Alcom-FT Technical Report Series
ALCOMFT-TR-03-58

Scheduling problems on two sets of identical machines *

Csanád Imreh †

Abstract

In this paper we investigate the following scheduling problem: We have two sets of identical machines, the jobs have two processing times one for each set of machines. We consider two different objective functions, in the first model the goal is to minimize the maximum of the makespans on the sets, in the second model we minimize the sum of the makespans. We consider the online, semi online and offline versions of these problems.

AMS subject classification: 90B35, 68W25
Keywords: Scheduling, online algorithms, approximation scheme.

1 Introduction

In this paper we examine the scheduling problem on the following machine enviroment. In the problem there are two sets $\mathcal{P}$ and $\mathcal{S}$ containing k and m identical machines. Each job j has two different processing times p_j and s_j, depending on which set of machines to schedule it. This machine enviroment models the situation where we have two types of machines. For the jobs we have to choose the set where we want to schedule it, and then we have to schedule it on one of the machines in the set. Let C_P and C_S denote the

*This research has been partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT) and by the Hungarian National Foundation for Scientific Research, Grant TO30074. During some part of this work the author had a postdoctoral scholarschip at MPI Informatics in Saarbrücken

†Institute of Informatics, University of Szeged, Àrpád tér 2, 6720 Szeged, Hungary, email:cimreh@inf.u-szeged.hu

makespans (maximal completion time) achieved at the sets of machines. We consider two different objective functions.

If our goal is to minimize the maximum completion time, then the objective function is the maximum of the makespans ($\max\{C_P, C_S\}$).We call this problem *Maximum Two Sets Scheduling* in short $MTSS(k,m)$. On the other hand if the sets are independent (the resources used by them are not common) and each set of machines has a cost proportional to the makespan, then we have to minimize the sum ($C_P + C_S$) of the makespans. Note that since the jobs may have different processing times belonging to the different sets of machines, hence considering any weighted sum of the makespans is reducible to this problem. (We can multiply the processing times with the weights.) We call this problem *Sum Two Sets Scheduling*, in short $STSS(k,m)$.

A closely related problem to $MTSS(k,m)$ is the unrelated machines scheduling problem. In the unrelated machines scheduling problem, there are n machines and the jobs have a load vector $(p_1(j),\ldots,p_n(j))$, $p_i(j)$ is the processing time if the job is scheduled on the i-th machine. The $MTSS(1,1)$ problem is the unrelated two machines problem, the general case is a restricted version of the unrelated $k+m$ machines scheduling problem. The online unrelated machines scheduling problem is investigated in [1], there a $O(\log n)$-competitive algorithm is presented, which is optimal up to a constant factor. The offline problem is stongly NP-hard. If the number of machines is constant an FPTAS for the solution of the problem is presented in [9]. The special case, where $p_i(j)$ is either $w(j)$ or ∞ is called the restricted assignment problem. The offline problem is also strongly NP-hard. For the online restricted assignment problem an $O(\log n)$-competitive algorithm is given in [3], and it is proven that this algorithm is optimal.

Another closely related problem is the problem of two-layered scheduling (see [10]). This scheduling model is a semi-online version of $MTSS(k,m)$, where the decision maker has to decide in an online fashion where to schedule the jobs, but it can produce an offline preemptive schedule at the end. In [10] a $1+\sqrt{2}$-competitive algorithm is presented for arbitrary k and m, furthermore a general lower bound of 1.781 is proven for the possible competitive ratios.

The $STSS(k,m)$ problem is a generalization of the scheduling with rejection model, the special case $k=1$ reduces to this scheduling problem. Scheduling with rejection was introduced in [4]. In this model, it is possible to reject the jobs. The jobs are characterized by a processing time and a penalty. The goal is to minimize the makespan of the schedule for the accepted jobs plus the sum of the penalties of the rejected jobs. In the offline case an FPTAS is presented for fixed number of machines, and a PTAS in the case where the number of machines is the part of the input. In the online case a 2.168-competitive algorithm is given for arbitrary number of machines, and an 1.618-competitive algorithm in the case of 2 machines. Matching lower bounds are also presented. The preemptive version of online scheduling with rejection is studied in [12]. In [12] a generalized version of the reject total penalty algorithm (see [4]) is analyzed, and it is proven that this generalized algorithm is 2.387-competitive for arbitrary number of machines. A general lower bound of 2.124, and a lower bound of 2.33 for the class of obliviously scheduling algorithms (the accepted jobs are scheduled without knowledge of the rejection penalties) are also presented. In [6] the offline scheduling problem with rejection is investigated in some more difficult identical machines models, in [7] an FPTAS is given for scheduling with rejection on related parallel machines.

We will consider both the offline version and the online version of $MTSS$ and $STSS$. Both problems are NP-hard, and we present FPTAS for the solution of $STSS$. Since the $MTSS$ problem is a special case of the unrelated machines scheduling problem, the FPTAS for the unrelated machines problem also works for $MTSS$. In the online case we characterize the competitive ratio of a load greedy algorithm which assigns every jobs to the set, where the job has smaller load. The competitive ratio of this algorithm tends to ∞ for both problems as the ratio m/k grows. We show that the post greedy algorithm has also unbounded competitive ratio as m/k grows. Finally we analyze a modified greedy algorithm which is defined in [10], and we show that it is constant competitive for arbitrary number of machines. A semi online case of $STSS$ where we have to decide in online way which set of machines is used for the jobs, but we do offline scheduling at the end is also considered. This version of $MTSS$ is investigated in [10]. In the semi online $STSS$ version the special case $k = 1$, is a semi online version of the scheduling with rejection model.

For the online and semi online problems we measure the performance of the algorithms by the competitive ratio. An online algorithm is called *c-competitive* if for each input the cost of the schedule produced by the algorithm is at most c times larger than the cost of the optimal schedule.

The paper is organized as follows. In the next section we introduce the basic notations. In Section 3 we examine some greedy algorithms. We present and analyze in Section 4 a better algorithm with constant competitive ratio. In Section 5 we consider the semi online problem with preemption. Later in Section 6 some lower bounds are presented for the possible competitive ratios. The offline FPTAS for $STSS$ is shown in Section 7. In Section 8 we summarize the results and list some further open problems related to the paper.

2 Notations and preliminaries

In the problem considered there are two sets of machines, $\mathcal{P}$ is the set of k and $\mathcal{S}$ is the set of m identical machines. In what follows we assume that $k \leq m$. Furthermore, each job j has two processing times, p_j if we schedule it on the machines of P and s_j if we schedule it on the machines of S. The vector (p_j, s_j) is called the size of the job.

