4OR-Q J Oper Res (2018) 16:95–104
https://doi.org/10.1007/s10288-017-0356-0

RESEARCH PAPER

Vector scheduling with rejection on a single machine

Weidong Li[1,2] · Qianna Cui[2]

Received: 15 March 2017 / Revised: 13 July 2017 / Published online: 16 October 2017

Abstract In this paper, we study a vector scheduling problem with rejection on a single machine, in which each job is characterized by a d-dimension vector and a penalty, in the sense that, jobs can be either rejected by paying a certain penalty or assigned to the machine. The objective is to minimize the sum of the maximum load over all dimensions of the total vector of all accepted jobs, and the total penalty of rejected jobs. We prove that the problem is NP-hard and design two approximation algorithms running in polynomial time. When d is a fixed constant, we present a fully polynomial time approximation scheme.

Keywords Vector scheduling · Approximation algorithms · Dynamic programming · Fully polynomial time approximation scheme

Mathematics Subject Classification 90B35

1 Introduction

Vector scheduling (VS) problem, which is a multi-dimensional generalization of the classical makespan minimization problem (Alon et al. 1998; Hochbaum and Shmoys 1987), is to schedule n d-dimensional jobs on m machines such that the maximum load over all dimensions and all machines is minimized. It finds many applications in multi-dimensional resource scheduling. When d is an arbitrary number, Chekuri and Khanna (2004) proved that it is NP-hard to approximate the VS problem

✉ Weidong Li
weidong@ynu.edu.cn

1 Yunnan University, Kunming 650091, People's Republic of China

2 Dianchi College of Yunnan University, Kunming 650228, People's Republic of China

within any constant factor, and presented an $O(\ln^2 d)$-approximation algorithm and an $O(\ln dm/\ln\ln dm)$-approximation with high probability for the VS problem. Meyerson et al. (2013) designed an improved $O(\log d)$-approximation algorithm for the VS problem. Recently, Im et al. (2015) designed an $O(\log d/\log\log d)$-approximation algorithm for the VS problem. When d is a constant, Chekuri and Khanna (2004) presented a $(1+\epsilon)$-approximation algorithm with running time $(nd/\epsilon)^{O((\frac{\ln(d/\epsilon)}{\epsilon})^d)}$. Bansal et al. (2016) proved that no $(1+\epsilon)$-approximation algorithm with running time $\exp(\lfloor 1/\epsilon \rfloor^{o(d)})$ for the VS problem, unless NP has subexponential time algorithm. They (Bansal et al. 2016) also designed a $(1+\epsilon)$-approximation algorithm with running time $\exp((1/\epsilon)^{O(d\log\log d)})$ for the VS problem, for sufficiently small $\epsilon > 0$, which is the best possible result. When $d = 1$, the VS problem is exactly the classical makespan minimization problem (Alon et al. 1998; Hochbaum and Shmoys 1987). Chen et al. (2014) proved that a $(1+\epsilon)$-approximation algorithm with running time $2^{O((1/\epsilon)^{1-\delta})} + n^{O(1)}$ for any $\delta > 0$ implies that exponential time hypothesis (ETH) fails. Recently, Jansen et al. (2016) designed a $(1+\epsilon)$-approximation algorithm with running time $2^{\tilde{O}(1/\epsilon)} + O(n\log n)$, which is tight under ETH up to logarithmic factors on the exponent. The VS problem has been generalized to a wide class of cost functions (Epstein and Tassa 2003, 2006), unrelated machines (Bonifaci and Wiese 2012), and online cases (Im et al. 2015; Zhu et al. 2014). More related results can be found in the recent survey (Christensen et al. 2017).

Scheduling with rejection, proposed by Bartal et al. (2000), is an another generalization of the classical makespan minimization problem, where a job is either rejected, in which case a rejection penalty has to be paid, or accepted and processed on the machine. The objective is to minimize the makespan plus the sum of the penalties of rejected jobs. They (Bartal et al. 2000) proposed a 2-approximation algorithm with running time $O(n\log n)$ and a polynomial time approximation scheme (PTAS) for scheduling with rejection on parallel machines. Ou et al. (2015) proposed a $(3/2+\epsilon)$-approximation algorithm with running time $O(n\log n + n/\epsilon)$ for the same problem, where ϵ is a small given positive constant. Zhang and Lu (2016) considered parallel-machine scheduling with release dates and rejection where the objective is to minimize the makespan plus the sum of the penalties of rejected jobs, and proposed a 2-approximation algorithm, which is improved by Zhong and Ou (2016) who designed a PTAS. Li et al. (2015) designed a PTAS for a variant of scheduling with rejection, where the rejection cost is bounded and the objective is to minimize the makespan.

