Practical Comparison of Approximation Algorithms for Scheduling Problems [*]

E. C. Xavier[†] F.K. Miyazawa[†]

April 23, 2003

Abstract

In this paper we consider an experimental study of approximation algorithms for scheduling problems in parallel machines minimizing the average weighted completion time. We implemented approximation algorithms for the following problems: $P|r_j|\sum C_j$, $P||\sum w_jC_j$, $P|r_j|\sum w_jC_j$, $R||\sum w_jC_j$ and $R|r_j|\sum w_jC_j$. We generated about 900 tests over more than 190 different instances and present some practical aspects of the implemented algorithms. We also made an experimental comparison on two lower bounds based in the formulations used by the algorithms. The first one is a semidefinite formulation for the problem $R||\sum w_jC_j$ and the other is a linear formulation for the problem $R|r_j|\sum w_jC_j$. For all tests, the algorithms obtained very good results. We note that algorithms using more refined techniques, when compared to algorithms with simple strategies, does not necessary leads to better results. We also present two heuristics, based in approximation algorithms, that generates solutions with better quality in almost all instances considered.

Key Words: Approximation algorithms, scheduling problems, practical analysis.

1 Introduction

In this paper, we consider an experimental study of approximation algorithms for scheduling problems. Several job scheduling problems in parallel machines that appears in practice are *NP*-hard. The scheduling problems we consider are all *NP*-hard and consists in scheduling a set of jobs, under some restrictions, in a set of machines minimizing the average weighted completion time. If we consider that $P \neq NP$, we cannot solve these problems to optimality efficiently. This motivates the development of approximation algorithms, that are efficient and produces results with quality guarantee. We implemented some approximation algorithms to scheduling jobs in parallel machines and study their computational performance.

Given an algorithm $\mathcal{A}$ for a minimization problem and an instance I of this problem, we denote by $\mathcal{A}(I)$ the value of the solution returned by $\mathcal{A}$ when applied to the instance I and we denote by $OPT(I)$ the value of an optimal solution to I. We say that an algorithm $\mathcal{A}$ has an approximation

[*]This research was partially supported by FAPESP project 01/04412-4, MCT/CNPq under PRONEX program (Proc. 664107/97-4) and CNPq (Proc. 470608/01–3, 464114/00–4, 300301/98–7).

[†]Instituto de Computação — Universidade Estadual de Campinas, Caixa Postal 6176 — 13084–971 — Campinas–SP — Brazil, {eduardo.xavier,fkm}@ic.unicamp.br.

factor α, or is an α-approximation, if $\mathcal{A}(I)/\mathrm{OPT}(I) \leq \alpha$, for all instances I. When the algorithm $\mathcal{A}$ is probabilistic and the inequality $E[\mathcal{A}(I)]/\mathrm{OPT}(I) \leq \alpha$ is valid for all instances I, where $E[\mathcal{A}(I)]$ is the expected value of the solution returned by algorithm $\mathcal{A}$, we say that $\mathcal{A}$ is a probabilistic α-approximation algorithm.

For all problems considered in this paper, we denote by $J = \{1, \ldots, n\}$ the set of jobs and $M = \{1, \ldots, m\}$ the set of machines. When machines are unrelated (e.g. have different processing speed) we denote by p_{ij} the processing time of job j when executed on machine i, and by p_j when all machines are identical. We denote by r_j the release date of job j, which represents the moment that a job can start and denote by w_j the importance weight of finishing the job earlier. The completion time of the job is denoted by C_j.

Since we consider several scheduling problems, we use the notation $\alpha|\beta|\gamma$ introduced by Graham, Lawler, Lenstra and Rinnooy Kan [3]. In the following we detail the terms used in this paper under this notation. The term α corresponds to the machine environment, P for identical machines or R for unrelated machines. The term β tell us some restrictions about jobs, if they have release dates, r_j, if the jobs can be interrupted and continued later, $pmtn$, etc. Finally the term γ indicates the objective function we want to minimize.

All problems we consider are non-preemptive, although algorithms for preemptive problems are used to find intermediate solutions.

There are a lot of work in the development of approximation algorithms, but very few consider practical performance analysis. In [5], Hepner and Stein present an implementation of a PTAS for a single machine scheduling with release dates. In [7] Savelsbergh et al., also present an experimental study of some approximation algorithms for scheduling problems in a single machine. They consider the problem $1|r_j|\sum w_j C_j$ and a variant of this problem when an average weighted flow time is minimized, i.e, problem $1|r_j|\sum w_j(C_j - r_j)$.

We implemented algorithms for the following problems: $P|r_j|\sum C_j$, $P||\sum w_j C_j$, $P|r_j|\sum w_j C_j$, $R||\sum w_j C_j$ and $R|r_j|\sum w_j C_j$. For the problem $P|r_j|\sum C_j$ we implemented the algorithm developed by Phillips et al. [8]. This algorithm is combinatorial and is based in a heuristic for the preemptive case. For the problem $P||\sum w_j C_j$ we implemented the algorithm of Kawaguchi and Kyan [6], that is based in a list scheduling heuristic. For the problems $P|r_j|\sum w_j C_j$ and $R|r_j|\sum w_j C_j$ we implemented algorithms of Schulz and Skutella [9]. The algorithm for the first problem is combinatorial and the algorithm for the second problem is based on a solution of a linear program. Both algorithms are probabilistic. For the problem $R||\sum w_j C_j$ we implemented the algorithm developed by Skutella [10] that is based on a solution of a semidefinite program.

We observe that there exists PTAS algorithms for parallel machines as presented by Afrati et al. [1], but their implementation requires extra efforts and results in algorithms with time complexity represented by very high degree polynomials.

To our knowledge, this paper is the first to consider practical analysis of approximation algorithms for parallel machines. All algorithms are implemented in C. For the algorithms that require solutions of linear or quadratic programs we use the Xpress-MP library, of Dash Optimization [2]. Based in the practical results, we propose a simple modification on the algorithm presented by Schulz and Skutella [9] for $R|r_j|\sum w_j C_j$ and on the algorithm of Kawaguchi and Kyan [6]. In the tests we consider, we show that these heuristics obtain solutions with better quality.

The paper is organized as follows. In section 2 we describe the implemented algorithms and give some insight of how they work. In section 3 we compare two lower bounds for problems

$R||\sum w_jC_j$ and some of its particular cases. In section 4 we present the computational results.

2 Algorithms

In this section we describe the algorithms and the way they are implemented. We do not show how their approximate factors are obtained. The interested reader can find more details about the approximation results of these algorithms in the references.

All algorithms are implemented in C. To solve the linear and semidefinite programs considered, we use the Xpress-MP library due to its recognized efficiency.

