

Experimental Comparison of Approximation Algorithms for Scheduling Unrelated Parallel Machines

Tjark Vredeveld • Cor Hurkens

Department of Mathematics and Computing Science, Technische Universiteit Eindhoven, P.O.Box 513,
5600 MB Eindhoven, The Netherlands
tjark@win.tue.nl • wscor@win.tue.nl

This paper presents an empirical comparison of polynomial-time approximation algorithms and local search heuristics for the problem of minimizing total weighted completion time on unrelated parallel machines. Algorithms with a worst-case performance guarantee are based on rounding a fractional solution to an LP-relaxation or to a convex quadratic-programming relaxation. We also investigate dominance relations among the lower bounds resulting from these relaxations.

(*Production-Scheduling: Multiple Machines; Linear Programming-Based Heuristics; Nonlinear Programming-Based Heuristics; Local Search*)

1. Introduction

In this paper, we make an experimental comparison of approximation algorithms for which we have constant performance bounds but no empirical evidence and local search heuristics that exhibit good empirical behavior but for which we have no performance bounds. The former algorithms are so-called *polynomial-time ρ -approximation algorithms*, which are algorithms that compute a feasible solution in polynomial time whose value is within a factor ρ of the optimal solution value; ρ is called the *performance guarantee* of the algorithm.

The problem under consideration is the problem of scheduling unrelated parallel machines so as to minimize the total weighted completion time. We are given a set of n jobs, J_1, \dots, J_n , each of which has to be scheduled without interruption on one of m machines, M_1, \dots, M_m , where m is part of the input. A machine can process at most one job at a time and all jobs and machines are available at time 0. If a job J_j is processed on a machine M_i , it will take a positive integral processing time p_{ij} . Furthermore, for each job we are

given a non-negative integral weight w_j . The objective is to schedule the jobs so that the sum of the weighted completion times of the jobs is minimized. Graham et al. (1979) denote this problem by $R\|\sum w_j C_j$. Bruno et al. (1974) and Lenstra et al. (1977) showed that the problem of minimizing total weighted completion time on two identical parallel machines is NP-hard; $R\|\sum w_j C_j$ is NP-hard in the strong sense. In the case that the jobs have the same weight, the problem can be formulated as a bipartite matching problem, which can be solved in polynomial time (Horn 1973, Bruno et al. 1974). In case there is only one machine the problem is also easy: Sequence the jobs in order of non-increasing ratios w_j/p_{1j} (Smith 1956). Hence, the problem reduces to assigning the jobs appropriately to the machines; once the jobs have been assigned to the machines, we sequence the jobs on each machine by the ratio rule.

The first polynomial-time approximation algorithm for minimizing the total weighted completion time on unrelated parallel machines was given by Phillips et al. (1997), who achieved a performance ratio $\mathcal{O}(\log^2 n)$. Hall et al. (1997) presented a

polynomial-time $16/3$ -approximation algorithm that uses an LP-relaxation in time-indexed variables and relies on the rounding technique of Shmoys and Tardos (1993) for the generalized assignment problem. This result was successively improved by Schulz and Skutella to performance guarantees $2 + \epsilon$ and $3/2 + \epsilon$, where ϵ is an arbitrary positive value (Schulz and Skutella 1997a,b). They also used LP-relaxations in time-indexed variables. Skutella presented a polynomial-time $3/2$ -approximation algorithm that was based on a convex quadratic relaxation (Skutella 1998). On the negative side, Hoogeveen et al. (1998) showed that the problem is APX-hard, i.e., there exists an $\epsilon > 0$ such that there does not exist a polynomial-time $(1 + \epsilon)$ -approximation for this problem, unless $P = NP$.

Little work has been done on experimental comparison of these kinds of approximation algorithms and local search heuristics. Previous work of this sort includes the following. Johnson and McGeoch (1997) made a case study in local optimization for the traveling-salesman problem, in which they also experimentally analyzed several constructive heuristics with a worst-case or probabilistic performance guarantee. In the area of scheduling Hariri and Potts (1991) examined the empirical behavior of the earliest completion time (ECT) heuristic and several two-phase heuristics for the problem of minimizing makespan on unrelated parallel machines, $R||C_{\max}$. In the first phase of the two-phase algorithms, an LP-relaxation is solved to generate a partial schedule; the second phase consists of an exact or heuristic method to schedule the remaining jobs. The performance guarantees of the two-phase heuristics they examined are 2 and $(2 + \lfloor \log_2(m-1) \rfloor)$, the performance guarantee of the ECT heuristic is m . Their conclusion was that these constructive methods are quite unsatisfactory, as deviations of more than 10% from the optimal solution value are common. They also applied an improvement heuristic to the obtained solutions, which achieved a significant reduction in the makespan at very small computational costs. Glass et al. (1994) extended this research by analyzing the empirical performance of other local search heuristics for this problem. Although it is difficult to compare the results of these two papers, as the

quality of the solution is measured as the relative deviation from the best obtained solution, which might not be the same in both papers, and the time spent on the local search heuristics is not equivalent to the time spent on the constructive heuristics, the results indicate that genetic descent, simulated annealing, and tabu search outperform the constructive heuristics; the local search heuristics have a deviation of about 1%. Savelsbergh et al. (1998) studied the quality of lower bounds for the problem of minimizing total weighted completion time on a single machine with release dates for the jobs, obtained by LP-relaxations, and they also studied the quality of upper bounds delivered by a number of approximation algorithms based on these relaxations. The best algorithms come within a few percent of the optimum. Although there are a few exceptions, they also concluded that the higher the quality of the solution to the LP-relaxation, the better the approximation algorithms perform. Uma and Wein (1998) made an analytical and empirical comparison of lower bounds for the same problem and also used some rounding techniques to obtain feasible solutions. They also applied simple local search heuristics to the solutions obtained by these approximation algorithms as well as to solutions from scratch. The best results were obtained by applying the local search heuristics to the solutions obtained by good approximation algorithms. Recently Van der Linden (2000) made a computational study on the empirical behavior of a convex quadratic approximation algorithm of Skutella (2001) for the problem of minimizing the total weighted completion time on unrelated parallel machines with release dates for the jobs. She compared the results to the solutions of an LP-based approximation algorithm by Schulz and Skutella (1997b). On all test instances, the LP-relaxation gave better lower bounds than the convex quadratic relaxation. In Section 3, we prove that for the case of trivial release dates the LP-relaxation is guaranteed to give better bounds than the convex quadratic relaxation. Van der Linden also used the convex quadratic relaxation in a branch-and-bound method, which was able to solve problem instances up to 20 jobs and 5 machines.

Our paper is organized as follows. In the following section, we discuss the heuristics for which we have a constant performance guarantee. In Section 3 we discuss the dominance relations of the lower bounds obtained by the relaxations discussed in Section 2, and in Section 4 we describe the local search heuristics. Next we make some remarks on implementation details and we discuss the results of our empirical evaluation. Finally, in Section 6 we make some concluding remarks.

2. Approximation Algorithms With a Constant Guarantee

The heuristics for which we have a constant guarantee are all based on rounding a fractional solution to some relaxation. The rounding of the fractional solutions is done in the same way for all three relaxations. The main idea is to exploit the values of the relaxation as probabilities with which jobs are assigned to machines: Each job is assigned to a machine using these probabilities. This way of randomized rounding can be derandomized using the method of conditional expectations, with no difference in performance guarantees, but at the cost of increased, but still polynomial, running times.