For an arbitrary list J of jobs and an algorithm A, we denote by $A(J)$ the cost of the schedule produced by algorithm A on list J, the cost of the optimal schedule is $OPT(J)$. For the online and semi online problems we measure the performance of the algorithms by the competitive ratio. An online algorithm A is called C-competitive if $A(J) \leq C \cdot OPT(J)$ is valid for arbitrary list J of jobs. The smallest such C for which the algorithm is C-competitive is called the competitive ratio of the algorithm.

For any subset I of the jobs, we define the following notations

$$\mathrm{S}_I = \sum_{j \in I} s_j, \quad \mathrm{P}_I = \sum_{j \in I} p_j, \quad \mathrm{MS}_I = \max_{j \in I} s_j, \quad \mathrm{MP}_I = \max_{j \in I} p_j.$$

As a subroutine we will use an online scheduling algorithm. Several algorithms are developed for the online scheduling problem on n identical machines (see the survey [13]), we will use the classical, greedy online scheduling algorithm LIST ([8]). This algorithm always schedules greedily the arriving job on the least loaded machine. The following statement is valid for the makespan achieved by LIST (here we denote the processing times by p_j).

Proposition 1 *[8]*

For the k identical machines scheduling problem on arbitrary sequence with job set J, LIST produces a schedule which has a makespan at most $\frac{k-1}{k}\ \mathrm{MP}_J + \frac{\mathrm{P}_J}{k}$.

3 The greedy algorithms

In this section we present and study two greedy algorithms. One approach to find a good schedule is to assign each job to the set where its load is smaller. We can define algorithm LG (Load Greedy) as follows.

Algorithm LG: *If a job j arrives, then it is assigned to $\mathcal{P}$ if $p_j/k \leq s_j/m$, otherwise it is assigned to $\mathcal{S}$. Schedule the jobs by LIST.*

The competitive ratios of this algorithm on the problems $MTSS(k,m)$ and $STSS(k,m)$ are determined by the following theorems.

Theorem 1 *The competitive ratio of LG for $STSS(k,m)$ is $2+\frac{m-2}{k}$.*

Proof: Consider a sequence J of jobs, fix an optimal offline schedule. Denote by P the set of jobs assigned to set $\mathcal{P}$ and by S the set of jobs assigned to $\mathcal{S}$. By Proposition 1 we have that

$$LG(J) \leq \frac{\mathrm{P}_P}{k} + \frac{k-1}{k}\,\mathrm{MP}_P + \frac{\mathrm{S}_S}{m} + \frac{m-1}{m}\,\mathrm{MS}_S.$$

Let PP and PS denote the sets of the jobs from P which are assigned to $\mathcal{P}$ and $\mathcal{S}$, and SP and SS denote the sets of the jobs from S which are assigned to $\mathcal{P}$ and $\mathcal{S}$ in the optimal solution, respectively. Then the optimal cost is at least $\frac{\mathrm{P}_{PP}}{k}+\frac{\mathrm{P}_{SP}}{k}+\frac{\mathrm{S}_{PS}}{m}+\frac{\mathrm{S}_{SS}}{m}$. On the other hand, by the definition of LG, we get that $\frac{\mathrm{P}_{PS}}{k} \leq \frac{\mathrm{S}_{PS}}{m}$ and $\frac{\mathrm{S}_{SP}}{m} \leq \frac{\mathrm{P}_{SP}}{k}$. This yields that $\frac{\mathrm{P}_P}{k}+\frac{\mathrm{S}_S}{m} \leq OPT(J)$.

Furthermore, by the definition of G it follows immediately that $\mathrm{MP}_P \leq OPT(J)$ and $\mathrm{MS}_S \leq \frac{m}{k}OPT(J)$. Substituting the above bounds into the upper bound of $LG(J)$ it follows that the algorithm is $2+\frac{m-2}{k}$-competitive.

To prove that this bound is tight in the case $k \geq 2$, consider the following sequence of jobs. Let N be an integer and $\varepsilon > 0$ small. The sequence consists of $(k-2)kN$ jobs of size $(1,\infty)$, one job of size (kN,∞), and it is finished by a job of size $(kN+\varepsilon, mN)$. Algorithm LG assigns the first $(k-2)kN+1$ jobs to $\mathcal{P}$, the last job to $\mathcal{S}$, and its cost is $((k-2)+k+m)N$. The optimal cost is $kN+\varepsilon$. As N tends to ∞ the ratio tends to $2+\frac{m-2}{k}$ which yields that our bound is tight. In the case $k=1$, consider the job $(1, m-\varepsilon)$. This job is assigned to set $\mathcal{S}$ the cost is $m-\varepsilon$, the optimal cost is 1, and the ratio tends to m as ε tends to 0. □

Theorem 2 *The competitive ratio of LG for $MTSS(k,m)$ is $2+\frac{m-1}{k}$.*

Proof: Consider a sequence J of jobs, fix an optimal offline schedule. Denote by P the set of jobs assigned to set $\mathcal{P}$ and by S the set of jobs assigned to $\mathcal{S}$. By Proposition 1 we have that

$$LG(J) \le \max\{\frac{\mathrm{P}_P}{k} + \frac{k-1}{k}\,\mathrm{MP}_P, \frac{\mathrm{S}_S}{m} + \frac{m-1}{m}\,\mathrm{MS}_S\}.$$

In the proof of Theorem 1 we showed that, $\frac{\mathrm{P}_P}{k} + \frac{\mathrm{S}_S}{m}$ is at most the sum of the optimal makespans, therefore $\frac{\mathrm{P}_P}{k} \le 2OPT(J)$ and $\frac{\mathrm{S}_S}{m} \le 2OPT(J)$ are valid in this case. Furthermore by the definition of LG we have that $\mathrm{MP}_P \le OPT(J)$ and $\mathrm{MS}_S \le \frac{m}{k}OPT(J)$ which yields the statement of the theorem. To see that the bound is tight for $k \ge 2$, consider the following sequence. The sequence contains $(k-1)m$ jobs of size $(1+\varepsilon, m/k)$ and m^2 jobs of size $(k/m+\varepsilon, 1)$ and one job of size $(m+\varepsilon, m^2/k)$. Algorithm LG assigns every jobs to $\mathcal{S}$ and it has a makespan of $(k-1)m/k + m + m^2/k$ on this set. The optimal algorithm assigns the first $(k-1)m$ and the last jobs to set $\mathcal{P}$, and the other jobs to $\mathcal{S}$, and its maximal makespan tends to m as ε tends to 0. The ratio tends to $2+\frac{m-1}{k}$ and this shows that the bound is tight. If $k=1$, consider the following sequence. The sequence contains m jobs of size $(\infty, 1)$ and one job of size $(1+\varepsilon, m)$. LG schedules all jobs on set $\mathcal{S}$ and it has there a makespan of $m+1$. The optimal algorithm schedules the last job on $\mathcal{P}$, and it has makespans $1+\varepsilon$ and 1. □

The load greedy algorithm has a small competitive ratio in the cases where m/k is small. On the other hand the competitive ratio of this algorithm tends to ∞ as m/k grows. The problem is caused by the fact that if m/k is large then some large jobs are assigned to $\mathcal{S}$, and yield a large makespan there. We can avoid this situation by using the post greedy algorithm, which was considered for the related machines scheduling problem in [5] and for the unrelated machines scheduling problem in [1]. We can define this algorithm as follows.