2.1 Algorithm PSW for $P|r_j|\sum C_j$

The algorithm of this section was developed by Phillips et al. [8] which we denote by PSW. It finds a solution in two phases. In the first phase, the algorithm finds an approximated solution for the preemptive version of this problem and in the second phase it uses an algorithm that converts the preemptive schedule to a non-preemptive schedule. The preemptive version of this problem is already *NP*-hard, and is approximated by a 2-approximation algorithm. The algorithm that converts the preemptive schedule to a non-preemptive one, produces a new schedule that is at most three times worse than the preemptive schedule. This leads to a 6-approximation algorithm for the problem $P|r_j|\sum C_j$ (see [8]).

The algorithm for the preemptive schedule is based in the following idea: at any time, execute m jobs with the shortest remaining amount of work. The time complexity of the implemented algorithm, which we denote by *Preemptive*, is $O(n(\log n + m))$.

The algorithm PSW, uses the preemptive scheduled generated by the algorithm *Preemptive* scheduling each job j in the machine which completed j in the preemptive schedule.

The algorithm generates a list M_i, for each machine i, of jobs ordered by their preemptive completion times C_j in the preemptive schedule. For each machine i, the algorithm PSW generates a non-preemptive schedule with jobs in the order specified by M_i, with the constraint that no job j starts before its release date.

The time complexity of the implemented algorithm is $O(n \log n + m)$ plus the time complexity to generate the preemptive schedule.

2.2 Algorithm KK for the problem $P||\sum w_jC_j$

The algorithm of this section is an extension of the problem $1||\sum w_jC_j$ for the problem $P||\sum w_jC_j$. The problem $1||\sum w_jC_j$ can be solved optimally with the following algorithm developed by Smith [11]: order jobs in non-decreasing order of $\frac{p_j}{w_j}$ and schedule the jobs in this order. The aproximation algorithm for the parallel machine case is an extension: order jobs in non-decreasing order of $\frac{p_j}{w_j}$ and schedule jobs in this order every time a machine becomes free. Kawaguchi and Kyan [6] have shown that this algorithm generates schedules with a factor of $(\frac{\sqrt{2}+1}{2})$ of the optimal. The implemented algorithm, which we denote by KK, have time complexity $O(n \log n + nm)$.

2.3 Algorithm SzSk for the problem $P|r_j|\sum w_jC_j$

The algorithm of this section was developed by Schulz and Skutella [9] and we denote it by SzSk. The algorithm is a probabilistic 2-approximation algorithm. For each instance, the algorithm SzSk is executed 100 times and the best generated schedule is returned. In our experiments, we observed that more executions leads to very small improvements. The algorithm is related to the linear formulation for a single machine problem presented below. We have variables y_{jt}, for each job j and for each time interval $(t, t+1]$ that a job can run. We also have variables C_j, that represents the finishing time of job j. The constant T is an upper bound for the completion time of any job. The relaxed linear program, denoted by *LPS*, is the following:

$$\text{Min} \ \sum_{j\in J} w_jC_j$$

$$\begin{array}{rcl}
\sum_{t=r_j}^{T} y_{jt} & = & p_j \quad \forall j \in J, \\
\sum_{j\in J} y_{jt} & \leq & 1 \quad t = 0, ..., T, \\
C_j & = & \frac{p_j}{2} + \frac{1}{p_j}\sum_{t=r_j}^{T} y_{jt}(t+\frac{1}{2}) \forall j \in J, \\
y_{jt} & = & 0 \quad \forall j \in J \text{ and } t = 0, ..., r_j - 1, \\
y_{jt} & \geq & 0 \quad \forall j \in J \text{ and } t = r_j, ..., T.
\end{array}$$

This linear program can be solved using a combinatorial algorithm [9]. Suppose we have just one machine m times faster than the machines considered. Consider the processing times of the jobs to be m times smaller. Construct a preemptive schedule for this single machine with the new processing times using the following rule: at any time, construct a preemptive schedule S on the new single machine by scheduling, among the available jobs, the one with the smallest $\frac{p_j}{w_j}$ ratio. The resulting schedule corresponds to an optimum solution to the formulation. Each variable y_{jt} receive value 1 if job j is processed during time $[t-1, t)$ in the generated schedule.

Notice that the algorithm *Preemptive* is easily modified to solve this formulation and can be implemented to run in $O(n \log n)$. After this, we construct a schedule based in probabilistic assignments. We choose for each job j, a variable α_j uniformly distributed from the interval $[0, 1]$. Then we consider the probabilistic finishing time, i.e., the first time in the schedule where the total amount of work done is $p_j\alpha_j$. We denote this value by $C_j(\alpha_j)$. The algorithm SzSk attributes each job j uniformly and independent to one of the m machines. For each machine the algorithm schedules jobs in nondecreasing order of values $C_j(\alpha_j)$.

The time complexity of the algorithm SzSk is $O(n \log n + m)$.

2.4 Algorithm Sk for $R||\sum w_jC_j$

The algorithm of this section is a probabilistic 2-approximation algorithm based on a semidefinite formulation and is presented by Skutella [10]. We denote the algorithm by Sk. The quadratic program for this problem has binary variables a_{ij}, that says that a job j is to be processed in machine i, if and only if $a_{ij} = 1$ and variables C_j that represents the finishing time of job j. We also have a function $\prec_i$ that specify the execution order of a job pair j, k in machine i. The job j must be processed before k in machine i if $\frac{w_j}{p_{ij}} \geq \frac{w_k}{p_{ik}}$. The quadratic program is the following:

$$\text{Min } \sum_{j\in J} w_j C_j$$

$$\begin{array}{rcl} C_j &=& \sum_{i=1}^{m} a_{ij} \cdot (p_{ij} + \sum_{k \prec_i j} a_{ik} p_{ik}) \;\; \forall j \in J, \\ a_{ij} &\in& \{0,1\} \quad \forall i \in M, \quad \forall j \in J. \end{array}$$

Skutella have shown that this formulation is equivalent to the following quadratic formulation:

$$\text{Min } c^T a + \tfrac{1}{2} a^T D a$$

$$\begin{array}{rcll} \sum_{i=1}^{m} a_{ij} &=& 1 & \forall j \in J, \\ a &\geq& 0, & \end{array}$$

where $a \in \mathbb{R}^{mn}$ is a vector of all variables a_{ij} lexicographically ordered with respect to the natural order $1, 2, \ldots, m$ of the machines and then, for each machine i, the jobs ordered according to $\prec_i$. The vector $c \in \mathbb{R}^{mn}$ is given by $c_{ij} = w_j p_{ij}$ and $D = (d_{(ij)(hk)})$ is a symmetric $mn \times mn$-matrix given by

$$d_{(ij)(hk)} = \begin{cases} 0 \text{ if } i \neq h \text{ or } j = k, \\ w_j p_{ik} \text{ if } i = h \text{ and } k \prec_i j, \\ w_k p_{ij} \text{ if } i = h \text{ and } j \prec_i k. \end{cases}$$

This problem can be solved in polynomial time if, and only if, matrix D is positive semidefinite. This motivates the construction of a new formulation, which we call *QSP*:

$$\text{Min } \tfrac{1}{2} c^T a + \tfrac{1}{2} a^T (D + \text{diag}(c)) a$$

$$\begin{array}{rcll} \sum_{i=1}^{m} a_{ij} &=& 1 & \forall j \in J, \\ a &\geq& 0, & \end{array}$$

where $(D + \text{diag}(c))$ is positive semidefinite and $\text{diag}(c)$ is a diagonal matrix with the vector c.