2.1. Convex Quadratic Programming Relaxation

The first relaxation we consider is due to Skutella (1998). He introduced a convex quadratic-programming relaxation that leads to a polynomial-time 3/2-approximation algorithm. The basic observation is that the problem is reduced to an assignment problem. Therefore, the problem can be formulated as an integer quadratic program in nm assignment variables, z_{ij} . The integer quadratic program, which forms the basis of the relaxation, is given in (IQP), where $k <_i j$ means that according to the ratio rule job J_k precedes job J_j on machine M_i . Constraints (2) ensure that the completion time of a job is its own processing time plus the processing times of its predecessors on the machine on which it is scheduled and constraints (1) ensure that each job is scheduled on

exactly one machine.

$$\begin{aligned} & \min \sum_j w_j C_j \\ \text{(IQP)} \quad & \text{s.t. } \sum_i z_{ij} = 1 \quad \forall j \end{aligned} \quad (1)$$

$$C_j = \sum_i z_{ij} \left(p_{ij} + \sum_{k <_i j} p_{ik} z_{ik} \right) \quad \forall j \quad (2)$$

$$z_{ij} \in \{0, 1\} \quad \forall i, j \quad (3)$$

In (6), the quadratic objective function is convexified by carefully raising the diagonal entries of the matrix determining the quadratic term until it becomes positive semidefinite and the function becomes convex. Adding constraint (5), which ensures that the sum of weighted completion times is at least the sum of weighted processing times, results in the convex quadratic programming problem given in (CQP). Solving this problem and then applying the randomized rounding procedure is a polynomial-time 3/2-approximation algorithm.

$$\begin{aligned} & \min Z \\ \text{s.t.} \quad & \sum_i z_{ij} = 1 \quad \forall j \end{aligned} \quad (4)$$

$$\text{(CQP)} \quad Z \geq \sum_{i,j} w_j p_{ij} z_{ij} \quad (5)$$

$$Z \geq \sum_{i,j} z_{ij} \left(1/2 w_j p_{ij} + 1/2 w_j p_{ij} z_{ij} + \sum_{k <_i j} w_j p_{ik} z_{ik} \right) \quad (6)$$

$$z_{ij} \geq 0 \quad \forall i, j \quad (7)$$

2.2. Time-Indexed Variables on Processing Intervals

Schulz and Skutella (1997a,b) generalized an LP-relaxation in time-indexed variables that was introduced by Dyer and Wolsey (1990) for the single-machine scheduling problem with release dates. It contains decision variables, y_{ijt} , indicating whether job J_j is being processed on machine M_i during the time interval $[t, t+1)$, for integral t , where t ranges from 0 to $T-1$, with T an upper bound on the length

of a schedule. The resulting LP-relaxation is a 3/2-relaxation of the scheduling problem under consideration, i.e., the value of an optimal schedule is within a factor 3/2 of the optimum LP value. Moreover, solving this LP-relaxation and applying the randomized rounding procedure yields a solution with expected value no more than 3/2 times the optimal value.

The LP-relaxation Schulz and Skutella used is the following:

$$\min \sum_j w_j C_j \quad (8)$$

$$\text{s.t. } \sum_{i,t} y_{ijt}/p_{ij} = 1 \quad \forall j \quad (9)$$

$$(LPY) \quad \sum_j y_{ijt} \leq 1 \quad \forall i, t \quad (10)$$

$$C_j \geq \sum_{i,t} (y_{ijt}/p_{ij}(t+1/2) + 1/2y_{ijt}) \quad \forall j \quad (11)$$

$$C_j \geq \sum_{i,t} y_{ijt} \quad \forall j \quad (12)$$

$$y_{ijt} \geq 0 \quad \forall i, j, t \quad (13)$$

To see that this is indeed a relaxation of $R\|\sum w_j C_j$, consider an arbitrary feasible schedule, where job J_j is being continuously processed between time S_j and $S_j + p_{hj}$ on machine M_h . Then $y_{ijt} = 1$ if $i = h$ and $t \in \{S_j, \dots, S_j + p_{hj} - 1\}$ and $y_{ijt} = 0$ otherwise. The right-hand side in (11) corresponds for these values of y_{ijt} to the completion time of J_j in the schedule. This relaxation can be strengthened by adding the constraint that a job can be processed by at most one machine during each time interval:

$$\sum_i y_{ijt} \leq 1 \quad \forall j, t. \quad (14)$$

As the time horizon, T , can be exponential in the input size, this relaxation may suffer from an exponential number of variables and constraints. One can overcome this drawback by turning to interval-indexed variables. The time intervals Schulz and Skutella used are of the form $I_0 = [0, 1)$ and $I_l = [(1 + \epsilon)^{l-1}, (1 + \epsilon)^l)$ for $l = 1, \dots, \log T / \log(1 + \epsilon)$, where $\epsilon > 0$ can be chosen arbitrarily small. The

LP-relaxation, including constraints (14), is then the following:

$$\min \sum_j w_j C_j \quad (15)$$

$$\text{s.t. } \sum_{i,l} y_{ijl}|I_l|/p_{ij} = 1 \quad \forall j \quad (16)$$

$$\sum_i y_{ijl} \leq 1 \quad \forall i, l \quad (17)$$

$$(LPY') \quad \sum_i y_{ijl} \leq 1 \quad \forall j, l \quad (18)$$

$$C_j \geq \sum_i \left((1/2p_{ij} + 1/2)y_{ij0} + \sum_{l \geq 1} (y_{ijl}|I_l|/p_{ij}(1 + \epsilon)^{l-1} + 1/2y_{ijl}|I_l|) \right) \quad \forall j \quad (19)$$

$$C_j \geq \sum_{i,l} y_{ijl}|I_l| \quad \forall j \quad (20)$$

$$y_{ijl} \geq 0 \quad \forall i, j, l \quad (21)$$

Any feasible solution, \bar{y} , of (LPY) plus constraints (14) can be transformed into a feasible solution, y , to (LPY') with the same or lower value. This is done by setting $y_{ijl} = \sum_t |[t, t+1) \cap I_l|/|I_l| \bar{y}_{ijt}$. Hence, the optimal value of (LPY') is at most equal to the optimal value of (LPY), including (14).

This leads to a $(3/2 + \epsilon)$ -relaxation of polynomial size and a polynomial-time $(3/2 + \epsilon)$ -approximation algorithm; in the subsequent sections, this method will be referred to as the *LPY approach*. Notice that the size of the relaxation still depends substantially on the time horizon and may be huge for small values of ϵ .

2.3. Time-Indexed Variables on Starting Times

The LP-relaxation used by Schulz and Skutella yields a poor lower bound, as even a 0–1 solution to this relaxation does not necessarily correspond to a feasible schedule. Therefore, we have implemented another relaxation. This is a generalization of a second LP-relaxation introduced by Dyer and Wolsey for the single-machine scheduling problem with release dates. The problem is formulated as an integer program in time-indexed variables, x_{ijt} , denoting whether

job J_j starts being processed on machine M_i at time t . This program is given in (IPX):

$$\begin{aligned} \min \sum_j w_j C_j \\ \text{s.t. } \sum_{i,t} x_{ijt} = 1 \quad \forall j \end{aligned} \quad (22)$$

$$(IPX) \quad \sum_j \sum_{s=t-p_{ij}+1}^t x_{ijs} \leq 1 \quad \forall i, t \quad (23)$$

$$C_j = \sum_{i,t} (t + p_{ij}) x_{ijt} \quad \forall j \quad (24)$$

$$x_{ijt} \in \{0, 1\} \quad \forall i, j, t \quad (25)$$

The LP-relaxation is obtained by relaxing the integrality constraints (25) to non-negativity constraints and will be denoted by (LPX).

As with LP problem (LPY), the time horizon, T , can be an exponential in the input size. Instead of trying to reduce the size of the LP problem, we solved this large problem by column generation, generalizing the work of Van den Akker et al. (2000) on the above-mentioned single-machine problem.

To reduce the number of constraints we apply Dantzig-Wolfe decomposition. The constraints (23) and the non-negativity constraints describe a polytope P . This polytope can be written as the Cartesian product $P = P_1 \times \dots \times P_m$, where $P_i = \{x \in \mathbb{R}_+^{nT} : \sum_j \sum_{s=t-p_{ij}+1}^t x_{jst} \leq 1, t = 1, \dots, T\}$. Hence a point $x \in P$ can be written as $x = (x^{(1)}, \dots, x^{(m)})$, where $x^{(i)} \in P_i$ ($i = 1, \dots, m$). The polytopes P_i have integral vertices (Van den Akker et al. 2000), and these vertices can be considered as schedules on machine M_i in which jobs do not have to be processed exactly once, as they do not necessarily satisfy constraints (22). We will call such schedules *semi-schedules*.

