Single product lot-sizing on unrelated parallel machines with non-decreasing processing times

A Eremeev¹³, M Kovalyov² and P Kuznetsov³

¹Sobolev Institute of Mathematics, Siberian Branch of Russian Academy of Sciences, Pevtsov 13, 644099, Omsk, Russia
²United Institute of Informatics Problems, National Academy of Sciences of Belarus, Surganova 6, 220012, Minsk, Belarus.
³Omsk State University n.a. F.M. Dostoevsky, Mira 55a, 644077, Omsk, Russia

Abstract. We consider a problem in which at least a given quantity of a single product has to be partitioned into lots, and lots have to be assigned to unrelated parallel machines for processing. In one version of the problem, the maximum machine completion time should be minimized, in another version of the problem, the sum of machine completion times is to be minimized. Machine-dependent lower and upper bounds on the lot size are given. The product is either assumed to be continuously divisible or discrete. The processing time of each machine is defined by an increasing function of the lot volume, given as an oracle. Setup times and costs are assumed to be negligibly small, and therefore, they are not considered. We derive optimal polynomial time algorithms for several special cases of the problem. An NP-hard case is shown to admit a fully polynomial time approximation scheme. An application of the problem in energy efficient processors scheduling is considered.

1. Introduction

In this paper, we study a problem which arises in scheduling workload on parallel processors or production planning of a single product on a number of parallel machines or reactors. There is a demand for at least $A$ units of a single product to be produced on $m$, $m \geq 2$, unrelated parallel machines in lots. The size of a lot is the number of units this lot includes. It can take either integer value or real value, in which cases the product is called either discrete or continuous, respectively. Machine $i$, $i=1,\ldots,m$, requires $p_i(x)$ time units to produce a lot of size $x$, where $p_i(x)$ is a non-decreasing function with non-negative finite values. If a lot of size $x$ is assigned to machine $i$, then the lot size must satisfy the relation $l_i \leq x \leq u_i$, where $l_i$ and $u_i$ are given lower and upper bounds, $i=1,\ldots,m$.

The problem is to partition at least $A$ units of the product into lots and assign these lots to the machines. In one version of the problem, the maximum machine completion time should be minimized, in another version of the problem the sum of machine completion times is to be minimized. Both criteria are related to the fair distribution of machine workloads. Setup times and costs are assumed to be negligibly small, and therefore, they are not considered. It is assumed that the values $A$ and $u_i$, $i=1,\ldots,m$, are positive rational numbers, and the values $l_i$, $i=1,\ldots,m$, are non-negative rational numbers.

The assumption that lot sizes must belong to the intervals $[l_i,u_i]$, or otherwise equal to zero, may reflect some technical constraints of the machines. In the computational applications, such constraints may be caused by limited RAM capacities on each of the processors (defines $u_i$), minimal cost-effective or minimal time-effective work load on each of the processors (defines $l_i$). In the industrial applications, these constraints may be due to a limited reactor size (defines $u_i$) and minimal amount of product that can be processed in one lot without damaging the equipment or due to economical reasons (defines $l_i$), see e.g. [9].
We denote this problem as $R|1, GT, \beta|\gamma$, where $\beta \in \{\text{cntn}, \text{dscr}\}$, $\gamma \in \{C_\Sigma, C_{\text{max}}\}$. Following the scheduling traditions, notation “R” here refers to unrelated parallel machines. Notation “GT” is used to indicate that at most one lot can be assigned on the same machine (GT stands for Group Technology). Abbreviations “cntn” and “dscr” specify continuous and discrete product, respectively. Maximum machine completion time and total machine completion time are denoted as “$C_{\text{max}}$” and “$C_\Sigma$”, respectively.