Given a solution for *QSP*, job j is assigned to machine i with probability a_{ij} and in each machine i the execution order is given by the function $\prec_i$. In our implementation, this assignment is performed 100 times returning the best generated schedule. For the special case of identical parallel machines, the optimal solution of the above formulation is given by $a_{ij} = \frac{1}{m}$. In this case, we implemented a combinatorial algorithm for this especial case attributing each job to a machine with probability $\frac{1}{m}$. This combinatorial algorithm is denoted by SK-C. The time complexity of the algorithm is $O(n \log n + m)$ plus the time complexity to solve the semidefinite program *QSP*.

2.5 Algorithm SZSK2 for $R|r_j|\sum w_j C_j$

The algorithm for problem $R|r_j|\sum w_j C_j$ is also a probabilistic algorithm and is, presented by Schulz and Skutella [9]. The algorithm, denoted by SZSK2, is based in the solution of a linear formulation and is a generalization of the algorithm SZSK. As in the linear program of the algorithm

SZSK, this formulation have variables C_j that represents the finishing time of job j and variables y_{ijt}, that says that job j is processed in machine i at time t, for each time interval $(t, t+1]$. The maximum time that a machine can execute is denoted by T. The formulation is exponential, but it can be made polynomial with a small loss in the objective function using interval times that increase exponentially in their size. In this case, we have binary variables y_{jl} indicating the execution of job j in machine i at interval $I_l = ((1+\beta)^{l-1}, (1+\beta)^l]$. The size of an interval I_l is denoted by $|I_l|$. For simplicity we denote $(1+\beta)^l$ by β_l. The relaxed formulation, which is denoted by *LPSS*, is the following:

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

$$\sum_{i=1}^{m}\sum_{l=0}^{L}\frac{y_{ijl}|I_l|}{p_{ij}} = 1 \quad \forall j \in J,$$

$$\sum_{j \in J} y_{ijl} \le 1 \quad \forall i \in M, \quad \text{e} \quad l = 0, ..., L,$$

$$C_j = \sum_{i=1}^{m}\sum_{l=0}^{L}(\frac{y_{ijl}|I_l|}{p_{ij}}\beta_{l-1} + \frac{1}{2}y_{ijl}|I_l|)\forall j \in J,$$

$$y_{ijl} = 0 \quad \forall i \in M, \quad \forall j \in J, \quad \beta_l \le r_{ij} - 1,$$

$$y_{ijl} \ge 0 \quad \forall i \in M, \quad \forall j \in J, \quad \forall l = 0, ..., L.$$

The algorithm solves the linear program *LPSS* and assign each job j to a machine-interval pair (i, I_l) at random with probability $\frac{y_{ijl} \cdot |I_l|}{p_{ij}}$. The jobs assigned to a machine i are scheduled in non-decreasing order of intervals assignment. If there are more than one job assigned to the same pair (i, I_l), the algorithm schedule them in order of their value j. For a given $\epsilon > 0$ setting $\beta = \frac{\epsilon}{2}$ this algorithm has a probabilistic $(2+\epsilon)$-approximation factor. As in the algorithm SK, the probabilistic assignment step is executed 100 times and the best generated schedule is returned. The time complexity of this algorithm is $O(nm \log_{(1+\epsilon)} T + n \log n)$ plus the time complexity to solve the linear program *LPSS*. Since this algorithm is executed with different values of ϵ, we denote by SZSK2$_\epsilon$ the algorithm SZSK2 with the given value of ϵ, that is, the algorithm SZSK2$_{0.1}$ is the algorithm SZSK2 with value of $\epsilon = 0.1$.

2.6 Two Heuristic Algorithms

In this section we present a new algorithm denoted by HE1 for the problem $R|r_j|\sum w_jC_j$, that is a simple modification of the algorithm SZSK2 and an extended heuristic of algorithm KK for the problem $P|r_j|\sum w_jC_j$, which we denote by HE2. In [4], Hariri and Potts present a simple heuristic algorithm for $1|r_j|\sum w_jC_j$ used to find an upper bound for a branch and bound algorithm. The algorithm is as follows:

1. Let S be the set of all (unsequenced) jobs, let $H = 0$ and $k = 0$ and find $T = min_{j\in S}\{r_j\}$.

2. Find the set $S' = \{j|j \in S, r_j \leq T\}$ and find a job i with $i \in S'$ and $\frac{w_i}{p_i} = max_{j\in S'}\{\frac{w_j}{p_j}\}$.

3. Set $k = k + 1$, sequence job i in position k, set $T = T + p_i$, set $H = H + w_iT$ and set $S = S - \{i\}$.

4. If $S = \emptyset$, then stop with the sequence generated having H as its cost. Otherwise set $T = max\{T, \min_{j\in S}\{r_j\}\}$ and go to step 2.

In algorithm SZSK2, the jobs are assigned to pairs machine-interval and them executed in each machine by the order of intervals assignments. In algorithm HE1, the assignment step is done as in algorithm SZSK2, but the jobs assigned to a machine i are scheduled using the algorithm of Hariri and Potts.

The algorithm HE2 executes an extended heuristic of algorithm KK: every time a machine becomes free, execute among the available jobs, the one with smallest ratio $\frac{p_j}{w_j}$. Notice that without the presence of release dates, this algorithm is essentially algorithm KK.

3 Study of Two Lower Bounds

In this section we present an experimental study of two formulations that provides lower bounds for our algorithms. The first formulation is the semidefinite formulation *QSP* presented for algorithm SK, and the second is a linear formulation *LPSS* presented for algorithm SZSK2. For problems that consider jobs with release dates we used the lower bounds provided by the linear program *LPSS*. For problems without release dates we performed a computational study to determine which formulation give lower bounds with better quality. For the most generic problem $R||\sum w_jC_j$, we consider three cases: $R2||\sum w_jC_j$, $R5||\sum w_jC_j$ and $R7||\sum w_jC_j$. We also tried to study the case $R10||\sum w_jC_j$ but we could not solve integer instances of this problem in a reasonable amount of time. We performed five tests with 100 jobs for each case. The processing times of jobs were taken uniformly from the interval $[1, 100]$ and w_j was chosen uniformly from the interval $[1, 10]$. The linear program *LPSS* is generated with time intervals defined with parameter $\epsilon = 0.3$ and $\epsilon = 0.1$. Besides the quality of the lower bound increases using $\epsilon = 0.1$ when compared with the solutions with $\epsilon = 0.3$, the solutions provided by the algorithms do not differ so much as we will see in the next sections. The lower bounds of these two formulations are compared with the value of an integer solution, which we obtained from the integral solutions of program *QSP*. The results of these tests can be seen in table 1.

