650	1.5	3.9	202170	$> 2h$
C75-20-2	75	20	77211	195896.6	198107	1.1	4.3	199036	1797.0
C75-20-3	75	20	64071	198076.0	201480	1.7	3.4	202960	1611.2
C75-20-4	75	20	68103	204039.8	206778	1.3	3.7	207560	1409.0
C75-20-5	75	20	79545	234250.8	237256	1.3	4.5	240269	3707.3
C75-30-1	75	30	99944	304541.6	309161	1.5	5.9	311284	4585.9
C75-30-2	75	30	106656	278421.3	283839	1.9	6.2	282531	5223.9
C75-30-3	75	30	104294	319551.3	322629	0.9	6.2	329943	$> 2h$
C75-30-4	75	30	104189	333039.6	338724	1.7	6.3	341159	6656.6
C75-30-5	75	30	102589	309565.2	316844	2.3	6.0	316323	4581.0
C75-40-1	75	40	147965	458579.4	464781	1.3	12.1	481204	$> 2h$
C75-40-2	75	40	144998	423853.4	429932	1.4	11.9	454237	$> 2h$
C75-40-3	75	40	144560	417154.4	425284	1.9	11.8	456559	$> 2h$
C75-40-4	75	40	149933	428511.3	435912	1.7	12.5	461271	$> 2h$
C75-40-5	75	40	163552	486133.0	494008	1.6	14.0	521586	$> 2h$
C75-50-1	75	50	171354	493840.1	500354	1.3	14.7	543857	$> 2h$
C75-50-2	75	50	170834	446598.1	458285	2.5	14.8	475395	$> 2h$
C75-50-3	75	50	167715	504205.0	514102	1.9	14.6	547678	$> 2h$
C75-50-4	75	50	165438	464884.3	472390	1.6	14.0	475878	5235.6
C75-50-5	75	50	178542	499036.1	502321	0.6	15.5	531653	$> 2h$

Table 2: Computational results for 75 jobs instances of $1|r_j|\sum w_j C_j$.

Name	njob	p_{max}	nvar	LB	UB	$\%gap$	Time (secs.)
C100-10-1	100	10	65912	193200.9	195902	1.4	3.8
C100-10-2	100	10	66126	170381.5	171923	0.9	3.9
C100-10-3	100	10	65277	192425.2	194604	1.1	3.8
C100-10-4	100	10	67272	177019.2	180762	2.0	4.0
C100-10-5	100	10	67786	193625.3	197089	1.7	4.1
C100-20-1	100	20	144766	420310.1	427623	1.7	10.8
C100-20-2	100	20	134940	335842.6	341252	1.6	10.0
C100-20-3	100	20	120217	328905.4	334140	1.6	7.2
C100-20-4	100	20	125402	355407.2	358789	0.9	7.1
C100-20-5	100	20	129343	394619.7	399516	1.2	8.0
C100-30-1	100	30	177646	519683.7	525799	1.2	14.7
C100-30-2	100	30	188813	485017.5	492452	1.5	15.6
C100-30-3	100	30	179593	586454.5	593065	1.1	14.3
C100-30-4	100	30	190397	576560.6	582335	1.0	15.5
C100-30-5	100	30	172976	495414.1	505649	2.0	13.7
C100-40-1	100	40	244558	687569.1	693168	0.8	22.2
C100-40-2	100	40	256782	711983.3	721082	1.3	29.4
C100-40-3	100	40	261111	797587.5	810334	1.6	33.2
C100-40-4	100	40	236137	696253.0	704904	1.2	21.8
C100-40-5	100	40	273688	782086.6	790894	1.1	26.8
C100-50-1	100	50	330298	866766.7	875000	0.9	34.1
C100-50-2	100	50	298382	845719.6	853757	0.9	30.2
C100-50-3	100	50	299436	876562.3	889010	1.4	31.3
C100-50-4	100	50	313384	880726.2	889420	1.0	32.1
C100-50-5	100	50	309648	925780.9	934973	1.0	32.4

Table 3: Computational results for 100 jobs instances of $1|r_j|\sum w_jC_j$.