1.1. Mathematical programming problem formulation
The following mathematical programming formulation for the problem $R|1, \alpha, \beta|\gamma$ can be given.

$$\min \sum_{i=1}^{m} p_i(x_i) , \quad \text{if } \gamma = C_\Sigma$$  \hspace{1cm} (1)

$$\min f_{\text{max}} , \quad \text{if } \gamma = C_{\text{max}}$$  \hspace{1cm} (1')

s.t.

$$f_{\text{max}} \geq p_i(x_i), \quad i = 1, \ldots, m , \quad \text{if } \gamma = C_{\text{max}}$$  \hspace{1cm} (2)

$$\sum_{i=1}^{m} x_i \geq A$$  \hspace{1cm} (3)

$$z_i l_i \leq x_i \leq z_i u_i, \quad i = 1, \ldots, m$$  \hspace{1cm} (4)

$$x_i \in Q_i, \text{ if } \beta = \text{cntn}$$  \hspace{1cm} (5)

$$x_i \in Z_i, \text{ if } \beta = \text{dscr}$$  \hspace{1cm} (5')

$$z_i \in \{0,1\}.$$  \hspace{1cm} (6)

Here $Q_i$ denotes the set of non-negative rational numbers and $Z_i$ is the set of non-negative integers. The variables are the maximum machine completion time $f_{\text{max}}$, the production volume $x_i$ on machine $i$, and the number of lots $z_i$ on machine $i$, $i = 1, \ldots, m$. Relation (3) ensures that the required quantity of the product is assigned to the machines. Relations (4) connect production volume $x_i$ with the number of lots $z_i$ on each machine $i$. Conditions (5) and (5') address the assumptions that the product is either continuously divisible (5) or discrete (5').

1.2. Related research
The problem formulated above is closely related to the problem studied by Dolgui et al. [4], in which there are several products, machine-dependent lot size lower bounds, machine and sequence dependent setup times, and the objective is to minimize $C_{\text{max}}$. This problem is strongly NP-hard due to a reduction from the well-known strongly NP-hard Traveling Salesman Problem. It was proved to be NP-hard in the ordinary sense even if the number of products $n=2$. Several dynamic programming algorithms for the special cases of the problem were developed.

The so-called Supply Scheduling Problem (SSP) is also closely related to the problem considered in this paper. In SSP, there are $m$ providers that supply a certain product to a manufacturing unit. If provider $i$ is not used, then the corresponding delivered quantity is $x_i=0$. If provider $i$ is used, then the delivered quantity $x_i$ must be between the given lower and upper bounds $l_i$ and $u_i$. The demand at the manufacturing unit is $A$. The delivery cost for sending a quantity $x_i$ from provider $i$ to the manufacturing unit is $c_i(x_i)$, where $c_i(\cdot)$ is a cost function which can be linear as e.g. in Chauhan et al. [2], or can be given by an oracle as in Chauhan et al. [3] and Ng et al. [8]. The goal is to minimize the total delivery cost, subject to the condition that the manufacturing demand is satisfied. The SSP is NP-hard in the ordinary sense and several FPTASes are proposed for different versions of this problem in [2,3,8]. A Fully Polynomial Time Approximation Scheme (FPTAS) is a collection of $(1+\varepsilon)$-
approximation algorithms Aε such that algorithm Aε guarantees the relative error ε and it runs in time polynomial in 1/ε and in the problem instance length in binary encoding. A generalization of SSP to the case of concave non-decreasing cost functions and several feasible intervals for the delivered quantity was studied by Eremeev et al. [5] and an FPTAS was developed.

In this paper, we derive optimal polynomial time algorithms for several special cases of the problem \( R[\alpha, \beta, \gamma] \). For an NP-hard case of the problem we provide a fully polynomial time approximation scheme. The obtained results extend and/or improve the algorithmic results on \( R[\alpha, \beta, \gamma] \) from [6], where the processing times were assumed to be proportional to the lot sizes and all the numeric parameters \( \alpha, \beta, \gamma \) and \( p_i \) were assumed to be integers. Note that under the assumption of processing times being proportional to lot sizes, the problem may be considered as a special case of an applied production scheduling problem in chemical industry [9] or as a special case of fuel supply management problem [1]. In [6], we also considered another version of the problem, denoted \( R[\alpha, \beta, \gamma] \), where each machine may produce more than one lot, assuming that all processing times are proportional to workload assigned to a machine. The exact and approximate algorithms obtained there for \( R[\alpha, \beta, \gamma] \) may be extended straightforwardly to the case of non-decreasing continuous processing times, therefore we do not give a detailed analysis of \( R[\alpha, \beta, \gamma] \) here.