Problem	Integer Optimal	*QSP*		*LPSS* $\epsilon = 0.3$		*LPSS* $\epsilon = 0.1$	
		Value	Ratio	Value	Ratio	Value	Ratio
$R2\|\|\sum w_j C_j$	163066	163052.92	0.999	152588.46	0.935	159313.82	0.976
	228766	228732.05	0.999	214004.87	0.935	223453.15	0.976
	223714	223673.35	0.999	209223.01	0.935	218505.52	0.976
	174802	174764.49	0.999	163503.50	0.935	170744.85	0.976
	180367	180337.23	0.999	168738.85	0.935	176189.91	0.976
		Value	Ratio	Value	Ratio	Value	Ratio
$R5\|\|\sum w_j C_j$	36767	36690.74	0.997	34506.58	0.938	35914.83	0.978
	33675	33636.31	0.998	31603.44	0.938	32925.71	0.977
	44130	44043.36	0.998	41379.21	0.937	43097.67	0.976
	36168	36104.74	0.998	33948.00	0.938	35340.76	0.977
	37343	37251.78	0.997	35028.43	0.938	36457.91	0.976
		Value	Ratio	Value	Ratio	Value	Ratio
$R7\|\|\sum w_j C_j$	26542	26473.41	0.997	24920.74	0.938	25924.81	0.976
	22429	22361.25	0.996	21074.51	0.939	21915.70	0.977
	26919	26857.20	0.997	25279.07	0.939	26302.32	0.977
	29093	29017.66	0.997	27314.76	0.938	28415.94	0.976
	25543	25440.19	0.995	23967.79	0.938	24928.01	0.975

Table 1: Comparison between formulations *QSP* and *LPSS*.

We also performed computational tests to compare the lower bounds for problem $P||\sum w_j C_j$. In this case we could solve only instances up to 20 jobs with 2 machines and 15 jobs with 5 machines. The next theorem, proved by Skutella [10], helps us to understand the hardness to obtain integer solutions for instances of this problem.

Theorem 3.1 *For instances of $Pm||\sum w_j C_j$, an optimum vector solution a of the quadratic program QSP is $a_{ij} = \frac{1}{m}$ for all i, j. This optimum solution is unique if all ratios $\frac{p_j}{w_j}$, are different and positive.*

In all instances the solution of the quadratic program is exactly the one provided in the theorem. Since the Xpress solver finds the optimal integer solution using a branch and bound tree, the number of nodes is exponential. We could not solve these kind of problems even if we use an upper bound provided by our approximation algorithms. We could solve only instances with 20 jobs for problem $P2||\sum w_j C_j$ and instances with 15 jobs for problem $P5||\sum w_j C_j$. The results of our tests are presented in table 2.

Problem	Integer Optimal	*QSP*		*LPSS* $\epsilon = 0.3$		*LPSS* $\epsilon = 0.1$	
		Value	Ratio	Value	Ratio	Value	Ratio
$P2\|\|\sum w_j C_j$	19615	19546.75	0.996	18413.86	0.938	19142.58	0.975
	19214	19164.00	0.997	18082.13	0.941	18773.20	0.977
	17199	17135.75	0.996	16158.01	0.939	16785.03	0.975
	16398	16322.50	0.995	15385.34	0.938	15992.72	0.975
	16415	16365.00	0.996	15451.82	0.941	16033.15	0.976
		Value	Ratio	Value	Ratio	Value	Ratio
$P5\|\|\sum w_j C_j$	5121	4913.60	0.959	4712.12	0.920	4842.30	0.945
	5467	5257.60	0.961	5035.09	0.920	5177.84	0.947
	6536	6312.40	0.965	6039.39	0.924	6215.11	0.950
	4875	4651.60	0.954	4468.68	0.916	4590.74	0.941
	5105	4895.80	0.959	4694.55	0.919	4824.64	0.945

Table 2: Comparison between formulations *QSP* and *LPSS*.

In all generated tests, the lower bounds provided by the formulation *QSP* are better than the lower bounds provided by formulation *LPSS*. Also note that when $\epsilon = 0.1$ the difference is not so large. We do not use smaller values of ϵ since the increase in the computational time to solve such formulations is very high.

4 Practical Analysis of the Implemented Algorithms

In this section, we present the results of our tests. Since some problems are particular cases of others, we performed several different tests. Each subsection is reserved for one case. Before presenting the computational results for each problem, we describe the proceeding to generate each test. For each test we generate 100 jobs with processing requirement uniformly chosen from the interval $[1, 100]$ and w_j chosen from the interval $[1, 10]$. When the problem require release dates, the data is generated using the same approach used by Hariri and Potts [4]. The release dates are uniformly chosen from the interval $[0, E[p]n\gamma]$. This simulates the arrival of n jobs from a stable queue according to a Poisson process with parameter γ [5]. The time in all tables is given in seconds. The ratio in the table corresponds to $\frac{V}{LB}$, where V is the value found by the algorithm and LB is a lower bound for the optimal solution. We perform tests with 2, 5, 7 and 10 machines. As was done in [5], we generated five different instances for each test problem, so the results in each line of the tables corresponds to the mean of five tests. The algorithms were tested on a AMD Athlon 1.2Ghz with 800 MB of RAM under Linux 2.4.2-2 kernel.

4.1 Tests for the problem $P||\sum w_jC_j$

In this problem we used the algorithms KK, SzSk, Sk-C, SzSk2 and HE1. We do not use the algorithm HE2 here because without the presence of release dates this algorithm generates the same solutions of algorithm KK. The table 3 presents the results of this tests. The LB column corresponds to the fractional optimal solution of the quadratic formulation QSP of the algorithm Sk.

Problem	LB	Algorithm	Value	Time	Ratio
$P2\|\|\sum w_jC_j$	382923.95	KK	383017.6	0.01	1.0002
		SzSk	383258.8	0.11	1.0008
		Sk-C	383319.6	0.05	1.0010
		SzSk2$_{0.1}$	383463.6	6.15	1.0010
		HE1$_{0.1}$	383265.0	6.18	1.0008
$P5\|\|\sum w_jC_j$	145821.3	KK	146088.2	0.01	1.0018
		SzSk	148134.2	0.17	1.0158
		Sk-C	147933.6	0.07	1.0144
		SzSk2$_{0.1}$	147929.6	16.85	1.0144
		HE1$_{0.1}$	147860.6	16.95	1.0139
$P7\|\|\sum w_jC_j$	115054.88	KK	115431.0	0.01	1.0032
		SzSk	117479.4	0.11	1.0210
		Sk-C	117882.6	0.07	1.0245
		SzSk2$_{0.1}$	117906.6	28.45	1.0247
		HE1$_{0.1}$	118298.4	28.14	1.0281
$P10\|\|\sum w_jC_j$	81997.11	KK	82516.2	0.01	1.0063
		SzSk	85775.2	0.11	1.0460
		Sk-C	85646.6	0.06	1.0445
		SzSk2$_{0.1}$	86259.0	43.45	1.0519
		HE1$_{0.1}$	86200.0	43.57	1.0512

Table 3: Comparison between KK, SzSk, SzSk2, HE1 and Sk-C.