Name	njob	p_{max}	nvar	LB	UB	$\%gap$	Time (secs.)
C200-10-1	200	10	270059	815001.6	827647	1.5	25.3
C200-10-2	200	10	267779	670085.3	678000	1.2	24.8
C200-10-3	200	10	271942	788518.8	797685	1.1	25.0
C200-10-4	200	10	263524	767919.8	781129	1.7	24.7
C200-10-5	200	10	275281	832218.6	840551	1.0	25.8
C200-20-1	200	20	525831	1506960.2	1527049	1.3	56.9
C200-20-2	200	20	520177	1478141.8	1507545	1.9	70.6
C200-20-3	200	20	534017	1535736.0	1558048	1.4	59.3
C200-20-4	200	20	520378	1466519.6	1489647	1.5	68.8
C200-20-5	200	20	533860	1595767.5	1623766	1.7	60.3
C200-30-1	200	30	778549	2321087.5	2353473	1.4	94.8
C200-30-2	200	30	794810	2179889.7	2207102	1.2	90.8
C200-30-3	200	30	764702	2156658.5	2183207	1.2	94.7
C200-30-4	200	30	807128	2232284.0	2257727	1.1	91.7
C200-30-5	200	30	715703	2037465.1	2078095	1.9	87.2
C200-40-1	200	40	1023077	2731490.0	2767215	1.3	144.7
C200-40-2	200	40	997686	2767169.4	2791934	0.9	144.3
C200-40-3	200	40	1032327	3130026.3	3164059	1.1	183.6
C200-40-4	200	40	963478	2675135.6	2705708	1.1	131.3
C200-40-5	200	40	1090136	3308648.5	3352171	1.3	162.0
C200-50-1	200	50	1243144	3539340.0	3628072	2.4	207.6
C200-50-2	200	50	1212412	3275700.5	3319334	1.3	189.8
C200-50-3	200	50	1197664	3541586.5	3580832	1.1	188.8
C200-50-4	200	50	1248578	3801306.2	3852853	1.3	199.1
C200-50-5	200	50	1307123	3854348.0	3878774	0.6	271.1

Table 4: Computational results for 200 jobs instances of $1|r_j|\sum w_j C_j$.

Name	njob	p_{max}	nvar	LB	UB	$\%gap$	Time (secs.)
C300-10-1	300	10	623083	1762715.7	1797535	1.9	97.3
C300-10-2	300	10	632886	1674060.9	1714605	2.3	105.1
C300-10-3	300	10	610751	1762244.9	1782216	1.1	98.7
C300-10-4	300	10	613112	1741343.1	1759408	1.0	101.9
C300-10-5	300	10	604353	1717539.4	1738848	1.2	99.7
C300-20-1	300	20	1200015	3169347.0	3182170	0.4	236.3
C300-20-2	300	20	1175600	3171612.2	3220311	1.5	247.0
C300-20-3	300	20	1162851	3348573.0	3370011	0.6	229.1
C300-20-4	300	20	1205640	3413916.5	3479448	1.9	236.5
C300-20-5	300	20	1237302	3468219.5	3489938	0.6	247.9
C300-30-1	300	30	1729934	4589222.0	4611495	0.5	386.1
C300-30-2	300	30	1791122	4485549.5	4506724	0.5	415.4
C300-30-3	300	30	1687186	4863097.2	4897029	0.7	383.4
C300-30-4	300	30	1852356	5112082.0	5175200	1.2	398.0
C300-30-5	300	30	1666116	4662721.7	4718405	1.2	367.6
C300-40-1	300	40	2323837	6067478.5	6101276	0.6	595.5
C300-40-2	300	40	2199066	5955684.0	6016862	1.0	548.8
C300-40-3	300	40	2271975	6777814.5	6832438	0.8	579.4
C300-40-4	300	40	2198457	6325858.2	6372482	0.7	544.3
C300-40-5	300	40	2468196	7064772.7	7170863	1.5	640.1
C300-50-1	300	50	2781100	7445458.5	7488298	0.6	785.0
C300-50-2	300	50	2675234	7041566.0	7091399	0.7	743.0
C300-50-3	300	50	2721201	7797337.5	7832145	0.4	762.8
C300-50-4	300	50	2938508	8272626.3	8320644	0.6	848.5
C300-50-5	300	50	2900708	8329975.5	8458770	1.5	817.2

Table 5: Computational results for 300 jobs instances of $1|r_j|\sum w_jC_j$.