Algorithm PG**:** *Schedule the job on the machine where it will be finished earliest.*

As the next observation shows, the competitive ratio of this algorithm also tends to ∞ as m/k grows.

Observation 1 *The competitive ratio of PG is at least $\lfloor m/k \rfloor$ for problem $MTSS(k,m)$ and at least $2\lfloor m/k \rfloor$ for the problem $STSS(k,m)$.*

Proof: Let $l = \lfloor m/k \rfloor$ and $\varepsilon > 0$. The following sequence of jobs proves the statement. The sequence consists of l phases. The i-th phase contains k jobs of size $(1-\varepsilon, 1)$ and m jobs of size $((i+1)\varepsilon, 1)$. Algorithm PG assigns the first part of each phase to set $\mathcal{P}$, and the second part to $\mathcal{S}$, and its makespans tends to l on both sets as ε tends to 0. The optimal algorithm assigns the first parts to set $\mathcal{S}$, and the second parts to set $\mathcal{P}$, and the makespan on $\mathcal{S}$ is 1, the makespan on $\mathcal{P}$ tends to 0 as ε tends to 0. $\square$

4 The modified greedy algorithm

Here we adapt to the problems the modified version of greedy algorithm from [10]. This algorithm is a generalization of the reject total penalty type algorithms for the scheduling with rejection presented in [4] and [12]. The algorithm has two parameters $0 < \alpha \leq 1$ and $0 < \gamma \leq 1$, and the competitive ratio depends on these parameters. The key idea is to modify LG to handle the large jobs in a better way (see Step (ii)). The algorithm is defined as follows.

Algorithm $MG(\alpha, \gamma)$

- 1. *Initialization.* Let $R := \emptyset$.
- 2. When job j arrives
  - (i) If $\frac{p_j}{k} \leq \gamma \cdot \frac{s_j}{m}$, then assign j to set $\mathcal{P}$.
  - (ii) Let $r = \max\{\frac{\mathrm{P}_{R\cup\{j\}}}{k}, \mathrm{MP}_{R\cup\{j\}}\}$ (this is a lower bound on the makespan of the optimal schedule of the jobs in $R \cup \{j\}$). If $r \leq \alpha \cdot s_j$, then
    - (a) Assign job j to $\mathcal{P}$,
    - (b) Set $R = R \cup \{j\}$.
  - (iii) Otherwise, assign job j to $\mathcal{S}$
- 3. Schedule the job by LIST.

First we consider $STSS(k, m)$. Denote the set of the jobs by J. Fix an optimal schedule of the jobs. Denote by OP and OS the sets of jobs which are assigned to set $\mathcal{P}$ and set $\mathcal{S}$ respectively in the optimal schedule. Let P_0 be the set of jobs with $\frac{p_j}{k} \leq \frac{\gamma}{m} \cdot s_j$, and P be the set of jobs assigned to $\mathcal{P}$ by the algorithm.

Let us observe that, by the definition of the algorithm, we have $P_0 \subseteq P$. Define the following sets

$$X = OS \setminus P, \quad Y = OP \setminus P,$$

$$Z = OP \cap (P \setminus P_0), \quad U = OP \cap P_0,$$

$$V = OS \cap P_0, \quad W = OS \cap (P \setminus P_0).$$

Let $LB_{OP} = \max\{\frac{\mathrm{P}_{OP}}{k},\ \mathrm{MP}_{OP}\}$ and $LB_{OS} = \max\{\frac{\mathrm{S}_{OS}}{m},\ \mathrm{MS}_{OS}\}$. Then we have that $OPT(J) \geq LB_{OP} + LB_{OS}$.

First we show some structural lemmas on the sets defined above. The first lemma is also presented in [10], but it is also true in this case since it depends on Steps 1 and 2 of the algorithm and on the definition of the sets. For the readers' convenience we also sketch the proof here.

Lemma 1 *The following inequalities are valid:*

$$(1) \quad \frac{\mathrm{P}_W}{k} \leq \alpha \cdot\ \mathrm{MS}_W,$$

$$(2) \quad \mathrm{MP}_W \leq \alpha \cdot\ \mathrm{MS}_W,$$

$$(3) \quad \alpha \cdot\ \mathrm{MS}_Y \leq \max\{\frac{\mathrm{P}_Z +\ \mathrm{P}_W +\ \mathrm{P}_Y}{k},\ \mathrm{MP}_{Z\cup W\cup Y}\}.$$

Proof: We first prove (1). Let j be the last job from W. At the time when it was assigned to set $\mathcal{P}$ by the algorithm, we had $r \leq \alpha \cdot s_j$. On the other hand, j is the last job in W, thus $\frac{\mathrm{P}_W}{k} \leq r$. Furthermore, obviously $s_j \leq\ \mathrm{MS}_W$, and the validity of (1) follows. We can prove (2) in the same way as (1).

There exists a job $j \in Y$ with $s_j =\ \mathrm{MS}_Y$. At the time when it was assigned to the set $\mathcal{S}$, we had $r > \alpha s_j$. On the other hand, $R \cup \{j\} \subseteq Z \cup W \cup Y$ was valid for the considered set R, thus $r \leq \max\{\frac{\mathrm{P}_Z + \mathrm{P}_W + \mathrm{P}_Y}{k},\ \mathrm{MP}_{Z\cup W\cup Y}\}$ was also valid by definition. Therefore, the required inequality holds. $\square$

The following technical lemmas will be used.