Let $\xi_1^{(i)}, \dots, \xi_{K_i}^{(i)}$ be the extreme points of P_i . Then any $x^{(i)} \in P_i$ can be written as a convex combination $x^{(i)} = \sum_{l=1}^{K_i} \lambda_l^{(i)} \xi_l^{(i)}$. The LP-relaxation can be reformulated using the variables $\lambda_l^{(i)}$ as follows:

$$\begin{aligned} \min \sum_i \sum_{l=1}^{K_i} \left(\sum_{j,t} c_{ijt} \xi_{l,jt}^{(i)} \right) \lambda_l^{(i)} \\ (LPX') \text{ s.t. } \sum_i \sum_{l=1}^{K_i} \left(\sum_t \xi_{l,jt}^{(i)} \right) \lambda_l^{(i)} = 1 \quad j = 1, \dots, n \end{aligned} \quad (26)$$

$$\sum_{l=1}^{K_i} \lambda_l^{(i)} = 1 \quad i = 1, \dots, m \quad (27)$$

$$\lambda_l^{(i)} \geq 0 \quad (28)$$

Note that the solutions of (LPX) and (LPX') are in a one-to-one correspondence.

Observe that the j th element of the column corresponding to $\lambda_l^{(i)}$, i.e., $\sum_t (\xi_l^{(i)})_{jt}$, is equal to the number of times that job J_j occurs in the semi-schedule $\xi_l^{(i)}$. This means that the column corresponding to the semi-schedule $\xi_l^{(i)}$ only indicates how many times each job occurs in this schedule. The cost coefficient of $\lambda_l^{(i)}$ is equal to the cost of the semi-schedule $\xi_l^{(i)}$.

By reformulating this way, the number of constraints is decreased significantly from $n + mT$ to $n + m$. The number of variables, however, has increased to the total number of extreme points of the polytopes P_i . Fortunately, this does not matter, since the problem can be solved through column generation. To apply column generation, we have to find an efficient way to determine a column with minimal reduced cost, i.e., to solve a pricing problem. We determine for each machine such a minimal column in the same way as Van den Akker et al. The reduced cost of the variable $\lambda_l^{(i)}$ is given by

$$\sum_{j,t} c_{ijt} \xi_{l,jt}^{(i)} - \sum_j \pi_j \left(\sum_t \xi_{l,jt}^{(i)} \right) - \alpha_i \quad (29)$$

where π_j denotes the dual variable of constraint (26) for job J_j , and α_i denotes the dual variable of constraints (27) for machine M_i .

Recall that each extreme point $\xi_l^{(i)}$ represents a semi-schedule on machine M_i . These semi-schedules can be represented by paths in a network in the following way. The nodes of the network correspond to time points $0, 1, \dots, T$. For each job J_j and each period s , with $s \leq T - p_{ij} + 1$, there is an arc from s to $s + p_{ij}$ that indicates the machine processes job J_j from time s to time $s + p_{ij}$; we say that this arc corresponds to the variable $(x^{(i)})_{js}$. Furthermore, for each time point t there is an idle time arc from t to $t + 1$ that indicates that the machine is idle in period $[t, t + 1)$. Any directed path from 0 to T corresponds to a semi-schedule $\xi_l^{(i)}$ on machine M_i , and vice versa.

If we set the length of the arc corresponding to $(x^{(i)})_{jt}$ equal to $c_{ijt} - \pi_j$, for all j and t , and the length of all idle time arcs equal to 0, then the reduced cost of the variable $\lambda_l^{(i)}$ is equal to the length of path corresponding to $\xi_l^{(i)}$ minus the dual variable α_i . Therefore, finding a column with minimal reduced costs boils down to finding the shortest path in a directed acyclic network with arbitrary weights. As the network is directed and acyclic, the shortest path problem can be solved in $\mathcal{O}(nT)$ time.

As we do not know of any analytical bounds on the number of columns that have to be generated and T might not be a polynomial in the input size, we have no polynomial-time guarantee for solving the LP problem.

As a corollary to the work of Schulz and Skutella, solving the LP problem and then applying the randomized rounding technique yields a performance guarantee of $3/2$. This method will be called the *LPX approach*.

3. Dominance Relations Among Lower Bounds

The LP-relaxations and the CQP-relaxation described in the previous section provide us with lower bounds. In the following theorems, we discuss the dominance relations between the lower bounds obtained by (LPX), (LPY), and (CQP).

THEOREM 1. *Let Z_{LPX} be the value of an optimal solution to (LPX) and (LPX') and let Z_{CQP} denote the value of an optimal solution to (CQP). Then $Z_{LPX} \geq Z_{CQP}$.*

PROOF. Consider a feasible solution $\lambda = (\lambda_l^{(i)})$ for (LPX'), and define, for $i = 1, \dots, m$, $z_i \in \mathbb{R}^n$ as

$$z_i = \sum_l \lambda_l n_l^{(i)},$$

where $n_l^{(i)}$ is the vector consisting of the first n elements of the column in (LPX') corresponding to variable $\lambda_l^{(i)}$, i.e., the j th element of $n_l^{(i)}$ is the number of copies of J_j occurring in semi-schedule $\xi_l^{(i)}$.

Let $c_i \in \mathbb{R}^n$ be given by $c_{ij} = w_j p_{ij}$ and $D_i = (d_{jk}^{(i)})_{jk} \in \mathbb{R}^{n \times n}$ be defined as $d_{jk}^{(i)} = \min(w_j p_{ik}, w_k p_{ij})$. If we define Z_i^C as

$$Z_i^C(z) = 1/2c_i^T z_i + 1/2z_i^T D_i z_i,$$

and $Z_{CQP}(z)$ as

$$Z_{CQP}(z) = \max \left(\sum_i c_i^T z_i, \sum_i Z_i^C(z) \right),$$

then $(z, Z) \in \mathbb{R}^{nm} \times \mathbb{R}$, with $z = (z_1, \dots, z_m)$ and $Z \geq Z_{CQP}(z)$, is a feasible solution for (CQP), as $z_{ij} \geq 0$ and $\sum_i z_{ij} = \sum_i \sum_l \lambda_l n_{lj}^{(i)} \stackrel{(26)}{=} 1$, for all j .

The sum of the completion times of all copies of J_j in semi-schedule $\xi_l^{(i)}$ is

$$C_{lj}^{(i)} = 1/2p_{ij}n_{lj}^{(i)} + 1/2p_{ij}(n_{lj}^{(i)})^2 + \sum_{k < i} p_{ik}n_{lk}^{(i)}n_{lj}^{(i)}.$$

Thus the weighted sum of completion times for a machine M_i in (LPX') is

$$\begin{aligned} Z_i^X(\lambda) &= \sum_{j,l} w_j \lambda_l^{(i)} C_{lj}^{(i)} \\ &= \sum_l \lambda_l^{(i)} (1/2c_i^T n_l^{(i)} + 1/2(n_l^{(i)})^T D_i n_l^{(i)}). \end{aligned}$$

As all elements of $n_l^{(i)}$ are integer, for all l and i , we know that

$$\begin{aligned} Z_i^X(\lambda) &= \sum_{j,l} w_j \lambda_l^{(i)} \\ &\times \left(1/2p_{ij}n_{lj}^{(i)} + 1/2p_{ij}(n_{lj}^{(i)})^2 + \sum_{k < i} p_{ik}n_{lk}^{(i)}n_{lj}^{(i)} \right) \\ &\geq \sum_{j,l} w_j \lambda_l^{(i)} (1/2p_{ij}n_{lj}^{(i)} + 1/2p_{ij}n_{lj}^{(i)}) \\ &= c_i^T \left(\sum_l \lambda_l^{(i)} n_l^{(i)} \right) = c_i^T z_i. \end{aligned} \quad (30)$$

Skutella (1998) proved that the matrix D_i is positive semi-definite, thus the function $f_i(x) = 1/2c_i^T x + 1/2x^T D_i x$ is convex. By constraint (27) $\sum_l \lambda_l^{(i)} = 1$ and thus

$$Z_i^X(\lambda) = \sum_l \lambda_l^{(i)} f_i(n_l^{(i)}) \geq f_i \left(\sum_l \lambda_l^{(i)} n_l^{(i)} \right) = Z_i^C(z). \quad (31)$$

By (30) we know that $\sum_i Z_i^X(\lambda) \geq \sum_i c_i^T z_i$ and by (31) we know that $\sum_i Z_i^X(\lambda) \geq \sum_i Z_i^C(z)$. Hence,

$$Z_{LPX}(\lambda) \geq \max \left(\sum_i c_i^T z_i, \sum_i Z_i^C(z) \right) = Z_{CQP}(z).$$

Thus, any feasible solution for (LPX') can be converted to a feasible solution for (CQP) with value at most equal to the value of the solution for (LPX'). Hence, $Z_{LPX} \geq Z_{CQP}$. \square

THEOREM 2. Let Z_{LPX} be the value of an optimal solution to (LPX) and (LPX') and let Z_{LPY} denote the optimal value to (LPY) plus constraints (14). Then $Z_{LPX} \geq Z_{LPY}$.