1.3. Application to energy efficient scheduling
The assumption that for each machine \( i \), \( p_i(x) \) is a non-decreasing function of the lot volume \( x \) allows us to consider an application of \( R[\alpha, \beta, \gamma] \) in the so-called energy efficient scheduling [7]. Suppose that the machines are speed-scalable processors, i.e. the processing time of each machine is \( x/s_i \), where \( s_i \) is a tunable processing speed (subject to optimization). When a processor runs at a speed \( s \), its energy consumption is \( s^\omega \) units of energy per time unit. The parameter \( a_i \) here is a constant for each processor (practical studies show that \( 1 < a_i \leq 3 \)). Suppose that the total amount of work (e.g. computation) \( A \) needs to be performed on a set of \( m \) unrelated parallel processors whose workloads are either zero or between \( l_i \) and \( u_i \) and all computations should be finished by a given deadline \( d \). The objective is to minimize the total energy consumption. Then allocation of a lot \( x_i \) to processor \( i \) implies that the minimal energy consumption \( E_i(x_i) \) over \( d \) time units on this processor is attained if it works at the minimal speed \( s=x_i/d \), therefore \( E_i(x_i) = (x_i/d)^\omega d \), which is a convex increasing function of \( x_i \) and may substitute \( p_i(x) \) in the problem formulation when \( R[\alpha, \beta, \gamma] \) is applied to energy efficient scheduling as described above. In this case, \( C_\Sigma \) criterion implies minimization of total energy consumption and \( C_{\max} \) implies minimization of maximal energy consumption over all processors.

The paper is structured as follows. In the next section, we briefly discuss the complexity and approximability of the problem with the criterion of total machine completion time minimization for discrete and continuous products. Section 3 is devoted to the makespan minimization and contains the main contributions of this paper, namely the polynomial-time exact algorithms of different time complexity for several special cases of the problem. Section 4 gives concluding remarks and further research directions.

2. Minimizing total machine completion time
For the problem \( R[\alpha, \beta, \gamma] \) with continuous non-decreasing non-negative functions \( p_i(x) \), we note that it is equivalent to the earlier studied Supply Scheduling Problem with continuous non-decreasing cost functions. The latter problem is NP-hard [3] but admits an FPTAS, as shown in [3,8]. In this case, we have to assume that all functions \( p_i(x) \) are given in the problem input as oracles, the same applies to the supplementary functions \( q_i(C) := \sup \{x; p_i(x) \leq C \} \), and all of these oracles are polynomial-time computable.

3. Minimizing maximum machine completion time
In this section, we describe a polynomial-time algorithm for optimal solving the problem \( R[\alpha, \beta, \gamma] \) under the assumption that the processing times are given by piece-wise linear increasing convex functions \( p_i(x), i=1, ..., m \).
3.1. Continuous product

First we describe an algorithm for problem $R[1,GT,cntn]C_{\text{max}}$. We begin with computing values $p(l_i)$ and $p(u_i)$, $i=1,\ldots,m$, and sorting these values in a non-decreasing order. Let $D_1,\ldots,D_t$, $i,\leq 2m$, denote the set of all distinct values $p(l_i)$ and $p(u_i)$, ordered so that $D_1<\ldots< D_t$. Denote the optimal makespan value by $C^*$. It is easy to see that there exists an index $k^*$ such that either (a) $C^*=D_{k^*}$, $1\leq k^* \leq t$, or (b) $D_{k^*}< C^*< D_{k^*+1}$, $1\leq k^* \leq t-1$. Assuming for a while that $k^*$ is known, let us partition the set of machines into the following three subsets: subset $Before_{k^*}$ consists of machines $i$, for which $p(u_i)<D_{k^*}$; subset $Between_{k^*}$ consists of machines $i$, for which $p(l_i) \leq D_{k^*}< p(u_i)$ in the case (a) and $p(l_i) \leq D_{k^*}< D_{k^*+1}\leq p(u_i)$ in the case (b); and subset $After_{k^*}$ consists of machines $i$, for which $D_{k^*+1}< p(l_i)$. These three definitions account for all possible locations of the interval $[D_k,D_{k+1}]$ with respect to the interval $[p(l_i),p(u_i)]$. In both cases (a) and (b), if $k^*$ is known, then there exists an optimal solution $x^*$ such that $x^*_i = u_i$ for all $i \in Before_{k^*}$, $x^*_i = 0$ for all $i \in After_{k^*}$ and $x^*_i = \max\{x_i | p(x_i) \leq C^*\}$ for all $i \in Between_{k^*}$. Furthermore, in the case (b), $\sum_{i=1}^{m} x^*_i = A$.