Lemma 2 *The following inequalities are valid:*

$$(4) \frac{\mathrm{P}_P}{k} \le \frac{\mathrm{P}_Z + \mathrm{P}_U}{k} + \alpha\ \mathrm{MS}_W + \gamma \frac{\mathrm{S}_V}{m}$$

$$(5) \frac{\mathrm{S}_S}{m} \le \frac{\mathrm{S}_X}{m} + \frac{1}{\gamma}\frac{\mathrm{P}_Y}{k}$$

$$(6)\ \mathrm{MP}_P \le \max\{\ \mathrm{MP}_{Z\cup U}, \alpha\ \mathrm{MS}_W, \gamma\frac{k}{m}\ \mathrm{MS}_V\}$$

$$(7)\ \mathrm{MS}_S \le LB_{OS} + LB_{OP}/\alpha$$

Proof: The lemma follows by substituting $P = Z \cup U \cup W \cup V$ and $S = X \cup Y$, and using Lemma 1 and the fact that $p_j/k \le \gamma s_j/m$ holds if and only if $j \in V \cup U$. □

Lemma 3 *The following inequalities are valid for* $a \ge 0$, $b \ge 0$*:*

$$(8)\ a\frac{\mathrm{P}_Y}{k} + b\ \mathrm{MP}_{Z\cup U} \le \max\{a, \frac{k-1}{k}a + b\}LB_{OP}$$

$$(9)\ a\frac{\mathrm{S}_{X\cup V}}{m} + b\ \mathrm{MS}_W \le \max\{a, \frac{m-1}{m}a + b\}LB_{OS}$$

Proof: Consider the following optimization problem with the unknowns P_Y, $\mathrm{MP}_{Z\cup U}$. Maximize $a + \frac{\mathrm{P}_Y}{k} + b\ \mathrm{MP}_{Z\cup U}$ subject to $\mathrm{P}_Y + \mathrm{MP}_{Z\cup U} \le kLB_{OP}$ and $\mathrm{MP}_{Z\cup U} \le LB_{OP}$. This linear programming problem has an optimal solution which is a vertex of the polytop of the feasible solutions. Checking the vertices of the feasible polytop, it follows that only the vertices $\mathrm{P}_Y = kLB_{OP}$, $\mathrm{MP}_{Z\cup U} = 0$, and $\mathrm{P}_Y = (k-1)LB_{OP}$, $\mathrm{MP}_{Z\cup U} = LB_{OP}$ might yield an optimum, and the inequality (8) follows. One can prove (9) in the same way. □

Now we are ready to prove the following result on the $STSS(k,m)$ problem.

Theorem 3 *Algorithm $MG(\alpha,\gamma)$ is C-competitive for $STSS(k,m)$, where*

$$C = \max\{\frac{m-1}{m\alpha} + M_1, \frac{m-1}{m} + \frac{k-1}{k}\alpha + M_2, \frac{m-1}{m} + \frac{k-1}{m}\gamma + M_2\}$$

and $M_1 = \max\{\frac{1}{\gamma}, 1 + \frac{k-1}{k\gamma}\}$, $M_2 = \max\{1, \frac{m-1}{m} + \alpha\}$.

Proof: By Proposition 1 it follows, that $MG(J) \leq B_P + B_S$, where $B_P = \mathrm{P}_P/k + \frac{k-1}{k}\ \mathrm{MP}_P$ and $B_S = \mathrm{S}_S/m + \frac{m-1}{m}\ \mathrm{MS}_S$ are upper bounds on the makespans. By Lemma 2 we can prove the statement considering the following three cases depending on the value of MP_P.

Case 1: If $\mathrm{MP}_P \leq \mathrm{MP}_{Z\cup U}$, then by Lemma 2 and Lemma 3 we have the following bound:

$$MG(J) \leq \frac{\mathrm{P}_{OP}}{k} + (1/\gamma - 1)\frac{\mathrm{P}_Y}{k} + \frac{k-1}{k}\ \mathrm{MP}_{Z\cup U} + \frac{m-1}{m}\frac{LB_{OP}}{\alpha} + \frac{\mathrm{S}_{X\cup V}}{m} +$$

$$\alpha\ \mathrm{MS}_W + \frac{m-1}{m}LB_{OS} \leq (\frac{m-1}{m\alpha} + M_1)LB_{OP} + (\frac{m-1}{m} + M_2)LB_{OS} \leq C\cdot OPT(J).$$

Case 2: If $\mathrm{MP}_P \leq \alpha\ \mathrm{MS}_W$, then by Lemma 2 and Lemma 3 we have the following bound:

$$MG(J) \leq \frac{1}{\gamma}\frac{\mathrm{P}_{OP}}{k} + \frac{m-1}{m}\frac{LB_{OP}}{\alpha} + \frac{\mathrm{S}_{X\cup V}}{m} + \alpha(\frac{2k-1}{k})\ \mathrm{MS}_W + \frac{m-1}{m}LB_{OS}$$

$$\leq (\frac{1}{\gamma} + \frac{m-1}{m\alpha})LB_{OP} + (\frac{m-1}{m} + \max\{1, \frac{m-1}{m} + \frac{2k-1}{k}\alpha\})LB_{OS} \leq C\cdot OPT(J).$$

Case 3: If $\mathrm{MP}_P \leq \gamma\frac{k}{m}\ \mathrm{MS}_V$, then by Lemma 2 and Lemma 3 we have the following bound:

$$MG(J) \leq \frac{1}{\gamma}\frac{\mathrm{P}_{OP}}{k} + \frac{m-1}{m}\frac{LB_{OP}}{\alpha} + \frac{\mathrm{S}_{X\cup V}}{m} + \alpha\ \mathrm{MS}_W + \gamma\frac{k-1}{m}\ \mathrm{MS}_V +$$

$$\frac{m-1}{m}LB_{OS} \leq (\frac{1}{\gamma} + \frac{m-1}{m\alpha})LB_{OP} + (\frac{m-1}{m} + M_2 + \gamma\frac{k-1}{m})LB_{OS} \leq C\cdot OPT(J).$$

□

The upper bound depends on k, m, α, γ. For $k = 1$ the makespan on $\mathcal{P}$ is the total load. In this case the optimal choice is $\gamma = 1$, and the algorithm reduces to the reject total penalty algorithm of [4]. For $k = m = 2$ algorithm LG is optimal (see section 7 for lower bounds). For the other small values of k and m we determined the optimal values of α, γ, these values are presented in the following table.

(k,m)	(2,3)	(2,4)	(2,5)	(3,3)	(3,4)	(3,5)	(4,4)	(4,5)	(5,5)
α	0.724	0.707	0.607	0.8	0.728	0.713	0.793	0.739	0.791
γ	1	1	1	0.8	0.971	1	0.793	0.924	0.791
C	2.419	2.563	2.647	2.666	2.713	2.788	2.887	2.893	3.023

Table 1: The upper bounds

For the general case we obtain the following result.

Corollary 1 *Algorithm* $MG(0.781, 0.781)$ *is 3.562-competitive for the* $STSS(k,m)$ *problem.*

Using the same technique we obtain the following result for the $MTSS(k,m)$ problem.