When the number of machines is 2, Zhong et al. (2017) developed a $(3/2+\epsilon)$-approximation algorithm for scheduling with release dates and rejection. When the number of machines is 1, Shabtay et al. (2012) presented a bicriteria approach to scheduling a single machine with job rejection and positional penalties. Zhang et al. (2009) proved that the problem of single machine scheduling with release dates and rejection is NP-hard and presented a 2-approximation algorithm with running time of $O(n^2)$, where the objective is to minimize the makespan plus the sum of the penalties of rejected jobs. Recently, He et al. (2016) and Ou et al. (2016) independently designed an improved approximation algorithm with running time of $O(n\log n)$. Zhang et al. (2010) also considered a variant of scheduling with rejection, called single-machine scheduling under the job rejection constraint, where the objective is to minimize the makespan, under the constraint that the sum of the penalties of rejected jobs is no more than a given bound. More results can be found in the surveys (Shabtay et al. 2013; Slotnick 2011).

Motivated by Bartal et al. (2000) and Chekuri and Khanna (2004), we propose a new scheduling model, called *vector scheduling with rejection* (VSR), which schedules n d-dimensional vectors on m machines, where each vector is either rejected, which incurs a rejection penalty, or accepted and processed by the machine. The objective is to minimize the maximum load over all dimensions and all machines plus the total cost of rejected vectors. Clearly, if $d = 1$, VSR is exactly the problem of scheduling with rejection (Bartal et al. 2000), and if the rejection penalty is infinite for each vector, VSR is exactly the VS problem (Chekuri and Khanna 2004). In this paper, we only consider the single machines case, as in He et al. (2016), Ou et al. (2016), Zhang et al. (2009, 2010). Actually, VSR on a single machine is a vector bi-partitioning problem and is not a proper "scheduling" problem, as there is neither a sequencing issue nor an allocation problem. However, for convenience, we still call VSR on a single machine as a scheduling problem.

The remainder of this paper is structured as follows. In Sect. 2 we provide a formal problem statement and prove that the VSR problem on a single machine is NP-hard even if $d = 2$. In Sect. 3, we propose two approximation algorithms for arbitrary d. In Sect. 4, we present an fully polynomial time approximation scheme for fixed d. We provide conclusion in Sect. 5.

2 Problem description and NP-hardness

VSR on a single machine is defined as follows. We are given a set of n jobs $J = \{J_1, J_2, \ldots, J_n\}$ and a machine. Associated with each job J_j are a d-dimensional vector $\mathbf{p}_j = (p_j^1, p_j^2, \ldots, p_j^d)$ where p_j^k is the amount of resource k needed by job J_j, and a rejection penalty cost $w_j > 0$. Job J_j can be either rejected with a rejection penalty cost w_j incurred, or accepted and processed on the given machine. A feasible solution is to determine a subset of jobs A to be accepted and a subset of jobs $R = J \backslash A$ to be rejected. Let $L^k = \sum_{J_j \in A} p_j^k$ be the load of resource k on the machine. VSR on a single machine is to minimize the objective function $Z = \max_k \sum_{J_j \in A} p_j^k + \sum_{J_j \in R} w_j$, i.e., the maximum load plus the total penalty of the rejection jobs. Clearly, if $d = 1$, this problem is exactly the problem of single machine scheduling with release dates and rejection $1|r_j, \text{reject}|C_{max} + \sum_{J_j \in R} w_j$ (Zhang et al. 2009) with $r_j = 0$ for each job J_j. In the other words, we extend the problem of single machine scheduling with rejection $1|\text{reject}|C_{max} + \sum_{J_j \in R} w_j$ to the multi-dimension case.

We introduce a binary variable x_j for each job J_j to indicate whether J_j is accepted, where $x_j = 1$ if and only if J_j is accepted. VSR on a single machine can be formulated as the following integer linear program (ILP).

$$\begin{cases} Min \;\; L + \sum_{j=1}^{n} w_j(1-x_j) \\ L^k = \sum_{j=1}^{n} p_j^k x_j \le L, \quad k = 1, 2, \ldots, d \\ x_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n. \end{cases} \tag{1}$$

When $d = 1$, it is easy to verify that VSR on a single machine can be solved optimally by accepting the jobs in $A = \{J_j | p_j^1 < w_j\}$. When $d \ge 2$, VSR on a single machine is NP-hard, which can be proved as follows.