PROOF. Consider a feasible solution for (LPX) and construct a feasible solution, $(y, C^Y) \in \mathbb{R}^{nmT} \times \mathbb{R}^n$, for (LPY) as follows:

$$y_{ijt} = \sum_{t'=t-p_{ij}+1}^t x_{ijt'}, \quad (32)$$

$$C_j^Y = \max \left(\sum_{i,t} y_{ijt}, \sum_{i,t} y_{ijt}/p_{ij}(t+1/2) + 1/2 y_{ijt} \right). \quad (33)$$

It is easy to verify that y satisfies constraints (9), (10), and (14). The right-hand side of constraint (12) is

$$\sum_{i,t} y_{ijt} = \sum_{i,t} \sum_{t'=t-p_{ij}+1}^t x_{ijt'} = \sum_{i,t} p_{ij} x_{ijt} \leq \sum_{i,t} (t+p_{ij}) x_{ijt},$$

and the right-hand side of constraint (11) is

$$\begin{aligned} & \sum_{i,t} (y_{ijt}/p_{ij}(t+1/2) + 1/2 y_{ijt}) \\ &= \sum_{i,t} \sum_{t'=t-p_{ij}+1}^t x_{ijt'} (t+1/2/p_{ij} + 1/2) \\ &= \sum_{i,t'} x_{ijt'} \sum_{t=t'}^{t'+p_{ij}-1} (t+1/2/p_{ij} + 1/2) \\ &= \sum_{i,t} (t+p_{ij}) x_{ijt}. \end{aligned}$$

Hence, $C_j^Y = \max(\sum_{i,t} y_{ijt}, \sum_{i,t} y_{ijt}/p_{ij}(t+1/2) + 1/2 \times y_{ijt}) \leq \sum_{i,t} (t+p_{ij}) x_{ijt}$ and the optimal value to (LPX) is at least as large as the optimal value to (LPY). \square

To establish the relation between (LPY) and (CQP), we need the following lemma, which specifies an optimal solution to (LPY) without constraints (12) for a given fractional assignment of the jobs to the machines.

LEMMA 1. Let $z = (z_{ij}) \in \mathbb{R}_+^{nm}$ and let $\bar{y} \in \mathbb{R}_+^{nmT}$ be

$$\bar{y}_{ijt} = |[t, t+1) \cap [S_{ij}, S_{ij} + p_{ij} z_{ij})|,$$

where $S_{ij} = \sum_{k<j} p_{ik} z_{ik}$. Then \bar{y} minimizes

$$f(y) = \sum_j w_j \sum_{it} (y_{ijt}/p_{ij}(t+1/2) + 1/2 y_{ijt})$$

over all $y \in Y(z) = \{y \in \mathbb{R}_+^{nmT} : \sum_j y_{ijt} \leq 1, \sum_t y_{ijt}/p_{ij} = z_{ij}\}$.

PROOF. First note that, by construction, \bar{y} satisfies $\sum_j \bar{y}_{ijt} \leq 1$ for all i, t , and $\sum_t y_{ijt}/p_{ij} = z_{ij}$, and so $\bar{y} \in Y(z)$. In addition \bar{y} has the property that for all i there exist an s_i and an $\alpha_i \in [0, 1)$ such that

$$\sum_j \bar{y}_{ijt} = \begin{cases} 1 & \text{if } t < s_i, \\ \alpha_i & \text{if } t = s_i, \\ 0 & \text{if } t > s_i. \end{cases}$$

Consider a solution $y \in Y(z)$, such that $y \neq \bar{y}$. Then there exists a triple (i, j, t) such that $y_{ijt} < \bar{y}_{ijt}$. Let (i, k, t') be such a triple, with minimal t' . Note that by construction of \bar{y} and minimality of t' , $y_{ijt} = \bar{y}_{ijt}$ for all $t < t'$, and thus there exists a $t'' > t'$ such that $y_{ikt''} > \bar{y}_{ikt''}$.

If $\sum_j y_{ijt'} < 1$, then we construct y' by adding $\beta = \min(1 - \sum_j y_{ijt'}, y_{ikt''})$ to $y_{ikt''}$ and subtracting the same value from $y_{ikt''}$. Clearly, $y' \in Y(z)$, and $f(y) - f(y') = w_k/p_{ik}\beta(t'' - t') > 0$.

If $\sum_j y_{ijt'} = 1$, then there is an l such that $y_{il't'} > \bar{y}_{il't'} \geq 0$. By construction of \bar{y} we have $k <_i l$. Let $\beta = \min(y_{il't'}, y_{ikt''}, \bar{y}_{ikt''} - y_{ikt''})$ and construct y' by adding β to $y_{ikt''}$ and $y_{il't''}$ and subtracting it from $y_{ikt''}$ and $y_{il't'}$. Then, clearly $y' \in Y(z)$ and

$$\begin{aligned} f(y) - f(y') &= w_k/p_{ik}\beta(t'' - t') + w_l/p_{il}\beta(t' - t'') \\ &= \beta(t'' - t')(w_k/p_{ik} - w_l/p_{il}) \geq 0. \end{aligned}$$

Hence, we have constructed a solution that has the same or lower value and is closer to \bar{y} . Setting $y = y'$ and repeating this procedure results in the solution \bar{y} , and it has the same or lower value than all intermediate solutions. That is, $f(\bar{y}) \leq f(y)$ for all $y \in Y(z)$. \square

THEOREM 3. Let Z_{LPY} be the value of an optimal solution to (LPY) and let Z_{CQP} denote the optimal solution value to (CQP). Then $Z_{LPY} \geq Z_{CQP}$.

PROOF. Let $(y, C^Y) \in \mathbb{R}^{nmT} \times \mathbb{R}^n$ be a feasible solution for (LPY) and let $z \in \mathbb{R}^{nm}$ be defined as $z_{ij} = \sum_t y_{ijt}/p_{ij}$, and $Z_{CQP}(z)$ as

$$\begin{aligned} Z_{CQP}(z) &= \max \left(\sum_{i,j} w_j p_{ij} z_{ij}, \sum_{i,j} 1/2 w_j p_{ij} z_{ij} \right. \\ & \quad \left. + 1/2 w_j p_{ij} z_{ij}^2 + \sum_{k<j} p_{ik} z_{ij} z_{ik} \right). \end{aligned}$$

Then $(z, Z_{CQP}(z))$ is a feasible solution for (CQP), and

$$\sum_{i,j} w_j p_{ij} z_{ij} = \sum_j w_j \sum_{i,t} y_{ijt} \leq \sum_j w_j C_j^Y.$$

Below, we prove that

$$\sum_{i,j} 1/2 w_j p_{ij} z_{ij} + 1/2 w_j p_{ij} z_{ij}^2 + \sum_{k < i,j} p_{ik} z_{ij} z_{ik} \leq \sum_j w_j C_j^Y(y).$$

Let \bar{y} be the feasible solution for (LPY) as defined in Lemma 1, i.e.,

$$\bar{y}_{ijt} = |[t, t+1) \cap [S_{ij}, S_{ij} + p_{ij} z_{ij})|,$$

where $S_{ij} = \sum_{k < i,j} p_{ik} z_{ik}$. Let $\alpha_{ij} = S_{ij} - \lfloor S_{ij} \rfloor$ and $\beta_{ij} = \lceil S_{ij} + p_{ij} z_{ij} \rceil - (S_{ij} + p_{ij} z_{ij})$. Then

$$\bar{y}_{ijt} = \begin{cases} 1 - \alpha_{ij} & \text{if } t = \lfloor S_{ij} \rfloor \text{ and} \\ & \lfloor S_{ij} \rfloor < \lceil S_{ij} + p_{ij} z_{ij} \rceil - 1, \\ 1 - \beta_{ij} & \text{if } t = \lceil S_{ij} + p_{ij} z_{ij} \rceil - 1 \text{ and} \\ & \lfloor S_{ij} \rfloor < \lceil S_{ij} + p_{ij} z_{ij} \rceil - 1, \\ 1 - \alpha_{ij} - \beta_{ij} & \text{if } t = \lfloor S_{ij} \rfloor = \lceil S_{ij} + p_{ij} z_{ij} \rceil - 1, \\ 1 & \text{if } \lfloor S_{ij} \rfloor < t < \lceil S_{ij} + p_{ij} z_{ij} \rceil - 1, \\ 0 & \text{otherwise.} \end{cases}$$