Index $k^*$ and the corresponding values $C^*$ and $x^*_i$, $i=1,\ldots,m$, can be found by the enumeration of the index $k=1,\ldots,t$. Consider iteration $k$ of this enumeration. To handle the case (a), calculate $x^*_i = u_i$, $i \in Before_{k^*}$, $x^*_i = 0$, $i \in After_{k^*}$ and $x^*_i = \max\{x_i | p(x_i) \leq D_k, x_i \in [l_i,u_i]\}$, $i \in Between_{k^*}$. If $\sum_{i=1}^{m} x^*_i \geq A$, then set $C^{(a,k)}:=D_k$, otherwise set $C^{(a,k)}:=\infty$. To handle the case (b), calculate $x^*_i = u_i$, $i \in Before_{k^*}$ and $x^*_i = 0$, $i \in After_{k^*}$. Values $C^*$ and $x^*_i$, $i \in Between_{k^*}$ are found as optimal solution values of the following mathematical programming problem, denoted as MMP.

$$\text{Min } C,$$

s.t.

$$p(x_i) \leq C,$$

$$\sum_{i \in Before_{k^*}} u_i + \sum_{i \in Between_{k^*}} x_i = A,$$

$$x_i \in [l_i,u_i], i \in Between_{k^*}.$$

If MMP has a solution, then we set $C^{(b,k)}$ to be equal to its optimal value. Otherwise we set $C^{(b,k)}=\infty$. It is easy to verify that

$$C^* = \min\{C^{(a,k)}, C^{(b,k)} | k=1,\ldots,t\}.$$

Note that the subsets $Before_{k+1}$, $Between_{k+1}$ and $After_{k+1}$ may be computed on the basis of subsets $Before_{k}$, $Between_{k}$ and $After_{k}$ so that the total time for finding all subsets $Before_{k}$, $Between_{k}$ and $After_{k}$, $k=1,\ldots,t-1$ is $O(m)$.

MMP can be converted into a linear programming problem if all $p(x_i)$ are convex increasing piecewise linear functions. Indeed, suppose that

$$p(x_i) = \sum_{0 \leq j \leq K_i} (\tau^{(j+1)}_i - \tau^{(j)}_i) p^{(j)}_i + (x_i - \tau^{(j)}_i) p^{(j)}_i$$

whenever $x_i \in [\tau^{(j)}_i, \tau^{(j+1)}_i]$, assuming that $\tau^{(1)}_i,...,\tau^{(K)}_i$ are the points that separate $K_i$ linear segments of $p(x_i)$, where $p^{(1)}_i < \ldots < p^{(K)}_i$ are the linear coefficients of the segments. We assume that $\tau^{(0)}_i = 0$ and
Let $r^{(K)}_i = u_i$. Then the supplementary problem reduces to the following linear programming problem, where each variable $x^{(j)}_i$ measures the usage of $j$-th segment $[r^{(j-1)}_i, r^{(j)}_i]$, $j = 1, \ldots, K_i, i \in \text{Between}_i$:

\[
\begin{align*}
\text{Min } C, \\
\text{s.t.} \\
\sum_{j=1}^{K_i} x^{(j)}_i p^{(j)}_i \leq C, & \quad i \in \text{Between}_i, \\
\sum_{i \in \text{Before}_i} u_i + \sum_{i \in \text{Between}_i} \sum_{j=1}^{K_i} x^{(j)}_i \geq A, \\
x^{(j)}_i \in [0, r^{(j+1)}_i - r^{(j)}_i], & \quad i \in \text{Between}_i, j = 1, \ldots, K_i - 1.
\end{align*}
\]

Since linear programming problems are polynomially solvable, the following proposition holds.

**Proposition 1.** If all processing times $p_i(x_i)$ are convex non-negative increasing piece-wise linear functions, then problem $R[1, GT, cntn]_{C_{\max}}$ can be solved in polynomial time.

If the processing times are proportional to the lot size, i.e. $p_i(x_i) = x_i p_i$ for some given positive scalars $p_1, \ldots, p_m$ then the supplementary linear programming problems are solvable analytically [6] which allows to improve the runtime in this special case as follows.