Theorem 1 *VSR on a single machine is NP-hard even if* $d = 2$.

Proof We use the NP-complete 2-Partition problem (Garey and Johnson 1979) for the reduction. Given $t+1$ positive integers $a_1, a_2, \ldots, a_t, B$, such that $\sum_{j=1}^{t} a_j = 2B$, 2-Partition problem is to find a subset $S \subset \{a_1, a_2, \ldots, a_t\}$, such that $\sum_{a_j \in S} a_j = B$.

For an instance I of the 2-Partition problem, we construct an instance $\tau(I)$ of VSR on a single machine with $n = t+1$ jobs as follows. For $1 \le j \le t$, $\mathbf{p}_j = (0, 2a_j)$ and $w_j = a_j$. For $j = t+1$, $\mathbf{p}_{t+1} = (2B, 0)$ and $w_{t+1} = 3B+1$. Obviously, this construction can be done in polynomial time. We will prove that instance I has a solution if and only if instance $\tau(I)$ has a feasible solution with objective value $3B$.

If instance I has a solution S such that $\sum_{a_j \in S} a_j = B$, we accept the jobs in $A = \{J_j | a_j \in S\} \cup \{J_{t+1}\}$ and reject the other jobs. The load of every resource is $2B$, and the total penalty of the rejected jobs is $\sum_{J_j \in R} w_j = \sum_{J_j \in R} a_j = \sum_{j=1}^{t} a_j - \sum_{a_j \in S} a_j = 2B - B = B$. It implies that instance $\tau(I)$ has a feasible solution with objective value $3B$.

If $\tau(I)$ has a feasible solution with objective value $3B$, job J_{t+1} must be accepted as $w_{t+1} = 3B+1 > 3B$. Thus, the load of resource 1 is $2B$, as $\mathbf{p}_{t+1} = (2B, 0)$. If the load of resource 2 is more than $2B$, i.e.,

$$\sum_{J_j \in A \backslash \{J_{t+1}\}} p_j^2 = \sum_{J_j \in A \backslash \{J_{t+1}\}} 2a_j > 2B, \tag{2}$$

the total penalty of the rejected jobs is

$$\sum_{J_j \in R} w_j = \sum_{j=1}^{t} a_j - \sum_{J_j \in A \backslash \{J_{t+1}\}} a_j, \tag{3}$$

as $R = \{J_1, J_2, \ldots, J_n\} \backslash (A \backslash \{J_{t+1}\})$. Therefore, the objective value of this solution for $\tau(I)$ is

$$\max\left\{2B, \sum_{J_j \in A \backslash \{J_{t+1}\}} p_j^2\right\} + \sum_{J_j \in R} w_j$$
$$= \sum_{J_j \in A \backslash \{J_{t+1}\}} 2a_j + \sum_{J_j \in R} w_j$$

$$
\begin{aligned}
&= \sum_{j=1}^{t} a_j + \sum_{J_j \in A \setminus \{J_{t+1}\}} a_j \\
&> 3B,
\end{aligned}
$$

a contradiction, where the first equality follows from (2), the second equality follows from (3), and the last inequality follows from (2) and the fact $\sum_{j=1}^{t} a_j = 2B$. Hence, the load of resource 2 is no more than $2B$. It implies that the total penalty of the rejected jobs is exactly $\sum_{J_j \in R} w_j = B$, as the maximum load is $2B$ and the objective value is $3B$. Let $S = \{a_j | J_j \in R\}$. Then, $\sum_{a_j \in S} a_j = \sum_{J_j \in R} w_j = B$, implying that S is a feasible solution of instance I. Since 2-Partition problem is NP-hard, Theorem 1 follows. □

3 Approximation algorithms for arbitrary d

3.1 A greedy *d*-approximation algorithm

For every $j = 1, 2, \ldots, n$, let $p_j = \sum_{k=1}^{d} p_j^k$ be the synthetical demand of job J_j. Our *greedy* algorithm is to reject all the jobs such that $w_j \le p_j$, i.e., $A = \{J_j | w_j > p_j\}$ and $R = \{J_j | w_j \le p_j\}$. Clearly, greedy algorithm can be implemented in $O(nd)$ time.