Let $C_{ij}^Y(\bar{y})$ be defined as

$$C_{ij}^Y(\bar{y}) = \sum_t (\bar{y}_{ijt} / p_{ij} (t + 1/2) + 1/2 \bar{y}_{ijt}).$$

If $\lfloor S_{ij} \rfloor = \lceil S_{ij} + p_{ij} z_{ij} \rceil - 1$, then $1 - \alpha_{ij} - \beta_{ij} = p_{ij} z_{ij}$ and

$$\begin{aligned} C_{ij}^Y(\bar{y}) &= 1 - \alpha_{ij} - \beta_{ij} / p_{ij} (S_{ij} - \alpha_{ij} + 1/2) + 1/2 p_{ij} z_{ij} \\ &= 1/2 p_{ij} z_{ij} + z_{ij} (S_{ij} + 1/2 p_{ij} z_{ij}) + 1 - \alpha_{ij} \\ &\quad - \beta_{ij} / p_{ij} (-1/2 (1 - \alpha_{ij} - \beta_{ij}) - \alpha_{ij} + 1/2) \\ &= 1/2 p_{ij} z_{ij} + 1/2 p_{ij} z_{ij}^2 + z_{ij} S_{ij} \\ &\quad + 1/p_{ij} 1/2 (\alpha_{ij} - \beta_{ij}) (\alpha_{ij} + \beta_{ij} - 1). \end{aligned} \quad (34)$$

If $\lfloor S_{ij} \rfloor < \lceil S_{ij} + p_{ij} z_{ij} \rceil - 1$, then

$$\begin{aligned} C_{ij}^Y(\bar{y}) &= 1 - \alpha_{ij} / p_{ij} (S_{ij} - \alpha_{ij} + 1/2) + 1 - \beta_{ij} / p_{ij} \\ &\quad \times (S_{ij} + p_{ij} z_{ij} + \beta_{ij} - 1/2) \\ &\quad + 1/p_{ij} (p_{ij} z_{ij} - 2 + \alpha_{ij} + \beta_{ij}) \\ &\quad \times (S_{ij} + 1/2 p_{ij} z_{ij} - 1/2 (\alpha_{ij} - \beta_{ij})) + 1/2 p_{ij} z_{ij} \\ &= 1/2 p_{ij} z_{ij} + 1/p_{ij} (p_{ij} z_{ij} + \alpha_{ij} + \beta_{ij}) \\ &\quad \times (S_{ij} + 1/2 p_{ij} z_{ij} - 1/2 (\alpha_{ij} - \beta_{ij})) \end{aligned}$$

$$\begin{aligned} &- \alpha_{ij} / p_{ij} (S_{ij} - \alpha_{ij} + 1/2) - \beta_{ij} / p_{ij} \\ &\quad \times (S_{ij} + p_{ij} z_{ij} + \beta_{ij} - 1/2) \\ &= 1/2 p_{ij} z_{ij} + 1/2 p_{ij} z_{ij}^2 + z_{ij} S_{ij} \\ &\quad + 1/p_{ij} 1/2 (\alpha_{ij} - \beta_{ij}) (\alpha_{ij} + \beta_{ij} - 1). \end{aligned} \quad (35)$$

Thus in both cases

$$\begin{aligned} C_{ij}^Y(\bar{y}) &= 1/2 p_{ij} z_{ij} + 1/2 p_{ij} z_{ij}^2 + z_{ij} S_{ij} \\ &\quad + 1/p_{ij} 1/2 (\alpha_{ij} - \beta_{ij}) (\alpha_{ij} + \beta_{ij} - 1). \end{aligned}$$

Consider a machine M_i and assume w.l.o.g. that $\{j : p_{ij} z_{ij} > 0\} = \{1, \dots, K\}$, and that $J_1 < \dots < J_K$. Then $\alpha_{i1} = 0$ and, as $S_{i,j+1} = S_{ij} + p_{ij} z_{ij}$, $\alpha_{i,j+1} = 1 - \beta_{ij}$, for $j = 1, \dots, K - 1$.

Then

$$\begin{aligned} &\sum_j w_j C_{ij}^Y(\bar{y}) \\ &= \sum_j \left(1/2 w_j p_{ij} z_{ij} + 1/2 w_j p_{ij} z_{ij}^2 + \sum_{k < i,j} w_j p_{ik} z_{ij} z_{ik} \right) \\ &\quad + \sum_j w_j / p_{ij} 1/2 (\alpha_{ij} - \beta_{ij}) (\alpha_{ij} + \beta_{ij} - 1) \\ &= \sum_j \left(1/2 w_j p_{ij} z_{ij} + 1/2 w_j p_{ij} z_{ij}^2 + \sum_{k < i,j} w_j p_{ik} z_{ij} z_{ik} \right) \\ &\quad + \sum_{j=1}^{K-1} 1/2 (\beta_{ij} - \beta_{ij}^2) (w_j / p_{ij} - w_{j+1} / p_{i,j+1}) \\ &\quad + 1/2 (\beta_{iK} - \beta_{iK}^2) w_K / p_{iK} \\ &\geq \sum_j \left(1/2 w_j p_{ij} z_{ij} + 1/2 w_j p_{ij} z_{ij}^2 + \sum_{k < i,j} w_j p_{ik} z_{ij} z_{ik} \right). \end{aligned} \quad (36)$$

The last inequality is true as $0 \leq \beta_{ij} < 1$ and thus $\beta_{ij} - \beta_{ij}^2 \geq 0$ and by the ordering of the jobs, we have that $w_j / p_{ij} \geq w_{j+1} / p_{i,j+1}$.

Hence, we have that

$$\begin{aligned} &\sum_{i,j} \left(1/2 w_j p_{ij} z_{ij} + 1/2 w_j p_{ij} z_{ij}^2 + \sum_{k < i,j} w_j p_{ik} z_{ij} z_{ik} \right) \\ &\leq \sum_{i,j} w_j C_{ij}^Y(\bar{y}) \leq \sum_{i,j} w_j C_{ij}^Y(y). \end{aligned}$$

The last inequality is due to Lemma 1. Thus $Z_{CQP}(z) \leq Z_{LPY}(y)$. \square

(LPY) plus constraints (14) yields a higher lower bound than (LPY), and thus also a higher bound than (CQP). However, as (LPY') is an underestimate of (LPY) plus constraints (14), it does not need to be the case that $Z_{LPY'} \geq Z_{CQP}$, where $Z_{LPY'}$ is the optimal solution to (LPY').

The inequalities in the above theorems are strict, as is shown in the following example.

EXAMPLE 1. Consider an instance with three jobs and two machines. Job J_1 can only be processed on machine M_1 and has processing time $p_{11} = 3$ and weight $w_1 = 11$. J_2 can only be processed on machine M_2 and has weight $w_2 = 1$ and processing time $p_{22} = 1$. Job J_3 can be processed on both machines. Its processing times are $p_{13} = 2$ and $p_{23} = 6$ and its weight is $w_3 = 7$.

The optimal solution to (LPX) is the optimal schedule: J_1 and J_3 are processed by M_1 and J_2 is scheduled on machine M_2 . The optimal value to (LPX) is $Z_{LPX} = 69$. The optimal solution to (LPY) with or without constraints (14) has value $Z_{LPY} = 671/2$; in this solution J_1 is assigned to the time slots $[0, 1)$, $[1, 2)$ and $[2, 3)$ on M_1 , J_2 is assigned to the time slot $[2, 3)$ on M_2 , and J_3 is fully assigned to the time slots $[3, 4)$ on M_1 and $[0, 1)$ and $[1, 2)$ on M_2 and for one third to time slot $[4, 5)$ on M_1 . Finally, the optimal solution to (CQP) assigns J_3 for $41/56$ to M_1 and for $15/56$ to M_2 . The optimal value to (CQP) is $Z_{CQP} = 7503/112 = 66.991$.