Name	njob	p_{max}	nvar	LB	UB	$\%gap$	Time (secs.)
C400-10-1	400	10	1098928	3031806.5	3044833	0.4	356.9
C400-10-2	400	10	1115627	2935076.2	2955324	0.3	345.2
C400-10-3	400	10	1079357	3181475.0	3189590	0.7	335.5
C400-10-4	400	10	1058930	2966744.9	2990206	0.8	322.1
C400-10-5	400	10	1093093	3056246.3	3087706	1.0	335.2
C400-20-1	400	20	2130248	5483277.4	5518855	0.6	722.9
C400-20-2	400	20	2107822	5497677.0	5531278	0.6	720.0
C400-20-3	400	20	2025556	6145341.7	6196956	0.8	686.3
C400-20-4	400	20	2121485	5981877.0	6029928	0.8	709.8
C400-20-5	400	20	2191540	6204123.5	6250196	0.5	752.6
C400-30-1	400	30	3040103	8129443.5	8175090	0.6	1126.4
C400-30-2	400	30	3113615	8406462.3	8493771	1.0	1162.0
C400-30-3	400	30	3017134	8634068.0	8710052	0.9	1161.1
C400-30-4	400	30	3248272	9011942.1	9087800	0.8	1230.2
C400-30-5	400	30	3050667	8382790.7	8449171	0.8	1154.5
C400-40-1	400	40	4007653	10833100.0	10984530	1.4	1646.7
C400-40-2	400	40	4008190	10235532.8	10379527	1.4	1626.0
C400-40-3	400	40	4017958	11702906.5	11777593	0.6	1633.1
C400-40-4	400	40	4002446	11452290.0	11519206	0.6	1663.8
C400-40-5	400	40	4317774	12004060.2	12178142	1.4	1902.2
C400-50-1	400	50	5064133	13244034.5	13330132	0.6	2273.4
C400-50-2	400	50	4879554	13546556.0	13640912	0.7	2190.5
C400-50-3	400	50	5041728	14696984.2	14826365	0.9	2288.4
C400-50-4	400	50	5162462	14576275.0	14638736	0.4	2331.9
C400-50-5	400	50	5270004	14904749.7	15038486	0.9	2418.5

Table 6: Computational results for 400 jobs instances of $1|r_j|\sum w_jC_j$.

Name	njob	p_{max}	nvar	LB	UB	$\%gap$	Time (secs.)
F100-10-1	100	10	65912	58184.2	59315	1.9	15.4
F100-10-2	100	10	66126	46667.3	47395	1.5	15.5
F100-10-3	100	10	65277	58214.7	59377	1.9	15.5
F100-10-4	100	10	67272	45894.5	47002	2.3	16.0
F100-10-5	100	10	67786	50652.8	51779	2.2	16.5
F100-20-1	100	20	144766	138086.8	140789	1.9	40.1
F100-20-2	100	20	134940	98757.3	101120	2.3	40.3
F100-20-3	100	20	120217	109327.8	112026	2.4	31.4
F100-20-4	100	20	125402	86022.7	88047	2.3	31.1
F100-20-5	100	20	129343	121782.7	124187	1.9	33.2
F100-30-1	100	30	177646	146363.4	149482	2.1	56.2
F100-30-2	100	30	188813	97744.6	100570	2.8	59.1
F100-30-3	100	30	179593	142768.2	145783	2.1	55.7
F100-30-4	100	30	190397	169527.7	172400	1.6	63.0
F100-30-5	100	30	172976	130767.7	133407	1.9	52.7
F100-40-1	100	40	244558	196842.5	200190	1.7	76.3
F100-40-2	100	40	256782	192098.4	196613	2.3	89.5
F100-40-3	100	40	261111	255530.8	260650	1.9	85.1
F100-40-4	100	40	236137	196190.8	200709	2.2	75.6
F100-40-5	100	40	273688	275613.2	280977	1.9	89.7
F100-50-1	100	50	330298	265584.0	271713	2.2	106.6
F100-50-2	100	50	298382	182484.5	187300	2.6	96.1
F100-50-3	100	50	299436	281166.8	287825	2.3	95.6
F100-50-4	100	50	313384	288180.3	291992	1.3	105.7
F100-50-5	100	50	309648	257234.5	261233	1.5	101.4

Table 7: Computational results for 100 jobs instances of $1|r_j|\sum w_j F_j$.