Theorem 4 *Algorithm* $MG(1,1)$ *is* $4-2/m$*-competitive for* $MTSS(k,m)$*.*

Proof: Let J be an arbitrary list of jobs. Define the sets $P_0, P, S, OP, OS, X, Y, Z, U, V, W$, and the values LB_{OP}, LB_{OS}, B_P, B_S in the same way as in the case of $STSS$. In this case $OPT(J) \geq \max\{LB_{OP}, LB_{OS}\}$. Since the technical lemmas proven above depend only on the definition of the sets and the algorithm, they also hold in this case. By Proposition 1 it follows, that $MG(J) \leq \max\{B_P, B_S\}$.
By Lemma 2 we obtain that $B_S \leq \frac{\mathrm{S}_X}{m} + \frac{1}{\gamma}\frac{\mathrm{P}_Y}{k} + \frac{m-1}{m}(LB_{OS} + \frac{1}{\alpha}LB_{OP}) \leq (2 + 2\frac{m-1}{m})OPT(J)$. To bound the value of B_P we distinguish the following cases:

Case 1: Suppose $\mathrm{MP}_P \leq \mathrm{MP}_{Z\cup U}$, then by Lemma 2 and Lemma 3 we obtain that

$B_P \leq (\max\{1, 2\frac{k-1}{k}\} + \max\{\gamma, \frac{m-1}{m}\gamma + \alpha\})OPT(J) \leq (4-2/m)OPT(J)$.

Case 2: Suppose $\mathrm{MP}_P \leq \alpha\ \mathrm{MS}_W$, then by Lemma 2 and Lemma 3 we obtain that

$B_P \leq (1 + \max\{\gamma, \frac{m-1}{m}\gamma + \frac{2k-1}{k}\alpha\})OPT(J) \leq (4-2/m)OPT(J)$.

Case 3: Suppose $\mathrm{MP}_P \leq \gamma\frac{k}{m}\ \mathrm{MS}_V$, then by Lemma 2 and Lemma 3 we obtain that

$B_P \leq (1 + \gamma\frac{k-1}{m} + \max\{\gamma, \frac{m-1}{m}\gamma + \alpha\})OPT(J) \leq (4-2/m)OPT(J)$. $\square$

5 The semi-online case

In this section we consider a semi-online version of $STSS$. In this version we have to decide in online way on which set of machines to schedule the jobs. Finally, when the stream of jobs has come to an end we schedule the jobs assigned to set $\mathcal{P}$ (respectively, the jobs assigned to set $\mathcal{S}$) on the machines of $\mathcal{P}$ (respectively, $\mathcal{S}$) so as to minimize the preemptive makespan in offline way. This problem is denoted by $SSTSS(k, m)$. This kind of semi online version of $MTSS$ is investigated in [10]. The special case when $k = 1$, is the semi online version of scheduling with rejection where we have to decide whether we reject or schedule the job in an online fashion, but we do offline scheduling at the end. In what follows we denote this problem by $SOSR(m)$.

With preemption a job may be scheduled on multiple machines from the choosen set. A *time slot* is a non-empty interval (q, t) with $q \geq 0$. In preemptive scheduling we have to assign time slots to each job on one or more machines. The sum of the sizes of the time slots must be the processing time of the job, and if time slots $(q_1, t_1), \ldots, (q_i, t_i)$ are assigned to a job, then $t_j \leq q_{j+1}$ must be valid for $j = 1, \ldots, i-1$. Furthermore, no two jobs may have overlapping time slots on the same machine.

We recall the following theorem about the preemptive offline makespan.

Proposition 2 ([11]) *The optimal preemptive makespan for any set of jobs J on k identical machines is* $\max\{\ \mathrm{P}_J/k,\ \mathrm{MP}_J\}$*, and a schedule with this makespan can be obtained by a linear time greedy type algorithm.*

We can adapt algorithms LG and MG in a natural way for this problem, the same method is used to determine the set of machines where the job is scheduled, and we do offline scheduling at the end. The analysis of these algorithms gives the following results.

Theorem 5 *The competitive ratio of LG for $SSTSS(k, m)$ is*

$$C = \begin{cases} 1 + \frac{m}{k} & \text{if } k \geq 2 \\ m & \text{if } k = 1. \end{cases}$$

Proof: Consider first the case $k \geq 2$. Let a and b denote the makespan obtained by LG on $\mathcal{P}$ and $\mathcal{S}$, respectively. Furthermore, let us fix an optimal solution. First assume that $a \geq b$. Suppose that makespan a is defined by the maximal processing time. Denote the job with this maximal processing time by j. Then $p_j = a$, and since LG assigned this job to $\mathcal{P}$ we get $s_j > \frac{m}{k}a$. Hence, the optimal cost is at least a and this yields that the competitive ratio is at most $\frac{a+b}{a} \leq 2 \leq 1 + \frac{m}{k}$. Now suppose that the makespan is defined by the load of the jobs. Then in the same way as in the proof of Theorem 1, we obtain that the optimal cost is at least a, and this proves the theorem for this case.

Let us assume that $a < b$. Now consider two cases depending on the makespan on $\mathcal{S}$. If the makespan is the load, then in the same way as in Theorem 1 we obtain that the optimal cost is at least b, which yields that the algorithm is $2 \leq 1 + \frac{m}{k}$ - competitive. Now suppose that the makespan is defined by a maximal processing time. Denote the job with this maximal processing time by j. Then $s_j = b$ and since the algorithm assigned this job to set $\mathcal{S}$, we obtain that $p_j > \frac{k}{m}b$. Hence, the optimal cost is at least $\frac{k}{m}b$. Furthermore, we observed above in the case $a \geq b$ that if the makespan on $\mathcal{P}$ is a, then the optimal cost is at least a. This yields that the optimal cost is at least $\max\{\frac{k}{m}b, a\}$. Hence, the algorithm

$$\frac{a+b}{\max\{\frac{k}{m}b, a\}} \leq 1 + \frac{m}{k}$$

competitive. We can handle the case $k = 1$ in a similar way. If $m = 1$, then the algorithm obviously 1-competitive. If $m \geq 2$, we consider four cases and we can obtain the upper bound 2, in the first three cases in the same way as we had for $k \geq 2$. The only difference is in the case when $a < b$ and b is defined by the processing time of a maximal job j. In this case, if the optimal algorithm assigns j to $\mathcal{S}$, we obtain the upper bound 2. If it assigns j to $\mathcal{P}$, then it causes $\frac{k}{m}b$ cost there. But in this case, since $k = 1$, we cannot schedule the jobs which cause the makespan on $\mathcal{P}$ on the others machines from $\mathcal{P}$. This yields that these jobs will cause at least a extra cost in the optimal solution. Thus, the algorithm is

$$\frac{a+b}{\frac{k}{m}b + a} \leq \frac{m}{k}$$

competitive.