Theorem 2 *The greedy algorithm produces a solution with objective value no more than dZ^{OPT}, where Z^{OPT} is the optimal value. The ratio d is tight.*

Proof Let A^* be the set of accepted jobs in the optimal solution, and $R^* = J \setminus A^*$. Clearly, the optimal value is

$$Z^{OPT} = \max_k \sum_{j \in A^*} p_j^k + \sum_{j \in R^*} w_j.$$

Since $A = (A \cap A^*) \cup (A \cap R^*)$, where $A \cap A^*$ and $A \cap R^*$ are two disjoint sets, for each $k = 1, \ldots, d$, the load of resource k in the greedy solution is

$$L^k = \sum_{J_j \in A} p_j^k = \sum_{J_j \in A \cap A^*} p_j^k + \sum_{J_j \in A \cap R^*} p_j^k,$$

Thus the total load is

$$
\begin{aligned}
\sum_{k=1}^{d} L^k &= \sum_{J_j \in A \cap A^*} \sum_{k=1}^{d} p_j^k + \sum_{J_j \in A \cap R^*} \sum_{k=1}^{d} p_j^k \\
&= \sum_{J_j \in A \cap A^*} p_j + \sum_{J_j \in A \cap R^*} p_j \\
&\le \sum_{J_j \in A \cap A^*} p_j + \sum_{J_j \in A \cap R^*} w_j,
\end{aligned}
$$

where the last inequality follows from the definition of A. Similarly, the total rejection penalty of the greedy solution is

$$\begin{aligned}\sum_{J_j\in R} w_j &= \sum_{J_j\in R\cap A^*} w_j + \sum_{J_j\in R\cap R^*} w_j\\ &\le \sum_{J_j\in R\cap A^*} p_j + \sum_{J_j\in R\cap R^*} w_j.\end{aligned}$$

The objective of the greedy solution is

$$\begin{aligned}\max_k L^k + \sum_{J_j\in R} w_j &\le \sum_{k=1}^{d} L^k + \sum_{J_j\in R} w_j\\ &\le \sum_{J_j\in A^*} p_j + \sum_{J_j\in R^*} w_j\\ &\le d\max_k \sum_{j\in A^*} p_j^k + \sum_{j\in R^*} w_j\\ &\le dZ^{OPT}.\end{aligned}$$

Consider an instance with one job, whose resource demand is $(1, 1, \ldots, 1)$ with synthetical demand $p_j = d$. Its reject penalty is d. Greedy algorithm will reject the job. However, the optimal solution will accept it. Thus, the approximation ratio is exactly d. It implies that the theorem holds. □

3.2 A randomized rounding method

In this subsection, we use a randomized (α, β)-rounding method to produce a feasible solution with objective value no more than $\frac{e}{e-1}Z^{OPT}$. Replacing the integrality conditions $x_j \in \{0, 1\}$ in ILP(1) by $0 \le x_j \le 1$, we obtain the relaxation of ILP(1), whose optimal solution $\tilde{x}_j$ can be found in $O(n^{3.5}|I|^2)$ time (Karmarkar 1984), where $|I|$ is the number of bits in the input instance I. Let α be a given constant in $(0, 1)$. We randomly choose a threshold β from the uniform distribution over $[\alpha, 1]$. Construct an integer solution $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$, where $\bar{x}_j = 1$ if and only if $\tilde{x}_j > \beta$. Clearly, $\bar{\mathbf{x}}$ is a feasible solution. Next, we will analyze the quality of the rounded solution $\bar{\mathbf{x}}$ and determine the constant α.

Lemma 3 *The expected objective value of the solution $\bar{x}$ is no more than $f(\alpha)Z^{OPT}$, where $f(\alpha) = \max\{\frac{\ln\alpha}{\alpha-1}, \frac{1}{1-\alpha}\}$.*