The algorithms obtained very good results for all tested instances. The algorithm KK is the most simple and obtained the best results generating solutions with values less than 0.7% of the lower bounds, besides the other algorithms use more advanced ideas. As we can see, the ratio grows when we use more machines. For algorithm KK the increase is very small. For the other ones the growth is more representative. We believe that with more jobs per machine the ratios obtained tends to decrease. This can be seen in graphics $(1, 2, 3$ and $4)$. We will describe more about this behavior in the next subsection.

4.2 Tests for the problem $P|r_j|\sum C_j$

To solve the problem $P|r_j|\sum C_j$ we used the algorithms PSW, SZSK, SZSK2, HE1 and HE2. Although the algorithm SZSK is the combinatorial version of the algorithm SZSK2 for identical machines, we also included the algorithm SZSK2 in the comparisons. The algorithms SZSK2 and HE1 were executed with parameter $\epsilon = 0.3$ and $\epsilon = 0.1$. We perform different tests using different values for parameter γ to generate the release dates. We used parameter $\gamma = 0.2$, $\gamma = 0.4$ and $\gamma = 0.6$. The LB column has the fractional optimal solution of the linear program of algorithm SZSK2 with $\epsilon = 0.1$. It is interesting to note that this lower bound may be far away from the integer optimal, since the value of an optimal integer solution for the program $LPSS$ is already a lower bound for the original problem $P|r_j|\sum C_j$. The tables $4, 5$ and 6 present the results obtained for these tests.

Problem with $\gamma = 0.2$	LB	Algorithm	Value	Time	Ratio
$P2\|r_j\|\sum C_j$	100161.56	PSW	109136.8	0.01	1.08
		SZSK	124153.6	0.01	1.23
		SZSK2$_{0.3}$	113298.6	5.08	1.13
		SZSK2$_{0.1}$	108391.2	78.45	1.08
		HE1$_{0.3}$	105280.4	5.79	1.05
		HE1$_{0.1}$	105276.4	79.72	1.05
		HE2	104981.4	0.01	1.04
$P5\|r_j\|\sum C_j$	52569.48	PSW	60768.4	0.01	1.15
		SZSK	75553.8	0.25	1.43
		SZSK2$_{0.3}$	63130.6	58.19	1.20
		SZSK2$_{0.1}$	62362.6	425.89	1.18
		HE1$_{0.3}$	60418.6	52.95	1.14
		HE1$_{0.1}$	60651.0	431.40	1.15
		HE2	57960.8	0.01	1.10
$P7\|r_j\|\sum C_j$	51345.58	PSW	57953.6	0.01	1.12
		SZSK	68479.4	0.01	1.33
		SZSK2$_{0.3}$	59530.8	90.13	1.15
		SZSK2$_{0.1}$	59246.2	807.20	1.15
		HE1$_{0.3}$	58486.2	90.17	1.13
		HE1$_{0.1}$	58983.2	819.64	1.14
		HE2	57204.4	0.01	1.11
$P10\|r_j\|\sum C_j$	47731.29	PSW	53526.4	0.01	1.12
		SZSK	62120.2	0.24	1.30
		SZSK2$_{0.3}$	55645.2	171.29	1.16
		SZSK2$_{0.1}$	54684.6	1584.29	1.14
		HE1$_{0.3}$	54845.2	183.71	1.14
		HE1$_{0.1}$	54618.4	1611.62	1.14
		HE2	53494.6	0.01	1.12

Table 4: Comparison between PSW, SZSK, SZSK2, HE2 and HE1.

The algorithm HE2 generate the best schedules in all tests. Notice that the algorithm HE1 obtain better results when we have few machines and small values of γ. The algorithms PSW and HE1 are the second best in all cases. For all tests, the algorithm PSW generate solutions that are at most 12% of the lower bound although its approximation factor is 6. Other interesting point is that the ratio for the algorithm SZSK get better results when we have more machines with bigger values of γ. The algorithm SZSK2 obtained better results than the algorithm SZSK for all cases, except when we have big values of γ and more machines as we can see in table 6. Analyzing the fractional solution of the linear program used by the algorithm SZSK2, we can see that the solver generates an optimal fractional solution using less machines in such a way that variables of some machines are

Problem with $\gamma = 0.4$	LB	Algorithm	Value	Time	Ratio
$P2\|r_j\|\sum C_j$	113585.05	PSW	126569.0	0.01	1.11
		SzSk	162040.8	1.73	1.42
		SzSk2$_{0.3}$	133894.2	6.52	1.17
		SzSk2$_{0.1}$	126938.2	86.24	1.11
		HE1$_{0.3}$	121682.4	6.11	1.07
		HE1$_{0.1}$	121691.0	95.55	1.07
		HE2	121034.2	0.01	1.06
$P5\|r_j\|\sum C_j$	94406.70	PSW	104561.0	0.01	1.10
		SzSk	124759.4	0.26	1.32
		SzSk2$_{0.3}$	107531.0	62.90	1.13
		SzSk2$_{0.1}$	104917.8	495.95	1.11
		HE1$_{0.3}$	105297.2	59.61	1.11
		HE1$_{0.1}$	104759.8	503.41	1.10
		HE2	103638.0	0.01	1.09
$P7\|r_j\|\sum C_j$	98514.48	PSW	108252.4	0.01	1.09
		SzSk	119628.8	2.15	1.21
		SzSk2$_{0.3}$	111726.6	112.95	1.13
		SzSk2$_{0.1}$	109100.8	952.18	1.10
		HE1$_{0.3}$	109449.6	107.21	1.11
		HE1$_{0.1}$	108890.2	967.13	1.10
		HE2	108235.0	0.01	1.09
$P10\|r_j\|\sum C_j$	94268.95	PSW	103936.8	0.01	1.10
		SzSk	107757.0	0.17	1.14
		SzSk2$_{0.3}$	107522.0	218.68	1.14
		SzSk2$_{0.1}$	104450.2	1912.04	1.10
		HE1$_{0.3}$	105247.4	221.41	1.11
		HE1$_{0.1}$	104419.4	1917.11	1.10
		HE2	103936.8	0.01	1.10

Table 5: Comparison between PSW, SzSk, SzSk2, HE2 and HE1.