To prove that the above analysis is tight in the case $k > 1$, consider the sequence of two jobs of size $(1, \frac{m}{k} - \varepsilon)$ and $(1, m)$. Algorithm LG assigns the first job to $\mathcal{S}$, the second job to $\mathcal{P}$ with cost $1 + \frac{m}{k} - \varepsilon$. The optimal cost is 1, thus as ε tends to 0, the competitive ratio on this sequence tends to $1 + \frac{m}{k}$. The tightness of the analysis for the case $k = 1$ follows if we consider one job of size $(1, \frac{m}{k} - \varepsilon)$. $\square$

For MG we obtain the following result.

Theorem 6 *Algorithm $MG(\alpha, \gamma)$ is C-competitive for $SSTSS(k, m)$, where*

$$C = \max\{\frac{1}{\gamma}, 1 + \frac{k-1}{k\gamma}, 1 + \frac{1}{\alpha}, 1 + \alpha + \frac{m-1}{m}\gamma\}$$

Proof: Consider an arbitrary set J of jobs. We can define the sets $P_0, P, S, OP, OS, X, Y, Z, U, V, W$, and the values LB_{OP}, LB_{OS} in the same way as in the completely online case. Since the technical lemmas proven there depend only on the definition of the sets and the algorithm, they also hold in this case. Furthermore, we have $OPT(J) \geq LB_{OP} + LB_{OS}$. By Proposition 2 we obtain, that $MG(J) = \max\{\frac{\mathrm{P}_P}{k}, \ \mathrm{MP}_P\} + \max\{\frac{\mathrm{S}_S}{m}, \ \mathrm{MS}_S\}$.

We can prove the theorem by considering the following cases.

Case 1: Suppose, that $\frac{\mathrm{P}_P}{k} \geq \ \mathrm{MP}_P$ and $\frac{\mathrm{S}_S}{m} \geq \ \mathrm{MS}_S$. In this case by Lemma 2 and Lemma 3 we obtain that

$$MG(J) \leq \frac{1}{\gamma}\frac{P_{OP}}{k} + \max\{1, \frac{m-1}{m} + \alpha\}LB_{OS} \leq C \cdot OPT(J).$$

Case 2: Suppose that $\frac{\mathrm{P}_P}{k} < \ \mathrm{MP}_P \leq \ \mathrm{MP}_{Z \cup U}$ and $\frac{\mathrm{S}_S}{m} \geq \ \mathrm{MS}_S$. In this case by Lemma 2 and Lemma 3 we obtain that

$$MG(J) \leq \max\{\frac{1}{\gamma}, \frac{k-1}{k}\frac{1}{\gamma} + 1\}LB_{OP} + \frac{\mathrm{S}_{OS}}{m} \leq C \cdot OPT(J)$$

Case 3: Suppose that $\frac{\mathrm{P}_P}{k} < \ \mathrm{MP}_P \leq \alpha \ \mathrm{MS}_W$ and $\frac{\mathrm{S}_S}{m} \geq \ \mathrm{MS}_S$. In this case by Lemma 2 and Lemma 3 we obtain that

$$MG(J) \leq \frac{1}{\gamma}\frac{P_{OP}}{k} + \max\{1, \frac{m-1}{m} + \alpha\}LB_{OS} \leq C \cdot OPT(J).$$

Case 4: Suppose that $\frac{\mathrm{P}_P}{k} < \ \mathrm{MP}_P \leq \gamma\frac{k}{m} \ \mathrm{MS}_V$ and $\frac{\mathrm{S}_S}{m} \geq \ \mathrm{MS}_S$. In this case by Lemma 2 and Lemma 3 we obtain that

$$MG(J) \leq \frac{1}{\gamma}\frac{P_{OP}}{k} + \max\{1, \frac{m-1}{m} + \gamma\frac{k}{m}\}LB_{OS} \leq C \cdot OPT(J).$$

Case 5: Suppose, that $\frac{\mathrm{P}_P}{k} \geq \mathrm{MP}_P$ and $\frac{\mathrm{S}_S}{m} < \mathrm{MS}_S$. In this case by Lemma 2 and Lemma 3 we obtain that

$$MG(J) \leq \frac{\mathrm{P}_{OP}}{k} + \frac{LB_{OP}}{\alpha} + LB_{OS} + \max\{\gamma, \frac{m-1}{m}\gamma + \alpha\}LB_{OS} \leq C \cdot OPT(J).$$

Case 6: Suppose that $\frac{\mathrm{P}_P}{k} < \mathrm{MP}_P \leq \mathrm{MP}_{Z \cup U}$ and $\frac{\mathrm{S}_S}{m} < \mathrm{MS}_S$. In this case by Lemma 2 and Lemma 3 we obtain that

$$MG(J) \leq \mathrm{MP}_{OP} + \frac{LB_{OP}}{\alpha} + LB_{OS} \leq C \cdot OPT(J).$$

Case 7: Suppose that $\frac{\mathrm{P}_P}{k} < \mathrm{MP}_P \leq \alpha\ \mathrm{MS}_W$ and $\frac{\mathrm{S}_S}{m} < \mathrm{MS}_S$. In this case by Lemma 2 and Lemma 3 we obtain that

$$MG(J) \leq \alpha\ \mathrm{MS}_{OS} + \frac{LB_{OP}}{\alpha} + LB_{OS} \leq C \cdot OPT(J).$$

Case 8: Suppose that $\frac{\mathrm{P}_P}{k} < \mathrm{MP}_P \leq \gamma\frac{k}{m}\ \mathrm{MS}_V$ and $\frac{\mathrm{S}_S}{m} < \mathrm{MS}_S$. In this case by Lemma 2 and Lemma 3 we obtain that

$$MG(J) \leq \gamma\frac{k}{m}\ \mathrm{MS}_{OS} + \frac{LB_{OP}}{\alpha} + LB_{OS} \leq C \cdot OPT(J).$$

□

To optimize the competitive ratio over the possibilities of α, γ in the general case, we have to choose $\alpha = \gamma = 1/\sqrt{2}$. Substituting these values into the above theorem, we obtain the following result.

Corollary 2 *Algorithm $MG(1/\sqrt{2}, 1/\sqrt{2})$ is $1 + \sqrt{2}$-competitive for $SSTSS(k, m)$ for arbitrary k and m.*

For the $SOSR$ problem, the algorithm has a better competitive ratio. It is worth noting that in this case the algorithm is reduced to the total rejection scheme of [12].

Theorem 7 *Algorithm $MG(\alpha, \gamma)$ is C-competitive for $SOSR(m)$, where*

$$C = \max\{\frac{1}{\gamma}, 1 + \frac{1}{\alpha}, 1 + \alpha + \frac{m-1}{m}\gamma\}.$$

Proof: To prove that the algorithm is C-competitive, we just have to simplify the upper bound proof of Theorem 6. Since $k = 1$, the makespan on set $\mathcal{P}$ is the sum of the jobs, therefore we have to consider only the cases 1 and 5. □

Minimizing the competitive ratio over (α, γ), we obtain the following result for the general case.