Proof By the definition of $\bar{\mathbf{x}}$, if $\tilde{x}_j > \beta$, $\bar{x}_j = 1 \le \tilde{x}_j/\beta$, and $\bar{x}_j = 0 \le \tilde{x}_j/\beta$, otherwise. Let $\bar{L}^k = \sum_{j=1}^{n} p_j^k \bar{x}_j$ be the load of resource k in the solution $\bar{x}$. Therefore, for every resource $k = 1, 2, \ldots, d$, we have

$$\begin{aligned}
E(\bar{L}^k) &= \sum_{j=1}^{n} p_j^k E\left(\bar{x}_j\right) \le \sum_{j=1}^{n} p_j^k E\left(\frac{\tilde{x}_j}{\beta}\right) \\
&= \sum_{j=1}^{n} p_j^k \tilde{x}_j E(\frac{1}{\beta}) = \tilde{L}^k \int_{\alpha}^{1} \frac{1}{\beta}\frac{1}{1-\alpha} d\beta \\
&= \frac{\ln\alpha}{\alpha-1}\tilde{L}^k,
\end{aligned}$$

where $\tilde{L}^k = \sum_{j=1}^{n} p_j^k \tilde{x}_j$ is the load of resource k in the solution $\tilde{x}$.

If $\tilde{x}_j \le \alpha$, we have $\bar{x}_j = 0$, which implies that

$$E(w_j(1-\bar{x}_j)) = w_j \le \frac{1}{1-\alpha} w_j(1-\tilde{x}_j).$$

If $\tilde{x}_j > \alpha$, we have

$$E(w_j(1-\bar{x}_j)) = w_j \cdot Pr(\tilde{x}_j \le \beta) + 0 \cdot Pr(\tilde{x}_j > \beta) = \frac{1}{1-\alpha} w_j(1-\tilde{x}_j).$$

Thus, the expected objection value of $\bar{\mathbf{x}}$ is

$$\begin{aligned}
E(\bar{Z}) &= E\left(\max_k \bar{L}^k + \sum_{j=1}^{n} w_j(1-\bar{x}_j)\right) \\
&\le \frac{\ln\alpha}{\alpha-1}\max_k \tilde{L}^k + \frac{1}{1-\alpha}\sum_{j=1}^{n} w_j(1-\tilde{x}_j) \\
&\le \max\left\{\frac{\ln\alpha}{\alpha-1}, \frac{1}{1-\alpha}\right\}\left(\max_k \tilde{L}^k + \sum_{j=1}^{n} w_j(1-\tilde{x}_j)\right) \\
&\le \max\left\{\frac{\ln\alpha}{\alpha-1}, \frac{1}{1-\alpha}\right\} Z^{OPT},
\end{aligned}$$

where the last inequality follows from that the optimal value of the relaxation of ILP(1) is a lower bound on Z^{OPT}. □

Theorem 4 *There is a $\frac{e}{e-1}$-approximation algorithm in deterministic polynomial time for the VSR problem on a single machine.*

Proof Consider the ratio $f(\alpha) = \max\{\frac{\ln\alpha}{\alpha-1}, \frac{1}{1-\alpha}\}$. By solving $\frac{\ln\alpha}{\alpha-1} = \frac{1}{1-\alpha}$, we have $\alpha = 1/e$. Thus, $f(\alpha)$ achieves its minimum value $\frac{e}{e-1}$ at $\alpha = 1/e$. Therefore, by setting $\alpha = 1/e$, we obtain a randomized polynomial time $\frac{e}{e-1}$-approximation algorithm.

Now, we show how to obtain a determinant polynomial time $\frac{e}{e-1}$-approximation algorithm. Without loss of generality, assume that $\tilde{x}_1 \le \tilde{x}_2 \le \cdots \le \tilde{x}_n$. Note that the randomized method will produce the same solution $(0, \ldots, \bar{x}_i = 0, \bar{x}_{i+1} = 1, \ldots, 1)$, when β lies in $(\tilde{x}_i, \tilde{x}_{i+1})$ with $\tilde{x}_i \geq 1/e$. In other words, there are at most $n+1$ different solutions with expected objective value no more than $\frac{e}{e-1}Z^{OPT}$. Thus, choosing the solution with minimum objective value, we obtain the desired solution in polynomial time. The theorem holds. □

4 An FPTAS for fixed d

We use dynamic programming to design a fully polynomial time approximation scheme for the VSR problem where d is fixed.

Theorem 5 *The VSR problem on a single machine can be solved in time $O(n^{d+1} p_{max}^d)$, where $p_{max} = \max_{j,k} p_j^k$.*

Proof For convenience, let $p_{max} = \max_{j,k} p_j^k$. Clearly, the load of any dimension is no more than np_{max}. We use $W_j(L_1, \ldots, L_d)$ to denote the minimum penalty cost when the load of machine is $(L_1, \ldots, L_d)$ among $J_1, J_2, \ldots, J_j$. Dynamic programming algorithm is described as follows.