**Proposition 2 [6].** In the special case where the processing times are proportional to lot sizes, the problem $R[1, GT, cntn]_{C_{\max}}$ is solvable in $O(m \log m)$ time.

### 3.2. Discrete Product

The discrete problem $R[1, GT, dscr]_{C_{\max}}$ can be solved by a bisection search w.r.t. $C_{\max}$ criterion. The initial lower bound for $C_{\max}$ is equal to 0 and the initial upper bound $UB_0$ for $C_{\max}$ is $\max_{i=1, \ldots, m} p_i(u_i)$. In case the processing times are given by polynomial-time computable oracles, the upper bound $UB_0$ is also upper bounded by $2^{\text{poly}(|I|)}$, where $|I|$ denotes the problem input length and $\text{poly}(\cdot)$ is some polynomial. On each iteration of the bisection algorithm, when a new tentative value of criterion $C$ is chosen by the bisection, in a polynomially bounded time we compute $m$ values $q(C) := \max \{x_i \in \mathbf{Z}_+ : p(x_i) \leq C\}$, $i = 1, \ldots, m$, to find the lot sizes corresponding to $C$. If some of these lot sizes turn out to be outside the feasible region, they need to be adjusted either by setting to 0 (if $q(C) < l$) or by setting to $u_i$ (if $q(C) > u_i$). After such adjustments, we compute the processing time of an adjusted lot in order to update the bisection search interval. The standard justification of the bisection search algorithms leads to the following proposition.

**Proposition 3.** If all processing times $p_i(x_i)$ are non-negative non-decreasing functions given by polynomial-time computable oracles, then problem $R[1, GT, dscr]_{C_{\max}}$ can be solved in polynomial time.

Let us now consider the special case of $R[1, GT, dscr]_{C_{\max}}$ where the processing times are proportional to lot sizes.

We will show now that the discrete problem $R[1, GT, dscr]_{C_{\max}}$ can be solved in $O(m^3)$ time by an iterative solution of $R[1, GT, cntn]_{C_{\max}}$ problem.
Let $LB^0$ and $C^*$ denote the makespan values of optimal solutions to a pair of identical instances of $R|1,GT,cntn|C_{\text{max}}$ and $R|1,GT,dscr|C_{\text{max}}$ respectively. Let $x^0$ be an optimal solution to the instance of $R|1,GT,cntn|C_{\text{max}}$. Obviously, $LB^0 \leq C^*$. At the same time, the vector $\lceil x^0 \rceil := (\lceil x_1^0 \rceil, \ldots, \lceil x_n^0 \rceil)$ is a feasible solution to the discrete instance, therefore the makespan of $\lceil x^0 \rceil$, denoted by $UB^0$, is not less than $C^*$.

Without a loss of generality, we assume that in $\lceil x^0 \rceil$ the makespan $UB^0$ is attained on machine 1. Two alternatives are possible: (A) $C^* = UB^0$ or (B) $LB^0 \leq C^* < UB^0$. In the case (A), the vector $\lceil x^0 \rceil$ is an optimal solution.

Consider alternative (B). In an optimal solution to the discrete problem, machine 1 completes at some moment $T_1 \leq C^*$. Note that there exists an optimal solution $x^*$ to the instance of $R|1,GT,dscr|C_{\text{max}}$, such that $x_i^* \geq \lceil x_i^0 \rceil$, since vector $x^0$ yields a lower bound $LB^0$ to the optimum $C^*$. In what follows, we will assume that an optimal solution $x^*$ satisfies this condition.

Let us round down $x_i^0$, fix it, and keep the rest of the components in $x^0$ unchanged for a while. In order to satisfy constraint (3), which may be violated by the modified vector $x^0$, we solve a new supplementary instance of $R|1,GT,cntn|C_{\text{max}}$ w.r.t. variables $x_2, \ldots, x_m$, where the right-hand side $A$ is decreased by $\lceil x_1^0 \rceil - x_1^*$. Now let $LB^1$ be an optimal makespan value in the new $R|1,GT,cntn|C_{\text{max}}$ subproblem. Note that $LB^1 \leq C^*$.

The iterative process continues. On the second iteration we round up the last optimal solution to the supplementary instance of $R|1,GT,cntn|C_{\text{max}}$ and denote the rounded solution as $\lceil x^1 \rceil$. Find the corresponding makespan value $UB^1$. Without loss of generality we assume that $UB^1$ is attained on machine 2 and continue as we did with machine 1.