4. Local Search

We compare the algorithms described in the previous section to local search heuristics. *Local search* is a family of methods that iteratively search through the set of feasible solutions. Starting from an initial solution, a local search procedure moves from one feasible solution to a neighboring solution until some stopping criteria are met.

The choice of a suitable neighborhood function has an important influence on the performance of local search methods. Recall that the problem can be reduced to the problem of assigning the jobs appropriately to the machines. Hence, we represent a schedule by the assignment of the jobs to the machines.

We consider two types of neighborhood functions. First, for the *jump* neighborhood, we select a job J_j

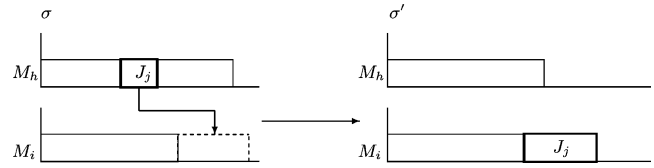


Figure 1 Jump

and a machine M_i such that J_j is not scheduled on machine M_i . A neighbor is formed by moving job J_j to machine M_i (see Figure 1).

The second neighborhood function is called *swap*. For this neighborhood, we select two jobs J_j and J_k , assigned to different machines, and the neighbor is obtained by interchanging their machine allocation (see Figure 2).

Besides applying the local search methods to the solutions obtained by the heuristics described in the previous section, they are also applied to randomly generated initial solutions. We have two strategies to generate these random solutions. The first strategy is the *completely random* strategy, in which we assign each job independently and uniformly to one of the m machines. In the second strategy, called *random greedy*, the jobs are, in random order, greedily assigned to the machines, that is, given a partial schedule and the first job that still has to be scheduled, the job is assigned to the machine for which the total weighted completion time of the new partial schedule is minimal.

We have implemented two local search heuristics: multi-start iterative improvement and tabu search.

4.1. Multi-Start Iterative Improvement

Iterative improvement is a simple form of local search. A neighbor is only accepted if this solution has lower cost. If no neighbor has lower cost, a *local minimum* has been found and the procedure terminates.

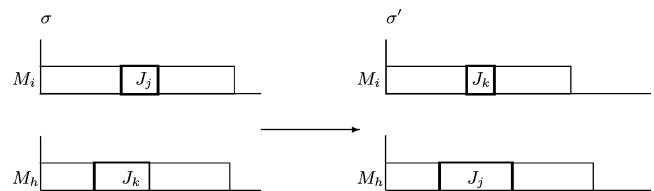


Figure 2 Swap

There are several ways to select the neighbor to move to. First, there is *first improvement*: We move to the first neighbor encountered that has lower cost. Another strategy for selecting the neighbor is *best improvement*: Move to the neighbor that has lowest cost among all neighbors. Initial tests showed that the neighbor selecting methods perform equally well.

Multi-start iterative improvement refers to a repeated application of the iterative improvement procedure on multiple initial solutions. Note that we only apply multi-start iterative improvement on randomly generated starting solutions. To make a fair comparison with the other methods, the number of repetitions is chosen such that the total time spent on this procedure is about the same as that for the others.

4.2. Tabu Search

In *tabu search*, moves to non-improving neighboring solutions are also allowed, so that we can get out of local minima. To avoid cycling, that is, returning to the same solutions, some information about solutions visited in past iterations are stored in a so-called *tabu list*. If a neighbor satisfies the properties of an entry in the tabu list, this solution is tabu and it may not be visited, unless it has value that is lower than the best found solution so far or satisfies some other aspiration criteria. The tabu list contains a limited number of entries and thus a limited number of forbidden properties.

The information stored in the tabu list due to a jump move is the job that has changed its machine allocation. If the move is due to a swap, we look at the contribution of the two swapped jobs to the objective function before and after the swap. The swapped job that has lowest decrease or highest increase in its contribution is stored in the tabu list. Hence, after each move, jump, or swap, we store one job in the tabu list. This job has to remain on its new machine for a number of iterations, unless moving it yields a better solution than found so far. The number of iterations during which a job has a fixed machine allocation is equal to the length of the tabu list which, after some initial experiments, we have chosen to be $n/2$ whenever $n < 40$ and 20 otherwise.

In our tabu search procedure, we use *best improvement* for finding the neighbor to move to. However, if

a move is made to a non-improving neighbor, we only allow jumps. The reason for this is that a jump creates more space on the machine from which the job is leaving than a swap and thus allows for better improvements in the subsequent iterations. Initial tests show indeed that tabu search with this feature gives better results than tabu search where non-improving swaps are also accepted.

Another feature of our tabu search procedure is the *backjump*; cf. the tabu search procedure of Nowicki and Smutnicki (1996) for the job-shop problem: If we have made 500 non-improving moves without improving the best found solution, we return to this solution and move to a neighbor that has not been visited yet directly from this solution.

In the case that tabu search is applied to the solutions obtained by the heuristics with constant performance guarantee, the procedure terminates when there have been too many, that is 20, backjumps to the same solution. In the case of randomly generated start solutions, we repeat the procedure of applying tabu search to a start solution until the total time spent is about the same as for the other procedures.

5. Computational Experience

5.1. Test Instances

Our heuristics have been tested on instances with size varying from 10 jobs and 5 machines to 100 jobs and 50 machines. The solution quality may depend on the structure of the test instances. To allow for possible variations in performance, three classes of test instances were considered, each of which is based on a different method of generating the processing time p_{ij} of job J_j on machine M_i .

- *No correlation*: All processing times p_{ij} are independently drawn from the uniform distribution over $[10, 100]$; w_j is an integer from the uniform distribution over $[1, 100]$.

- *Machine correlation*: p_{ij} is an integer from the uniform distribution over $[\alpha_i, \alpha_i + 10]$ where α_i is an integer from the uniform distribution over $[1, 100]$; w_j is an integer from the uniform distribution over $[1, 100]$.

- *Favorite machines*: Each job J_j has two favorite machines, $M_{i_1(j)}$ and $M_{i_2(j)}$, which are randomly

Table 1 Lower Bounds: Average (Maximum) Relative Deviation in From Best Lower Bound

$n \times m$	10×10	20×20	50×50	10×5	20×10	50×20	100×50	Average
<i>CQP</i>	-5.13 (-11.82)	-5.17 (-10.23)	-4.35 (-9.02)	-3.69 (-8.02)	-3.67 (-8.46)	-2.76 (-4.98)	-3.35 (-5.69)	-4.02
<i>LPY'</i>	-4.48 (-10.28)	-4.41 (-9.20)	-3.85 (-8.65)	-5.01 (-9.08)	-5.10 (-8.65)	-4.95 (-7.14)	-5.00 (-7.81)	-4.68

selected; $p_{i_1(j),j}$ and $p_{i_2(j),j}$ are drawn from the uniform distribution over $[\beta_j, \beta_j + 4]$, where β_j is an integer from the uniform distribution over $[15, 25]$, and p_{ij} ($i \neq i_1(j), i_2(j)$) is drawn from the uniform distribution over $[70, 90]$; w_j is an integer from the uniform distribution over $[1, 100]$.

For each value of m and n , 50 instances have been generated for each of the three instance classes.

5.2. Implementational Details

Multi-start iterative improvement, tabu search, and both LP approaches have been implemented in C, using CPLEX for solving the LP problems. The *CQP* approach has been coded in MATLAB, and we use SEDUMI (Sturm 1999) for solving the *CQP* problem. The reason for implementing the *CQP* approach in MATLAB instead of C is that SEDUMI is only available in MATLAB. The tests have been run on a Sun Ultra-1, 140 MHz, with 256 MB memory.

5.3. Computational Results

Before looking how good the schedules, obtained by the several algorithms, are, we first look at the quality of the lower bounds. In Table 1, we show the average relative deviation of the lower bounds obtained by the *CQP* and *LPY'*-relaxation from the best obtained lower bounds, which are all obtained by the *LPX*-relaxation as shown in Section 3. The average and maximum are taken over all instances.