Name	njob	p_{max}	nvar	LB	UB	$\%gap$	Time (secs.)
F200-10-1	200	10	270059	225659.1	229096	1.5	85.0
F200-10-2	200	10	267779	175351.4	179183	2.1	82.1
F200-10-3	200	10	271942	223569.2	228025	1.9	84.3
F200-10-4	200	10	263524	196818.9	201195	2.2	81.3
F200-10-5	200	10	275281	221918.7	226901	2.2	84.8
F200-20-1	200	20	525831	415004.9	422654	1.8	175.7
F200-20-2	200	20	520177	436679.6	441844	1.2	188.5
F200-20-3	200	20	534017	465683.3	476421	2.2	177.6
F200-20-4	200	20	520378	385356.8	392069	1.7	184.2
F200-20-5	200	20	533860	466986.8	477233	2.1	170.7
F200-30-1	200	30	778549	625748.8	635919	1.6	271.9
F200-30-2	200	30	794810	608514.5	624915	2.6	255.8
F200-30-3	200	30	764702	633726.0	649419	2.4	244.3
F200-30-4	200	30	807128	543021.3	557199	2.5	267.6
F200-30-5	200	30	715703	560528.8	574842	2.5	235.9
F200-40-1	200	40	1023077	765913.5	777847	1.5	364.2
F200-40-2	200	40	997686	682824.8	698609	2.2	352.4
F200-40-3	200	40	1032327	875236.1	893145	2.0	363.9
F200-40-4	200	40	963478	794030.5	809643	1.9	349.4
F200-40-5	200	40	1090136	900544.7	918434	1.9	371.9
F200-50-1	200	50	1243144	844417.0	859293	1.7	488.5
F200-50-2	200	50	1212412	791763.1	814834	2.8	493.4
F200-50-3	200	50	1197664	936089.2	960325	2.5	501.1
F200-50-4	200	50	1248578	1094760.7	1113143	1.6	477.6
F200-50-5	200	50	1307123	1163152.2	1177507	1.2	516.4

Table 8: Computational results for 200 jobs instances of $1|r_j|\sum w_j F_j$.

Name	njob	p_{max}	nvar	LB	UB	$\%gap$	Time (secs.)
F300-10-1	300	10	623083	452680.2	460465	1.7	209.4
F300-10-2	300	10	632886	429288.5	438663	2.1	205.4
F300-10-3	300	10	610751	526857.2	539279	2.3	204.2
F300-10-4	300	10	613112	397427.7	405825	2.1	206.2
F300-10-5	300	10	604353	475918.0	484166	1.7	205.8
F300-20-1	300	20	1200015	910168.7	921859	1.2	430.9
F300-20-2	300	20	1175600	868733.3	890512	2.4	418.7
F300-20-3	300	20	1162851	880516.8	903580	2.5	394.1
F300-20-4	300	20	1205640	998567.5	1016669	1.8	413.5
F300-20-5	300	20	1237302	1061925.0	1082689	1.9	440.1
F300-30-1	300	30	1729934	1349564.7	1373277	1.7	633.9
F300-30-2	300	30	1791122	1253441.0	1277197	1.8	628.2
F300-30-3	300	30	1687186	1278834.7	1309558	2.3	627.8
F300-30-4	300	30	1852356	1536144.5	1559613	1.5	650.9
F300-30-5	300	30	1666116	1295319.2	1328195	2.4	626.2
F300-40-1	300	40	2323837	1692304.5	1735108	2.5	941.8
F300-40-2	300	40	2199066	1419792.0	1454080	2.3	864.7
F300-40-3	300	40	2271975	1704247.7	1748053	2.5	909.5
F300-40-4	300	40	2198457	1671591.7	1713367	2.4	874.3
F300-40-5	300	40	2468196	2130808.0	2175553	2.0	994.0
F300-50-1	300	50	2781100	1981381.0	2029958	2.4	1209.3
F300-50-2	300	50	2675234	1584205.8	1626232	2.6	1263.6
F300-50-3	300	50	2721201	2029880.5	2082966	2.5	1302.0
F300-50-4	300	50	2938508	2182288.7	2232831	2.3	1291.6
F300-50-5	300	50	2900708	2272852.0	2312720	1.7	1243.9

Table 9: Computational results for 300 jobs instances of $1|r_j|\sum w_j F_j$.