Problem with $\gamma = 0.6$	LB	Algorithm	Value	Time	Ratio
$P2\|r_j\|\sum C_j$	151285.05	PSW	168188.8	0.01	1.11
		SzSk	216423.0	1.55	1.43
		SzSk2$_{0.3}$	175249.6	7.24	1.15
		SzSk2$_{0.1}$	168341.6	100.95	1.11
		HE1$_{0.3}$	165981.0	6.60	1.09
		HE1$_{0.1}$	165819.6	86.39	1.09
		HE2	164796.0	0.01	1.08
$P5\|r_j\|\sum C_j$	141989.53	PSW	155055.8	0.01	1.09
		SzSk	176075.4	0.10	1.24
		SzSk2$_{0.3}$	162046.8	62.90	1.14
		SzSk2$_{0.1}$	156466.4	497.253	1.10
		HE1$_{0.3}$	156834.4	67.48	1.10
		HE1$_{0.1}$	155801.2	490.57	1.09
		HE2	154996.4	0.01	1.09
$P7\|r_j\|\sum C_j$	144311.06	PSW	157384.2	0.01	1.09
		SzSk	165377.2	2.21	1.14
		SzSk2$_{0.3}$	162347.6	120.04	1.12
		SzSk2$_{0.1}$	158264.0	1023.78	1.09
		HE1$_{0.3}$	158676.2	116.25	1.09
		HE1$_{0.1}$	157997.0	1177.65	1.09
		HE2	157384.2	0.01	1.09
$P10\|r_j\|\sum C_j$	139258.28	PSW	152096.4	0.01	1.09
		SzSk	153544.8	0.07	1.10
		SzSk2$_{0.3}$	158748.8	246.22	1.13
		SzSk2$_{0.1}$	152670.2	2051.54	1.09
		HE1$_{0.3}$	153621.8	243.68	1.10
		HE1$_{0.1}$	152384.2	1955.04	1.09
		HE2	152096.4	0.01	1.09

Table 6: Comparison between PSW, SzSk, SzSk2, HE2 and HE1.

little used. Consequently, the generated schedule have some machines that are almost unused. The algorithm SzSk is the combinatorial version of SzSk2 but the jobs are attributed to all machines uniformly. This also explains why the algorithm HE1 when compared to the algorithm PSW, get better results using two machines than 7 and 10 machines. Based on this observation we try to solve the linear program *LPSS* under the algorithm HE1 with an increase in the number of jobs per machine. Note that the algorithm HE1 is based in the algorithm SZSK2 and we can expect the same behavior in both algorithms. We perform several tests that can be seen in tables $7, 8, 9$ and 10. The interesting point to note is that when we get a ratio of approximately 60 jobs per machine the algorithm HE1 produces better schedules. The solution of the linear program has better attribution when this happens. We also present some graphics (1, 2, 3 and 4) that summarizes these results. As we mentioned in the previous subsection, the algorithms get better results when we use more jobs per machine. This can be easily verified in these graphics. Notice that we could not solve all instances of the problem with a given $\epsilon = 0.3$. When we solved the problem with ten machines for example, we used $\epsilon = 0.8$ and the time to solve the linear program *LPSS* is still high. With such values, the lower bound provided by the linear program becomes worse and the ratios obtained in these tests are bigger than the ones of the previous tests. We believe that such ratios could be much better.

$P2\|r_j\|\sum C_j$ with $\gamma = 0.6$	LB	Algorithm	Value	Time	Ratio
		PSW	28791.2	0.01	1.24
$\|J\| = 20$	23153.11	$\text{HE1}_{0.3}$	28870.0	0.310	1.24
		PSW	64322.2	0.01	1.25
$\|J\| = 40$	51281.51	$\text{HE1}_{0.3}$	64647.0	1.11	1.26
		PSW	96616.0	0.01	1.25
$\|J\| = 60$	76944.32	$\text{HE1}_{0.3}$	97042.4	2.35	1.26
		PSW	122591.8	0.01	1.24
$\|J\| = 80$	98227.78	$\text{HE1}_{0.3}$	122696.8	3.30	1.24
		PSW	165469.8	0.01	1.23
$\|J\| = 100$	133911.11	$\text{HE1}_{0.3}$	164117.6	4.20	1.22
		PSW	204703.2	0.01	1.21
$\|J\| = 120$	167971.42	$\text{HE1}_{0.3}$	200258.0	10.26	1.19
		PSW	443288.6	0.01	1.16
$\|J\| = 200$	381645.09	$\text{HE1}_{0.3}$	429205.6	45.11	1.12

Table 7: Comparison between PSW, and HE1.

$P5\|r_j\|\sum C_j$ with $\gamma = 0.6$	LB	Algorithm	Value	Time	Ratio
		PSW	75690.0	0.01	1.24
$\|J\| = 50$	60878.68	$\text{HE1}_{0.3}$	75998.6	9.920	1.24
		PSW	156073.2	0.01	1.24
$\|J\| = 100$	125460.43	$\text{HE1}_{0.3}$	157288.4	62.87	1.25
		PSW	248198.8	0.01	1.25
$\|J\| = 150$	198061.29	$\text{HE1}_{0.3}$	250604.0	138.54	1.26
		PSW	319717.4	0.01	1.24
$\|J\| = 200$	256076.57	$\text{HE1}_{0.3}$	322656.8	246.97	1.26
		PSW	507942.0	0.01	1.23
$\|J\| = 300$	411426.11	$\text{HE1}_{0.3}$	498390.2	579.96	1.21
		PSW	763550.2	0.01	1.20
$\|J\| = 400$	635731.46	$\text{HE1}_{0.3}$	735126.4	1066.96	1.15
		PSW	1100436.4	0.01	1.17
$\|J\| = 500$	936920.79	$\text{HE1}_{0.3}$	1064286.6	1648.83	1.13

Table 8: Comparison between PSW, and HE1.

$P7\|r_j\|\sum C_j$ with $\gamma = 0.6$	LB	Algorithm	Value	Time	Ratio
$\|J\| = 100$	114575.89	PSW	168822.6	0.01	1.47
		$\mathrm{HE1}_{0.6}$	170932.8	31.11	1.49
$\|J\| = 200$	213171.61	PSW	317334.2	0.01	1.48
		$\mathrm{HE1}_{0.6}$	321711.4	137.540	1.50
$\|J\| = 300$	308844.77	PSW	458481.0	0.01	1.48
		$\mathrm{HE1}_{0.6}$	465342.2	308.92	1.50
$\|J\| = 400$	468532.64	PSW	643419.0	0.01	1.37
		$\mathrm{HE1}_{0.6}$	644531.8	559.20	1.37
$\|J\| = 500$	671739.66	PSW	904658.2	0.01	1.34
		$\mathrm{HE1}_{0.6}$	877560.8	912.38	1.30
$\|J\| = 600$	921405.31	PSW	1201110.0	0.01	1.30
		$\mathrm{HE1}_{0.6}$	1168692.4	1306.86	1.26
$\|J\| = 800$	1575752.90	PSW	1973292.2	0.01	1.25
		$\mathrm{HE1}_{0.6}$	1913924.2	2357.64	1.21

Table 9: Comparison between PSW, and HE1.