Initially, $W_0(0, 0, \ldots, 0) = 0$, and $W_0(L_1, L_2, \ldots, L_d) = +\infty$, for $(L_1, L_2, \ldots, L_d) \neq (0, 0, \ldots, 0)$.

For $j = 1, 2, \ldots, n$,

$W_j(L_1, \ldots, L_d) = \min\{w_j + W_{j-1}(L_1, \ldots, L_n), W_{j-1}(L_1 - p_j^1, \ldots, L_d - p_j^d)\}$;

Let $Z_n(L_1, \ldots, L_d) = W_n(L_1, \ldots, L_d) + \max_k L_k$ be the objective value when the load of machine is $(L_1, \ldots, L_d)$. The optimal value is given by $\min\{Z_n(L_1, \ldots, L_d) : 0 \leq L_k \leq np_{max}\}$.

Since L_k belongs to $\{0, 1, 2, \ldots, np_{max}\}$ at every iteration, we need to compute the value $W_j(L_1, \ldots, L_d)$ in $O((np_{max} + 1)^d)$ time for each job j. Thus, VSR on a single machine can be solved in time $O(n(np_{max} + 1)^d) = O(n^{d+1} p_{max}^d)$. □

Theorem 6 *There is a FPTAS with running time $O(n^{d+1}(d/\epsilon)^d)$ for VSR on a single machine when d is fixed.*

Proof Let Z be the objective value of the solution produced by the greedy algorithm in Sect. 3.1. By Theorem 2, we have

$$Z^{OPT} \leq Z \leq dZ^{OPT}. \tag{4}$$

Construct an instance $\hat{I}$ with

$$\hat{p}_j^k = \left\lfloor \frac{p_j^k}{\frac{\epsilon Z}{dn}} \right\rfloor \frac{\epsilon Z}{dn}, \quad \hat{w}_j = \left\lfloor \frac{w_j}{\frac{\epsilon Z}{dn}} \right\rfloor \frac{\epsilon Z}{dn}$$

Then, we use the dynamic programming algorithm showed in the proof of Theorem 5, and obtain an optimal solution $(\hat{A}, \hat{R})$ for instance $\hat{I}$. Clearly, $Z^{OPT}(\hat{I}) \leq Z^{OPT}$, as $\hat{p}_j^k \leq p_j^k$ and $\hat{w}_j \leq w_j$ for every j, k. Replacing the modified demand $\hat{p}_j^k$ (rejection penalty $\hat{w}_j$) by the original resource demand p_j^k (rejection penalty w_j), we obtain that the objective value of the solution $(\hat{A}, \hat{R})$ for instance I is

$$\begin{aligned}\max_k \sum_{J_j\in\hat{A}} p_j^k + \sum_{J_j\in\hat{R}} w_j &\le \max_k \sum_{J_j\in\hat{A}} \left(\hat{p}_j^k + \frac{\epsilon Z}{dn}\right) + \sum_{J_j\in\hat{R}} \left(\hat{w}_j + \frac{\epsilon Z}{dn}\right)\\ &\le Z^{OPT}(\hat{I}) + n\cdot\frac{\epsilon Z}{dn}\\ &\le (1+\epsilon)Z^{OPT},\end{aligned}$$

where the last inequality follows from (4).

In the dynamic programming algorithm, we only need to consider the solution with maximum load no more than Z as $Z^{OPT}(\hat{I}) \le Z^{OPT} \le Z$, and each value L_k is an integer multiple of $\frac{\epsilon Z}{dn}$. It implies that instance $\hat{I}$ can be solved in time $O(n(dn/\epsilon+1)^d) = O(n^{d+1}(d/\epsilon)^d)$. Thus, the theorem holds. □

5 Conclusion

In this paper, we proposed the VSR problem on a single machine. We proved this problem is NP-hard and designed some approximation algorithms. It is challenging to design a PTAS for VSR on a single machine or prove that it is APX-hard when d is arbitrary. Also, it is desired to design approximation algorithms for VSR on parallel machines.

Acknowledgements We are grateful to the anonymous referees for numerous helpful comments and suggestions which helped to improve the presentation of our work. The work is supported in part by the National Natural Science Foundation of China [Nos. 61662088, 11301466], the Natural Science Foundation of Yunnan Province of China [No. 2014FB114], IRTSTYN, and Program for Excellent Young Talents, Yunnan University.