Name	njob	p_{max}	nvar	LB	UB	$\%gap$	Time (secs.)
F400-10-1	400	10	1098928	793732.0	814387	2.5	385.3
F400-10-2	400	10	1115627	703552.1	719594	2.2	381.5
F400-10-3	400	10	1079357	824805.5	839226	1.7	379.1
F400-10-4	400	10	1058930	742532.3	758006	2.0	360.4
F400-10-5	400	10	1093093	751402.1	769055	2.3	370.3
F400-20-1	400	20	2130248	1691062.5	1717227	1.5	812.8
F400-20-2	400	20	2107822	1561106.7	1598610	2.3	772.7
F400-20-3	400	20	2025556	1552715.0	1584486	2.0	739.0
F400-20-4	400	20	2121485	1700282.7	1732716	1.9	783.1
F400-20-5	400	20	2191540	1675038.3	1712906	2.2	842.9
F400-30-1	400	30	3040103	2216747.9	2266577	2.2	1217.1
F400-30-2	400	30	3113615	2070670.5	2118536	2.2	1278.8
F400-30-3	400	30	3017134	2263365.5	2317435	2.3	1216.4
F400-30-4	400	30	3248272	2698355.7	2738466	1.5	1350.3
F400-30-5	400	30	3050667	2455580.0	2505861	2.0	1254.3
F400-40-1	400	40	4007653	2662386.8	2719678	2.1	1774.0
F400-40-2	400	40	4008190	2683765.0	2745439	2.2	1754.5
F400-40-3	400	40	4017958	3275855.7	3349425	2.1	1783.4
F400-40-4	400	40	4002446	2921514.0	2996636	2.5	1773.1
F400-40-5	400	40	4317774	3500427.8	3575182	2.1	1956.0
F400-50-1	400	50	5064133	3692456.5	3768805	2.0	2458.5
F400-50-2	400	50	4879554	3259512.0	3319666	1.8	2535.1
F400-50-3	400	50	5041728	3973331.7	4062679	2.2	2677.4
F400-50-4	400	50	5162462	3806510.2	3901621	2.4	2733.3
F400-50-5	400	50	5270004	4062836.0	4168188	2.5	2721.1

Table 10: Computational results for 400 jobs instances of $1|r_j|\sum w_j F_j$

Name	njob	nvar	LB	UB	%gap	Time (secs.)	LP-LB	Opt	Cplex Time
H75-1	75	31887	77097.1	78215	1.4	6.6	77097.5	78016	10.9
H75-2	75	44387	117417.0	120296	2.4	7.6	117417.2	119305	19.4
H75-3	75	34935	99854.9	102348	2.4	6.3	99855.3	101342	26.2
H75-4	75	48936	135222.6	142281	4.9	7.1	135222.9	142125	3423.2
H75-5	75	35880	97254.4	100850	3.6	6.5	97255.8	99011	29.1

Table 11: Computational results for *hard instances* of $1|r_j|\sum w_jC_j$.

Name	njob	p_{max}	nvar	LB	UB	%gap	Time (secs.)
H100-1	100	50	47756	108502.4	108856	0.3	9.8
H100-2	100	50	90993	234642.9	244287	3.9	20.1
H100-3	100	50	83940	203180.3	212421	4.3	18.5
H100-4	100	50	72728	163159.7	168921	3.4	15.6
H100-5	100	50	113772	299870.0	308238	2.7	26.1
H200-1	200	50	220471	448173.5	456466	1.8	56.0
H200-2	200	50	280532	610283.8	629618	3.0	71.8
H200-3	200	50	314190	765021.2	798307	4.1	84.6
H200-4	200	50	279469	672410.5	689469	2.5	75.3
H200-5	200	50	266985	616976.3	631387	2.3	72.5
H300-1	300	50	581890	1278915.6	1345798	4.9	157.3
H300-2	300	50	629712	1493301.5	1554953	3.9	178.7
H300-3	300	50	879334	1999378.8	2088324	4.2	227.6
H300-4	300	50	578156	1328882.1	1375906	3.4	166.7
H300-5	300	50	869699	1930130.7	2001966	3.6	221.5
H400-1	400	50	1089906	2392531.0	2514889	4.8	325.3
H400-2	400	50	1525775	3496497.7	3654330	4.3	489.9
H400-3	400	50	1299314	2861252.2	2968565	3.6	498.3
H400-4	400	50	1583552	3620487.5	3733772	3.0	521.0
H400-5	400	50	1146905	2575363.2	2651500	2.8	337.1

Table 12: Computational results for *hard instances* of $1|r_j|\sum w_jC_j$.

References

[1] J.M. van den Akker, C.P.M. van Hoesel and M.W.P. Savelsbergh, A polyhedral approach to single-machine scheduling problems, Mathematical Programming 85, 1999, 541-572.

[2] J.M. Ven den Akker, C.Q.J. Hurkens and M.W.P. Savelsbergh, A time-indexed formulation for single-machine scheduling problems: column generation, INFORMS Journal on Computing 12/2, 2000, 111-124.