$P10\|r_j\|\sum C_j$ with $\gamma = 0.6$	LB	Algorithm	Value	Time	Ratio
$\|J\| = 100$	96960.46	PSW	160832.6	0.01	1.65
		$\mathrm{HE1}_{0.8}$	162182.8	38.420	1.67
$\|J\| = 200$	187381.96	PSW	312235.6	0.01	1.66
		$\mathrm{HE1}_{0.8}$	315302.8	174.60	1.68
$\|J\| = 400$	384035.05	PSW	635543.6	0.01	1.65
		$\mathrm{HE1}_{0.8}$	644026.2	720.43	1.67
$\|J\| = 600$	677843.38	PSW	995554.0	0.01	1.46
		$\mathrm{HE1}_{0.8}$	996502.2	1614.99	1.47
$\|J\| = 800$	1113753.2	PSW	1535404.0	0.01	1.37
		$\mathrm{HE1}_{0.8}$	1494949.8	2901.76	1.34
$\|J\| = 1000$	1666006.6	PSW	2214162.8	0.01	1.32
		$\mathrm{HE1}_{0.8}$	2168788.2	4648.66	1.30
$\|J\| = 1200$	2269898.8	PSW	2994767.4	0.01	1.31
		$\mathrm{HE1}_{0.8}$	2920309.8	6843.41	1.28

Table 10: Comparison between PSW, and HE1.

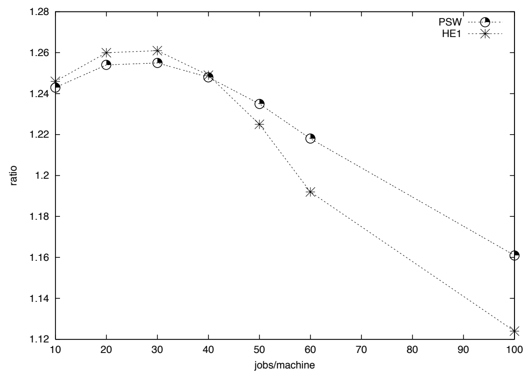

Figure 1: Graphics for 2 machines

4.3 Tests for the problem $R||\sum w_j C_j$

In this problem we use the algorithms SK, SZSK2 and HE1. For the tests in table 11 we choose p_{ij} uniformly from the interval $[1, \ldots, 100]$. In the tests presented in table 12 the processing times are chosen from different intervals to give the idea that we have machines with different speeds. Us-

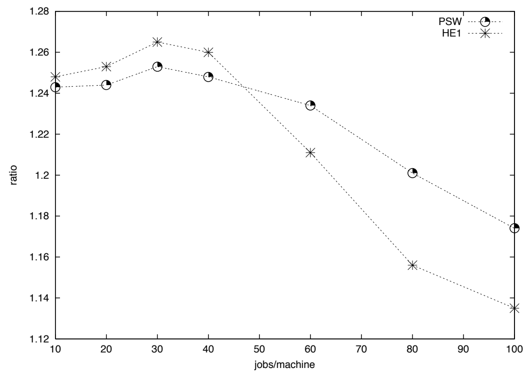

Figure 2: Graphics for 5 machines

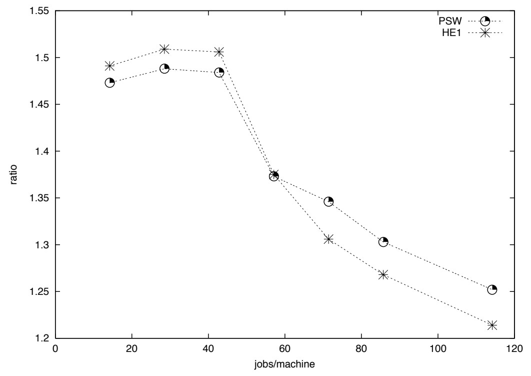

Figure 3: Graphics for 7 machines

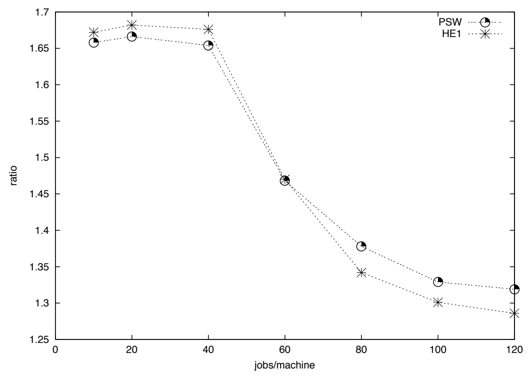

Figure 4: Graphics for 10 machines

ing two machines the processing times are chosen from the interval $[1, \ldots, 50]$ for the first machine and from $[50, \ldots, 100]$ for the second machine. Using five machines the processing times are chosen from intervals, $[1, \ldots, 20]$, $[20, \ldots, 40]$, $\ldots$, $[80, \ldots, 100]$. Using seven machines the processing times are chosen from intervals, $[1, \ldots, 15]$,$[15, \ldots, 30]$,$\ldots$,$[90, \ldots, 100]$. In the tests with ten machines, the processing times are chosen from intervals $[1,\ldots,10][10,\ldots,20][20,\ldots,30],\ldots,$ $[90,\ldots,100]$. We use $\epsilon = 0.1$ and $\epsilon = 0.3$ in the algorithms SzSk2 and HE1. The LB column corresponds to the fractional solution found by the quadratic formulation *QSP* of the algorithm Sk.

Problem	LB	Algorithm	Value	Time	Ratio
$R2\|\|\sum w_jC_j$	194112.01	Sk	194216.4	1.13	1.0005
		SzSk2$_{0.3}$	194149.8	1.40	1.0001
		SzSk2$_{0.1}$	194156.4	6.14	1.0002
		HE1$_{0.3}$	194143.8	1.21	1.0001
		HE1$_{0.1}$	194143.8	6.78	1.0001
$R5\|\|\sum w_jC_j$	37616.6	Sk	37763.0	38.31	1.0038
		SzSk2$_{0.3}$	37644.0	4.08	1.0007
		SzSk2$_{0.1}$	37635.8	21.31	1.0005
		HE1$_{0.3}$	37627.4	3.81	1.0002
		HE1$_{0.1}$	37625.8	21.99	1.0002
$R7\|\|\sum w_jC_j$	26049.94	Sk	26305.0	89.36	1.0097
		SzSk2$_{0.3}$	26154.2	5.15	1.0040
		SzSk2$_{0.1}$	26149.8	25.43	1.0038
		HE1$_{0.3}$	26140.2	5.06	1.0034
		HE1$_{0.1}$	26145.6	26.21	1.0036
$R10\|\|\sum w_jC_j$	11337.05	Sk	11666.2	200.79	1.0290
		SzSk2$_{0.3}$	11474.6	8.15	1.0121
		SzSk2$_{0.1}$	11463.0	40.96	1.0111
		HE1$_{0.3}$	11450.2	8.79	1.0099
		HE1$_{0.1}$	11465.2	39.56	1.0113

Table 11: Comparison between Sk, SzSk2 and HE1.