We see that for the square instances, where the number of jobs is equal to the number of machines,

the *LPY'*-relaxation yields somewhat better lower bounds, whereas the *CQP*-relaxation yields the better lower bounds for the rectangular instances. The reason can be found in the fact that the schedule length for square instances is on average shorter than that for the rectangular instances with the same number of machines and, therefore, the *LPY'*-relaxation is closer to the *LPY*-relaxation. On average, the *CQP* and *LPY'*-relaxation are about 4% to 5% from the lower bounds obtained by the *LPX*-relaxation. For the uncorrelated and favorite machine instances, there is the same behavior, whereas for the machine correlated instances the *CQP*-relaxation is better for the square as well as the rectangular instances; these results are found in Table 2. In the machine correlated instances all jobs have the same favored machine, which results in a somewhat higher schedule length. Because of this higher schedule length, the *LPY'*-relaxation has a larger deviation from the *LPY*-relaxation, resulting in a worse lower bound.

The upper bounds, i.e., values of schedules, obtained by the *CQP*, *LPY*, and *LPX* approach are given in Table 3.

The *LPX* approach yields the best upper bounds of these three methods, on average around 0.13% from the best lower bound. The reason that the *LPX* approach yields better solutions is that the solution to the *LPX*-relaxation is much closer to a real schedule than the solutions to the *CQP* and *LPY'*-relaxations, that is, the number of jobs J_j for which there is a machine M_i such that $\sum_t x_{ijt} = 1$ is much larger than

Table 2 Lower Bounds: Average Relative Deviation in % From Best Lower Bound— Machine-Correlated Instances

$n \times m$	10×10	20×20	50×50	10×5	20×10	50×20	100×50	Average
<i>CQP</i>	-6.27	-6.04	-5.27	-4.12	-4.57	-3.68	-3.83	-4.83
<i>LPY'</i>	-7.17	-7.31	-7.12	-6.26	-6.61	-6.25	-6.52	-6.75

Table 3 Heuristics With Constant Guarantee: Average (Maximum) Relative Deviation in % From Best Lower Bound

$n \times m$	10 × 10	20 × 20	50 × 50	10 × 5	20 × 10	50 × 20	100 × 50	Average
<i>CQP</i>	1.20 (9.38)	1.23 (6.57)	1.48 (3.56)	1.02 (5.61)	1.19 (4.55)	1.19 (3.46)	1.37 (3.26)	1.24
<i>LPY</i>	0.69 (7.05)	0.77 (2.99)	0.81 (2.68)	0.84 (4.77)	0.96 (4.10)	1.05 (3.07)	1.12 (3.01)	0.89
<i>LPX</i>	0.07 (1.51)	0.08 (1.25)	0.06 (0.71)	0.15 (3.48)	0.16 (2.65)	0.20 (0.91)	0.17 (1.24)	0.13

the number of jobs for which there is a machine M_i such that $\sum_t y_{ijt}/p_{ij} = 1$ or $z_{ij} = 1$, respectively.

In Table 4 we report on the average relative deviation from the best lower bound obtained by the *CQP*, *LPY*, and *LPX* approaches for the machine correlated instances. Although the lower bounds for the *LPY'*-relaxation are worse than those for the *CQP*-relaxation, we see that the values of the schedules obtained by the *LPY* approach are not worse than those obtained by the *CQP* approach. The reason for this is that the average number of fractional assignments obtained by the *CQP*-relaxation is not less than the number of fractional assignment obtained by the *LPY'*-relaxation.

Table 5 shows the results of applying iterative improvement, denoted by II, and tabu search, denoted by TS, to the solutions of the heuristics with a constant guarantee. We see that applying a simple iterative improvement procedure to the *CQP* and *LPY* solutions, although improving the solutions significantly, still yields results that are not better than the solution of the *LPX* approach. Applying tabu search to the solutions of the *CQP* and *LPY* approach yields results that are on average as good as the *LPX* approach; note that the tabu search solutions are all very close to optimal as their values are only about 0.1% away from the best lower bound. Tabu search

applied to the *LPX* solution yields the best results: on average less than 0.1% from the best lower bound.

In Table 6, the results of the local search heuristics are given. We have used two different time bounds as stopping criteria for the local search heuristics. The first one is that the time is equal to the time used by the *LPX* approach; the second time bound is that the time is equal to the time used by the *LPX* approach and applying tabu search to these solutions. The latter will be denoted by the extension "long." We see that tabu search applied to a good starting solution yields better results than applying it to random start solutions and the multi-start iterative improvement procedures. The multi-start iterative improvement procedures perform better than the tabu search applied to the randomly generated start solutions. This implies that it is hard to get out of the local optima and it is better to have many start solutions and do a simple improvement procedure than to do a more sophisticated improvement procedure on few start solutions.

In Table 7, we report on the performance of the several heuristics on the hardest instances for the *LPX* approach. For each size, we have chosen from all the instances the one for which the *LPX* approach has the worst performance ratio. We see that for these instances the tabu search procedure applied to the schedule obtained by the *LPX* approach yields a

Table 4 Heuristics With Constant Guarantee: Average Relative Deviation in % From Best Lower Bound—Machine-Correlated Instances

$n \times m$	10 × 10	20 × 20	50 × 50	10 × 5	20 × 10	50 × 20	100 × 50	Average
<i>CQP</i>	1.54	1.56	1.69	1.16	1.63	1.75	1.68	1.57
<i>LPY</i>	0.94	1.30	1.43	1.15	1.36	1.63	1.61	1.34
<i>LPX</i>	0.05	0.09	0.11	0.07	0.04	0.06	0.07	0.07

Table 5 Heuristics With Constant Guarantee Plus Local Search: Average Relative Deviation in % From Best Lower Bound

$n \times m$	10×10	20×20	50×50	10×5	20×10	50×20	100×50	Average
<i>CQP</i>	1.20	1.23	1.48	1.02	1.19	1.19	1.37	1.24
<i>CQP + II</i>	0.50	0.45	0.68	0.34	0.40	0.44	0.59	0.48
<i>CQP + TS</i>	0.06	0.06	0.18	0.12	0.11	0.18	0.27	0.14
<i>LPY</i>	0.69	0.77	0.81	0.84	0.96	1.05	1.12	0.89
<i>LPY + II</i>	0.29	0.23	0.36	0.25	0.35	0.40	0.47	0.34
<i>LPY + TS</i>	0.06	0.07	0.15	0.12	0.11	0.16	0.23	0.13
<i>LPX</i>	0.07	0.08	0.06	0.15	0.16	0.20	0.17	0.13
<i>LPX + II</i>	0.06	0.07	0.04	0.13	0.13	0.14	0.11	0.10
<i>LPX + TS</i>	0.04	0.05	0.03	0.12	0.09	0.11	0.08	0.08

schedule that is the best found or close to the best found schedule. Moreover, we see that the *CQP* and *LPY* approaches on these hard instances for the *LPX* approach do not perform much better, except for the *CQP* approach on the instance of size 10×10 .

Finally, Table 8 reports on the time it takes to find the solutions for the heuristics with guarantees. As the local search procedures that are applied to the randomly generated start solutions take about the same time as the *LPX* approach, we do not report on the time usage of these heuristics. Applying iterative improvement to the solutions obtained by the good start solutions takes about 0.02 seconds for the largest instances. Therefore, the time for obtaining the good start solution hardly differs from the time of obtaining it and then applying iterative improvement to it. Although solving the *LPX*-relaxation has no polynomial-time guarantee, we see that this method is fastest. The smallest instances are solved in a few seconds. The *CQP* approach needs on average more than 15 minutes, and the *LPY* approach even needs around 40 minutes for solving instances of size 100×50 . The *LPX* approach is much faster: It takes about 10

minutes. The application of tabu search takes about another 3 to 5 minutes for the largest instances.

6. Concluding Remarks

Our main goal in this paper was to make an empirical comparison of approximation algorithms with performance guarantees and local search heuristics for $R \parallel \sum w_j C_j$. The algorithms with worst-case performance guarantees are based on rounding solutions to relaxations of the scheduling problem; these relaxations also provide us with lower bounds. In Section 3, we proved that the best lower bounds are obtained by the *LPX*-relaxation. In Section 5, we saw that rounding the solution to this relaxation and then applying tabu search on this feasible schedule also yields the best upper bounds.