[3] R. Anbil and F. Barahona, The Volume Algorithm: Producing Primal Solutions With a Subgradient Method, Mathematical Programming 87, 2000, 385-399.

[4] E. Balas, On the facial structure of scheduling polyhedron, Mathematical Programming 24, 1985, 179-218.

[5] H. Belouadah, M.E. Posner and C.N. Potts, Scheduling with release dates on a single machine to minimize total weighted completion time, Discrete Applied Mathematics 36, 1992, 213-231.

[6] P.M. Camerini, L. Fratta, F. Maffioli, On improving relaxation methods by modified gradient techniques, Mathematical Programming Study 3, 1975, 26-34.

[7] H.C. Chen, D. Elizandro, Q. Liu and D.M. Miller, A Hybrid genetic algorithm for the simple machine scheduling problem, Journal of Heuristics, 1999, 437-454.

[8] Y. Crama and F.C.R Spieksma, Scheduling jobs of equal length: complexity, facet and computational results. E. Balas and J. Clausen (eds.) Procedeengs of the 4th International IPCO Conferenze, Denmark, Lecture Notes in Computer Science 920, Springer, Berlin, 1995, 277-291.

[9] 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.

[10] P.C. Fishburn, Interval orders and interval graphs, Wiley, New York, 1985.

[11] M. Goemans, Improved approximation algorithm for scheduling with release dates, in procedeengs af the 8th ACM-SIAM Symposium on Discrete Algorithms, 1997, 591-598.

[12] M. Goemans, M. Queiranne, A. Schulz, M. Skutella and Y. Wang, Single machine scheduling with release dates, SIAM Journal on Discrete Mathematics 15, 2002, 165-192.

[13] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, 1980, Academic Press.

[14] L.A. Hall, Approximation algorithms for scheduling, in D.S. Hochbaum editor, *Approximation algorithms for NP-hard Problems*, PWS Publishing Company, 1997, 1-43.

[15] L.A. Hall, A.S. Schulz, D.B. Shmoys and J. Wein, Scheduling to minimize average completion time: off-line and on-line algorithms, Mathematics of Operations Research 22, 1997, 513-544.

[16] A.M. Laguna, A.J. Barnes and A.F. Glover, Tabu Search Methodology for a Single Machine Scheduling Problem, Journal of Int. Manufacturing 2, 1991, 63-74.

[17] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics 1, 1977, 343-362.

[18] C. Mannino, G. Oriolo, Solving stability problems on a superclass of interval graphs, Technical Report n. 155, Centro Vito Volterra.

[19] R.H. Mőhring, A. S. Schulz, F. Stork, and M. Uetz, Solving Project Scheduling Problems by Minimum Cut Computations, Extended abstract appeared in J. Nesetril (ed.): Algorithms - ESA '99, Lecture Notes in Computer Science 1643, Springer, Berlin, 1999, 139-150.

[20] R.H. Möhring, A. S. Schulz, F. Stork, and M. Uetz, On Project Scheduling with Irregular Starting Time Costs, Operations Research Letters 28, 2001, 149-154.

[21] M. Queyranne, Structure of a simple scheduling polyhedron,Mathematical Programming, 1993, (58):262-285.

[22] M. Queyranne and A.S. Schulz, Polyhedral approaches to machine scheduling, technical report 408/1994, Technical University of Berlin, 1994.

[23] M.W.P. Savelsbergh, R.N. Uma and J. Wein, An Experimental Study of LP-Based Approximation Algorithm for Scheduling Problems, Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms, January 1998.

[24] A.S. Schulz, Scheduling to Minimize Total Weighted Completion Time: Performance Guarantees of LP-Based Heuristics and Lower Bounds, in W.H. Cunningham, S.T. McCormick, and M. Queyranne (eds.): Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science 1084, Springer: Berlin, 1996, 301-315.

[25] J.P. De Sousa and L.A. Wolsey, A time-indexed formulation of non-preemptive single-machine scheduling problems, Mathematical Programming 54, 1992, 353-367.

[26] R.N. Uma, Theoretical and Experimental Perspectives on Hard Scheduling Problems, Ph.D. Thesis, Polytechnic University, New York, 2000.

[27] R.N. Uma, Personal communication.

[28] S.L. van de Velde, Dual decomposition of a single-machine scheduling problem, Mathematical Programming 69, 1995, 413-428

[29] H. Waterer, E.L. Johnson and M.W.P. Savelsbergh, The relation of time indexed formulation of single machine scheduling problems to the node packing problem, CORE Discussion Paper 9, 2002.