Problem *	LB	Algorithm	Value	Time	Ratio
$R2\|\|\sum w_jC_j$	246745.49	Sk	246873.6	0.97	1.0005
		SzSk2$_{0.3}$	246811.2	1.12	1.0002
		SzSk2$_{0.1}$	246818.6	6.24	1.0002
		HE1$_{0.3}$	246783.8	1.13	1.0001
		HE1$_{0.1}$	246783.8	6.78	1.0001
$R5\|\|\sum w_jC_j$	73513.03	Sk	73934.6	37.41	1.0057
		SzSk2$_{0.3}$	73670.0	3.80	1.0021
		SzSk2$_{0.1}$	73673.0	16.78	1.0022
		HE1$_{0.3}$	73659.6	3.71	1.0019
		HE1$_{0.1}$	73667.6	17.21	1.0021
$R7\|\|\sum w_jC_j$	51544.92	Sk	52211.6	98.06	1.0129
		SzSk2$_{0.3}$	51828.4	5.90	1.0054
		SzSk2$_{0.1}$	51808.0	26.31	1.0051
		HE1$_{0.3}$	51849.2	5.14	1.0059
		HE1$_{0.1}$	51834.2	26.71	1.0056
$R10\|\|\sum w_jC_j$	28453.73	Sk	30516.2	207.62	1.0724
		SzSk2$_{0.3}$	29472.2	8.07	1.0357
		SzSk2$_{0.1}$	29451.0	40.95	1.0350
		HE1$_{0.3}$	29415.8	8.16	1.0338
		HE1$_{0.1}$	29449.4	41.08	1.0349

Table 12: Comparison between Sk, SzSk2 and HE1.

As we can see all algorithms produces schedules very close to the optimal. For all tests, the algorithms produced solutions with values that are at most 3% of the lower bound except for the algorithm Sk that generated a solution with value 7% of the lower bound. In general the algorithm HE1 generate better schedules. Other point is that although the semidefinite program *QSP* generate fractional solutions that are closer to the optimal, the algorithm Sk generate the worst schedules even if compared with the algorithm SzSk2$_{0.3}$.

4.4 Comparison for problem $R|r_j|\sum w_jC_j$

In this case we use the algorithms SZSK2 and HE1 to solve problem $R|r_j|\sum w_jC_j$. Table 13 show the results of the tests. The processing times were chosen uniformly from the interval $[1,\ldots,100]$ and we use $\gamma = 0.2$ to generate release dates. The LB is the optimal fractional solution of the linear program $LPSS$ with $\epsilon = 0.1$. We emphasize that this lower bound may be far away from the integer optimal solution, since an integer optimal solution to $LPSS$ is already a relaxation of the problem $R|r_j|\sum w_jC_j$. The algorithm HE1 get better results in all tests.

Problem	LB	Algorithm	Value	Time	Ratio
$R2\|r_j\|\sum w_jC_j$	306490.12	SZSK2$_{0.3}$	352346.2	5.9	1.15
		SZSK2$_{0.1}$	339533.6	80.6	1.11
		HE1$_{0.3}$	332137.8	5.6	1.08
		HE1$_{0.1}$	331878.4	74.4	1.08
$R5\|r_j\|\sum w_jC_j$	230274.71	SZSK2$_{0.3}$	257145.8	51.8	1.12
		SZSK2$_{0.1}$	252826.8	420.8	1.10
		HE1$_{0.3}$	251919.2	53.7	1.09
		HE1$_{0.1}$	251915.8	364.7	1.09
$R7\|r_j\|\sum w_jC_j$	229843.17	SZSK2$_{0.3}$	254033.2	109.32	1.1
		SZSK2$_{0.1}$	250782.6	840.1	1.09
		HE1$_{0.3}$	250165.8	94.7	1.09
		HE1$_{0.1}$	250146.2	777.6	1.09
$R10\|r_j\|\sum w_jC_j$	233510.85	SZSK2$_{0.3}$	256892.4	186.3	1.10
		SZSK2$_{0.1}$	253754.4	1603.6	1.09
		HE1$_{0.3}$	253180.0	185.5	1.08
		HE1$_{0.1}$	253132.8	1491.8	1.08

Table 13: Comparison between SZSK2 and HE1.

5 Conclusion

We present computational results for some approximation algorithms for scheduling on parallel machines. As expected, the practical solutions yields ratios better than the approximation factor of the presented algorithms. We also note that algorithms with more refined techniques does not lead to better results. In fact, for problems $P||\sum w_jC_j$ and $P|r_j|\sum C_j$ algorithms PSW and KK obtained the best results even when compared to algorithms with advanced ideas. We also note that the solutions provided by the algorithm SK is worst than the solutions provided by the algorithm SZSK2 despite the semidefinite program generate fractional solutions with better quality. Finally we present two heuristics that get better results in almost all cases studied.

6 Bibliography

1. F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, M. Sviridenko and C. Stein. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS'99)* pp. 32-44, 1999

2. Dash Optimization. Xpress-MP Release 13. *Xpress-MP Manual*, 2002.

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

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

5. C. Hepner and C. Stein. Implementation of a PTAS for Scheduling with Release Dates. In *3rd Workshop on Algorithm Engineering and Experiments (ALENEX 2001). Lecture Notes in Computer Sciense* 2513 pp. 202-215, 2001.

6. T. Kawaguchi and S. Kyan. Worst Case Bound of an LRF Schedule for the Mean Weighted Flow-Time Problem. *SIAM J. Computing* 4, 1986.

7. M. W. P. Savelsbergh, R. N. Uma, and J. M. Wein. An experimental study of LP-based approximation algorithms for scheduling problems. *Proceedings of the 9th Annual ACM–SIAM Symposium on Discrete Algorithms* pp. 453–462, 1998.

8. C. Phillips and C. Stein and J. Wein. Minimizing Average Completion time in the Presence of Release Dates. *Mathematical Programming B* 82, 1998.

9. A. S. Schulz and M. Skutella. Scheduling Unrelated Machines by Randomized Rounding. *SIAM Journal on Discrete Mathematics* Volume 15, Number 4, 2002.

10. M. Skutella. Semidefinite Relaxations for Parallel Machine Scheduling. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS'98)* pp. 472-481, 1998.

11. W. E. Smith. Various optimizers for single-stage production. *Naval Res. Logist. Quart.*, 1956, 3 pp. 58-66.