Comparing our results with the work of Savelsbergh et al. (1998) and Uma and Wein (1998) on $1|r_j| \sum w_j C_j$, we come to the same conclusions: Rounding a better relaxation yields a better schedule, and the best schedules are obtained by combining a heuristic based on a good relaxation and local search.

Table 6 LPX Versus Local Search: Average Relative Deviation in % From Best Lower Bound

$n \times m$	10×10	20×20	50×50	10×5	20×10	50×20	100×50	Average
<i>LPX</i>	0.07	0.08	0.06	0.15	0.16	0.20	0.17	0.13
<i>LPX + TS</i>	0.04	0.05	0.03	0.12	0.09	0.11	0.08	0.08
<i>II</i>	0.06	0.07	0.27	0.12	0.09	0.10	0.38	0.16
<i>II long</i>	0.05	0.05	0.22	0.12	0.09	0.09	0.37	0.14
<i>TS</i>	0.09	0.30	0.56	0.13	0.19	0.30	0.39	0.28
<i>TS long</i>	0.08	0.19	0.45	0.12	0.12	0.20	0.35	0.22

Table 7 Hardest Instances for LPX Approach

$n \times m$	10×10	20×20	50×50	10×5	20×10	50×20	100×50
<i>CQP</i>	0.03	0.62	1.51	4.11	2.25	1.52	1.43
<i>LPY</i>	1.51	0.62	2.21	4.11	1.38	1.49	1.40
<i>LPX</i>	1.51	1.25	0.71	3.48	2.65	0.91	1.24
<i>CQP + TS</i>	0.03	0.33	0.34	3.48	1.14	0.46	0.61
<i>LPY + TS</i>	0.03	0.33	0.64	3.48	1.14	0.46	0.55
<i>LPX + TS</i>	0.03	0.33	0.26	3.48	1.14	0.52	0.60
<i>II</i>	0.03	0.33	0.04	3.48	1.19	0.48	0.48
<i>II long</i>	0.03	0.33	0.03	3.48	1.14	0.48	0.48
<i>TS</i>	0.77	0.79	0.24	3.48	1.57	0.52	0.57
<i>TS long</i>	0.77	0.33	0.24	3.48	1.35	0.52	0.57

With the work of Hariri and Potts (1991) and Glass et al. (1994) on $R||C_{\max}$ there is some difference. In an empirical evaluation of heuristics with a constant performance guarantee, Hariri and Potts concluded that the performance of these heuristics is unsatisfactory, as a deviation from the optimal solution of more than 10% was normal. Applying a simple improvement procedure resulted in a significant improvement of the schedules. Glass et al. showed that some good local search heuristics yield schedules within 1% of the best found solution. The procedures with a constant performance guarantee that we evaluated resulted in solutions that were within 1 or 2% of optimal and for the LPX approach even within 0.2%. The reason that the heuristics with constant performance guarantee we considered are much better can be found in the objective function. We considered the sum of weighted completion times and each job contributes to the objective function. Some jobs will have lower completion time compared to that of an optimal schedule, and other jobs will have higher completion time. In $R||C_{\max}$ the objective is makespan minimization, so we only look at the last job to

be completed: Of course this will never be lower than in an optimal schedule. Also the objective value for the sum of weighted completion times is much larger than for makespan minimization, and the same absolute difference yields a lower relative difference for the problem considered in this paper than for makespan minimization problems.

Acknowledgments

The authors thank Jan Karel Lenstra for his useful comments. This research was supported by the project "High Performance Methods for Mathematical Optimization" of The Netherlands Organization for Scientific Research (NWO).

References

- Bruno, J. L., E. G. Coffman, Jr., R. Sethi. 1974. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM* **17** 382–387.
- Dyer, M. E., L. A. Wolsey. 1990. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics* **26** 255–270.
- Glass, C. A., C. N. Potts, P. Shade. 1994. Unrelated parallel machine scheduling using local search. *Mathematical and Computer Modelling* **20** 41–52.

Table 8 Time Usage in Seconds: Average Over All Instances

$n \times m$	10×10	20×20	50×50	10×5	20×10	50×20	100×50
<i>CQP</i>	1.66	6.72	153.92	1.12	3.46	41.22	985.98
<i>CQP + TS</i>	1.75	8.92	225.79	1.21	5.25	95.66	1278.72
<i>LPY</i>	2.00	13.33	442.15	0.81	5.93	68.58	2407.76
<i>LPY + TS</i>	2.08	15.40	507.83	0.90	7.70	122.55	2684.35
<i>LPX</i>	0.05	0.41	14.36	0.08	0.59	28.56	684.89
<i>LPX + TS</i>	0.12	2.04	42.47	0.16	1.98	66.57	883.09

- Graham, R. L., E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics* 5 287–326.
- Hall, L. A., A. S. Schulz, D. B. Shmoys, J. Wein. 1997. Scheduling to minimize average completion time: off-line and on-line algorithms. *Mathematics of Operations Research* 22 513–544.
- Hariri, A. M. A., C. N. Potts. 1991. Heuristics for scheduling unrelated parallel machines. *Computers and Operations Research* 18 323–331.
- Hoogeveen, J. A., P. Schuurman, G. J. Woeginger. 1998. Non-approximability results for scheduling problems with minsum criteria. *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization (IPCO'98)*, LNCS 1412, Springer, Berlin, Germany. 353–366.
- Horn, W. A. 1973. Minimizing average flow time with parallel machines. *Operations Research* 21 846–847.
- Johnson, D. S., L. A. McGeoch. 1997. The traveling salesman problem: a case study. E. H. L. Aarts and J. K. Lenstra, eds. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Chichester, U.K.
- Lenstra, J. K., A. H. G. Rinnooy Kan, P. Brucker. 1977. Complexity of machine scheduling problems. *Annals of Discrete Mathematics* 1 343–362.
- Nowicki, E., C. Smutnicki. 1996. A fast taboo search algorithm for the job shop problem. *Management Science* 42 797–913.
- Phillips, C., C. Stein, J. Wein. 1997. Task scheduling in networks. *SIAM Journal on Discrete Mathematics* 10 573–598.
- Savelsbergh, M. W. P., R. N. Uma, J. Wein. 1998. An experimental study of linear programming-based scheduling heuristics. *Proceedings of 8th ACM-SIAM Symposium on Discrete Algorithms*. SIAM, Philadelphia, PA. 453–461.
- Schulz, A. S., M. Skutella. 1997a. Scheduling-LPs bear probabilities: randomized approximations for min-sum criteria. R. Burkard and G. Woeginger, eds. *Algorithms—ESA'97*, LNCS 1284, Springer, Berlin, Germany. 416–429.
- Schulz, A. S., M. Skutella. 1997b. Random-based scheduling: new approximations and LP lower bounds. J. Rolim, ed. *Randomization and Approximation Techniques in Computer Science*, LNCS 1296, Springer, Berlin, Germany. 119–133.
- Shmoys, D. B., E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming* 62 461–474.
- Skutella, M. 1998. *Approximation and Randomization in Scheduling*. Ph.D. thesis, Fachbereich Mathematik, Technische Universität Berlin, Berlin, Germany.
- Skutella, M. 2001. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM* 48 206–242.
- Smith, W. E. 1956. Various optimizers for single-stage production. *Naval Research Logistics Quarterly* 3 59–66.
- Sturm, J. F. 1999. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software* 11–12 625–653.
- Uma, R. N., J. Wein. 1998. On the relationship between combinatorial and LP-based approaches to NP-hard scheduling problems. *IPCO: 6th Integer Programming and Combinatorial Optimization Conference*, LNCS 1412, Springer, Berlin, Germany.
- Van den Akker, J. M., C. A. J. Hurkens, M. W. P. Savelsbergh. 2000. Time-indexed formulations for machine scheduling problems: column generation. *INFORMS Journal on Computing* 12 111–124.
- Van der Linden, S. J. 2000. Convex quadratic relaxations and approximation algorithms: a computational study. Master's thesis, Department of Operational Research and Management, University of Amsterdam, Amsterdam, The Netherlands.

Accepted by W. David Kelton; received March 2001; revised November 2001; accepted November 2001.