Corollary 3 *Algorithm $MG(0.802, 0.445)$ is 2.247-competitive for $SOSR(m)$ for every m.*

6 Lower bounds

Theorem 8 *If an online algorithm is c-competitive for the $MTSS(k, m)$ problem, then $c \geq 2$.*

Proof: Consider a job of size $(1, 1)$. If the algorithm assigns it to set $\mathcal{P}$, then k jobs of size $(1, 2)$ follows, otherwise m jobs of size $(2, 1)$. For this sequence the cost of the algorithm is 2. The optimal schedule has cost 1. □

Theorem 9 *If an online algorithm for any of the problems $STSS(k, m)$, $SSTSS(k, m)$ with $k \geq 2$ is c-competitive, then $c \geq 2$.*

Proof: Consider a job of size $(1, 1)$. If the algorithm assigns it to set $\mathcal{P}$, then the second job is $(2, 1)$, otherwise it is $(1, 2)$. This yields, that the cost of the algorithm is 2. The optimal algorithm assigns the two jobs to the same set and it has cost 1. □

By this lower bound we obtain the following corollary.

Corollary 4 *Algorithm LG is optimal for STSS(2,2), and for SSTSS(m,m) for every m.*

In the case $k = 1$ we can use the following lower bound from online scheduling with rejection which is also valid with the same proof for the semi online case.

Theorem 10 ([4]) *If an online algorithm is c-competitive for $SOSR(m)$, then $c^{m-1} + c^{m-2} + \ldots + 1 \leq c^m$.*

In the case $k \geq 5$ we can obtain a stronger bound for the $STSS(k,m)$ problem.

Theorem 11 *If an online algorithm for any of the problems $STSS(k,m)$, with $k \geq 5$ is c-competitive, then $c \geq \frac{3+\sqrt{3}}{2} \approx 2.366$.*

Proof: By contrary suppose that we have an algorithm which has smaller competitive ratio than $\frac{3+\sqrt{3}}{2}$. Consider such algorithm. Denote by x the value $\frac{1+\sqrt{3}}{2}$. Consider a sequence of jobs which started by $k-1$ jobs of size $(1,x)$. If any of these jobs is assigned to $\mathcal{S}$, then let the k-th job have size $(1,\infty)$. It will be assigned to $\mathcal{P}$, thus the algorithm has a cost of at least $1+x$ while the optimal cost is 1 by scheduling all of the k jobs on the machines of $\mathcal{P}$. The ratio $(1+x)/1 = \frac{3+\sqrt{3}}{2}$, thus in this case we have a contradiction. If the jobs are scheduled on the machines of $\mathcal{P}$, and there is a machine containing more than one job, then let the next job have size (∞, x). In this case, the cost of the algorithm is at least $2+x$, the optimal algorithm schedules every jobs on set $\mathcal{S}$, (it is possible since the number of jobs is $k \leq m$), and has cost x. Since $\frac{2+x}{x} > \frac{3+\sqrt{3}}{2}$, this case also leads to a contradiction.

Therefore, we can suppose that the first $k-1$ jobs are scheduled on $k-1$ machines from set $\mathcal{P}$. The next job has size $(2x, 2x)$. If the algorithm assigns it to $\mathcal{S}$, then it is followed by two jobs of size $(2x, \infty)$. If the $(2x, 2x)$ job is assigned to $\mathcal{P}$, then it is followed by a new job of size $(2x, 2x)$. If the algorithm assigns this new job to $\mathcal{S}$ then the last job has size $(2x, \infty)$, otherwise the last job has size $(\infty, 2x)$. In all of these cases, the algorithm has to schedule two new jobs of size $2x$ on $\mathcal{P}$, and one job of size $2x$ on $\mathcal{S}$, therefore its cost is at least $4x+1$. The offline algorithm can assign the jobs in each cases to the same set of machines, and it can schedule the first $k-1$ jobs in pairs on $(k-1)/2$ machines, and the last three jobs on other machines and its cost is $2x$. (This is a valid schedule since $\lceil (k-1)/2 \rceil + 3 \leq k \leq m$ by $k \geq 5$.) Since $\frac{4x+1}{2x} = \frac{3+\sqrt{3}}{2}$, we have also obtained a contradiction in this case, and the statement of the theorem follows. □

7 The offline problems

The special case of the $STSS$ problem where $m = 2$ and $p_j = \infty$ for every jobs leads to the two identical machines scheduling problem therefore this problem is NP-hard. The $MTSS(1,1)$ problem is the two unrelated machines scheduling problem, thus this problem is also NP-hard. Since the problems are NP-hard, it is an interesting question to determine their approxamibility. The $MTSS$ problem is a restricted case of the unrelated machines scheduling problem, therefore the approximation schemes developed for the unrelated machines also work for this problem. In this part we present an FPTAS for the $STSS$ problem provided that k and m are constant. We use the dynamic programming and rounding technique which was first used for scheduling problems in [9]. First we show that the problem can be solved by dynamic programming if the processing times are integers.

Lemma 4 *The STSS(k,m) problem with integer processing times can be solved in time polynomial in n and $OPT(J)$, where n is the number of jobs.*

Proof: The problem can be solved by dynamic programming. Let $M_1, \ldots, M_k$ represent the loads on the machines of $\mathcal{P}$, $M_{k+1}, \ldots, M_{m+k-1}$ represent the loads on the first $m-1$ machines of $\mathcal{S}$. For each $M_i \leq OPT(J)$ we calculate the minimal value of the load on the last machine of $\mathcal{S}$ that can be achieved with the loads $M_1, \ldots, M_{k+m-1}$. We denote this value after the j-th job by $S_j(M_1, \ldots, M_{k+m-1})$, this function is defined to be ∞ if $M_i < 0$ for some i. Using this information we obtain the minimal cost which can be achieved with the loads $M_1, \ldots, M_{k+m-1}$, this value is $Z(M_1, \ldots, M_{k+m-1}) = A + B$ where $A = \max\{M_1, \ldots, M_k\}$ and $B = \max\{M_{k+1}, \ldots, M_{k+m-1}, S_n(M_1, \ldots, M_{k+m-1})\}$.
For $M_1, \ldots, M_{k+m-1} \geq 0$ the value of S_j can be computed recursively as follows

$$S_0(M_1, \ldots, M_{k+m-1}) = 0$$

$$S_j(M_1, \ldots, M_{k+m-1}) = \min\{s_j + S_{j-1}(M_1, \ldots, M_{k+m-1}),$$

$$\min_{1 \leq i \leq k} S_{j-1}(M_1, \ldots, M_{i-1}, M_i - p_j, M_{i+1} \ldots M_{k+m-1}),$$

$$\min_{k+1 \le i \le k+m-1} S_{j-1}(M_1, \ldots, M_{i-1}, M_i - s_j, M_{i+1} \ldots M_{k+m-1})\}.$$

We can compute all the values as $\max_i M_i$ increases, if it reaches the smallest value of Z which we obtained so far, then the algorithm stops and we obtain the optimum. □

We can transform the dynamic programming algorithm presented above into FPTAS by a rounding technique which was introduced in [9] for scheduling problems.