The total number of iterations is at most $m$. Note that in case at some iteration alternative (A) takes place, all subsequent subproblems $R|1,GT,cntn|C_{\text{max}}$ may turn out to be infeasible.

An optimal solution to $R|1,GT,dscr|C_{\text{max}}$ is now obtained either as a vector consisting of all fixed values of coordinates $x_i^*$ or as one of the vectors $\lceil x^l \rceil$, if at some iteration the alternative (A) actually took place. It is sufficient to compare the makespan values of all these vectors and find the least of them to identify an optimal solution.

Note that in the algorithm described above, it is sufficient to sort the values $D_i$ just once, which takes $O(m \log m)$ operations. After that, we use the remaining part of the algorithm from Proposition 2 at most $m$ times, which has the overall time complexity $O(m)$. So we have

**Proposition 4.** In the special case, where processing times are proportional to lot sizes, the problem $R|1,GT,dscr|C_{\text{max}}$ is solvable in time $O(m^3)$.

4. Conclusions and discussion
We have extended the previous results, known for $R|1,GT,f\alpha|\gamma$ problem with integer input data and processing times proportional to lot sizes, to more general cases where the processing times are given by oracle computable functions and the numeric parameters of the problem are rational numbers. The obtained results are applicable in logistics, production scheduling and scheduling parallel processors, in particular, for the speed-scalable processors.

The further research might be undertaken in analysis of computational complexity and development of efficient exact and approximation algorithms for the case of more than one product and other generalizations of the problem $R|1,GT,f\alpha|\gamma$. Some technical assumptions made in this paper may be possible to relax. In particular, it is plausible that the exact algorithm proposed for $R|1,GT,cntn|C_{\text{max}}$ in Section 3 under the assumption of convex and piece-wise linear processing times may be generalized to a wider class of convex functions, still keeping the runtime polynomially bounded. We also expect that the continuity assumption on functions for processing times, used in Section 2, may be slightly relaxed.

Acknowledgments
The first author is supported by the Program of fundamental scientific research of the State Academies of Sciences, task I.5.1.6. The second author is supported by a joint project of Belarusian Fund for
Fundamental Research and Siberian Branch of Russian Academy of Sciences "Combinatorial models and methods of Scheduling Theory and Their Applications in Decision Support Systems".

References

[1] Austin L M and Hogan W W 1976 Optimizing the procurement of aviation fuels Management Science 22 5 pp 515–527
[2] Chauhan S S, Eremeev A V, Romanova A A and Servakh V V 2007 Approximation solution of the supply management problem J. of Appl. and Industrial Math. 1 4 pp 433–441
[3] Chauhan S S, Eremeev A V, Romanova A A, Servakh V V and Woeginger G J 2005 Approximation of the supply scheduling problem Oper. Res. Lett. 33 3 pp 249–254
[4] Dolgui A, Eremeev A V, Kovalyov M Y and Kuznetsov P M 2010 Multi-product lot sizing and scheduling on unrelated parallel machines IIE Transactions 42 7 pp 514–524
[5] Eremeev A V, Kovalyov M Y and Kuznetsov P M 2008 Approximate solution of the control problem of supplies with many intervals and concave cost functions Automation and Remote Control 69 7 pp 1181–1187
[6] Eremeev A V, Kovalyov M Y and Kuznetsov P M 2016 Scheduling the single-item production on parallel machines with lot size bounds (in Russian) Proc. 7th Int. Conf. “Tanaev’s Readings” (28–29 March 2016, Minsk) pp 72–76
[7] Gerards M E T, Hurink J L and Holzenspies P K F 2016 A survey of offline algorithms for energy minimization under deadline constraints J. of Sched. 19 pp 3–19
[8] Ng C T, Kovalyov M Y and Cheng T C E 2008 An FPTAS for a supply scheduling problem with non-monotone cost functions Naval Res. Logistics 55 pp 194–199
[9] Shaik M A, Floudas C A, Kallrath J and Pitz H-J 2009 Production scheduling of a large-scale industrial continuous plant: short-term and medium-term scheduling Computers and Chem. Engineering 33 8 pp 670–686