References

Alon N, Azar Y, Woeginger GJ, Yadid T (1998) Approximation schemes for scheduling on parallel machines. J Sched 1:55–66

Bansal N, Oosterwijk T, Vredeveld T, Zwaan R (2016) Approximating vector scheduling: almost matching upper and lower bounds. Algorithmica 76(4):1077–1096

Bartal Y, Leonardi S, Spaccamela AM, Sgall J, Stougie L (2000) Multiprocessor scheduling with rejection. SIAM J Discrete Math 13(1):64–78

Bonifaci V, Wiese A (2012) Scheduling unrelated machines of few different types. arXiv:1205.0974vl,

Chekuri C, Khanna S (2004) On multidimensional packing problems. SIAM J Comput 33(4):837–851

Chen L, Jansen K, Zhang G (2014) On the optimality of approximation schemes for the classical scheduling problem. In: Proceedings of the 25th annual ACM-SIAM symposium 19 on discrete algorithms (SODA), pp 657–668

Christensen HI, Khan A, Pokutta S, Tetali P (2017) Approximation and online algorithms for multidimensional bin packing: a survey. Comput Sci Rev 24:63–79

Epstein L, Tassa T (2003) Vector assignment problems: a general framework. J Algorithms 48(2):360–384

Epstein L, Tassa T (2006) Vector assignment schemes for asymmetric settings. Acta Inform 42(6–7):501–514

Garey MR, Johnson DS (1979) Computers and intractablity: a guide to the theory of NP-completeness. Freeman, San Francisco

He C, Leung JYT, Lee K, Pinedo ML (2016) Improved algorithms for single machine scheduling with release deates and rejections. 4OR Q J Oper Res 14(1):41–55

Hochbaum DS, Shmoys DB (1987) Using dual approximation algorithms for scheduling theoretical and practical results. J ACM 34(1):144–162

Im S, Kell N, Kulkarni J, Panigrahi D (2015) Tight bounds for online vector scheduling. In: IEEE 56th annual symposium on foundations of computer science (FOCS), pp 525–544

Jansen K, Klein KM, Verschae J (2016) Closing the gap for makespan scheduling via sparsification techniques. arXiv preprint arXiv:1604.07153

Karmarkar N (1984) A new polynomial-time algorithm for linear programming. In: Proceedings of the sixteenth annual ACM symposium on theory of computing (STOC), pp 302–311

Li W, Li J, Zhang X, Chen Z (2015) Penalty cost constrained identical parallel machine scheduling problem. Theoret Comput Sci 607:181–192

Meyerson A, Roytman A, Tagiku B (2013) Online multidimensional load balancing. Lect Notes Comput Sci 8096:287–302

Ou J, Zhong X, Li C-L (2016) Faster algorithms for single machine scheduling with release dates and rejection. Inf Process Lett 116(8):503–507

Ou J, Zhong X, Wang G (2015) An improved heuristic for parallel machine scheduling with rejection. Eur J Oper Res 241(3):653–661

Shabtay D, Gaspar N, Kaspi M (2013) A survey on offline scheduling with rejection. J Sched 16(1):3–28

Shabtay D, Gaspar N, Yedidsion L (2012) A bicriteria approach to scheduling a single machine with job rejection and positional penalties. J Comb Optim 23(4):395–424

Slotnick SA (2011) Order acceptance and scheduling: a taxonomy and review. Eur J Oper Res 212:1–11

Zhang L, Lu L (2016) Parallel-machine scheduling with release dates and rejection. 4OR Q J Oper Res 14:165–172

Zhang L, Lu L, Yuan J (2009) Single machine scheduling with release dates and rejection. Eur J Oper Res 198(3):975–978

Zhang L, Lu L, Yuan J (2010) Single-machine scheduling under the job rejection constraint. Theoret Comput Sci 411(16–18):1877–1882

Zhong X, Ou J (2016) Improved approximation algorithms for parallel machine scheduling with release dates and job rejection. 4OR Q J Oper Res. doi:10.1007/s10288-016-0339-6.

Zhong X, Pan Z, Jiang D (2017) Scheduling with release times and rejection on two parallel machines. J Comb Optim 33(3):934–944

Zhu X, Li Q, Mao W, Chen G (2014) Online vector scheduling and generalized load balancing. J Parallel Distrib Comput 74(4):2304–2309