Theorem 12 *For any $\varepsilon > 0$, there exists ε-approximation algorithm for the $STSS(k, m)$ problem that runs in time polynomial in n^{k+m} and $1/\varepsilon^{k+m}$.*

Proof: Define the approximation algorithm $APPR_\varepsilon$ for the $STSS(k, m)$ problem. Let I be an instance of the $STSS(k, m)$ problem with n jobs. First we can use the polynomial time online algorithm $MG(1, 1)$ which gives a 4 approximate solution with cost Z. Define a modified instance I' from I by $p'_i = \lfloor p_j/l \rfloor$ and $s'_i = \lfloor p_j/l \rfloor$, where $l = \varepsilon Z/4n$. We obtain a solution of this modified problem by Lemma 4, and we can change it into an approximate solution of the original problem scheduling the jobs on the same machines as in the optimal solution of I'.

Since $|APPR_\varepsilon(I) - OPT(I)| \le nl$, we have that

$$\frac{|APPR_\varepsilon(I) - OPT(I)|}{OPT(I)} \le \frac{nl}{OPT(I)} \le \frac{\varepsilon Z/4}{OPT(I)} \le \varepsilon.$$

Using the observation that $OPT(I') \le OPT(I)/l \le 4Z/l = n/\varepsilon$, by Lemma 4 it also follows, that the running time is polynomial in n^{k+m} and $1/\varepsilon^{k+m}$. □

8 Summary and open problems

In this paper we investigated the following scheduling problem: There are two sets of identical machines, the jobs have two processing times one for each set of machines. We considered two different objective functions.

In the first model the goal is to minimize the maximum of the makespans on the sets. The model is a restricted version of the unrelated machines scheduling problem. First we investigated the online version of the problem, we have analyzed two greedy type algorithms and a more sophisticated modified greedy algorithm. The latter algorithm is $4 - 2/m$-competitive if the sets contain at most m machines. In the online unrelated machines scheduling problem there is no constant competitive algorithm for the general case, therefore we obtained that it is easier to approximate this restricted version. The FPTAS for the unrelated machines problem also works in the investigated model, and the semi online version of the problem is studied in [10], therefore we did not considered the offline and the semi online versions.

In the second model the objective is to minimize the sum of the makespans. The problem is a generalization of the scheduling with rejection problem. We consider the online, semi online and offline versions of this model. In the online problem we analyzed two greedy type and a more sophisticated modified greedy algorithm. The latter algorithm is 3.562-competitive for arbitrary number of machines, better results have been obtained for smaller sets of machines. A lower bound of 2.366 is also proven if the sets contain at least 5 machines. We also considered the semi online version where we have to decide in online way which set of machines is used for the jobs, but we do offline scheduling at the end. Here the modified greedy algorithm is 2.41-competitive for arbitrary number of machines. The case where one of the sets contains only one machine is a semi online version of scheduling with rejection. In this restricted case the algorithm is 2.247-competitive. The algorithm belongs to the class of obliviously scheduling algorithms, and in the completely online case there is a lower bound of 2.33 for the possible competitive ratios of the obliviously scheduling algorithms, therefore we also obtained that it helps to allow the algorithm to do offline scheduling at the end. Concerning the offline problem we presented an FPTAS provided that the number of machines is constant.

Related to the problem there are some further questions to investigate. First it would be nice to decrease the gaps between the lower and upper bounds, we have matching bounds only in the cases $STSS(2,2)$, and $SSTSS(m,m)$ with arbitrary m. In this paper we considered the cost functions obtained by taking the l_1 and l_∞ norms of the two makespans. It could be interesting to investigate the other l_p norms.

A straightforward generalization of the problem is to consider k-sets ($k > 2$) of machines. One can easily modify the FPTAS for the offline case to this more general version. Concerning the online case it has been recently proven ([2]) that there is an $O(\log k)$ competitive algorithm for the Maximum k-Sets Scheduling problem and an $O(k)$-competitive algorithm for the Sum k-Sets Scheduling problem, furthermore $\Omega(\log k)$ and $\Omega(k)$ lower bounds for the possible competitive ratios are also proven.

References

[1] Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., Waarts, O.: On-line load balancing with application to machine scheduling and virtual circuit routing, *Journal of the ACM*, **44**,486–504, 1997.

[2] Azar J., Personal communication

[3] Azar, J., Naor, J., Rom, R.: The competitiveness of on-line assignments, *Journal of Algorithms*, **18**, 221–237, 1995.

[4] Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., Stougie, L.: Multiprocessor scheduling with rejection, *SIAM Journal on Discrete Mathematics*, **13**, 64–78, 2000.

[5] Cho, Y., Sahni, S.: Bounds for list schedules on uniform processors, *SIAM Journal on Computing*, **9**, 91–103, 1988.

[6] Engels, D. W., Karger D. R., Kolliopoulos, S. G., Sengupta, S., Uma, R. N., Wein, J.: Techniques for scheduling with rejection, In *Proceedings of the 6th Annual European Symposium on Algorithms, LNCS 1461*, Springer-Verlag 490–501, 1998

[7] Epstein, L., Sgall, J.: Approximation schemes for scheduling on uniformly related and identical parallel machines, *Proceedings of the 7th Annual European Symposium on Algorithms, LNCS 1643*, Springer-Verlag Berlin, 1999, 151–162.

[8] Graham, R. L.: Bounds for certain multiprocessor anomalies, *Bell System Technical J.*, **45**, 1563–1581, 1966.

[9] Horowitz, E., Sahni, S.: Exact and approximate algorithms for scheduling non identical processors, *Journal of the ACM*, **23**, 317–327, 1976.

[10] Imreh, Cs.: An online scheduling algorithm for a two-layer multiprocessor architecture, *Acta Cybernetica*, **15**, (2001), 163–172

[11] Mc Naughton, R.: Scheduling with deadlines and loss functions, *Management Science 6*, 1–12, 1959.

[12] Seiden, S. S.: Preemptive Multiprocessor Scheduling with Rejection, *Theoretical Computer Science*, **262**, 437–458, 2001

[13] Sgall, J.: On-line scheduling, in *Online algorithms: The State of the Art* (A. Fiat, and G. J. Woeginger (eds.)), LNCS 1442, Springer-Verlag Berlin, 196–231, 1998